aws

ユーザーガイド

Amazon EKS

Amazon EKS: ユーザーガイド

Table of Contents

Amazon EKS とは

Amazon Elastic Kubernetes Service (Amazon EKS) は、Amazon Web Services (AWS) 上で、独自の Kubernetes コントロールプレーンをインストール、運用、保守する必要がないマネージド型サービスです。Kubernetes は、コンテナ化されたアプリケーションの管理、スケーリング、デプロイを自動化するオープンソースシステムです。

Amazon EKS の機能

Amazon EKS の主な特徴を以下に挙げます。

安全なネットワーキングと認証

Amazon EKS は Kubernetes ワークロードと AWS ネットワーキングおよびセキュリティサービスを統合します。また、AWS Identity and Access Management (IAM) と統合し、Kubernetes クラスターに認証を提供します。

クラススケーリングが簡単

Amazon EKS では、Kubernetes クラスターをワークロードの需要に応じて、簡単にスケールアップおよびスケールダウンできます。Amazon EKS は CPU またはカスタムメトリクスに基づく水平方向の Pod のオートスケーリングや、ワークロード全体の需要に基づくクラスターのオートスケーリングをサポートします。

マネージド Kubernetes エクスペリエンス

`eksctl`、AWS Management Console、AWS Command Line Interface (AWS CLI)、API、`kubectl`、Terraform を使用して、Kubernetes クラスターを変更できます。

高可用性

Amazon EKS は複数のアベイラビリティーゾーンにまたがるコントロールプレーンの高可用性を提供します。

AWS サービスとの統合

Amazon EKS は他の AWS サービスと統合し、コンテナ化されたアプリケーションをデプロイおよび管理するための包括的なプラットフォームを提供します。また、Kubernetes ワークロードをさまざまなオブザーバビリティツールを使用して、より簡単にトラブルシューティングできます。

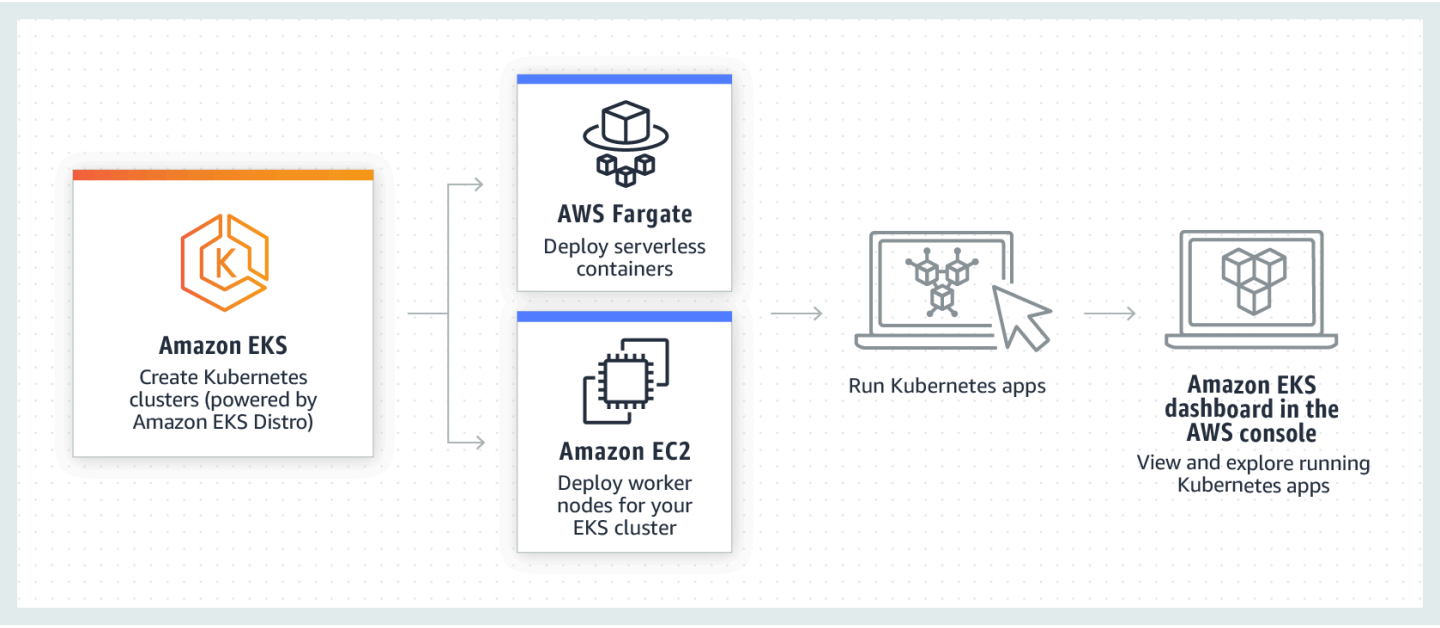
Amazon EKS の他の機能の詳細については、「Amazon EKS の機能」を参照してください。

Amazon EKS の使用を開始する

最初のクラスターとそれに関連するリソースを作成するには、「Amazon EKS の使用開始」を参照してください。一般的に、Amazon EKS の使用を開始するには、以下のステップを実行します。

1. クラスターを作成 — `eksctl`、AWS Management Console、AWS CLI、またはいずれかの AWS SDK を使用してクラスターを作成することから始めます。
2. コンピューティングリソースへのアプローチを選択 — AWS Fargate、Karpenter、マネージド型ノードグループ、セルフマネージド型ノードのいずれかを選択します。
3. セットアップ — 必要なコントローラー、ドライバー、サービスをセットアップします。
4. ワークロードをデプロイ — Kubernetes ワークロードを調節し、選択したノードタイプのリソースと機能を最大限に活用します。
5. 管理 — ワークロードを監視し、AWS サービスを統合して、オペレーションを合理化し、ワークロードのパフォーマンスを向上させます。AWS Management Console を使用して、ワークロードに関する情報を表示できます。

以下の図は、Amazon EKS をクラウドで実行する基本的なフローを示しています。その他の Kubernetes のデプロイオプションについては、「デプロイオプション」を参照してください。



Amazon EKS の料金

Amazon EKS クラスターは、コントロールプレーンと、Pods を実行する Amazon Elastic Compute Cloud (Amazon EC2) または Fargate コンピューティングで構成されます。コントロールプレーンの料金の詳細については、「Amazon EKS の料金表」を参照してください。Amazon EC2 とFargate では以下をご利用いただけます。

オンデマンドインスタンス

秒単位で使用するインスタンスに対して支払いを行い、長期的な確約や前払い金は不要です。詳細については、「Amazon EC2 オンデマンド料金」および「AWS Fargate 料金」を参照してください。

, Savings Plans

1〜3 年の期間、1 時間 につき USD で、定期的な使用量を守ることによりコストを削減できます。詳細については、「Pricing with Savings Plans」を参照してください。

Amazon EKS の一般的なユースケース

Amazon EKS は、AWS 上で堅牢なマネージド型の Kubernetes サービスを提供しています。このサービスは、コンテナ化されたアプリケーションを最適化するように設計されています。以下は、Amazon EKS の強みを活用して特定のニーズに対応するのに役立つ、最も一般的ないくつかのユースケースを示しています。

高可用性アプリケーションのデプロイ

Elastic Load Balancing を使用することで、複数のアベイラビリティゾーンをまたいでアプリケーションの高可用性を確保できます。

マイクロサービスアーキテクチャの構築

Kubernetes サービス検出機能と AWS Cloud Map または Amazon VPC Lattice を使用して、回復力のあるシステムを構築します。

ソフトウェアリリースプロセスの自動化

自動化されたアプリケーションの構築、テスト、デプロイのプロセスを簡素化する、継続的インテグレーションおよび継続的デプロイ (CICD) パイプラインを管理します。

サーバーレスアプリケーションの実行

AWS Fargate と Amazon EKS を使用して、サーバーレスアプリケーションを実行します。つまり、Amazon EKS と Fargate が基盤となるインフラストラクチャを処理している間に、お客様はアプリケーション開発のみに集中できます。

機械学習ワークロードの実行

Amazon EKS は、TensorFlow、MXNet、PyTorch などの一般的な機械学習フレームワークと互換性があります。。GPU サポートにより、複雑な機械学習タスクでも効果的に処理できます。

オンプレミスとクラウドで一貫性のあるデプロイ

Amazon EKS Anywhere を使用すると、クラウド内の Amazon EKS と一貫性のあるツールを使用して、独自のインフラストラクチャ上で Kubernetes クラスターを操作できます。

費用対効果の高いバッチ処理とビッグデータワークロードの実行

スポットインスタンスを使用すると、わずかなコストで Apache Hadoop や Spark などのバッチ処理およびビッグデータワークロードを実行できます。これにより、未使用の Amazon EC2 キャパシティを割引価格で活用できます。

アプリケーションの保護とコンプライアンスの確保

AWS Identity and Access Management (IAM)、Amazon Virtual Private Cloud (Amazon VPC)、AWS Key Management Service (AWS KMS) などの AWS セキュリティサービスと統合されている Amazon EKS を使用して、強力なセキュリティ対策を実施し、コンプライアンスを維持します。これにより、業界標準に準拠したデータのプライバシーと保護が保証されます。

Amazon EKS アーキテクチャ

Amazon EKS は、Kubernetes の一般的なクラスターアーキテクチャに準拠しています。詳細については、Kubernetes ドキュメントの「Kubernetes のコンポーネント」をご参照ください。以下のセクションでは、Amazon EKS のその他のアーキテクチャの詳細を要約しています。

コントロールプレーン

Amazon EKS では、すべてのクラスターで独自の Kubernetes コントロールプレーンが使用されることが保証されます。この設計により、各クラスターのインフラストラクチャーは常に分離され、クラスターや AWS アカウントの間で重複が生じることはありません。セットアップには以下が含まれます。

分散コンポーネント

コントロールプレーンは、AWS リージョン内の 3 つの AWS アベイラビリティーゾーンをまたいで、少なくとも 2 つの API サーバーインスタンスと 3 つの etcd インスタンスを配置します。

最適なパフォーマンス

Amazon EKS はコントロールプレーンインスタンスを積極的にモニタリングして調整し、ピークパフォーマンスを維持します。

耐障害性

コントロールプレーンインスタンスに問題が生じた場合、Amazon EKS は必要に応じて別のアベイラビリティーゾーンを使用して、迅速にそのインスタンスを置き換えます。

一貫した稼働時間

複数のアベイラビリティーゾーンにまたがってクラスターを稼働させることにより、信頼性の高い API サーバーエンドポイントの可用性のサービスレベルアグリーメント (SLA) を実現しています。

Amazon EKS は Amazon Virtual Private Cloud (Amazon VPC) を使用して、単一クラスター内のコントロールプレーンコンポーネント間のトラフィックを制限します。Kubernetes のロールベースのアクセスコントロール (RBAC) ポリシーで許可されている場合を除き、クラスターコンポーネントは、他のクラスターまたは AWS アカウントからの通信を表示したり受信したりすることはできません。

コンピューティング

Amazon EKS クラスターには、コントロールプレーンに加えて、ノードと呼ばれる一連のワーカーマシンがあります。特定の要件を満たし、リソース使用率を最適化するには、適切なタイプの Amazon EKS クラスターノードを選択することが重要です。Amazon EKS には、以下のタイプのプライマリノードが用意されています。

AWS Fargate

Fargate はコンテナ用のサーバーレスコンピューティングエンジンで、基盤となるインスタンスの管理が不要になります。Fargate でアプリケーションのリソースニーズを指定すると、AWS はインフラストラクチャーのプロビジョニング、スケーリング、メンテナンスを自動的に行います。このオプションは、使いやすさを優先し、インフラストラクチャの管理よりもアプリケーションの開発とデプロイに集中したいユーザーに最適です。

Karpenter

Karpenter は柔軟で高性能な Kubernetes クラスターオートスケーラーで、アプリケーションの可用性とクラスター効率の向上に役立ちます。Karpenter では、アプリケーション負荷の変化に応じて、適切なサイズのコンピューティングリソースを起動します。このオプションでは、ワークロードの要件を満たすジャストインタイムのコンピューティングリソースをプロビジョニングできます。

マネージド型ノードグループ

マネージド型ノードグループは、Amazon EKS クラスター内の Amazon EC2 インスタンスのコレクションを管理するための自動化とカスタマイズを組み合わせたものです。AWS がノードのパッチ適用、更新、スケーリングなどのタスクを処理することで、運用面を容易にします。それと同時に、カスタム `kubelet` 引数がサポートされていることで、高度な CPU およびメモリ管理ポリシーの可能性が広がります。さらに、サービスアカウントの AWS Identity and Access Management (IAM) ロールによってセキュリティを強化しながら、クラスターごとに個別のアクセス許可を設定する必要性を抑えます。

セルフマネージド型ノード

セルフマネージド型ノードでは、Amazon EKS クラスター内の Amazon EC2 インスタンスを完全に制御できます。ユーザーは、ノードの管理、スケーリング、メンテナンスを担当することで、基盤となるインフラストラクチャを完全に制御できます。このオプションは、ノードをきめ細かく制御およびカスタマイズする必要があり、インフラストラクチャの管理と保守に時間を費やす準備ができているユーザーに適しています。

Kubernetes の概念

Amazon Elastic Kubernetes Service (Amazon EKS) は、オープンソースの Kubernetes プロジェクトをベースとした AWS マネージドサービスです。Amazon EKS サービスが AWS クラウドとどのように統合するのかについて、(特に Amazon EKS クラスターを初めて作成する場合は)、理解しておくべき点がいくつかありますが、Amazon EKS クラスター一度起動して実行した後の使用方法は、他の Kubernetes クラスターを使う場合とほとんど同じです。そのため、Kubernetes クラスターの管理とワークロードのデプロイを開始するには、少なくとも Kubernetes の概念の基本について理解しておく必要があります。

このページでは、Kubernetes の概念を、Kubernetes を利用する理由、クラスター、ワークロードの 3 つのセクションに分けています。1 つめのセクションでは、Kubernetes のサービス、特に Amazon EKS のようなマネージドサービスとしてこれを実行する場合の利点について説明します。ワークロードのセクションでは、Kubernetes アプリケーションの構築、保存、実行、管理の方法について

説明します。クラスターのセクションでは、Kubernetes クラスターを構成しているさまざまなコンポーネントと、Kubernetes クラスターを作成して管理するときのユーザーの責任について説明します。

トピック

このコンテンツに含まれているリンクをクリックすると、Amazon EKS と Kubernetes の両方のドキュメントに記載されている Kubernetes の概念について解説したページが開き、ここで取り上げたトピックをさらに深く掘り下げることができます。Amazon EKS で Kubernetes コントロールプレーンとコンピューティング機能を実装する方法の詳細については、「Amazon EKS アーキテクチャ」を参照してください。

Kubernetes を利用する理由

Kubernetes は、ミッションクリティカルで本番稼働品質のコンテナ化アプリケーションを実行するときの可用性とスケーラビリティを向上させることを目的に設計されました。1 台のマシン上で Kubernetes を実行するのではなく (実行可能ではありますが)、需要に応じて拡張または縮小する複数のコンピュータを横断してアプリケーションを実行できるようにすれば、Kubernetes は上記の目標を達成できます。Kubernetes には、以下を簡単に行えるようにする機能があります。

- アプリケーションを複数のマシンにデプロイする (ポッドにデプロイされたコンテナを使用)
- コンテナの状態をモニタリングし、障害が発生したコンテナを再起動する
- 負荷に基づいてコンテナをスケールアップ/スケールダウンする
- コンテナを新しいバージョンで更新する
- コンテナ間でリソースを割り当てる
- マシン間でトラフィックを分散する

Kubernetes を使ってこの種の複雑なタスクを自動化すれば、アプリケーションデベロッパーは、インフラストラクチャのことを心配せずに、アプリケーションワークロードの構築と改善に専念できます。デベロッパーは通常、YAML ファイルとしてフォーマットされた設定ファイルを作成し、そこにアプリケーションの望ましい状態を記述します。これには、実行するコンテナ、リソースの上限、ポッドレプリカの数、CPU/メモリの割り当て、アフィニティールールなどが含まれます。

Kubernetes の属性

上記の目標を達成するため、Kubernetes は次の属性を使用します。

- コンテナ化 - Kubernetes はコンテナオーケストレーションツールです。Kubernetes を使用するには、最初にアプリケーションをコンテナ化する必要があります。アプリケーションの種類に応じて、これはマイクロサービスのセット、バッチジョブ、その他形式のいずれかになります。それにより、アプリケーションで、ツールの巨大なエコシステムを含む Kubernetes ワークフローを利用することができます。そこでは、コンテナをイメージとしてコンテナレジストリに保存したり、Kubernetes クラスターにデプロイしたり、利用可能なノード上で実行したりすることができます。個々のコンテナを Kubernetes クラスターにデプロイする前に、Docker または別のコンテナランタイムを使ってローカルのコンピュータ上で構築しテストすることができます。
- スケーラブル - アプリケーションの需要が、それらのアプリケーションを実行しているインスタンスの容量を超える場合、Kubernetes はスケールアップすることができます。必要に応じて、Kubernetes は、アプリケーションに追加の CPU やメモリが必要かどうかを判断し、自動的に利用可能な容量を拡張するか、既存の容量を追加で利用します。アプリケーションでより多くのインスタンスを実行するための十分なコンピューティングがある場合は、ポッドレベルでスケーリングを実行することができます (水平ポッド自動スケーリング)。あるいは、容量の増加に対応するためにより多くのノードを起動する必要がある場合は、ノードレベルでスケーリングを実行することができます (Cluster Autoscaler または Karpenter)。容量が不要になると、これらのサービスは不要なポッドを削除して、不要なノードをシャットダウンすることができます。
- 使用可能 - アプリケーションまたはノードが異常な状態になったり使用できなくなったりすると、Kubernetes で実行中のワークロードを、使用可能な別のノードに移動できます。この問題は、ワークロードを実行しているワークロードまたはノードの実行中のインスタンスを削除すれば、強制的に解決できます。ここでのポイントは、ワークロードを現在の場所で実行できなくなったときは、他の場所で起動できるという点です。
- 宣言型 - Kubernetes はアクティブな照合を使用して、クラスターに宣言した状態が、実際の状態と一致していることを常にチェックします。Kubernetes オブジェクトを、オブジェクトをクラスターに適用すると、通常は YAML 形式の設定ファイルを通じて、例えば、クラスター上で実行したいワークロードの起動をリクエストすることができます。設定は後で変更可能で、新しいバージョンのコンテナを使用したり、メモリをさらに割り当てたりすることができます。Kubernetes は、望ましい状態にするために必要なことを実行します。例えば、ノードの起動や停止、ワークロードの停止および再起動、更新したコンテナの取得などを実行します。
- コンポーザブル - アプリケーションは通常、複数のコンポーネントで構成されているため、これら一連のコンポーネント (通常は複数のコンテナで表されます) をまとめて管理できるようにしておくことをお勧めします。Docker Compose には、Docker を使ってこれを直接実行する方法がありますが、Kubernetes Kompose コマンドを使うと、Kubernetes を使ってこれを実行できます。そ

の方法の例については、「Translate a Docker Compose File to Kubernetes Resources」を参照してください。

- 拡張可能 - 独自のソフトウェアとは異なり、オープンソースの Kubernetes プロジェクトは、ニーズに合わせて Kubernetes を自由に拡張できるように設計されています。API と設定ファイルは、直接変更することができます。サードパーティーは、インフラストラクチャとエンドユーザー Kubernetes 機能の両方を拡張するときは、独自の Controller を作成することが推奨されています。Webhook を使用すると、クラスタールを設定して、ポリシーを適用し、状況の変化に対応することができます。Kubernetes クラスターを拡張する方法の詳細については、「Kubernetes を拡張する」を参照してください。
- ポータブル - 多くの組織では、アプリケーションのニーズをすべて同じ方法で管理できることから、自社での Kubernetes の運用を標準化しています。デベロッパーは、コンテナ化されたアプリケーションを同じパイプラインを使って構築し、保存することができます。これらのアプリケーションは、オンプレミス、クラウド上、レストランの POS 端末、企業のリモートサイトに分散されている IOT デバイスなどで実行している Kubernetes クラスターにデプロイできます。オープンソースであるため、このような特殊な Kubernetes ディストリビューションを、その管理に必要なツールと併せて開発することが可能です。

Kubernetes の管理

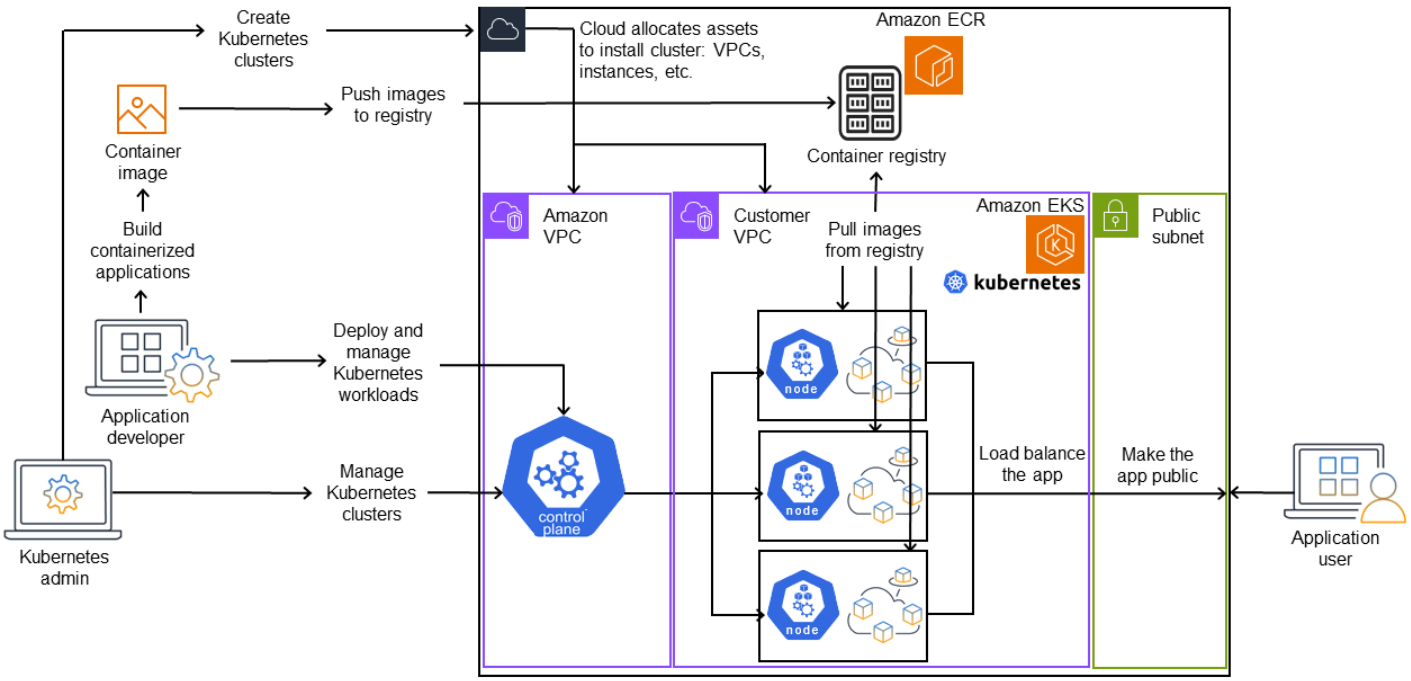
Kubernetes ソースコードは無料で入手できるため、Kubernetes を自分の機器を使ってインストールし、管理することができます。ただし、セルフマネージド型の Kubernetes には運用に関する専門知識が必要であり、維持していくには時間と労力がかかります。そのため、本番環境のワークロードをデプロイする人の大半が、独自のテスト済み Kubernetes ディストリビューションと、Kubernetes の専門家のサポートを備えた、クラウドプロバイダー (Amazon EKS など) またはオンプレミスプロバイダー (Amazon EKS Anywhere など) を利用しています。それらを利用することで、以下のようなクラスターのメンテナンスに必要な未分化の作業の多くを軽減しています。

- ハードウェア - 要件に応じて Kubernetes を実行できるハードウェアがない場合、AWS Amazon EKS などのクラウドプロバイダーを利用すると初期費用を節約できます。Amazon EKS では、コンピュータインスタンス (Amazon Elastic Compute Cloud)、独自のプライベート環境 (Amazon VPC)、一元的なアイデンティティと権限管理 (IAM)、ストレージ (Amazon EBS) など、AWS で提供されている最善のクラウドリソースを利用できます。AWS は、コンピュータ、ネットワーク、データセンター、および、Kubernetes の実行に必要なその他すべての物理コンポーネントを管理します。同様に、需要が最も高い日に最大容量を処理できるようにデータセンターを計画しておく必要はありません。Amazon EKS Anywhere やその他のオンプレミスの Kubernetes クラスターの場合、Kubernetes デプロイに使用されるインフラストラクチャの管理はお客様の責任となりますが、引き続き AWS を使って Kubernetes を最新の状態に保つことができます。

- コントロールプレーン管理 - Amazon EKS は、AWS がホストしている Kubernetes コントロールプレーンのセキュリティと可用性を管理します。コンテナのスケジュール設定、アプリケーションの可用性の管理、その他の重要なタスクはこのコントロールプレーンが行うため、ユーザーはアプリケーションのワークロードに専念することができます。クラスターが故障した場合、AWS には、実行状態に復元するための方法がいくつかあります。Amazon EKS Anywhere の場合、コントロールプレーンを自分で管理することになります。
- テスト済みのアップグレード - クラスターをアップグレードするときは、Amazon EKS または Amazon EKS Anywhere を使って、Kubernetes ディストリビューションのテスト済みバージョンを用意することができます。
- アドオン - 拡張して Kubernetes と連携するように構築された数百ものプロジェクトがあり、クラスターのインフラストラクチャに追加したり、ワークロードの実行を支援するために使用したりすることができます。これらのアドオンを自分で構築して管理する代わりに、AWS には、クラスターで使用できる Amazon EKS アドオンが用意されています。Amazon EKS Anywhere には、多くの一般的なオープンソースプロジェクトのビルドを含む、キュレーションパッケージがあります。そのため、ユーザーは、ソフトウェアを自分で構築したり、重要なセキュリティパッチ、バグ修正、アップグレードを管理したりする必要はありません。同様に、デフォルトの設定がニーズを満たしていれば、通常、それらのアドオンの設定はほとんど必要ありません。アドオンを使用してクラスターを拡張する方法については、「クラスターの拡張」を参照してください。

実行中の Kubernetes

以下の図は、Kubernetes 管理者またはアプリケーションデベロッパーが Kubernetes クラスターを作成して使用する際に実行する主要なアクティビティを示したものです。そのプロセスにおいて、基盤となるクラウドプロバイダーの例として AWS クラウドを使用し、Kubernetes のコンポーネントがどのように相互作用するかを説明しています。



Kubernetes 管理者は、クラスターが構築されるプロバイダーのタイプに固有のツールを使用して、Kubernetes クラスターを作成します。この例では、Amazon EKS という マネージドの Kubernetes サービスを提供する AWS クラウドをプロバイダーとして使用します。このマネージドサービスは、クラスターの作成に必要なリソースを自動的に割り当てます。これには、クラスター用の 2 つの新しい Virtual Private Cloud (Amazon VPC) の作成、ネットワークのセットアップ、クラウド内のアセットを管理するための Kubernetes アクセス許可のマッピング、コントロールプレーンサービスを実行する場所があることの確認、ワークロードを実行する Kubernetes ノードとしての 0 個以上の Amazon EC2 インスタンスの割り当てなどが含まれます。AWS は、コントロールプレーン用に Amazon VPC 自体を 1 つ管理し、もう 1 つの Amazon VPC には、ワークロードを実行するカスタマーノードが含まれます。

Kubernetes 管理者が今後行うタスクの多くは、kubectl などの Kubernetes ツールを使って行われます。このツールは、クラスターのコントロールプレーンに直接サービスをリクエストします。その場合、クラスターに対してクエリや変更を行う方法は、任意の Kubernetes クラスターに対して行う場合の方法と非常によく似ています。

このクラスターにワークロードをデプロイする場合、アプリケーションデベロッパーは、いくつかのタスクを実行することができます。デベロッパーは、アプリケーションを 1 つ以上のコンテナイメージに構築し、そのイメージを、Kubernetes クラスターにアクセスできるコンテナレジストリ

にプッシュする必要があります。AWS には、それを行うための Amazon Elastic Container Registry (Amazon ECR) というレジストリがあります。

このアプリケーションを実行するには、デベロッパーは、レジストリからどのコンテナを取得するか、それらのコンテナをポッドにどのようにラップするかなど、アプリケーションの実行方法をクラスターに指示する YAML 形式の設定ファイルを作成します。コントロールプレーン (スケジューラー) は、コンテナを 1 つ以上のノードにスケジュールし、各ノードのコンテナランタイムが必要なコンテナーを実際にプルして実行します。また、デベロッパーは Application Load Balancer を設定することで、各ノードで実行されている利用可能なコンテナにトラフィックを分散し、パブリックネットワーク上で外部から利用できるようにアプリケーションを公開することもできます。これで、アプリケーションを使用したいユーザーが、アプリケーションエンドポイントに接続してアクセスできるようになります。

以下のセクションでは、これらの各機能について、Kubernetes のクラスターとワークロードの観点から詳しく説明します。

クラスター

ジョブに Kubernetes クラスターの起動および管理が含まれる場合は、Kubernetes クラスターの作成、強化、管理、削除の方法を理解しておく必要があります。また、クラスターを構成しているコンポーネントと、それらのコンポーネントを維持するために行うべきことについても、知っておく必要があります。

クラスターを管理するツールは、Kubernetes のサービスと基盤となるハードウェアプロバイダーとの重複を処理します。そのため、これらのタスクの自動化は、Kubernetes プロバイダー (Amazon EKS や Amazon EKS Anywhere など) が、そのプロバイダーに固有のツールを使って行うのが一般的です。例えば、Amazon EKS クラスターを起動する場合は `eksctl create cluster` を使用できますが、Amazon EKS Anywhere には `eksctl anywhere create cluster` を使用できます。これらのコマンドは Kubernetes クラスターを作成しますが、プロバイダー固有のものであるため、Kubernetes プロジェクト自体には含まれていないことに注意してください。

クラスターの作成および管理ツール

Kubernetes プロジェクトには、Kubernetes クラスターを手動で作成するためのツールが用意されています。したがって、Kubernetes を単一のマシンにインストールしたり、マシン上でコントロールプレーンを実行したり、ノードを手動で追加したりする場合は、「Kubernetes Install Tools」に一覧表示されている kind、minikube、kubeadm などの CLI ツールを使用できます。クラスターの作成と管理のライフサイクル全体を簡素化して自動化するには、Amazon EKS や Amazon EKS Anywhere

など、定評のある Kubernetes プロバイダーがサポートしているツールを使用するとはるかに簡単になります。

AWS クラウドでは、Amazon EKS クラスターを、eksctl などの CLI ツールや、Terraform などのより宣言的なツールを使用して作成することができます (「Amazon EKS Blueprints for Terraform」を参照)。また、AWS 管理コンソールからクラスターを作成することもできます。Amazon EKS で利用できる機能の一覧は「Amazon EKS の機能」を参照してください。Amazon EKS がお客様に代わって引き受ける Kubernetes の責任には、以下が含まれます。

- マネージドコントロールプレーン - AWS は、コントロールプレーンをユーザーに代わって管理し、AWS アベイラビリティーゾーン全体で利用可能するため、Amazon EKS クラスターが利用可能かつスケーラブルになります。
- ノード管理 - ノードを手動で追加する代わりに、マネージド型ノードグループまたは Karpenter を使用して、必要に応じて Amazon EKS でノードを自動的に作成することができます。マネージドノードグループは、Kubernetes クラスター自動スケーリングと統合されています。ノード管理ツールを使用すると、スポットインスタンスやノード統合などによりコストを節約したり、ワークロードのデプロイ方法やノードの選択方法を設定するスケジューリング機能を使って、可用性を活用することができます。
- クラスターネットワーク - `eksctl` は、CloudFormation テンプレートを使用して、Kubernetes クラスター内のコントロールプレーンとデータプレーン (ノード) コンポーネント間に、ネットワークを設定します。また、内部と外部の通信を行うためのエンドポイントも設定します。詳細については、「De-mystifying cluster networking for Amazon EKS worker nodes」を参照してください。Amazon EKS 内のポッド間の通信は、Amazon EKS Pod Identity を使用して行われます。これによりポッドは、認証情報やアクセス許可を管理する AWS クラウドの方法を利用できるようになります。
- アドオン - Amazon EKS を使用すると、Kubernetes クラスターをサポートするために一般的に使用される、ソフトウェアコンポーネントを構築して追加する必要がなくなります。例えば、AWS マネジメントコンソールから Amazon EKS クラスターを作成すると、Amazon EKS kube-proxy、Kubernetes 用の Amazon VPC CNI プラグイン、CoreDNS アドオンが、自動的に追加されます。利用可能なアドオンも含む、これらのアドオンの詳細については、「Amazon EKS アドオン」参照してください。

クラスターをオンプレミスのコンピュータとネットワーク上で実行できるように、Amazon は Amazon EKS Anywhere を提供しています。AWS クラウドをプロバイダーにする代わりに、Amazon EKS Anywhere を VMware vSphere、ベアメタル (Tinkerbell プロバイダー)、Snow、CloudStack、Nutanix の各プラットフォームで、独自の機器を使用して実行することを選択できます。

Amazon EKS Anywhere は、Amazon EKS で使用されているものと同じ Amazon EKS Distro ソフトウェアをベースにしています。ただし、Amazon EKS Anywhere は、Amazon EKS Anywhere クラスター内のマシンのライフサイクル全体を管理するために、Kubernetes クラスター API (CAPI) インターフェイスのさまざまな実装に依存しています (vSphere の場合は CAPV、CloudStack の場合は CAPC など)。クラスター全体がユーザーの機器上で実行されるため、ユーザーは、コントロールプレーンの管理とデータのバックアップの責任を追加で負うことになります (本ドキュメントで後述する etcd を参照)。

クラスターコンポーネント

Kubernetes クラスターコンポーネントは、コントロールプレーンとワーカーノードという 2 つの主要な領域に分かれています。コントロールプレーンコンポーネントはクラスターを管理し、その API へのアクセスを提供します。ワーカーノード (単にノードとも呼ばれます) は、実際のワークロードを実行する場所です。ノードコンポーネントは、各ノード上で実行されるサービスで構成され、コントロールプレーンと通信してコンテナを実行します。クラスターのワーカーノードのセットは、データプレーンと呼ばれます。

コントロールプレーン

コントロールプレーンは、クラスターを管理する一連のサービスで構成されています。これらのサービスはすべて 1 台のコンピュータで実行される場合もあれば、複数のコンピュータに分散される場合もあります。内部的には、これらはコントロールプレーンインスタンス (CPI) と呼ばれます。CPI の実行方法は、クラスターの規模と、高可用性の要件に応じて異なります。クラスターの需要が増えると、コントロールプレーンサービスはスケールしてそのサービスのインスタンスを増やし、リクエストはインスタンス間で負荷分散されます。

Kubernetes コントロールプレーンのコンポーネントが実行するタスクには、以下が含まれます。

- クラスターコンポーネント (API サーバー) との通信 (API サーバー) - API サーバー (kube-apiserver) は、クラスターへのリクエストをクラスターの内部と外部の両方から行えるようにするために Kubernetes API を公開します。つまり、クラスターのオブジェクト (ポッド、サービス、ノードなど) を追加または変更するリクエストは、ポッドを実行する `kubectl` のリクエストなどの外部コマンドから送信することができます。同様に、ポッドのステータスを `kubelet` サービスにクエリするときのように、API サーバーからクラスター内のコンポーネントに対してリクエストを実行することができます。
- クラスターに関するデータの保存 (etcd キーバリューストア) - etcd サービスは、クラスターの現在の状態を追跡するという重要な役割を担っています。etcd サービスにアクセスできなくなると、ワークロードはしばらく実行し続けますが、クラスターの状態を更新したりクエリしたりする

ことはできなくなります。そのため、重要なクラスターは通常、複数の負荷分散された etcd サービスのインスタンスを同時に実行し、データの損失や破損に備えて、etcd キーバリューストアのバックアップを定期的に行っています。Amazon EKS では、これはすべてデフォルトで自動的に処理されます。Amazon EKS Anywhere は、etcd のバックアップと復元の手順を公開しています。etcd によるデータ管理の方法については、etcd の「Data Model」を参照してください。

- ノードへのポッドのスケジュール (スケジューラー) - Kubernetes での、ポッドの起動または停止のリクエストは、Kubernetes スケジューラー (kube-scheduler) に送信されます。クラスターにはポッドを実行できるノードが複数含まれている場合があるため、どのノード (レプリカの場合は複数のノード) でポッドを実行するかは、スケジューラーが決定します。リクエストされたポッドを既存のノードで実行するための十分な容量がない場合、他のポッドをプロビジョニングしていなければそのリクエストは失敗します。プロビジョニングには、新しいノードを自動的に起動してワークロードを処理するマネージド型ノードグループや Karpenter などのサービスを有効化する作業が含まれます。
- コンポーネントを目的の状態に維持する (コントローラーマネージャー) -Kubernetes コントローラーマネージャーは、クラスターの状態をモニタリングしたり、クラスターに変更を加えて期待される状態に戻したりするデーモンプロセス (kube-controller-manager) として実行されます。特に、コントローラーは、node-lifecycle-controller、statefulset-controller、endpoint-controller、cronjob-controller など、さまざまな Kubernetes オブジェクトをモニタリングするコントローラーが複数あります。
- クラウドリソースの管理 (Cloud Controller Manager) - Kubernetes と、基盤となるデータセンターリソースのリクエストを実行するクラウドプロバイダーとのやり取りは、Cloud Controller Manager(cloud-controller-manager) が処理します。Cloud Controller Manager が管理するコントローラーには、ルートコントローラー (クラウドネットワークルートを設定するため)、サービスコントローラー (クラウドのロードバランシングサービスを使用するため)、ノードコントローラー (クラウド API を使用して Kubernetes ノードをクラウドノードと同期させるため) が含まれます。

ワーカーノード (データプレーン)

単一ノード Kubernetes クラスターの場合、ワークロードはコントロールプレーンと同じマシンで実行します。ただし、より一般的な構成では、Kubernetes ワークロードを実行するための専用の個別のコンピュータシステム (「Nodes」) を 1 つ以上用意します。

Kubernetes クラスターを初めて作成する場合、作成ツールによっては、(既存のコンピュータシステムを識別するか、プロバイダーに新しいコンピュータシステムを作成させることによって) 特定の数のノードをクラスターに追加するように設定できるものがあります。これらのシステムにワークロードを追加する前に、以下の機能を実装するためのサービスを各ノードに追加します。

- 各ノードの管理 (kubelet) - API サーバーは、各ノードで実行されている kubelet サービスと通信し、ノードが正しく登録され、スケジューラーからリクエストされたポッドが実行されていることを確認します。kubelet は、ポッドマニフェストを読み取り、ローカルシステム上で Pod が必要とするストレージボリュームやその他の機能を設定できます。また、ローカルで実行されているコンテナの状態もチェックすることができます。
- ノードでコンテナを実行する (コンテナランタイム) - 各ノードのコンテナランタイムは、ノードに割り当てられた各ポッドに対してリクエストされたコンテナを管理します。つまり、適切なレジストリからコンテナイメージをプルし、そのコンテナを実行して停止し、コンテナに関するクエリに応答することができます。デフォルトのコンテナランタイムは containerd です。Kubernetes 1.24 以降、コンテナランタイムとして使用できる Docker (Dockershim) の特別統合は、Kubernetes から削除されました。ローカルシステム上でコンテナをテストおよび実行する際は、引き続き Docker を使用することができますが、Kubernetes で Docker を使用するには、各ノードに Docker Engine をインストールして Kubernetes で使用する必要があります。
- コンテナ間のネットワークの管理 (kube-proxy) - サービスを使用してポッド間の通信をサポートする場合、Kubernetes に、ポッドネットワークを設定して、それらのポッドに関連づけられた IP アドレスとポートを追跡する方法が必要でした。kube-proxy サービスは、各ノードで実行し、ポッド間でその通信を可能にします。

拡張クラスター

クラスターをサポートするために Kubernetes に追加できるサービスがいくつかありますが、それらはコントロールプレーンでは実行されません。それらのサービスは、多くの場合、kube-system 名前空間のノードで直接実行されるか、独自の名前空間で実行されます (サードパーティーのサービスプロバイダーではよく行われています)。一般的な例が、クラスターに DNS サービスを提供する CoreDNS サービスです。クラスターの kube-system でどのクラスターサービスが実行されているのかを確認する方法については、「Discovering builtin services」を参照してください。

クラスターに追加できるアドオンには、さまざまな種類があります。クラスターを健全な状態に保つには、ログ記録、監査、メトリクスなどの作業を実行できる、オブザーバビリティ機能を追加します。この情報があれば、多くの場合、同じオブザーバビリティインターフェイスを使用して、発生した問題をトラブルシューティングすることができます。こうしたタイプのサービスには、Amazon GuardDuty、CloudWatch、AWS Distro for OpenTelemetry、Kubernetes 用の Amazon VPC CNI プラグイン、Grafana Kubernetes Monitoring などがあります。ストレージでは、Amazon EKS のアドオンには Amazon Elastic Block Store CSI ドライバー (ブロックストレージデバイスの追加用)、Amazon Elastic File System CSI ドライバー (ファイルシステムストレージの追加用)、サードパーティー製ストレージのアドオン (Amazon FSx for NetApp ONTAP CSI ドライバーなど) などがあります。

使用可能な Amazon EKS アドオンの詳細については、「Amazon EKS アドオン」を参照してください。

ワークロード

Kubernetes では、ワークロードは「Kubernetes で実行中のアプリケーション」と定義されています。そのアプリケーションは、ポッド内のコンテナとして実行される、一連のマイクロサービスで構成されていますが、バッチジョブやその他の種類のアプリケーションとして実行される場合もあります。Kubernetes は、それらのオブジェクトを設定またはデプロイするために行われたリクエストを確実に実行する役割を担います。アプリケーションをデプロイするユーザーは、コンテナの構築方法、ポッドの定義方法、デプロイに使用する方法を学ぶ必要があります。

コンテナ

Kubernetes にデプロイして管理するアプリケーションワークロードの最も基本的な要素がポッドです。ポッドは、アプリケーションのコンポーネントを格納する方法であるだけでなく、ポッドの属性を記述する仕様を定義する方法でもあります。RPM や Deb パッケージは、Linux システム用のソフトウェアをまとめてパッケージ化しますが、それ自体はエンティティとしては動作しません。

ポッドはデプロイ可能な最小のユニットであるため、通常は 1 つのコンテナしか格納できません。しかし、コンテナが密結合している場合は、複数のコンテナを 1 つのポッドに含めることができます。例えばウェブサーバーコンテナは、ログ記録、モニタリング、またはウェブサーバーコンテナと密結合したその他のサービスを提供できるサイドカータイプのコンテナとともに、ポッドにパッケージ化できます。この場合、同じポッド内にあることで、そのポッドが実行している各インスタンスでは、両方のコンテナが常に同じノードで実行されることになります。同様に、ポッド内のすべてのコンテナは同じ環境を共有し、ポッド内のコンテナは、隔離された同じホストにあるかのように実行されます。これにより、各コンテナはポッドへのアクセスを可能にする 1 つの IP アドレスを共有し、コンテナはあたかも独自のローカルホスト上で実行しているかのように相互に通信することができます。

ポッドの仕様 (PodSpec) は、ポッドの望ましい状態を定義します。ワークロードリソースを使用してポッドテンプレートを管理することで、個別のポッドまたは複数のポッドをデプロイできます。ワークロードリソースには、Deployments (複数のポッドレプリカを管理する)、StatefulSets (データベースポッドなどの一意にする必要があるポッドをデプロイする)、DaemonSets (ポッドをすべてのノードで継続的に実行する必要がある場合) が含まれます。詳細は後ほど説明します。

ポッドはデプロイできる最小のユニットですが、コンテナは構築して管理できる最小のユニットです。

コンテナの構築

ポッドは、実際には 1 つ以上のコンテナを取り囲む構造に過ぎず、各コンテナ自体にファイルシステム、実行ファイル、設定ファイル、ライブラリ、その他アプリケーションを実際に実行するためのコンポーネントが格納されています。コンテナを普及させた会社の名前が Docker Inc. だったため、Docker コンテナと呼ばれることもあります。しかし、コンテナのランタイム、イメージ、ディストリビューション方法を業界向けに定義してきたのは Open Container Initiative です。また、コンテナは、Linux に既に存在していた多くの機能から作成されていることから、OCI コンテナ、Linux コンテナ、あるいはただ単にコンテナとも呼ばれます。

コンテナを構築するときは、通常は Dockerfile (会社名から命名) から始めます。Dockerfile 内では、以下を識別します。

- ベースイメージ - ベースのコンテナイメージは、通常は、オペレーティングシステムのファイルシステム (Red Hat Enterprise Linux や Ubuntu など) の最小バージョン、または特定の種類のアプリケーション (nodejs や python アプリケーションなど) を実行するソフトウェアを提供するように拡張された最小システムのいずれかから構築されるコンテナです。
- アプリケーションソフトウェア - Linux システムに追加するときとほぼ同じ方法で、アプリケーションソフトウェアをコンテナに追加できます。例えば、Dockerfile では、npm や `yarn` を実行して Java アプリケーションをインストールするか、`yum` や `dnf` を実行して RPM パッケージをインストールすることができます。つまり、Dockerfile で RUN コマンドを使用すると、ベースイメージのファイルシステムで使用可能な任意のコマンドを実行して、生成されたコンテナイメージの内部でソフトウェアをインストールしたり、設定したりできます。
- インストラクション - Dockerfile reference には、Dockerfile の設定時に追加できるインストラクションが記載されています。これには、コンテナ自体の中にあるもの (ローカルシステムの `ADD` または `COPY` ファイル) の構築、コンテナを実行するときに実行するコマンドの特定 (`CMD` または `ENTRYPOINT`)、コンテナを実行するシステムへのコンテナの接続 (実行する `USER`、マウントするローカル `VOLUME`、`EXPOSE` するポート) などに使用する指示が含まれています。

`docker` コマンドとサービスは、従来、コンテナ (`docker build`) の構築に使用されてきましたが、コンテナイメージの構築に使用できるその他ツールには podman や nerdctl などがあります。コンテナの構築方法については、「Building Better Container Images」または「Build with Docker」参照してください。

コンテナの保存

コンテナイメージを作成したら、ワークステーションのコンテナディストリビューションレジストリ、またはパブリックコンテナレジストリに保存できます。ワークステーションでプライベートコン

テナレジストリを実行すると、コンテナイメージをローカルに保存して、すぐに利用できるようにすることができます。

コンテナイメージを、よりパブリックな方法で保存するときは、イメージをパブリックコンテナレジストリーにプッシュします。パブリックコンテナレジストリは、コンテナイメージの保存と配布を一元的に行える場所です。代表的なパブリックコンテナレジストリには、Amazon Elastic Container Registry、Red Hat Quay レジストリ、Docker Hub レジストリなどがあります。

Amazon Elastic Kubernetes Service (Amazon EKS) でコンテナ化されたワークロードを実行するときは、Amazon Elastic Container Registry に保存されている Docker 公式イメージのコピーをプルすることをお勧めします。AWSAmazon ECR は、2021 年からこれらのイメージを保存しています。人気の高いコンテナイメージは Amazon ECR Public Gallery で検索でき、特に、Docker Hub イメージは Amazon ECR DockerGallery で検索できます。

コンテナの実行

コンテナは標準フォーマットで構築されているため、コンテナランタイムを実行でき (Docker など)、コンテンツがローカルマシンのアーキテクチャ (`x86_64` や `arm` など) に一致するマシンであれば、どのマシンでもコンテナを実行できます。コンテナをテストしたり、ローカルのデスクトップで実行したりするには、`docker run` または `podman run` コマンドを使用して、ローカルホストでコンテナを起動します。ただし Kubernetes では、各ワーカーノードにコンテナランタイムがデプロイされており、そのノードにコンテナの実行をリクエストするかどうかは Kubernetes が決定します。

コンテナがノード上で実行されるように割り当てられると、そのノードは、リクエストされたバージョンのコンテナイメージがノード上にすでに存在するかどうかを確認します。存在しなければ、Kubernetes が、適切なコンテナレジストリからそのコンテナをプルしてローカルで実行するように、コンテナランタイムに指示します。コンテナイメージとは、ノートパソコン、コンテナレジストリー、Kubernetes ノード間を移動するソフトウェアパッケージのことを指します。コンテナとは、そのイメージの実行中のインスタンスを指します。

ポッド

コンテナの準備が整ったら、ポッドの操作として、ポッドの設定、デプロイ、ポッドをアクセス可能にするなどの作業を行います。

ポッドの設定

ポッドを定義するときは、一連の属性をポッドに割り当てます。これらの属性には、少なくともポッド名と実行するコンテナイメージが含まれている必要があります。ただし、ポッドの定義を使

用して設定する必要のある項目は他にも多数あります (ポッドに追加できる内容の詳細については「PodSpec」のページを参照してください)。具体的には次のとおりです。

- ストレージ - 実行中のコンテナを停止して削除すると、より永続的なストレージを設定しない限り、そのコンテナ内のデータストレージは削除されます。Kubernetes はさまざまなストレージタイプをサポートしており、ボリューム内でこれらを抽象化します。ストレージタイプには、CephFS、NFS、iSCSI などがあります。ローカルのコンピュータからローカルブロックデバイスを使用することもできます。クラスターからこれらのストレージタイプのいずれかを使用すれば、ストレージボリュームをコンテナのファイルシステム内の選択したマウントポイントにマウントできます。永続ボリュームは、ポッドが削除された後も残り、エフェメラルボリュームはポッドが削除されると、削除されます。クラスター管理者がクラスター用に異なる StorageClasses を作成した場合、使用後にボリュームを削除するか、再利用するか、容量がさらに必要になった場合に拡張するか、また、特定のパフォーマンス要件を満たすようにするかなど、使用するストレージの属性を選択できることがあります。
- シークレット - Pod 仕様のコンテナでシークレットを使用できるようにすると、ファイルシステム、データベース、その他保護されているアセットにアクセスするために必要な権限をそれらのコンテナに付与することができます。キー、パスワード、トークンは、シークレットとして保存できるアイテムです。シークレットを使用すると、この情報をコンテナイメージに保存する必要はなく、実行中のコンテナでシークレットを利用できるようにすることができます。シークレットに似ているのが ConfigMaps です。ConfigMap には一般に、サービスを設定するためのキーと値のペアなど、重要度が低い情報が格納されます。
- コンテナリソース - コンテナを詳細に設定するためのオブジェクトは、リソース設定の形式をとることができます。コンテナごとに、そのコンテナが使用できるメモリと CPU の量をリクエストできるほか、コンテナが使用できるリソースの合計量の上限を設定できます。例については、「Resource Management for Pods and Containers」を参照してください。
- 中断 - ポッドは、意図せずに (ノードがダウンしたとき)、または意図的に (アップグレードが必要なとき) 中断することがあります。ポッド中断の予算を設定することで、中断が起きた場合のアプリケーションの可用性をある程度制御することができます。アプリケーションの例については、「Specifying a Disruption Budget」を参照してください。
- 名前空間 - Kubernetes には、Kubernetes コンポーネントとワークロードを互いに分離するさまざまな方法が用意されています。特定のアプリケーションのすべてのポッドを、同じ名前空間で実行することは、それらのポッドをまとめて保護し管理する一般的な方法です。独自の名前空間を作成して使用することも、名前空間を指定しない (Kubernetes が `default` の名前空間を使用する) ように選択することもできます。Kubernetes コントロールプレーンのコンポーネントは、通常、kube-system 名前空間で実行されます。

これらの設定は、通常、YAML ファイルにまとめられ、Kubernetes クラスターに適用されます。パーソナルな Kubernetes クラスターの場合は、これらの YAML ファイルをローカルシステムに保存します。ただし、よりクリティカルなクラスターやワークロードの場合は、GitOps が、ストレージを自動化し、ワークロードと Kubernetes インフラストラクチャリソースの両方を更新する方法として一般的に使用されています。

ポッド情報の収集とデプロイに使用されるオブジェクトは、以下のデプロイ方法のいずれかによって定義されます。

ポッドのデプロイ

ポッドをデプロイする際に選択すべき方法は、そのポッドで実行する予定のアプリケーションのタイプに応じて異なります。この方法には、以下のようなものがあります。

- ステートレスアプリケーション - ステートレスアプリケーションは、クライアントのセッションデータを保存しないため、その次のセッションは、前のセッションで起きたことを参照する必要がありません。これにより、ポッドが異常な状態になった場合は新しいポッドと交換するか、状態を保存せずにポッドを移動させるだけで済みます。ステートレスアプリケーション (ウェブサーバーなど) を実行している場合は、Deployment を使用して Pods と ReplicaSets をデプロイできます。ReplicaSet は、同時に実行したいポッドのインスタンス数を定義します。ReplicaSet は直接実行することもできますが、1 回に実行するポッドのレプリカの数を定義するため、デプロイ内でレプリカを直接実行するのが一般的です。
- ステートフルアプリケーション - ステートフルアプリケーションとは、ポッドの ID とポッドの起動順序が重要であるアプリケーションのことです。これらのアプリケーションには、安定した永続的ストレージが必要であり、一貫した方法でデプロイしスケールインする必要があります。ステートフルアプリケーションを Kubernetes にデプロイするには、StatefulSets を使用します。通常、StatefulSet として実行されるアプリケーションの一例が、データベースです。StatefulSet 内では、レプリカ、ポッドとそのコンテナ、マウントするストレージボリューム、データを保存するコンテナ内の場所を定義できます。ReplicaSet としてデプロイされるデータベースの例については、「Run a Replicated Stateful Application」を参照してください。
- ノードあたりのアプリケーション - アプリケーションを、Kubernetes クラスター内のすべてのノードで実行する必要がある場合があります。例えば、データセンターで、すべてのコンピュータでモニタリングアプリケーションまたは特定のリモートアクセスサービスを実行する必要がある場合です。Kubernetes では、DaemonSet を使用することで、選択したアプリケーションがクラスター内のすべてのノードで実行されるようにすることができます。
- 完了まで実行するアプリケーション - 特定のタスクを完了するために、複数のアプリケーションを実行する必要がある場合があります。例えば、毎月のステータスレポートを実行したり、古いデータを消去したりする場合です。Job オブジェクトを使用すると、アプリケーションを起動して実

行し、タスクが完了したら、終了するように設定することができます。CronJob オブジェクトを使用すると、Linux の crontab 形式で定義された構造を使用して、特定の時間、分、日、月、曜日に、アプリケーションを実行するように設定できます。

ネットワークからアプリケーションにアクセスできるようにする

アプリケーションはさまざまな場所を移動するマイクロサービスのセットとしてデプロイされることが多いため、Kubernetes には、それらのマイクロサービスが相互に検索できる方法が必要でした。また、Kubernetes クラスターの外にあるアプリケーションにアクセスするユーザーのために、Kubernetes には、そのアプリケーションを外部のアドレスとポートで公開する方法が必要でした。これらのネットワーク関連の機能は、それぞれ Service オブジェクトと Ingress オブジェクトを使って実行されます。

- サービス - ポッドは異なるノードやアドレスに移動できるため、最初のポッドと通信する必要のある別のポッドがそのポッドの場所を特定することが困難になる場合があります。この問題を解決するため、Kubernetes では、アプリケーションを Service として表すことができます。Service を使用すると、特定の名前を持つポッドまたはポッドのセットを識別し、そのアプリケーションのサービスをポッドから公開するポートと、他のアプリケーションがそのサービスにアクセスする際に使用できるポートを指定することができます。クラスター内の別のポッドは名前で Service をリクエストすることができ、Kubernetes は、そのリクエストをそのサービスを実行しているポッドのインスタンスの適切なポートに転送します。
- Ingress – Ingress は、Kubernetes Services によって示されたアプリケーションを、クラスター外のクライアントが利用できるようにします。Ingress の基本機能には、ロードバランサー (Ingress が管理)、Ingress コントローラー、コントローラーから Service にリクエストを送信するルールなどがあります。Kubernetes で選択できる Ingress Controllers は複数あります。

次のステップ

Kubernetes の基本的な概念と、それらが Amazon EKS とどのように関連するのかを理解することで、Amazon EKS ドキュメントと Kubernetes のドキュメントの両方を参照しながら、Amazon EKS クラスターを管理したりワークロードをクラスターにデプロイしたりする際に必要になる情報をすぐに見つけられるようになります。Amazon EKS の使用を開始するには、以下から選択してください。

- シンプルなクラスターを作成する
- より複雑なクラスターを作成する
- サンプルアプリケーションをデプロイする

- クラスターの管理方法を知る

デプロイオプション

以下の任意のオプション使用して、Amazon EKS をデプロイできます。

クラウド内の Amazon EKS

ユーザーは、独自の Kubernetes コントロールプレーンやノードをインストール、運用、メンテナンスしなくても、AWS クラウドで Kubernetes を実行できます。このオプションについてはこのガイドで説明しています。

Amazon EKS on Outposts

AWS Outposts を使用すると、オンプレミス施設でネイティブ AWS のサービス、インフラストラクチャ、運用モデルを使用できるようになります。Amazon EKS on Outposts では、拡張クラスターとローカルクラスターのどちらを実行するかを選択できます。拡張クラスターでは、Kubernetes コントロールプレーンは AWS リージョンで実行され、ノードは Outposts で実行されます。ローカルクラスターでは、Kubernetes コントロールプレーンとノードの両方を含め、Kubernetes クラスター全体が Outposts でローカルに実行されます。詳細については、「AWS Outposts における Amazon EKS」を参照してください。

Amazon EKS Anywhere

Amazon EKS Anywhere は、Amazon EKS のデプロイオプションであり、Kubernetes クラスターをオンプレミスで簡単に作成および運用できるようになります。Amazon EKSとAmazon EKS Anywhere はどちらも Amazon EKS Distro上に構築されています。Amazon EKS Anywhere と Amazon EKS との相違点については、概要および Amazon EKS Anywhere と Amazon EKS を比較する Amazon EKS Anywhere のドキュメントを参照してください。よく寄せられる質問に対する回答については、「Amazon EKS Anywhere のよくある質問」を参照してください。

Amazon EKS Distro

Amazon EKS Distro は、Amazon EKS がクラウドにデプロイしている同じオープンソースの Kubernetes ソフトウェアおよび依存関係のディストリビューションです。Amazon EKS Distro は Amazon EKS と同じ Kubernetes バージョンのリリースサイクルに従っており、オープンソースプロジェクトとして提供されています。詳細については、Amazon EKS Distro を参照してください。また、GitHub で Amazon EKS Distro のソースコードを表示およびダウンロードすることもできます。

Kubernetes クラスターに使用するデプロイオプションを選択する場合は、以下の点を考慮してください。

機能	Amazon EKS	Amazon EKS on Outposts	Amazon EKS Anywhere	Amazon EKS Distro
ハードウェア	AWS - による提供	AWS - による提供	お客様が提供する	お客様が提供する
デプロイの場所	AWS クラウド	お客様のデータセンター	お客様のデータセンター	お客様のデータセンター
Kubernetes コントロールプレーンの位置	AWS クラウド	AWS クラウドまたはデータセンター	お客様のデータセンター	お客様のデータセンター
Kubernetes データプレーンの位置	AWS クラウド	お客様のデータセンター	お客様のデータセンター	お客様のデータセンター
サポート	AWS Support	AWS Support	AWS Support	OSS コミュニティのサポート

Amazon EKS のセットアップ

AWS リソースには通常、リソースを作成した AWS エンティティへのアクセスを制限するアクセス制限があります。そのため、最初から AWS Command Line Interface で適切なユーザー設定を確立することが重要です。さらに、Amazon EKS クラスターをコマンドラインで効率的に管理するための必須ツールをローカルマシンに装備する必要があります。このトピックは、クラスターのコマンドライン管理の準備に役立ちます。

ステップ 1: AWS CLI の設定

AWS CLI は、Amazon EKS など AWS のサービスを操作するためのコマンドラインツールです。また、ローカルマシンから Amazon EKS クラスターやその他の AWS リソースにアクセスするための IAM ユーザーまたはロールの認証にも使用されます。コマンドラインから AWS のリソースをプロビジョニングするには、コマンドラインで使用する AWS アクセスキー ID とシークレットキーを取得する必要があります。次に、これらの認証情報を AWS CLI で設定する必要があります。AWS CLI をまだインストールしていない場合は、「AWS Command Line Interface ユーザーガイド」の「AWS CLI の最新バージョンをインストールまたは更新する」を参照してください。

アクセスキーを作成するには

1. AWS Management Console にサインインします。
2. 右上の AWS ユーザー名を選択し、ナビゲーションメニューを開きます。例えば、[**webadmin**] を選択します。次に、[セキュリティ認証情報] を選択します。
3. [アクセスキー] の下で、[アクセスキーの作成] を選択します。
4. [コマンドラインインターフェイス (CLI)] を選択し、[次へ] を選択します。
5. [Create access key] (アクセスキーの作成) を選択します。
6. [.csv ファイルをダウンロード] を選択します。

AWS CLI を設定するには

AWS CLI をインストールしたら、以下の手順に従って設定を行います。詳細については、「AWS Command Line Interface ユーザーガイド」の「AWS CLI を設定する」を参照してください。

1. ターミナルウィンドウで、以下のコマンドを入力します。

```
aws configure
```

オプションで、**--profile cluster-admin** などの名前付きプロファイルを設定できます。AWS CLI で名前付きプロファイルを設定する場合、以降のコマンドでは必ずこのフラグを渡す必要があります。

2. AWS 認証情報を入力します。例:

```
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Default region name [None]: region-code
Default output format [None]: json
```

セキュリティトークンを取得するには

必要に応じて、次のコマンドを実行して AWS CLI の新しいセキュリティトークンを取得します。詳細については、AWS CLI コマンドリファレンスの get-session-token を参照してください。

デフォルトでは、トークンは 15 分間有効です。デフォルトのセッションタイムアウトを変更するには、**--duration-seconds** フラグを渡します。例:

```
aws sts get-session-token --duration-seconds 3600
```

このコマンドは、AWS CLI セッションの一時的なセキュリティ認証情報を返します。以下のようなレスポンス出力が表示されます。

```
{
    "Credentials": {
        "AccessKeyId": "ASIA5FTRU3LOEXAMPLE",
        "SecretAccessKey": "JnKgvwfqUD9mNsPoi9IbxAYEXAMPLE",
        "SessionToken": "VERYLONGSESSIONTOKENSTRING",
        "Expiration": "2023-02-17T03:14:24+00:00"
    }
}
```

ユーザー ID を確認するには

必要に応じて、次のコマンドを実行して、ターミナルセッションの IAM ユーザー ID (*ClusterAdmin* など) の AWS 認証情報を確認します。

```
aws sts get-caller-identity
```

このコマンドは、AWS CLI に設定されている IAM エンティティの Amazon リソースネーム (ARN) を返します。以下のようなレスポンス出力例が表示されます。

```
{
    "UserId": "AKIAIOSFODNN7EXAMPLE",
    "Account": "01234567890",
    "Arn": "arn:aws:iam::01234567890:user/ClusterAdmin"
}
```

ステップ 2: Kubernetes ツールのインストール

Kubernetes クラスターと通信するには、Kubernetes API を操作するツールが必要です。さらに、ローカルマシン上の Kubernetes 環境を管理するためのものなど、他にもいくつかのツールが必要です。

AWS リソースを作成するには

- Amazon EKS クラスターリソース — AWS を初めて使用する場合は、`eksctl` をインストールすることをお勧めします。`eksctl` は、AWS CloudFormation を使用して Amazon EKS クラスターを簡単に作成する Infrastructure as Code (IaC) ユーティリティです。また、サービスアカウントなどの追加の Kubernetes リソースも作成します。`eksctl` のインストール手順については、`eksctl` ドキュメントの「インストール」を参照してください。
- AWS リソース — AWS インフラストラクチャのプロビジョニングとデプロイを自動化することに慣れている場合は、Terraform をインストールすることをお勧めします。Terraformは、HashiCorp によって開発されたオープンソースの Infrastructure as Code (IaC) ツールです。HashiCorp 構成言語 (HCL) や JSON などの高レベルの構成言語を使用してインフラストラクチャを定義およびプロビジョニングできます。Terraform のインストール手順については、Terraform ドキュメントの「Terraform のインストール」を参照してください。

`kubectl` をインストールするには

`kubectl` は、Amazon EKS クラスター上の Kubernetes API サーバーとの通信に使用されるオープンソースのコマンドラインツールです。ローカルマシンにまだインストールしていない場合は、次のオプションから選択してください。

- AWS バージョン — Amazon EKS がサポートする kubectl バージョンをインストールするには、「kubectl のインストールまたは更新」を参照してください。
- コミュニティバージョン — kubectl の最新のコミュニティバージョンをインストールするには、Kubernetes ドキュメントの「Install tools」ページを参照してください。

開発環境をセットアップするには

- ローカルデプロイツール — Kubernetes を初めて使用する場合は、minikube や kind などのローカルデプロイツールをインストールすることを検討してください。これらのツールを使用すると、ローカルマシンで Amazon EKS クラスターを管理できます。
- パッケージマネージャー — Helm は複雑なパッケージのインストールと管理を簡素化する Kubernetes の一般的なパッケージマネージャーです。Helm を使用すると、Amazon EKS クラスターの AWS ロードバランサーコントローラーなどのパッケージを簡単にインストールして管理できます。

次のステップ

- Amazon EKS の使用開始

kubectl のインストールまたは更新

Kubectl は、Kubernetes API サーバーと通信するために使用するコマンドラインツールです。kubectl バイナリは、多くのオペレーティングシステムパッケージマネージャーで利用できます。インストールにパッケージマネージャーを使用する方が、多くの場合、手動のダウンロードおよびインストールプロセスより簡単です。

このトピックは、デバイス上の kubectl バイナリのダウンロードおよびインストール、または更新に役立ちます。バイナリは アップストリームコミュニティのバージョン と同様です。バイナリは Amazon EKS または AWS に固有のものではありません。

> **Note**
>
> Amazon EKS クラスターコントロールプレーンとのマイナーバージョンの相違が 1 つ以内である kubectl バージョンを使用する必要があります。例えば、1.29 kubectl クライアントは Kubernetes、1.28、1.29 および 1.30 クラスターで動作します。

kubectl をインストールまたは更新するには

1. デバイスに kubectl が既にインストールされているかどうかを判断します。

   ```
   kubectl version --client
   ```

   kubectl をデバイスのパスにインストールした場合、出力例には以下のような情報が含まれます。現在インストールされているバージョンを新しいバージョンで更新する場合は、次のステップを完了し、新しいバージョンを現在のバージョンと同じ場所にインストールするようにします。

   ```
   Client Version: v1.30.X-eks-1234567
   ```

   出力が表示されない場合は、kubectl がインストールされていないか、デバイスのパス内の場所にインストールされていません。

2. macOS、Linux、および Windows のオペレーティングシステムで、kubectl を インストールまたは更新します。

   macOS

   kubectl を macOS にインストールまたは更新するには

   1. クラスターの Kubernetes バージョンの バイナリを Amazon S3 からダウンロードします。

      - Kubernetes 1.30

        ```
        curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.0/2024-05-12/bin/darwin/amd64/kubectl
        ```

      - Kubernetes 1.29

        ```
        curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.3/2024-04-19/bin/darwin/amd64/kubectl
        ```

      - Kubernetes 1.28

        ```
        curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.8/2024-04-19/bin/darwin/amd64/kubectl
        ```

- Kubernetes 1.27

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.12/2024-04-19/bin/darwin/amd64/kubectl
```

- Kubernetes 1.26

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-04-19/bin/darwin/amd64/kubectl
```

- Kubernetes 1.25

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.25.16/2024-04-19/bin/darwin/amd64/kubectl
```

- Kubernetes 1.24

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.24.17/2024-04-19/bin/darwin/amd64/kubectl
```

- Kubernetes 1.23

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.23.17/2024-04-19/bin/darwin/amd64/kubectl
```

- Kubernetes 1.22

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.22.17/2024-04-19/bin/darwin/amd64/kubectl
```

- Kubernetes 1.21

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.21.14/2024-04-19/bin/darwin/amd64/kubectl
```

2. (オプション) ダウンロードしたバイナリを、バイナリの SHA-256 チェックサムで検証します。

   a. クラスターの Kubernetes バージョンの SHA-256 チェックサムをダウンロードします。

      - Kubernetes 1.30

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.0/2024-05-12/
bin/darwin/amd64/kubectl.sha256
```

- Kubernetes 1.29

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.3/2024-04-19/
bin/darwin/amd64/kubectl.sha256
```

- Kubernetes 1.28

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.8/2024-04-19/
bin/darwin/amd64/kubectl.sha256
```

- Kubernetes 1.27

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.12/2024-04-19/
bin/darwin/amd64/kubectl.sha256
```

- Kubernetes 1.26

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-04-19/
bin/darwin/amd64/kubectl.sha256
```

- Kubernetes 1.25

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.25.16/2024-04-19/
bin/darwin/amd64/kubectl.sha256
```

- Kubernetes 1.24

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.24.17/2024-04-19/
bin/darwin/amd64/kubectl.sha256
```

- Kubernetes 1.23

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.23.17/2024-04-19/
bin/darwin/amd64/kubectl.sha256
```

- Kubernetes 1.22

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.22.17/2024-04-19/bin/darwin/amd64/kubectl.sha256
```

- Kubernetes 1.21

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.21.14/2024-04-19/bin/darwin/amd64/kubectl.sha256
```

b. ダウンロードしたバイナリの SHA-256 チェックサムを確認します。

```
openssl sha1 -sha256 kubectl
```

c. 出力で生成されたチェックサムが、ダウンロードした kubectl.sha256 ファイルのチェックサムと一致することを確認してください。

3. バイナリへの実行アクセス許可を適用します。

```
chmod +x ./kubectl
```

4. バイナリを PATH のフォルダにコピーします。既に kubectl のバージョンが既にインストールされている場合、$HOME/bin/kubectl を作成し、$HOME/bin が $PATH の最初にあることを確認することをお勧めします。

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
```

5. (オプション) シェルを開いたときに設定されるように、シェルの初期化ファイルに $HOME/bin パスを追加します。

```
echo 'export PATH=$HOME/bin:$PATH' >> ~/.bash_profile
```

Linux (amd64)

Linux (**amd64**) で **kubectl** をインストールまたは更新する方法

1. クラスターの Kubernetes バージョンの kubectl バイナリを Amazon S3 からダウンロードします。

- Kubernetes 1.30

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.0/2024-05-12/bin/linux/amd64/kubectl
```

- Kubernetes `1.30`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.0/2024-05-12/bin/linux/amd64/kubectl
```

- Kubernetes `1.29`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.3/2024-04-19/bin/linux/amd64/kubectl
```

- Kubernetes `1.28`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.8/2024-04-19/bin/linux/amd64/kubectl
```

- Kubernetes `1.27`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.12/2024-04-19/bin/linux/amd64/kubectl
```

- Kubernetes `1.26`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-04-19/bin/linux/amd64/kubectl
```

- Kubernetes `1.25`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.25.16/2024-04-19/bin/linux/amd64/kubectl
```

- Kubernetes `1.24`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.24.17/2024-04-19/bin/linux/amd64/kubectl
```

- Kubernetes `1.23`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.23.17/2024-04-19/
bin/linux/amd64/kubectl
```

- Kubernetes `1.22`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.22.17/2024-04-19/
bin/linux/amd64/kubectl
```

- Kubernetes `1.21`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.21.14/2024-04-19/
bin/linux/amd64/kubectl
```

2. (オプション) ダウンロードしたバイナリを、バイナリの SHA-256 チェックサムで検証します。

   a. デバイスのハードウェアプラットフォームのコマンドを使用して、クラスターの Kubernetes バージョンの SHA-256 チェックサムを Amazon S3 からダウンロードします。各バージョンの最初のリンクは amd64 用で、二つ目のリンクは `arm64` 用です。

   - Kubernetes `1.30`

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.0/2024-05-12/
   bin/linux/amd64/kubectl.sha256
   ```

   - Kubernetes `1.30`

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.0/2024-05-12/
   bin/linux/amd64/kubectl.sha256
   ```

   - Kubernetes `1.29`

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.3/2024-04-19/
   bin/linux/amd64/kubectl.sha256
   ```

   - Kubernetes `1.28`

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.8/2024-04-19/
   bin/linux/amd64/kubectl.sha256
   ```

   - Kubernetes `1.27`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.12/2024-04-19/bin/linux/amd64/kubectl.sha256
```

- Kubernetes 1.26

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-04-19/bin/linux/amd64/kubectl.sha256
```

- Kubernetes 1.25

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.25.16/2024-04-19/bin/linux/amd64/kubectl.sha256
```

- Kubernetes 1.24

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.24.17/2024-04-19/bin/linux/amd64/kubectl.sha256
```

- Kubernetes 1.23

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.23.17/2024-04-19/bin/linux/amd64/kubectl.sha256
```

- Kubernetes 1.22

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.22.17/2024-04-19/bin/linux/amd64/kubectl.sha256
```

- Kubernetes 1.21

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.21.14/2024-04-19/bin/linux/amd64/kubectl.sha256
```

b. 次のいずれかのコマンドを使用して、ダウンロードしたバイナリの SHA-256 チェックサムを確認します。

-

```
sha256sum -c kubectl.sha256
```

このコマンドを使用するときは、次の出力が表示されることを確認してください。

```
kubectl: OK
```

- ```
  openssl sha1 -sha256 kubectl
  ```

  このコマンドを使用する場合は、出力で生成されたチェックサムが、ダウンロードした kubectl.sha256 ファイルのチェックサムと一致することを確認してください。

3. バイナリへの実行アクセス許可を適用します。

```
chmod +x ./kubectl
```

4. バイナリを PATH のフォルダにコピーします。既に kubectl のバージョンが既にインストールされている場合、$HOME/bin/kubectl を作成し、$HOME/bin が $PATH の最初にあることを確認することをお勧めします。

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
```

5. (オプション) シェルを開いたときに設定されるように、シェルの初期化ファイルに $HOME/bin パスを追加します。

> **Note**
>
> このステップでは、Bash シェルを使用していることを前提としています。別のシェルを使用している場合は、特定のシェル初期化ファイルを使用するよう、コマンドを変更します。

```
echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc
```

Linux (arm64)

Linux (**arm64**) で **kubectl** をインストールまたは更新する方法

1. クラスターの Kubernetes バージョンの kubectl バイナリを Amazon S3 からダウンロードします。
   - Kubernetes 1.30
```

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.0/2024-05-12/bin/linux/arm64/kubectl
```

- Kubernetes `1.30`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.0/2024-05-12/bin/linux/arm64/kubectl
```

- Kubernetes `1.29`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.3/2024-04-19/bin/linux/arm64/kubectl
```

- Kubernetes `1.28`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.8/2024-04-19/bin/linux/arm64/kubectl
```

- Kubernetes `1.27`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.12/2024-04-19/bin/linux/arm64/kubectl
```

- Kubernetes `1.26`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-04-19/bin/linux/arm64/kubectl
```

- Kubernetes `1.25`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.25.16/2024-04-19/bin/linux/arm64/kubectl
```

- Kubernetes `1.24`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.24.17/2024-04-19/bin/linux/arm64/kubectl
```

- Kubernetes `1.23`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.23.17/2024-04-19/
bin/linux/arm64/kubectl
```

- Kubernetes `1.22`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.22.17/2024-04-19/
bin/linux/arm64/kubectl
```

- Kubernetes `1.21`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.21.14/2024-04-19/
bin/linux/arm64/kubectl
```

2. (オプション) ダウンロードしたバイナリを、バイナリの SHA-256 チェックサムで検証します。

   a. デバイスのハードウェアプラットフォームのコマンドを使用して、クラスターの Kubernetes バージョンの SHA-256 チェックサムを Amazon S3 からダウンロードします。各バージョンの最初のリンクは amd64 用で、二つ目のリンクは `arm64` 用です。

   - Kubernetes `1.30`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.0/2024-05-12/
bin/linux/arm64/kubectl.sha256
```

   - Kubernetes `1.30`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.0/2024-05-12/
bin/linux/arm64/kubectl.sha256
```

   - Kubernetes `1.29`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.3/2024-04-19/
bin/linux/arm64/kubectl.sha256
```

   - Kubernetes `1.28`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.8/2024-04-19/
bin/linux/arm64/kubectl.sha256
```

   - Kubernetes `1.27`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.12/2024-04-19/bin/linux/arm64/kubectl.sha256
```

- Kubernetes 1.26

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-04-19/bin/linux/arm64/kubectl.sha256
```

- Kubernetes 1.25

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.25.16/2024-04-19/bin/linux/arm64/kubectl.sha256
```

- Kubernetes 1.24

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.24.17/2024-04-19/bin/linux/arm64/kubectl.sha256
```

- Kubernetes 1.23

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.23.17/2024-04-19/bin/linux/arm64/kubectl.sha256
```

- Kubernetes 1.22

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.22.17/2024-04-19/bin/linux/arm64/kubectl.sha256
```

- Kubernetes 1.21

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.21.14/2024-04-19/bin/linux/arm64/kubectl.sha256
```

b. 次のいずれかのコマンドを使用して、ダウンロードしたバイナリの SHA-256 チェックサムを確認します。

-

```
sha256sum -c kubectl.sha256
```

このコマンドを使用するときは、次の出力が表示されることを確認してください。

```
kubectl: OK
```

- ```
  openssl sha1 -sha256 kubectl
  ```

  このコマンドを使用する場合は、出力で生成されたチェックサムが、ダウンロードした kubectl.sha256 ファイルのチェックサムと一致することを確認してください。

3. バイナリへの実行アクセス許可を適用します。

```
chmod +x ./kubectl
```

4. バイナリを PATH のフォルダにコピーします。既に kubectl のバージョンが既にインストールされている場合、$HOME/bin/kubectl を作成し、$HOME/bin が $PATH の最初にあることを確認することをお勧めします。

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
```

5. (オプション) シェルを開いたときに設定されるように、シェルの初期化ファイルに $HOME/bin パスを追加します。

> **Note**
>
> このステップでは、Bash シェルを使用していることを前提としています。別のシェルを使用している場合は、特定のシェル初期化ファイルを使用するよう、コマンドを変更します。

```
echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc
```

Windows

kubectl を Windows にインストールまたは更新するには

1. PowerShell ターミナルを開きます。
2. クラスターの Kubernetes バージョンの kubectl バイナリを Amazon S3 からダウンロードします。
```

- Kubernetes 1.30

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.0/2024-05-12/bin/windows/amd64/kubectl.exe
```

- Kubernetes 1.29

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.3/2024-04-19/bin/windows/amd64/kubectl.exe
```

- Kubernetes 1.28

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.8/2024-04-19/bin/windows/amd64/kubectl.exe
```

- Kubernetes 1.27

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.12/2024-04-19/bin/windows/amd64/kubectl.exe
```

- Kubernetes 1.26

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-04-19/bin/windows/amd64/kubectl.exe
```

- Kubernetes 1.25

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.25.16/2024-04-19/bin/windows/amd64/kubectl.exe
```

- Kubernetes 1.24

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.24.17/2024-04-19/bin/windows/amd64/kubectl.exe
```

- Kubernetes 1.23

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.23.17/2024-04-19/bin/windows/amd64/kubectl.exe
```

- Kubernetes 1.22

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.22.17/2024-04-19/bin/windows/amd64/kubectl.exe
```

- Kubernetes `1.21`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.21.14/2024-04-19/bin/windows/amd64/kubectl.exe
```

3. (オプション) ダウンロードしたバイナリを、バイナリの SHA-256 チェックサムで検証します。

   a. Windows 用のクラスターの Kubernetes バージョンの SHA-256 チェックサムをダウンロードします。

   - Kubernetes `1.30`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.30.0/2024-05-12/bin/windows/amd64/kubectl.exe.sha256
```

   - Kubernetes `1.29`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.29.3/2024-04-19/bin/windows/amd64/kubectl.exe.sha256
```

   - Kubernetes `1.28`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.28.8/2024-04-19/bin/windows/amd64/kubectl.exe.sha256
```

   - Kubernetes `1.27`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.27.12/2024-04-19/bin/windows/amd64/kubectl.exe.sha256
```

   - Kubernetes `1.26`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.26.15/2024-04-19/bin/windows/amd64/kubectl.exe.sha256
```

   - Kubernetes `1.25`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.25.16/2024-04-19/bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes 1.24

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.24.17/2024-04-19/bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes 1.23

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.23.17/2024-04-19/bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes 1.22

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.22.17/2024-04-19/bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes 1.21

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.21.14/2024-04-19/bin/windows/amd64/kubectl.exe.sha256
```

b. ダウンロードしたバイナリの SHA-256 チェックサムを確認します。

```
Get-FileHash kubectl.exe
```

c. 出力で生成されたチェックサムが、ダウンロードした kubectl.sha256 ファイルのチェックサムと一致することを確認してください。PowerShell の出力は、大文字の同じ文字列である必要があります。

4. バイナリを PATH のフォルダにコピーします。コマンドラインユーティリティで使用する PATH 内に既存のディレクトリがある場合、そのディレクトリにバイナリをコピーします。それ以外の場合は、以下の手順を完了します。

a. コマンドラインのバイナリ用に新しいディレクトリ (C:\bin など) を作成します。

b. kubectl.exe バイナリを新しいディレクトリにコピーします。

c. ユーザーまたはシステムの PATH 環境変数を編集し、新しいディレクトリを PATH に追加します。

d. 新しい PATH 可変を取得するには、PowerShell ターミナルを閉じ、新しいターミナルを開きます。

3. kubectl をインストールしたら、そのバージョンを確認できます。

```
kubectl version --client
```

kubectl を初めてインストールする場合、まだどのサーバーとも通信するように設定されていません。この設定については、必要に応じて他の手順で説明します。特定のクラスターと通信するために設定を更新する必要が生じた場合は、以下のコマンドを実行できます。*region-code* をクラスターのある AWS リージョン に置き換えます。*my-cluster* を自分のクラスター名に置き換えます。

```
aws eks update-kubeconfig --region region-code --name my-cluster
```

Amazon EKS の使用開始

入門ガイドを読む前に、Amazon EKS を使用するように設定されていることを確認してください。詳細については、「Amazon EKS のセットアップ」を参照してください。

Amazon EKS でノードを使用して新しい Kubernetes クラスターを作成するために利用できる 2 つの入門ガイドがあります。

- Amazon EKS の開始方法 – eksctl – この入門ガイドは、eksctl (Amazon EKS 上に Kubernetes クラスターを作成および管理するためのシンプルなコマンドラインユーティリティ) を使用して、Amazon EKS の開始に必要なすべてのリソースをインストールする方法を解説しています。チュートリアルを完了すると、アプリケーションのデプロイが可能な Amazon EKS クラスターが実行状態になります。このガイドの利用が、Amazon EKS の使用を開始するための最も迅速かつシンプルな手段です。
- Amazon EKS の開始方法 – AWS Management Console と AWS CLI – この入門ガイドは、AWS Management Console と AWS CLI を使用して Amazon EKS を開始するために必要な、すべてのリソースの作成に役立ちます。チュートリアルを完了すると、アプリケーションのデプロイが可能な Amazon EKS クラスターが実行状態になります。このガイドでは、Amazon EKS クラスターに必要なリソースを手動で個別に作成していきます。ここでの手順に従うことで、各リソースがどのように作成されたか、および、リソース間のやり取りがどのように行われているかなどを、把握できるようになります。

また、以下のリファレンスも提供しています。

- 厳選された実践的なチュートリアルのコレクションについては、AWS コミュニティで「Amazon EKS のナビゲーション」を参照してください。
- コード例については、「Code examples for Amazon EKS using AWS SDKs」を参照してください。

Amazon EKS の開始方法 – eksctl

このガイドでは、Amazon EKS 上に Kubernetes クラスターを作成および管理するためのシンプルなコマンドラインユーティリティである eksctl を使用して、Amazon Elastic Kubernetes Service (Amazon EKS) の開始に必要なリソースのすべてを作成する方法を解説していきます。このチュートリアルの終わりには、アプリケーションのデプロイが可能な、実行状態の Amazon EKS クラスターが完成します。

AWS Management Console からクラスターを作成する場合には手動で作成する必要のあるリソースのいくつかが、このガイドの手順に従うことで自動的に作成されます。リソース間での連携方法について良く理解するために、大半のリソースを手動で作成する場合には、クラスターと計算機能の作成に AWS Management Console を使用します。詳細については、「Amazon EKS の開始方法 – AWS Management Console と AWS CLI」を参照してください。

前提条件

このチュートリアルを開始する前に、Amazon EKS クラスターの作成と管理に必要な次のツールとリソースを、インストールおよび設定しておく必要があります。

- **`kubectl`** - Kubernetes クラスターを操作するためのコマンドラインツール。詳細については、「kubectl のインストールまたは更新」を参照してください。
- **`eksctl`** – EKS クラスターで多くの個別のタスクを自動化するために使用するコマンドラインツール。詳細については、`eksctl` ドキュメントの「インストール」を参照してください。
- 必要な IAM 許可 – 使用している IAM セキュリティプリンシパルには、Amazon EKS の IAM ロール、サービスにリンクされたロール、AWS CloudFormation、VPC、その関連リソースを操作するための権限が必要になります。詳細については、「IAM ユーザーガイド」の「Amazon Elastic Container Service for Kubernetes のアクション、リソース、および条件キー」、および「サービスにリンクされたロールの使用」を参照してください。このガイドのすべての手順は、1 つのユーザーとして実行する必要があります。現在のユーザーを確認するには、次のコマンドを実行します。

```
aws sts get-caller-identity
```

ステップ 1: Amazon EKS クラスターとノードを作成する

> **Important**
>
> 可能な限りシンプルかつ迅速に使用を開始するため、このトピックでは、クラスターとノードをデフォルト設定で作成する手順について説明します。実稼働で使用するクラスターとノードを作成する際には、すべての設定内容に習熟した上で、ご自身の要件を満たす設定でクラスターとノードをデプロイし直すことをお勧めします。詳細については、Amazon EKS クラスターの作成およびAmazon EKS ノードを参照してください。一部の設定は、クラスターとノードの作成時にのみ有効にできます。

クラスターの作成には、次のいずれかのノードタイプが使用できます。各タイプの詳細については、「Amazon EKS ノード」を参照してください。クラスターをデプロイした後に、他のノードタイプを追加できます。

- Fargate - Linux - AWS Fargate で Linux アプリケーションを実行する場合は、このタイプのノードを選択します。Fargate は、Amazon EC2 インスタンスを管理せずに Kubernetes Pods をデプロイできるサーバーレスコンピューティングエンジンです。
- マネージド型ノード - Linux - Amazon EC2 インスタンスで Amazon Linux アプリケーションを実行する場合は、このタイプのノードを選択します。このガイドでは説明しませんが、Windows セルフマネージド型ノードと Bottlerocket ノードをクラスターに追加することもできます。

次のコマンドを使用して、Amazon EKS クラスターを作成します。*my-cluster* は独自の値に置き換えることができます。この名前には、英数字 (大文字と小文字が区別されます) とハイフンのみを使用できます。先頭の文字は英数字である必要があります。また、100 文字より長くすることはできません。名前は、クラスターを作成する AWS リージョン および AWS アカウント 内で一意である必要があります。*region-code* を Amazon EKS でサポートされている AWS リージョン に置き換えます。AWS リージョン の一覧については、AWS 全般的なリファレンスガイドの Amazon EKS エンドポイントとクォータ を参照してください。

Fargate – Linux

```
eksctl create cluster --name my-cluster --region region-code --fargate
```

Managed nodes – Linux

```
eksctl create cluster --name my-cluster --region region-code
```

クラスターの作成には数分かかります。作成中に、数行の出力が表示されます。出力の最後の行は、次のサンプル行のようになります。

```
[...]
[#]  EKS cluster "my-cluster" in "region-code" region is ready
```

eksctl により、~/.kube 内に kubectl の config ファイルが作成されるか、コンピュータ上の ~/.kube 内に既存の config ファイルの中に新しいクラスター設定が追加されます。

クラスターの作成が完了したら、AWS CloudFormation コンソール (https://console.aws.amazon.com/cloudformation) で、eksctl-*my-cluster*-cluster という名前の AWS CloudFormation スタックを表示して、作成されたすべてのリソースを確認します。

ステップ 2: Kubernetes リソースの表示

1. クラスターノードを表示します。

```
kubectl get nodes -o wide
```

出力例は次のとおりです。

Fargate – Linux

```
NAME                                                STATUS   ROLES    AGE
   VERSION              INTERNAL-IP   EXTERNAL-IP   OS-IMAGE         KERNEL-
VERSION                   CONTAINER-RUNTIME
fargate-ip-192-0-2-0.region-code.compute.internal   Ready    <none>
  8m3s    v1.2.3-eks-1234567   192.0.2.0     <none>        Amazon Linux 2
  1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
fargate-ip-192-0-2-1.region-code.compute.internal   Ready    <none>
  7m30s   v1.2.3-eks-1234567   192-0-2-1     <none>        Amazon Linux 2
  1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
```

Managed nodes – Linux

```
NAME                                          STATUS   ROLES    AGE    VERSION
          INTERNAL-IP   EXTERNAL-IP   OS-IMAGE         KERNEL-VERSION
       CONTAINER-RUNTIME
ip-192-0-2-0.region-code.compute.internal   Ready    <none>   6m7s
   v1.2.3-eks-1234567   192.0.2.0     192.0.2.2     Amazon Linux 2
  1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
ip-192-0-2-1.region-code.compute.internal   Ready    <none>   6m4s
   v1.2.3-eks-1234567   192.0.2.1     192.0.2.3     Amazon Linux 2
  1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
```

出力に表示される内容の詳細については、「Kubernetes リソースを表示する」を参照してください。

2. クラスターで実行されているワークロードを表示します。

```
kubectl get pods -A -o wide
```

出力例は次のとおりです。

Fargate – Linux

```
NAMESPACE     NAME                          READY   STATUS    RESTARTS   AGE   IP
          NODE                                                NOMINATED NODE
 READINESS GATES
kube-system   coredns-1234567890-abcde   1/1     Running   0          18m
  192.0.2.0   fargate-ip-192-0-2-0.region-code.compute.internal   <none>
   <none>
kube-system   coredns-1234567890-12345   1/1     Running   0          18m
  192.0.2.1   fargate-ip-192-0-2-1.region-code.compute.internal   <none>
   <none>
```

Managed nodes – Linux

```
NAMESPACE     NAME                          READY   STATUS    RESTARTS   AGE     IP
          NODE                                        NOMINATED NODE   READINESS
 GATES
kube-system   aws-node-12345             1/1     Running   0          7m43s
   192.0.2.1   ip-192-0-2-1.region-code.compute.internal   <none>
 <none>
kube-system   aws-node-67890             1/1     Running   0          7m46s
   192.0.2.0   ip-192-0-2-0.region-code.compute.internal   <none>
 <none>
kube-system   coredns-1234567890-abcde   1/1     Running   0          14m
   192.0.2.3   ip-192-0-2-3.region-code.compute.internal   <none>
 <none>
kube-system   coredns-1234567890-12345   1/1     Running   0          14m
   192.0.2.4   ip-192-0-2-4.region-code.compute.internal   <none>
 <none>
kube-system   kube-proxy-12345           1/1     Running   0          7m46s
   192.0.2.0   ip-192-0-2-0.region-code.compute.internal   <none>
 <none>
kube-system   kube-proxy-67890           1/1     Running   0          7m43s
   192.0.2.1   ip-192-0-2-1.region-code.compute.internal   <none>
 <none>
```

出力に表示される内容の詳細については、「Kubernetes リソースを表示する」を参照してください。

ステップ 3: クラスターとノードを削除する

このチュートリアルのために作成したクラスターとノードの使用が終了したら、クリーンアップのために、次のコマンドを使用してそれらのクラスターとノードを削除する必要があります。クリーンアップせずに、他の目的でこのクラスターを使用する場合は、「次のステップ」を参照してください。

```
eksctl delete cluster --name my-cluster --region region-code
```

次のステップ

以下のトピックは、クラスターの機能を拡張するのに役立ちます。

- サンプルアプリケーションをクラスターにデプロイします。
- クラスターを作成した IAM プリンシパルは、kubectl または AWS Management Console を使用して Kubernetes API サーバーを呼び出すことができる唯一のプリンシパルです。他の IAM プリンシパルがクラスターにアクセスできるようにする場合は、それらを追加する必要があります。詳細については、Kubernetes API へのアクセスを許可する および必要なアクセス許可を参照してください。
- 本番用にクラスターをデプロイする前に、クラスターとノードのすべての設定を理解しておくことをお勧めします。Amazon EC2 ノードへの SSH アクセスの有効化など一部の設定は、クラスターの作成時に行う必要があります。
- クラスターのセキュリティを強化するには、サービスアカウントの IAM ロールを使用する Amazon VPC コンテナネットワークインターフェイスプラグインの設定を行ってください。

Amazon EKS の開始方法 – AWS Management Console と AWS CLI

このガイドでは、AWS Management Console と AWS CLI を使用して、Amazon Elastic Kubernetes Service (Amazon EKS) を使用開始するために必要なすべてのリソースを作成する方法を解説しま

す。ここでは、各リソースを手動で作成します。このチュートリアルの終わりには、アプリケーションのデプロイが可能な、実行状態の Amazon EKS クラスターが完成します。

このガイドの手順に従うと、各リソースがどのように作成され、リソース間でどのようにやり取りするかを全面的に把握できます。大半のリソースを自動的に作成させたい場合には、クラスターとノードの作成に `eksctl` CLI を使用します。詳細については、「Amazon EKS の開始方法 – eksctl」を参照してください。

前提条件

このチュートリアルを開始する前に、Amazon EKS クラスターの作成と管理に必要な次のツールとリソースを、インストールおよび設定しておく必要があります。

- AWS CLI – Amazon EKS など AWS のサービスを操作するためのコマンドラインツールです。詳細については、 ユーザーガイドの、AWS CLI のインストール、更新、およびアンインストール AWS Command Line Interfaceを参照してください。AWS CLI のインストール後は、設定も行っておくことをお勧めします。詳細については、 ユーザーガイドの`aws configure` でクイック設定 AWS Command Line Interfaceを参照してください。
- **`kubectl`** - Kubernetes クラスターを操作するためのコマンドラインツール。詳細については、「`kubectl` のインストールまたは更新」を参照してください。
- 必要な IAM アクセス許可 – 使用している IAM セキュリティプリンシパルには、Amazon EKS の IAM ロール、サービスにリンクされたロール、AWS CloudFormation、VPC、その関連リソースを操作するための権限が必要になります。詳細については、「IAM ユーザーガイド」の「Amazon Elastic Kubernetes サービスのアクション、リソース、および条件キー」および「サービスにリンクされたロールの使用」を参照してください。このガイドのすべての手順は、1 つのユーザーとして実行する必要があります。現在のユーザーを確認するには、次のコマンドを実行します。

  ```
  aws sts get-caller-identity
  ```

- このトピック内の手順は、 Bash シェル内で実行することが推奨されます。Bash シェルを使用していない場合、行継続文字や、変数の設定と使用に関する方法など、一部のスクリプトコマンドのためにシェルの調整が必要となります。さらに、シェルの引用規則とエスケープ規則は異なる場合があります。詳細については、「AWS Command Line Interface ユーザーガイド」の「AWS CLI での文字列への引用符の使用」を参照してください。

ステップ 1: Amazon EKS クラスターを作成する

> **Important**
>
> 可能な限りシンプルかつ迅速に使用を開始するため、このトピックでは、クラスターをデフォルトの設定で作成するステップについて説明します。本番で使用するクラスターを作成する前に、すべての設定内容に習熟した上で、要件を満たす設定でクラスターをデプロイすることをお勧めします。詳細については、「Amazon EKS クラスターの作成」を参照してください。一部の設定は、クラスターの作成時にのみ有効にできます。

クラスターを作成するには

1. Amazon EKS の要件を満たすように、パブリックサブネットとプライベートサブネットを持つ Amazon VPC を作成します。*region-code* を Amazon EKS でサポートされている AWS リージョン に置き換えます。AWS リージョン の一覧については、「AWS 全般的なリファレンスガイド」の「Amazon EKS エンドポイントとクォータ」を参照してください。*my-eks-vpc-stack* は、好みの任意の名前で置き換えることができます。

   ```
   aws cloudformation create-stack \
     --region region-code \
     --stack-name my-eks-vpc-stack \
     --template-url https://s3.us-west-2.amazonaws.com/amazon-
   eks/cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml
   ```

   > **Tip**
   >
   > 前のコマンドが作成したすべてのリソースの一覧については、AWS CloudFormation コンソール (https://console.aws.amazon.com/cloudformation) を開きます。*my-eks-vpc-stack* スタックを選択し、[リソース] タブを選択します。

2. クラスター IAM ロールを作成し、必要な Amazon EKS IAM マネージドポリシーをそれにアタッチします。Amazon EKS によって管理される Kubernetes クラスターは、サービスで使用するリソースを管理するために、ユーザーに代わって他の AWS サービスを呼び出します。

   a. 次の内容を *eks-cluster-role-trust-policy.json* という名前のファイルにコピーします。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

b. ロールを作成します。

```
aws iam create-role \
  --role-name myAmazonEKSClusterRole \
  --assume-role-policy-document file://"eks-cluster-role-trust-policy.json"
```

c. このロールに、必要な Amazon EKS 管理の IAM ポリシーをアタッチします。

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy \
  --role-name myAmazonEKSClusterRole
```

3. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。

   コンソールの右上に表示されている AWS リージョン が、クラスターを作成する AWS リージョン であることを確認します。異なる場合は、AWS リージョン 名の横にあるドロップダウンを展開し、使用する AWS リージョン を選択します。

4. [クラスターを追加]、[作成] の順に選択します。このオプションが表示されない場合は、まず左のナビゲーションペインの [クラスター] を選択します。

5. [クラスターの設定] ページで、次の手順を実行します。

   a. [名前] に、クラスターの名前 (**my-cluster** など) を入力します。この名前には、英数字 (大文字と小文字が区別されます) とハイフンのみを使用できます。先頭の文字は英数字である必要があります。また、100 文字より長くすることはできません。名前は、クラスターを作成する AWS リージョン および AWS アカウント 内で一意である必要があります。

   b. [クラスターサービスロール] で、「*myAmazonEKSClusterRole*」を選択します。

c. その他の設定はデフォルト値のままにし、[次へ] をクリックします。

6. [ネットワーキングを指定] ページで、以下の作業を行います。

a. 前のステップで [VPC] ドロップダウンリストから作成した VPC の ID を選択します。この ID は *vpc-00x0000x000x0x000* | *my-eks-vpc-stack-VPC* のような値です。

b. その他の設定はデフォルト値のままにし、[次へ] をクリックします。

7. [オブザーバビリティの設定]ページで、[次へ] を選択します。

8. [アドオンの選択] ページで、[次へ] を選択します。

アドオンの詳細については、「Amazon EKS アドオン」を参照してください。

9. [選択したアドオンセッティングの設定] ページで、[次へ] を選択します。

10. [確認して作成] ページで、[作成] をクリックします。

クラスターのプロビジョニングプロセスが完了するまで、数分の間、クラスター名の右側でステータスが [作成中] と表示されます。ステータスが [アクティブ] になるまで、次のステップに進まないでください。

> **Note**
>
> リクエストで指定したアベイラビリティーゾーンのいずれかに、Amazon EKS クラスターの作成に十分な容量がない場合には、エラーが表示されることがあります。このエラー出力には、新しいクラスターをサポートできるアベイラビリティーゾーンが表示されます。アカウント向けにサポートされているアベイラビリティーゾーンにある 2 つ以上のサブネットを使用して、クラスターを作成します。詳細については、「容量不足」を参照してください。

ステップ 2: 自分のコンピュータでクラスターとの通信を設定する

このセクションでは、クラスター用の kubeconfig ファイルを作成します。このファイルの設定により、kubectl CLI からクラスターへの通信ができるようになります。

クラスターと通信するようにコンピュータを設定するには

1. クラスター用の kubeconfig ファイルを作成もしくは更新します。*region-code* を、クラスターを作成する AWS リージョン に置き換えます。*my-cluster* を自分のクラスター名に置き換えます。

```
aws eks update-kubeconfig --region region-code --name my-cluster
```

デフォルトでは、config ファイルが ~/.kube に作成されるか、config ファイルが既に ~/.kube に存在する場合には、その中に新しいクラスター設定が追加されます。

2. 設定をテストします。

```
kubectl get svc
```

> **Note**
>
> 認証またはリソースタイプのエラーが発生した場合は、トラブルシューティングトピックの「許可されていないか、アクセスが拒否されました (kubectl)」を参照してください。

出力例は次のとおりです。

```
NAME             TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
svc/kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   1m
```

ステップ 3: ノードを作成する

> **Important**
>
> 可能な限りシンプルかつ迅速に使用を開始するため、このトピックでは、ノードをデフォルトの設定で作成するステップについて説明します。本番で使用するノードを作成する前に、すべての設定内容に習熟した上で、要件を満たす設定でノードをデプロイすることをお勧めします。詳細については、「Amazon EKS ノード」を参照してください。一部の設定は、ノードの作成時にのみ有効にできます。

クラスターの作成には、次のいずれかのノードタイプが使用できます。各タイプの詳細については、「Amazon EKS ノード」を参照してください。クラスターをデプロイした後に、他のノードタイプを追加できます。

- Fargate - Linux - AWS Fargate で Linux アプリケーションを実行する場合は、このタイプのノードを選択します。Fargate は、Amazon EC2 インスタンスを管理せずに Kubernetes Pods をデプロイできるサーバーレスコンピューティングエンジンです。
- マネージド型ノード - Linux - 実行する場合は、このタイプのノードを選択します。このガイドでは説明しませんが、Windows セルフマネージド型 ノードと Bottlerocket ノードをクラスターに追加することもできます。

Fargate – Linux

Fargate プロファイルを作成します。デプロイされる Kubernetes Pods が、このプロファイルで定義される基準を満たしている場合、その Pods は Fargate にデプロイされます。

Fargate プロファイルを作成するには

1. IAM ロールを作成して、必要な Amazon EKS IAM 管理ポリシーをアタッチします。クラスターが Fargate インフラストラクチャ上で Pods を作成する場合、Fargate インフラストラクチャ上で実行されているコンポーネントは、ユーザーに代わって AWS API にコールを実行する必要があります。これは、Amazon ECR からコンテナイメージをプルしたり、ログを他の AWS サービスにルーティングしたりするなどのアクションを実行できるようにするためです。Amazon EKS の Pod 実行ロールにより、これらを行うための IAM アクセス許可が付与されます。

   a. 次の内容を *pod-execution-role-trust-policy*.json という名前のファイルにコピーします。*region-code* をクラスターのある AWS リージョン に置き換えます。アカウントで、すべての AWS リージョン で同じロールを使用する場合は、*region-code* を * に置き換えます。*111122223333* をアカウント ID に置き換え、*my-cluster* を自分のクラスター名に置き換えます。アカウント内のすべてのクラスターに同じロールを使用する場合は、*my-cluster* を * に置き換えます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Condition": {
         "ArnLike": {
            "aws:SourceArn": "arn:aws:eks:region-
code:111122223333:fargateprofile/my-cluster/*"
         }
```

```
      },
      "Principal": {
        "Service": "eks-fargate-pods.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

b. Pod 実行 IAM ロールを作成します。

```
aws iam create-role \
  --role-name AmazonEKSFargatePodExecutionRole \
  --assume-role-policy-document file://"pod-execution-role-trust-policy.json"
```

c. このロールに、必要な Amazon EKS 管理の IAM ポリシーをアタッチします。

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSFargatePodExecutionRolePolicy \
  --role-name AmazonEKSFargatePodExecutionRole
```

2. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。
3. [クラスター] ページで、*my-cluster* クラスターを選択します。
4. [***my-cluster***] ページで、次の操作を行います。

   a. [コンピューティング] タブを開きます。

   b. [Fargate プロファイル] で、[Fargate プロファイルを追加] を選択します。

5. [Fargate プロファイルを設定] ページで、次の操作を行います。

   a. [名前] に、Fargate プロファイルの一意の名前 (***my-profile*** など) を入力します。

   b. [ポッド実行ロール] で、以前のステップで作成した [AmazonEKSFargatePodExecutionRole] を選択します。

   c. [サブネット] ドロップダウンを展開し、名前に Public を含むすべてのサブネットの選択を解除します。Fargate で実行される Pods では、プライベートサブネットのみがサポートされます。

   d. [Next] を選択します。

6. [Pod の選択を設定] ページで、次の操作を行います。

   a. [名前空間] に **default** と入力します。

   b. [Next] を選択します。

7. [確認と作成] ページで、Fargate プロファイルの情報を確認し、[作成] を選択します。

8. 数分後、[Fargate プロファイル設定] セクションにある [ステータス] が [作成中] から [アクティブ] に変わります。ステータスが [アクティブ] になるまで、次のステップに進まないでください。

9. すべての Pods を Fargate にデプロイする場合 (Amazon EC2 ノードにはデプロイしない場合)、次の手順を実行して別の Fargate プロファイルを作成し、Fargate でデフォルトのネームリゾルバー (CoreDNS) を実行します。

   > **Note**
   >
   > これを行わな場合、現時点では利用可能なノードはありません。

   a. [Fargate プロファイル] ページで *[my-profile]* を選択します。

   b. [Fargate プロファイル] で、[Fargate プロファイルを追加] を選択します。

   c. [名前] に ***CoreDNS*** と入力します。

   d. [ポッド実行ロール] で、以前のステップで作成した [AmazonEKSFargatePodExecutionRole] を選択します。

   e. [サブネット] ドロップダウンを展開し、名前に `Public` を含むすべてのサブネットの選択を解除します。Fargate で実行される Pods では、プライベートサブネットのみがサポートされます。

   f. [Next] を選択します。

   g. [名前空間] に **kube-system** と入力します。

   h. [ラベルを一致させる] を選択し、次に [ラベルを追加] をクリックします。

   i. [キー] に **k8s-app** を入力し、値として **kube-dns** を入力します。これは、デフォルト名のリゾルバ (CoreDNS) を Fargate にデプロイするために必要です。

   j. [Next] を選択します。

   k. [確認と作成] ページで、Fargate プロファイルの情報を確認し、[作成] を選択します。

   l. 次のコマンドを実行して、CoreDNS Pods からデフォルトの `eks.amazonaws.com/compute-type : ec2` アノテーションを削除します。

```
kubectl patch deployment coredns \
    -n kube-system \
    --type json \
    -p='[{"op": "remove", "path": "/spec/template/metadata/annotations/
eks.amazonaws.com~1compute-type"}]'
```

> **Note**
>
> 追加した Fargate プロファイルのラベルに基づいて、システムにより 2 つのノードが作成されデプロイされます。これらはいずれも Fargate ノードに適用できないため、[ノードグループ] のリストには表示されません。新しいノードは [概要] タブ内のリストに表示されます。

Managed nodes – Linux

前の手順で作成したサブネットとノード IAM ロールを指定しながら、マネージド型ノードグループを作成します。

Amazon EC2 Linux マネージド型ノードグループを作成するには

1. ノードの IAM ロールを作成して、必要な Amazon EKS IAM 管理ポリシーをアタッチします。Amazon EKS ノード `kubelet` デーモンが、ユーザーに代わって AWS API への呼び出しを実行します。ノードは、IAM インスタンスプロファイルおよび関連ポリシーを通じて、これらの API コールのアクセス許可を受け取ります。

   a. 次の内容を *`node-role-trust-policy.json`* という名前のファイルにコピーします。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
```

```
  ]
}
```

b. ノードの IAM ロールを作成します。

```
aws iam create-role \
  --role-name myAmazonEKSNodeRole \
  --assume-role-policy-document file://"node-role-trust-policy.json"
```

c. ロールに、必要なマネージド IAM ポリシーをアタッチします。

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name myAmazonEKSNodeRole
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name myAmazonEKSNodeRole
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name myAmazonEKSNodeRole
```

2. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。
3. ステップ 1: Amazon EKS クラスターを作成する で作成したクラスターの名前 (*my-cluster* など) を選択します。
4. [***my-cluster***] ページで、次の操作を行います。
   a. [コンピューティング] タブを開きます。
   b. [ノードグループを追加] をクリックします。
5. [ノードグループの設定] ページで以下を実行します。
   a. [名前] に、マネージド型ノードグループの一意の名前 (***my-nodegroup*** など) を入力します。ノードグループ名は 63 文字以下である必要があります。先頭は文字または数字でなければなりませんが、残りの文字にはハイフンおよびアンダースコアを含めることもできます。
   b. [ノード IAM ロール名] で、前のステップで作成した *myAmazonEKSNodeRole* ロールを選択します。各ノードグループには独自の一意の IAM ロールを使用することをお勧めします。
   c. [Next] を選択します。

6. [コンピューティングとスケーリングの設定] ページではデフォルトの値を受け入れ、[次へ] をクリックします。
7. [ネットワーキングの指定] ページでは、デフォルトの値を受け入れ、[次へ] をクリックします。
8. [確認と作成] ページで、マネージド型ノードグループの設定を確認し、[作成] を選択します。
9. 数分後、[ノードグループの設定] セクションにある [ステータス] の表示が、[作成中] から [アクティブ] に変わります。ステータスが [アクティブ] になるまで、次のステップに進まないでください。

ステップ 4: リソースを表示する

ノードと Kubernetes のワークロードを表示することができます。

ノードとワークロードを表示するには

1. 左のナビゲーションペインで [クラスター] を選択します。[クラスター] のリストで *my-cluster* など、作成したクラスターの名前を選択します。
2. [***my-cluster***] ページで、次の項目を選択します。
   a. [コンピューティング] タブ – クラスターにデプロイした [ノード] が一覧表示されます。ノードの名前を選択すると、そのノードに関するより詳細な情報が表示されます。
   b. [リソース] タブ - Amazon EKS クラスターにデフォルトでデプロイされたすべての Kubernetes リソースが表示されます。コンソールでリソースタイプを選択すると、その詳細を確認できます。

ステップ 5: リソースを削除する

このチュートリアルのために作成したクラスターとノードの使用が終了したら、作成したリソースを削除する必要があります。リソースを削除する前に、他の目的でこのクラスターを使用する場合は、次のステップ を参照してください。

このガイドで作成したリソースを削除するには

1. ここで作成した、いずれかのノードグループもしくは Fargate プロファイルを削除します。

a. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。

b. 左のナビゲーションペインで [クラスター] を選択します。クラスターのリストから *my-cluster* を選択します。

c. [コンピューティング] タブを開きます。

d. ノードグループを作成している場合は、 ノードグループ *my-nodegroup* を した上で、[削除] を選択します。「***my-nodegroup***」と入力し、[削除] を選択します。

e. 作成した各 Fargate プロファイルを選択した後に、[削除] をクリックします。プロファイルの名前を入力した後に、[削除] をクリックします。

> **Note**
>
> 続いて 2 つ目の Fargate プロファイルを削除する場合、最初のプロファイルの削除が完了するまで待機します。

f. ノードグループまたは Fargate プロファイルが削除されるまで続行しないでください。

2. クラスターを削除します。

a. 左のナビゲーションペインで [クラスター] を選択します。クラスターのリストから *my-cluster* を選択します。

b. [クラスターの削除] を選択します。

c. ***my-cluster*** と入力し、[削除] を選択します。クラスターが削除されるまで続行しないでください。

3. 作成した VPC AWS CloudFormation スタックを削除します。

a. https://console.aws.amazon.com/cloudformation で AWS CloudFormation コンソール を開きます。

b. [*my-eks-vpc-stack*] スタックを選択してから、[削除] を選択します。

c. [***my-eks-vpc-stack*** の削除] 確認ダイアログボックスで、[スタックを削除] をクリックします。

4. 作成した IAM ロールを削除します。

a. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。

b. 左のナビゲーションペインで、[ロール] を選択します。

c. リスト ([*myAmazonEKSClusterRole*]、および、[AmazonEKSFargatePodExecutionRole] または *[myAmazonEKSNodeRole]*) から、作成した各ロールを選択します。[削除] を選択し、要求された確認テキストを入力した後で、[削除] をクリックします。

次のステップ

以下のトピックは、クラスターの機能を拡張するのに役立ちます。

- クラスターを作成した IAM プリンシパルは、`kubectl` または AWS Management Console を使用して Kubernetes API サーバーを呼び出すことができる唯一のプリンシパルです。他の IAM プリンシパルがクラスターにアクセスできるようにする場合は、それらを追加する必要があります。詳細については、Kubernetes API へのアクセスを許可する および必要なアクセス許可を参照してください。
- サンプルアプリケーションをクラスターにデプロイします。
- 本番用にクラスターをデプロイする前に、クラスターとノードのすべての設定を理解しておくことをお勧めします。Amazon EC2 ノードへの SSH アクセスの有効化など一部の設定は、クラスターの作成時に行う必要があります。
- クラスターのセキュリティを強化するには、サービスアカウントの IAM ロールを使用する Amazon VPC コンテナネットワークインターフェイスプラグインの設定を行ってください。

Amazon EKS クラスター

Amazon EKS クラスターは、主要な 2 つのコンポーネントで構成されています。

- Amazon EKS コントロールプレーン
- コントロールプレーンに登録された Amazon EKS ノード

Amazon EKS コントロールプレーンは、`etcd` や Kubernetes API サーバーなどの Kubernetes ソフトウェアを実行するコントロールプレーンノードで設定されています。コントロールプレーンは AWS によって管理されるアカウントで実行され、Kubernetes API はクラスターに関連付けられた Amazon EKS エンドポイントを介して公開されます。各 Amazon EKS クラスターのコントロールプレーンは、一意かつシングルテナントであり、固有の Amazon EC2 インスタンスセット上で実行されます。

`etcd` ノードおよび関連する Amazon EBS ボリュームによって格納されているすべてのデータに対しては、AWS KMS を使用した暗号化が行われます。クラスターコントロールプレーンは、複数のアベイラビリティーゾーンに渡ってプロビジョニングされ、Elastic Load Balancing Network Load Balancer によって前面に置かれます。また、Amazon EKS は、VPC サブネット内に Elastic Network Interface もプロビジョニングします。これにより、コントロールプレーンインスタンスからノードへの接続が可能になります (例えば、`kubectl exec logs proxy` のデータフローをサポートします)。

> **Important**
>
> Amazon EKS 環境では、アップストリームガイダンスのとおり、`etcd` ストレージは 8 GiB に制限されます。次のコマンドを実行して、現在のデータベースサイズのメトリックを監視できます。クラスターの Kubernetes バージョンが `1.28` 以下の場合は、*`apiserver_storage_size_bytes`* を以下のバージョンに置き換えます。
>
> - Kubernetes バージョン `1.27` および `1.26` — **`apiserver_storage_db_total_size_in_bytes`**
> - Kubernetes バージョン `1.25` 以下 — **`etcd_db_total_size_in_bytes`**
>
> ```
> kubectl get --raw=/metrics | grep "apiserver_storage_size_bytes"
> ```

Amazon EKS ノードは、ユーザーの AWS アカウントで実行されます。クラスターのコントロールプレーンへの接続は、API サーバーエンドポイントや、クラスターのために作成された証明書ファイル経由で行われます。

> **Note**
>
> - 「Amazon EKS ネットワーク」で、Amazon EKS のさまざまなコンポーネントがどのように機能するかについて、ご確認いただけます。
> - 接続されたクラスタについては、Amazon EKS Connectorをご覧ください。

トピック

Amazon EKS クラスターの作成

このトピックでは、使用可能なオプションの概要と、Amazon EKS クラスターの作成時に考慮すべき点を説明します。AWS Outpost でクラスターを作成する必要がある場合は、「AWS Outposts の Amazon EKS のローカルクラスター」を参照してください。Amazon EKS クラスターを初めて作成する場合は、Amazon EKS の使用開始 ガイドのいずれかに従うことをお勧めします。これらのガイドは、使用可能なすべてのオプションを展開することなく、シンプルでデフォルトのクラスターを作成するのに役立ちます。

前提条件

- Amazon EKS の要件 を満たす既存の VPC とサブネット。本番用にクラスターをデプロイする前に、VPC とサブネットの要件を十分に理解しておくことをお勧めします。VPC とサブネットがない場合は、Amazon EKS に用意されている AWS CloudFormation テンプレート を使用して作成できます。
- デバイスまたは AWS CloudShell に、kubectl コマンドラインツールがインストールされていること。バージョンは、ご使用のクラスターの Kubernetes バージョンと同じか、1 つ前のマイナーバージョン以前、あるいはそれより新しいバージョンが使用できます。例えば、クラスターのバージョンが 1.29 である場合、kubectl のバージョン 1.28、1.29、または 1.30 が使用できます。kubectl をインストールまたはアップグレードする方法については、「kubectl のインストールまたは更新」を参照してください。
- ご使用のデバイスまたは AWS CloudShell で、バージョン 2.12.3 以降、または AWS Command Line Interface (AWS CLI) のバージョン 1.27.160 以降がインストールおよび設定されていること。現在のバージョンを確認するには、「**aws --version | cut -d / -f2 | cut -d ' ' -f1**」を参照してください。macOS の yum、apt-get、または Homebrew などのパッケージマネージャは、AWS CLI の最新バージョンより数バージョン遅れることがあります。最新バージョンをインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI のインストール、更新、およびアンインストール」と「aws configure でのクイック設定」を参照してください。AWS CloudShell にインストールされている AWS CLI バージョンは、最新バージョンより数バージョン遅れている可能性もあります。更新するには、「AWS CloudShell ユーザーガイド」の「ホームディレクトリへの AWS CLI のインストール」を参照してください。
- Amazon EKS クラスターを create および describe するための許可を持つ IAM プリンシパル。詳細については、Outpost にローカル Kubernetes クラスターを作成しますおよびすべてのクラスターの一覧表示または説明を参照してください。

Amazon EKS クラスターを作成するには

1. 既にクラスター IAM ロールがある場合、または eksctl を使用してクラスターを作成する場合は、このステップはスキップできます。デフォルトでは、eksctl により、ロールが自動的に作成されます。

   Amazon EKS クラスター IAM ロールを作成するには

   1. IAM 信頼ポリシー用の JSON ファイルを作成するには、次のコマンドを実行します。

      ```
      cat >eks-cluster-role-trust-policy.json <<EOF
      ```

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

2. Amazon EKS クラスターの IAM ロールを作成します。必要であれば、前のステップでファイルを書き込んだコンピュータ上のパスを *eks-cluster-role-trust-policy.json* の前につけます。このコマンドは、前のステップで作成した信頼ポリシーをロールに関連付けます。IAM ロールを作成するには、ロールを作成する IAM プリンシパルに `iam:CreateRole` アクション (許可) を割り当てる必要があります。

   ```
   aws iam create-role --role-name myAmazonEKSClusterRole --assume-role-policy-
   document file://"eks-cluster-role-trust-policy.json"
   ```

3. Amazon EKS 管理のポリシーを割り当てるか、独自のカスタムポリシーを作成できます。カスタムポリシーで使用する必要がある最小限の許可については、「Amazon EKS クラスターの IAM ロール」を参照してください。

   このロールに、Amazon EKS 管理の IAM ポリシー (AmazonEKSClusterPolicy) をアタッチします。IAM ポリシーを IAM プリンシパルにアタッチするには、ポリシーのアタッチを行っているプリンシパルに、次のいずれかの IAM アクション (許可) を割り当てる必要があります: `iam:AttachUserPolicy` または `iam:AttachRolePolicy`。

   ```
   aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
   AmazonEKSClusterPolicy --role-name myAmazonEKSClusterRole
   ```

2. Amazon EKS クラスターを作成します。

   `eksctl`、AWS Management Console、または AWS CLI を使用してクラスターを作成できます。

eksctl

前提条件

デバイスまたは AWS CloudShell にインストールされている `eksctl` コマンドラインツールのバージョン `0.183.0` 以降。`eksctl` をインストールまたはアップグレードするには、`eksctl` ドキュメントの「インストール」を参照してください。

クラスターを作成するには

デフォルトの AWS リージョン に、Amazon EKS デフォルトの Kubernetes バージョンを使用して、Amazon EKS `IPv4` クラスターを作成します。コマンドを実行する前に、次の置き換えを行います。

- *`region-code`* を、クラスターを作成する AWS リージョン に置き換えます。
- *`my-cluster`* をクラスターの名前に置き換えます。この名前には、英数字 (大文字と小文字が区別されます) とハイフンのみを使用できます。先頭の文字は英数字である必要があります。また、100 文字より長くすることはできません。名前は、クラスターを作成する AWS リージョン および AWS アカウント 内で一意である必要があります。
- *`1.29`* を Amazon EKS がサポートする任意のバージョンに置き換えます。

  > **Note**
  >
  > この時点で 1.30 クラスターをデプロイするには、AWS Management Console または AWS CLI を使用する必要があります。

- 要件を満たすように `vpc-private-subnets` の値を変更します。さらに ID を追加することもできます。少なくとも 2 つのサブネット ID を指定する必要があります。パブリックサブネットを指定する場合は、`--vpc-private-subnets` を `--vpc-public-subnets` に変更できます。パブリックサブネットには、インターネットゲートウェイへのルートに関連付けられたルートテーブルがありますが、プライベートサブネットには関連付けられたルートテーブルがありません。可能な限り、プライベートサブネットを使用することをお勧めします。

  選択するサブネットは Amazon EKS サブネットの要件を満たす必要があります。サブネットを選択する前に、Amazon EKS VPC およびサブネットの要件と考慮事項をすべて理解しておくことをお勧めします。

```
eksctl create cluster --name my-cluster --region region-code --version 1.29 --
vpc-private-subnets subnet-ExampleID1,subnet-ExampleID2 --without-nodegroup
```

クラスターのプロビジョニングには数分かかります。クラスターの作成中は、数行の出力が表示されます。出力の最後の行は、次のサンプル行のようになります。

```
[#]  EKS cluster "my-cluster" in "region-code" region is ready
```

> **Tip**
>
> eksctl を使用してクラスターを作成するときに指定できるほとんどのオプションを表示するには、**eksctl create cluster --help** コマンドを使用します。使用可能なオプションをすべて表示するには、config ファイルを使用します。詳細については、「eksctl ドキュメント」の「Using config files」(設定ファイルの使用) と「設定ファイルのスキーマ」を参照してください。設定ファイルの例は、GitHub で見つけることができます。

オプション設定

必要に応じて前のコマンドに追加する必要があるオプションの設定を次に示します。これらのオプションは、クラスターの作成時にのみ有効にすることができ、作成後は有効にできません。これらのオプションを指定する必要がある場合は、前のコマンドを使用するのではなく、eksctl config ファイルを使用してクラスターを作成し、設定を指定する必要があります。

- Amazon EKS が作成するネットワークインターフェイスに割り当てる 1 つまたは複数のセキュリティグループを指定する場合は、securityGroup オプションを指定します。

  セキュリティグループを選択するかどうかにかかわらず、Amazon EKS はクラスターと VPC 間の通信を可能にするセキュリティグループを作成します。Amazon EKS は、このセキュリティグループおよびユーザーが選択したセキュリティグループを、作成するネットワークインターフェイスに関連付けます。Amazon EKS が作成するクラスターセキュリティグループの詳細については、「the section called “セキュリティグループの要件”」を参照してください。Amazon EKS が作成するクラスターセキュリティグループのルールを変更できます。

- Kubernetes がサービス IP アドレスを割り当てる IPv4 Classless Inter-Domain Routing (CIDR) ブロックを指定する場合は、`serviceIPv4CIDR` オプションを指定します。

  独自の範囲を指定すると、Kubernetes サービスと VPC にピアリングまたは接続されたその他のネットワークとの間の競合を防ぐことができます。CIDR 表記で範囲を入力します。例: `10.2.0.0/16`。

  この　CIDR ブロックでは、以下の要件を満たす必要があります。

  - `10.0.0.0/8`、`172.16.0.0/12`、または `192.168.0.0/16` のいずれかの範囲内にある。
  - 最小サイズが `/24`、最大サイズが `/12`。
  - Amazon EKS リソースの VPC の範囲と重複しない。

  このオプションを指定できるのは、IPv4 アドレスファミリーを使用してクラスターを作成するときのみです。これを指定しない場合、Kubernetes は、`10.100.0.0/16` または `172.20.0.0/16` のいずれかの CIDR ブロックからサービス IP アドレスを割り当てます。

- クラスターを作成していて、そのクラスターで IPv4 アドレスではなく IPv6 アドレスを Pods とサービスに割り当てるようにする場合は、`ipFamily` オプションを指定します。

  Kubernetes は、デフォルトで IPv4 アドレスを Pods とサービスに割り当てます。IPv6 ファミリーの使用を決定する前に、the section called “VPC の要件と考慮事項”、the section called “サブネットの要件と考慮事項”、the section called “セキュリティグループの要件”、および the section called “IPv6” のトピックの考慮事項と要件をすべて理解していることを確認します。IPv6 ファミリーを選択すると、Kubernetes が IPv6 サービスアドレスを割り当てる範囲を IPv4 ファミリーで指定できるようには指定できません。Kubernetes は特定のローカルアドレス範囲 (`fc00::/7`) からサービスアドレスを割り当てます。

AWS Management Console

クラスターを作成するには

1. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。

2. [クラスターを追加]、[作成] の順にクリックします。
3. [クラスターの設定] ページで、次のフィールドに入力します。
   - [名前] - クラスターの名前。この名前には、英数字 (大文字と小文字が区別されます)、ハイフン、下線のみを使用できます。先頭の文字は英数字である必要があります。また、100 文字より長くすることはできません。名前は、クラスターを作成する AWS リージョンおよび AWS アカウント内で一意である必要があります。
   - Kubernetes バージョン - クラスターに使用する Kubernetes のバージョン。以前のバージョンが必要でない限り、最新バージョンを選択することをお勧めします。
   - [クラスターサービスロール] - ユーザーに代わって AWS リソースを管理することを Kubernetes コントロールプレーンに許可するために作成した Amazon EKS クラスター IAM ロールを選択します。
   - [シークレット暗号化] - (オプション) KMS キーを使用して Kubernetes シークレットのシークレット暗号化を有効にするよう選択します。クラスターを作成した後で、これを有効にすることもできます。この機能を有効にする前に、既存のクラスター上でシークレット暗号化を有効にする の情報をよく理解していることを確認してください。
   - [タグ] - (オプション) クラスターにタグを追加します。詳細については、「Amazon EKS リソースのタグ付け」を参照してください。

     このページを読み終えたら、[次へ] を選択します。
4. [ネットワーキングの指定] ページで、次のフィールドの値を選択します。
   - [VPC] – Amazon EKS VPC 要件を満たす既存の VPC を選択し、そこでクラスターを作成します。VPC を選択する前に、Amazon EKS VPC およびサブネットの要件と考慮事項 の要件と考慮事項をすべて理解しておくことをお勧めします。クラスターの作成後は、使用する VPC を変更できません。VPC が表示されていない場合は、まず作成する必要があります。詳細については、「Amazon EKS クラスター VPC の作成」を参照してください。
   - [サブネット] - デフォルトで、前のフィールドで指定した VPC 内の利用可能なすべてのサブネットがあらかじめ選択されています。少なくとも 2 つ選択する必要があります。

     選択するサブネットは Amazon EKS サブネットの要件を満たす必要があります。サブネットを選択する前に、Amazon EKS VPC およびサブネットの要件と考慮事項をすべて理解しておくことをお勧めします。

     [セキュリティグループ] - (オプション) Amazon EKS が作成するネットワークインターフェイスに関連付ける 1 つまたは複数のセキュリティグループを指定します。

セキュリティグループを選択するかどうかにかかわらず、Amazon EKS はクラスターと VPC 間の通信を可能にするセキュリティグループを作成します。Amazon EKS は、このセキュリティグループおよびユーザーが選択したセキュリティグループを、作成するネットワークインターフェイスに関連付けます。Amazon EKS が作成するクラスターセキュリティグループの詳細については、「the section called “セキュリティグループの要件”」を参照してください。Amazon EKS が作成するクラスターセキュリティグループのルールを変更できます。

- クラスターの IP アドレスファミリーの選択 – IPv4 と IPv6 のどちらかを選択できます。

  Kubernetes は、デフォルトで IPv4 アドレスを Pods とサービスに割り当てます。IPv6 ファミリーの使用を決定する前に、the section called “VPC の要件と考慮事項”、the section called “サブネットの要件と考慮事項”、the section called “セキュリティグループの要件”、および the section called “IPv6” のトピックの考慮事項と要件をすべて理解していることを確認します。IPv6 ファミリーを選択すると、Kubernetes が IPv6 サービスアドレスを割り当てる範囲を IPv4 ファミリーで指定できるようには指定できません。Kubernetes は特定のローカルアドレス範囲 (fc00::/7) からサービスアドレスを割り当てます。

- (オプション) [Kubernetes サービス IP アドレスの範囲を設定する] を選択し、[サービスの **IPv4** 範囲] を指定します。

  独自の範囲を指定すると、Kubernetes サービスと VPC にピアリングまたは接続されたその他のネットワークとの間の競合を防ぐことができます。CIDR 表記で範囲を入力します。例: 10.2.0.0/16。

  この　CIDR ブロックでは、以下の要件を満たす必要があります。

  - 10.0.0.0/8、172.16.0.0/12、または 192.168.0.0/16 のいずれかの範囲内にある。
  - 最小サイズが /24、最大サイズが /12。
  - Amazon EKS リソースの VPC の範囲と重複しない。

  このオプションを指定できるのは、IPv4 アドレスファミリーを使用してクラスターを作成するときのみです。これを指定しない場合、Kubernetes は、10.100.0.0/16 または 172.20.0.0/16 のいずれかの CIDR ブロックからサービス IP アドレスを割り当てます。

- [クラスターエンドポイントのアクセス] で、オプションを選択します。クラスターを作成した後で、このオプションを変更できます。デフォルト以外のオプションを選択する前に、オプションとその意味を理解しておいてください。詳細については、「Amazon EKS クラスターエンドポイントアクセスコントロール」を参照してください。

  このページを読み終えたら、[次へ] を選択します。

5. (オプション) [オブザーバビリティの設定] ページで、有効にする [メトリクス] と [コントロールプレーンのロギング] オプションを選択します。デフォルトでは、それぞれのログタイプは無効化されています。
   - Prometheus メトリクスの詳細については、「クラスターを作成するときは、Prometheus メトリクスを有効にしてください。」を参照してください。
   - [トラフィック設定] のオプションの詳細については、「Amazon EKS コントロールプレーンのログ記録」を参照してください。

   このページを読み終えたら、[次へ] を選択します。
6. [アドオンの選択] ページで、クラスターに追加するアドオンを選択します。[Amazon EKS アドオン] と [AWS Marketplace アドオン] は必要な数だけ選択できます。インストールする [AWS Marketplace アドオン] が一覧にない場合は、検索ボックスにテキストを入力して、利用可能な [AWS Marketplace アドオン] を検索できます。[カテゴリ]、[ベンダー]、または [価格モデル] で検索し、検索結果からアドオンを選択することもできます。このページを読み終えたら、[次へ] を選択します。
7. [選択したアドオン設定の構成] ページで、インストールするバージョンを選択し、[次へ] を選択します。クラスターを作成した後は、いつでも新しいバージョンに更新できます。クラスターの作成後に、各アドオンの設定を更新できます。アドオンの設定の詳細については、「アドオンの更新」を参照してください。このページを読み終えたら、[次へ] を選択します。
8. [確認と作成] ページで、前のページで入力または選択した情報を確認します。変更する必要がある場合は、[編集] を選択します。そのままでよければ、[作成] を選択します。クラスターがプロビジョニングされている間、[ステータス] フィールドに [作成中] と表示されます。

> **Note**
>
> リクエストで指定したアベイラビリティーゾーンのいずれかに、Amazon EKS クラスターの作成に十分な容量がない場合には、エラーが表示されることがあります。このエラー出力には、新しいクラスターをサポートできるアベイラビリ

ティーゾーンが表示されます。アカウント向けにサポートされているアベイラビリティーゾーンにある 2 つ以上のサブネットを使用して、クラスターを作成します。詳細については、「」を参照してください容量不足

クラスターのプロビジョニングには数分かかります。

AWS CLI

クラスターを作成するには

1. 下記のコマンドを使用して、クラスターを作成します。コマンドを実行する前に、次の置き換えを行います。
   - *region-code* を、クラスターを作成する AWS リージョン に置き換えます。
   - *my-cluster* をクラスターの名前に置き換えます。この名前には、英数字 (大文字と小文字が区別されます)、ハイフン、下線のみを使用できます。先頭の文字は英数字である必要があります。また、100 文字より長くすることはできません。名前は、クラスターを作成する AWS リージョン および AWS アカウント 内で一意である必要があります。
   - *1.30* を Amazon EKS がサポートする任意のバージョンに置き換えます。
   - *111122223333* をアカウント ID に置き換え、*myAmazonEKSClusterRole* をクラスター IAM ロールに置き換えます。
   - subnetIds の値を独自の値に置き換えます。さらに ID を追加することもできます。少なくとも 2 つのサブネット ID を指定する必要があります。

     選択するサブネットは Amazon EKS サブネットの要件を満たす必要があります。サブネットを選択する前に、Amazon EKS VPC およびサブネットの要件と考慮事項をすべて理解しておくことをお勧めします。
   - セキュリティグループ ID を指定しない場合は、コマンドから ,securityGroupIds=sg-*ExampleID1* を削除します。1 つまたは複数のセキュリティグループ ID を指定する場合は、securityGroupIds の値を独自の値に置き換えます。さらに ID を追加することもできます。

     セキュリティグループを選択するかどうかにかかわらず、Amazon EKS はクラスターと VPC 間の通信を可能にするセキュリティグループを作成します。Amazon EKS は、このセキュリティグループおよびユーザーが選択したセキュリティグループを、作成する

ネットワークインターフェイスに関連付けます。Amazon EKS が作成するクラスターセキュリティグループの詳細については、「the section called “セキュリティグループの要件”」を参照してください。Amazon EKS が作成するクラスターセキュリティグループのルールを変更できます。

```
aws eks create-cluster --region region-code --name my-cluster --kubernetes-
version 1.30 \
   --role-arn arn:aws:iam::111122223333:role/myAmazonEKSClusterRole \
   --resources-vpc-config
 subnetIds=subnet-ExampleID1,subnet-ExampleID2,securityGroupIds=sg-ExampleID1
```

> **Note**
>
> リクエストで指定したアベイラビリティーゾーンのいずれかに、Amazon EKS クラスターの作成に十分な容量がない場合には、エラーが表示されることがあります。このエラー出力には、新しいクラスターをサポートできるアベイラビリティーゾーンが表示されます。アカウント向けにサポートされているアベイラビリティーゾーンにある 2 つ以上のサブネットを使用して、クラスターを作成します。詳細については、「容量不足」を参照してください。

オプション設定

必要に応じて前のコマンドに追加する必要があるオプションの設定を次に示します。これらのオプションは、クラスターの作成時にのみ有効にすることができ、作成後は有効にできません。

- Kubernetes がサービス IP アドレスを割り当てる IPv4 Classless Inter-Domain Routing (CIDR) ブロックを指定する場合は、次のコマンドに **`--kubernetes-network-config serviceIpv4Cidr=`*`CIDR block`*** を追加して指定する必要があります。

  独自の範囲を指定すると、Kubernetes サービスと VPC にピアリングまたは接続されたその他のネットワークとの間の競合を防ぐことができます。CIDR 表記で範囲を入力します。例: `10.2.0.0/16`。

  この　CIDR ブロックでは、以下の要件を満たす必要があります。

  - `10.0.0.0/8`、`172.16.0.0/12`、または `192.168.0.0/16` のいずれかの範囲内にある。
  - 最小サイズが `/24`、最大サイズが `/12`。

- Amazon EKS リソースの VPC の範囲と重複しない。

このオプションを指定できるのは、IPv4 アドレスファミリーを使用してクラスターを作成するときのみです。これを指定しない場合、Kubernetes は、10.100.0.0/16 または 172.20.0.0/16 のいずれかの CIDR ブロックからサービス IP アドレスを割り当てます。

- クラスターを作成していて、そのクラスターで IPv4 アドレスではなく IPv6 アドレスを Pods とサービスに割り当てるようにする場合は、次のコマンドに **--kubernetes-network-config ipFamily=ipv6** を追加します。

Kubernetes は、デフォルトで IPv4 アドレスを Pods とサービスに割り当てます。IPv6 ファミリーの使用を決定する前に、the section called “VPC の要件と考慮事項”、the section called “サブネットの要件と考慮事項”、the section called “セキュリティグループの要件”、および the section called “IPv6” のトピックの考慮事項と要件をすべて理解していることを確認します。IPv6 ファミリーを選択すると、Kubernetes が IPv6 サービスアドレスを割り当てる範囲を IPv4 ファミリーで指定できるようには指定できません。Kubernetes は特定のローカルアドレス範囲 (fc00::/7) からサービスアドレスを割り当てます。

2. クラスターがプロビジョニングされるまでに数分かかります。クラスターのステータスのクエリを実行するには、次のコマンドを使用します。

```
aws eks describe-cluster --region region-code --name my-cluster --query
 "cluster.status"
```

出力が「ACTIVE」を返すまで、次のステップに進まないでください。

3. eksctl を使用してクラスターを作成した場合、このステップはスキップできます。eksctl によってこのステップはすでに完了しているからです。新しいコンテキストを kubectl config ファイルに追加して、kubectl がクラスターと通信できるようにします。ファイルを作成および更新する方法の詳細については、「Amazon EKS クラスターの kubeconfig ファイルを作成または更新する」を参照してください。

```
aws eks update-kubeconfig --region region-code --name my-cluster
```

出力例は次のとおりです。

```
Added new context arn:aws:eks:region-code:111122223333:cluster/my-cluster to /home/
username/.kube/config
```

4. 次のコマンドを実行して、クラスターとの通信を確認します。

```
kubectl get svc
```

出力例は次のとおりです。

```
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   28h
```

5. (推奨) Amazon EKS アドオンを使用するか、固有の AWS Identity and Access Management (IAM) アクセス許可を個々の Kubernetes ワークロードに付与できるようにするには、クラスター用に IAM OpenID Connect (OIDC) プロバイダーを作成します。クラスター用に IAM OIDC プロバイダーを作成する必要があるのは 1 回だけです。Amazon EKS アドオンの詳細については、「Amazon EKS アドオン」を参照してください。ワークロードに特定の IAM アクセス許可を割り当てる方法については、「サービスアカウントの IAM ロール」を参照してください。

6. (推奨) Amazon EC2 ノードをクラスターにデプロイする前に Amazon VPC CNI plugin for Kubernetes プラグイン用にクラスターを設定します。デフォルトでは、プラグインはクラスターとともにインストールされています。Amazon EC2 ノードをクラスターに追加すると、プラグインは追加する各 Amazon EC2 ノードに自動的にデプロイされます。プラグインでは、次の IAM ポリシーのいずれかを IAM ロールにアタッチする必要があります。

   AmazonEKS_CNI_Policy マネージド IAM ポリシー

   クラスターが IPv4 ファミリーを使用している場合

   ユーザーが作成する IAM ポリシー

   クラスターが IPv6 ファミリーを使用している場合

   ポリシーをアタッチする IAM ロールは、ノード IAM ロール、またはプラグインにのみ使用される専用ロールです。このロールにポリシーをアタッチすることをお勧めします。ロールの作成の詳細については、「サービスアカウントの IAM ロールを使用する Amazon VPC CNI plugin for Kubernetes の設定」または「Amazon EKS ノードの IAM ロール」を参照してください。

7. AWS Management Console を使用してクラスターをデプロイした場合、このステップはスキップできます。AWS Management Console では、デフォルトで、Amazon VPC CNI plugin for Kubernetes、CoreDNS、および kube-proxy Amazon EKS アドオンがデプロイされます。

   eksctl または AWS CLI のいずれかを使用してクラスターをデプロイする場合、Amazon VPC CNI plugin for Kubernetes、CoreDNS、および kube-proxy セルフマネージド型アドオンがデプロイされます。クラスターとともに Amazon EKS アドオンにデプロイされる Amazon VPC CNI plugin for Kubernetes、CoreDNS、および kube-proxy セルフマネージド型アドオンを移行できます。詳細については、「Amazon EKS アドオン」を参照してください。
8. (オプション) まだ作成していない場合は、クラスターの Prometheus メトリクスを有効にできます。詳細については、「Amazon Managed Service for Prometheus ユーザーガイド」の「スクレイパーの作成」を参照してください。
9. Prometheus メトリクスを有効にした場合は、aws-auth ConfigMap スクレイパーにクラスター内のアクセス権限を付与するように設定する必要があります。詳細については、「Amazon Managed Service for Prometheus ユーザーガイド」の「Amazon EKS クラスターの設定」を参照してください。
10. Amazon EBS ボリュームを使用するクラスターにワークロードをデプロイする予定で、1.23 以降のクラスターを作成した場合、ワークロードをデプロイする前に、Amazon EBS CSI ドライバー をクラスターにインストールする必要があります。

推奨される次の手順は以下の通りです。

- クラスターを作成した IAM プリンシパルは、クラスターにアクセスできる唯一のプリンシパルです。他の IAM プリンシパルに許可を付与して、クラスターにアクセスできるようにします。
- クラスターを作成した IAM プリンシパルが、前提条件で参照されている最低限の IAM 許可しか持たない場合、そのプリンシパルに Amazon EKS 許可を追加することができます。Amazon EKS 許可を IAM プリンシパルに付与する方法については、「Amazon EKS の Identity and Access Management」を参照してください。
- クラスターを作成した IAM プリンシパル、またはその他のプリンシパルに Amazon EKS コンソールで Kubernetes リソースを表示させる場合は、必要なアクセス許可 をエンティティに付与します。
- ノードと IAM プリンシパルが VPC 内からクラスターにアクセスできるようにする場合、クラスターのプライベートエンドポイントを有効にします。デフォルトでは、パブリックエンドポイントは有効です。プライベートエンドポイントを有効にした後、必要に応じてパブリックエンドポイントを無効にできます。詳細については、「Amazon EKS クラスターエンドポイントアクセスコントロール」を参照してください。

- クラスターのシークレット暗号化を有効にする。
- クラスターのログ記録を設定する。
- クラスターにノードを追加する。

クラスターのインサイト

Amazon EKS クラスターインサイトは、Amazon EKS と Kubernetes のベストプラクティスに従うのに役立つ推奨事項を提供します。すべての Amazon EKS クラスターは、Amazon EKS がキュレートしたインサイトのリストと照合して、自動的かつ定期的にチェックされます。これらのインサイトチェックは Amazon EKS によって完全に管理され、どのような結果にも対処するための推奨事項を提示します。

クラスターインサイトの推奨されている使用方法:

- クラスターの Kubernetes バージョンを更新する前に、EKS コンソールでクラスターインサイトを確認します。
- クラスターに問題が見つかった場合は、確認して適切な修正を行います。問題には、Amazon EKS および Kubernetes へのリンクも含まれます。
- 問題を修正したら、クラスターインサイトが更新されるまで待機します。すべての問題が解決したら、クラスターを更新します。

現在、Amazon EKS は Kubernetes バージョンアップグレードの準備状況に関するインサイトのみを返します。

アップグレードインサイトは、Kubernetes クラスターのアップグレードに影響する可能性のある問題を特定します。これにより、管理者がアップグレードの準備に費やす労力が最小限に抑えられ、新しい Kubernetes バージョンでのアプリケーションの信頼性が高まります。クラスターは、Amazon EKS によって Kubernetes バージョンアップグレードに影響する可能性のある問題のリストと照合して自動的にスキャンされます。Amazon EKS は、各 Kubernetes バージョンリリースで行われた変更のレビューに基づいて、インサイトチェックのリストを頻繁に更新します。

Amazon EKS のアップグレードインサイトにより、新しいバージョンのテストと検証のプロセスがスピードアップします。また、クラスター管理者やアプリケーション開発者は、懸念事項を浮き彫りにし、修正に関するアドバイスを提供することで、最新の Kubernetes 機能を活用できます。実行されたインサイトチェックのリストと、Amazon EKS によって特定された関連する問題を確認するには、Amazon EKS `ListInsights` API オペレーションを呼び出すか、Amazon EKS コンソールを調べることができます。

クラスターのインサイトは定期的に更新されます。クラスターインサイトは、手動では更新できません。クラスターの問題を修正すると、クラスターインサイトが更新されるまでしばらく時間がかかります。修正が完了したかどうかを判断するには、変更を実施した時刻とクラスターインサイトの「最終更新時刻」とを比較します。

クラスターインサイトを表示する (コンソール)

Amazon EKS クラスターのインサイトを確認するには

a. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。

b. クラスターのリストから、インサイトを確認する Amazon EKS クラスターの名前を選択します。

c. [アップグレードインサイト] タブを選択します。

d. 「アップグレードインサイト」ページには、以下のフィールドが表示されます。

- 名前 — Amazon EKS がクラスターに対して実行したチェック。
- インサイトステータス — ステータスが「エラー」のインサイトは、通常、影響を受ける Kubernetes バージョンが現在のクラスターバージョンの N+1 であることを意味し、ステータスが「警告」とは、インサイトが将来の Kubernetes バージョン N+2 以上に適用されることを意味します。ステータスが「合格」のインサイトは、Amazon EKS がこのインサイトチェックに関連する問題をクラスター内で発見しなかったことを意味します。インサイトステータスが「不明」の場合は、Amazon EKS はクラスターがこのインサイトチェックの影響を受けているかどうかを判断できないことを意味します。
- バージョン — インサイトにより問題の可能性がチェックされた Kubernetes バージョン。
- 最終更新時間 (UTC-5:00) — このクラスターのインサイトのステータスが最後に更新された時刻。
- 最終移行時間 (UTC-5:00) — このインサイトのステータスが最後に変更された時刻。
- 説明 — アラートと修復のための推奨アクションを含むインサイトチェックの情報。

クラスターインサイトを表示する (AWS CLI)

Amazon EKS クラスターのインサイトを確認するには

a. インサイトを確認したいクラスターを決定します。次のコマンドは、指定されたクラスターのインサイトをリスト表示しています。必要に応じてコマンドに次の変更を加え、変更したコマンドを実行します。

- *region-code* をAWS リージョン のコードに置き換えます。

- *my-cluster* を自分のクラスター名に置き換えます。

```
aws eks list-insights --region region-code --cluster-name my-cluster
```

出力例は次のとおりです。

```
{
    "insights": [
        {
            "category": "UPGRADE_READINESS",
            "name": "Deprecated APIs removed in Kubernetes v1.29",
            "insightStatus": {
                "status": "PASSING",
                "reason": "No deprecated API usage detected within the last 30 days."
            },
            "kubernetesVersion": "1.29",
            "lastTransitionTime": 1698774710.0,
            "lastRefreshTime": 1700157422.0,
            "id": "123e4567-e89b-42d3-a456-579642341238",
            "description": "Checks for usage of deprecated APIs that are scheduled
 for removal in Kubernetes v1.29. Upgrading your cluster before migrating to the
 updated APIs supported by v1.29 could cause application impact."
        }
    ]
}
```

b. インサイトの詳細を表示するには、次のコマンドを実行します。必要に応じてコマンドに次の変更を加え、変更したコマンドを実行します。

- *region-code* をAWS リージョン のコードに置き換えます。
- *123e4567-e89b-42d3-a456-579642341238* をクラスターのインサイトの一覧から取得したインサイト ID に置き換えます。
- *my-cluster* を自分のクラスター名に置き換えます。

```
aws eks describe-insight --region region-code --id 123e4567-e89b-42d3-
a456-579642341238 --cluster-name my-cluster
```

出力例は次のとおりです。

```
{
    "insight": {
```

```
        "category": "UPGRADE_READINESS",
        "additionalInfo": {
            "EKS update cluster documentation": "https://docs.aws.amazon.com/eks/
latest/userguide/update-cluster.html",
            "Kubernetes v1.29 deprecation guide": "https://kubernetes.io/docs/
reference/using-api/deprecation-guide/#v1-29"
        },
        "name": "Deprecated APIs removed in Kubernetes v1.29",
        "insightStatus": {
            "status": "PASSING",
            "reason": "No deprecated API usage detected within the last 30 days."
        },
        "kubernetesVersion": "1.29",
        "recommendation": "Update manifests and API clients to use newer Kubernetes
 APIs if applicable before upgrading to Kubernetes v1.29.",
        "lastTransitionTime": 1698774710.0,
        "lastRefreshTime": 1700157422.0,
        "categorySpecificSummary": {
            "deprecationDetails": [
                {
                    "usage": "/apis/flowcontrol.apiserver.k8s.io/v1beta2/
flowschemas",
                    "replacedWith": "/apis/flowcontrol.apiserver.k8s.io/v1beta3/
flowschemas",
                    "stopServingVersion": "1.29",
                    "clientStats": [],
                    "startServingReplacementVersion": "1.26"
                },
                {
                    "usage": "/apis/flowcontrol.apiserver.k8s.io/v1beta2/
prioritylevelconfigurations",
                    "replacedWith": "/apis/flowcontrol.apiserver.k8s.io/v1beta3/
prioritylevelconfigurations",
                    "stopServingVersion": "1.29",
                    "clientStats": [],
                    "startServingReplacementVersion": "1.26"
                }
            ]
        },
        "id": "f6a11fe4-77f7-48c6-8326-9a13f022ecb3",
        "resources": [],
        "description": "Checks for usage of deprecated APIs that are scheduled for
 removal in Kubernetes v1.29. Upgrading your cluster before migrating to the updated
 APIs supported by v1.29 could cause application impact."
```

```
    }
}
```

Amazon EKS クラスターの Kubernetes バージョンの更新

Amazon EKS で利用可能な新しい Kubernetes バージョンがある場合には、Amazon EKS クラスターを最新バージョンに更新できます。

> **Important**
>
> クラスターをアップグレードすると、以前のバージョンにダウングレードすることはできません。新しい Kubernetes バージョンに更新する前に、「Amazon EKS Kubernetes のバージョン」で情報を確認し、さらに本トピック内の更新手順でも確認することをお勧めします。

新しい Kubernetes バージョンでは、大幅な変更が加えられている場合があります。このため、本稼働用クラスターで更新を行う前に、新しいバージョンの Kubernetes に対するアプリケーションの動作をテストしておくことをお勧めします。この際は、継続的な統合ワークフローを構築し、新しい Kubernetes バージョンに移行する前にアプリケーションの動作をテストします。

更新プロセスに含まれている Amazon EKS が、更新された Kubernetes バージョンを使用しながら新しい API サーバーノードを起動することで、既存のバージョンを置き換えます。Amazon EKS は、これらの新しいノードで、ネットワークトラフィックの標準インフラストラクチャと準備状況に関するヘルスチェックを実行し、想定どおりに動作していることを確認します。ただし、クラスターのアップグレードを開始すると、一時停止または停止することはできません。これらのヘルスチェックのいずれかが失敗すると、Amazon EKS はインフラストラクチャのデプロイを元に戻します。クラスターは前の Kubernetes バージョンのままになります。この際も、実行中のアプリケーションは影響を受けません。また、クラスターが不確定または回復不可能な状態のままになることもありません。Amazon EKS は定期的にすべてのマネージド型クラスターをバックアップします。さらに、必要に応じてクラスターを復元するメカニズムも存在します。Kubernetes インフラストラクチャの管理プロセスは、継続的に評価、改善されています。

クラスターをアップグレードする際、Amazon EKS には、クラスターの作成時に指定したサブネット内に、最大で 5 つの使用可能な IP アドレスが必要となります。Amazon EKS は、指定したサブネットのいずれかに、新しいクラスターの Elastic Network Interface (ネットワークインターフェイス) を作成します。新しいネットワークインターフェイスは、既存のネットワークインターフェイス

があるのとは b サブネット内に作成される場合があります。ですので、クラスター作成時に指定したサブネットのいずれでも必要なクラスターとの通信が許可されるよう、セキュリティグループルールを設定します。クラスターの作成時に指定したサブネットのいずれかが存在しない、使用できる十分な IP アドレスがない、または必要なクラスターとの通信を許可するセキュリティグループルールがない場合、更新が失敗する可能性があります。

> **Note**
>
> クラスターの API サーバーエンドポイントに常にアクセスできるように、Amazon EKS では高可用性を備えた Kubernetes コントロールプレーンを提供しており、アップデート実行中に API サーバーインスタンスのローリングアップデートを行います。Kubernetes API サーバーエンドポイントをサポートする API サーバーインスタンスの IP アドレスの変更を考慮するために、API サーバークライアントが再接続を適切に管理しているか確認する必要があります。`kubectl` の最新バージョンと公式にサポートされている Kubernetes クライアントライブラリは、この再接続プロセスを透過的に実行します。

Amazon EKS クラスターに必要な Kubernetes バージョンを更新する

クラスターに必要な Kubernetes のバージョンの更新方法

1. クラスターコントロールプレーンの Kubernetes バージョンと、ノードの Kubernetes バージョンを比較します。

   - クラスターコントロールプレーンの Kubernetes バージョンを取得します。

     ```
     kubectl version
     ```

   - ノードの Kubernetes バージョンを取得します。このコマンドでは、セルフマネージド型およびマネージド型の Amazon EC2 および Fargate ノードがすべて表示されます。各 Fargate Pod は、独自のノードとしてリストされます。

     ```
     kubectl get nodes
     ```

   コントロールプレーンを新しい Kubernetes バージョンに更新する前に、クラスター内のマネージド型ノードと Fargate ノードの双方の Kubernetes マイナーバージョンが、コントロールプレーンのバージョンと同じであることを確認してください。例えば、コントロールプレーンがバージョン `1.29` を実行し、かつノードの 1 つがバージョン `1.28` を実行している場合、コン

トロールプレーンを 1.30 に更新する前に、ノードをバージョン 1.29 に更新する必要があります。また、コントロールプレーンを更新する前に、セルフマネージド型ノードをコントロールプレーンと同じバージョンに更新することをお勧めします。詳細については、マネージド型ノードグループの更新およびセルフマネージド型ノードの更新を参照してください。Fargate ノードのマイナーバージョンがコントロールプレーンのバージョンよりも古い場合、まずノードの示す Pod を削除します。次に、コントロールプレーンを更新します。残りの Pods は、再デプロイ後に新しいバージョンに更新されます。

2. 最初にクラスターにデプロイした Kubernetes バージョンが Kubernetes 1.25 以降だった場合、このステップをスキップしてください。

   デフォルトでは、Amazon EKS クラスターで Pod セキュリティポリシーのアドミッションコントローラーが有効化されています。クラスターを更新する前に、適切な Pod セキュリティポリシーが指定されていることを確認してください。これは、潜在的なセキュリティの問題を回避するためのものです。**kubectl get psp eks.privileged** コマンドを使用して、デフォルトのポリシーを確認できます。

   ```
   kubectl get psp eks.privileged
   ```

   以下のエラーが表示された場合は、先に進む前に「Amazon EKS での デフォルトの Pod セキュリティポリシー」を参照してください。

   ```
   Error from server (NotFound): podsecuritypolicies.extensions "eks.privileged" not
    found
   ```

3. 最初にクラスターにデプロイした Kubernetes バージョンが Kubernetes 1.18 以降だった場合、このステップをスキップしてください。

   CoreDNS マニフェストから、廃止された単語を削除する必要がある場合があります。

   a. CoreDNS マニフェストに、upstream というワードのみの行があるかどうかを確認します。

      ```
      kubectl get configmap coredns -n kube-system -o jsonpath='{$.data.Corefile}' |
       grep upstream
      ```

      出力が返されない場合は、マニフェストにこの行は含まれていません。この場合は、次のステップに進みます。upstream というワードが返された場合は、その行を削除します。

b. ConfigMap ファイルで、ファイルの先頭付近にある、upstream のみを含む行を削除します。このファイル内の他の部分は変更しないでください。行を削除したら、変更を保存します。

```
kubectl edit configmap coredns -n kube-system -o yaml
```

4. eksctl、AWS Management Console、またはAWS CLI を使用してクラスターを更新します。

> **Important**
>
> - バージョン 1.23 に更新してクラスターで Amazon EBS ボリュームを使用する場合は、ワークロードの中断を避けるために、クラスターをバージョン 1.23 に更新する前にクラスターに Amazon EBS CSI ドライバーをインストールする必要があります。詳細については、Kubernetes 1.23およびAmazon EBS CSI ドライバーを参照してください。
> - Kubernetes 1.24 およびそれ以降では、デフォルトのコンテナランタイムとして containerd が使用されます。containerd ランタイムに切り替える予定で、すでに Fluentd が Container Insights 用に構成されている場合は、クラスターを更新する前に Fluentd を Fluent Bit に移行する必要があります。Fluentd パーサーは JSON 形式のログメッセージのみを解析するように構成されています。dockerd とは異なり、containerd コンテナランタイムには JSON 形式ではないログメッセージがあります。Fluent Bit に移行しないと、構成された Fluentd's パーサーの一部が Fluentd コンテナー内で大量のエラーを生成します。移行の詳細については、「CloudWatch Logs へログを送信する DaemonSet として Fluent Bit を設定する」を参照してください。
> - Amazon EKS は可用性の高いコントロールプレーンを実行しているため、一回に更新できるのはマイナーバージョン 1 つのみです。この要件の詳細については、「Kubernetes のバージョンおよびバージョンスキューのサポートポリシー」を参照してください。現在のクラスターバージョンが 1.28 であり、1.30 に更新することを仮定します。最初にバージョン 1.28 クラスターをバージョン 1.29 に更新し、次にバージョン 1.29 クラスターをバージョン 1.30 に更新する必要があります。
> - ノード上の Kubernetes kube-apiserver と kubelet 間のバージョンスキューを確認してください。
>   - Kubernetes バージョン 1.28 以降、kubelet は kube-apiserver より最大 3 マイナーバージョン古い場合があります。「Kubernetes アップストリームバージョンのスキューポリシー」を参照してください。

  - マネージド型ノードと Fargate ノードの kubelet が Kubernetes バージョン 1.25 以降の場合は、kubelet バージョンを更新せずに最大 3 バージョン先までクラスターを更新できます。例えば、kubelet がバージョン 1.25 である場合、kubelet のバージョンを 1.25 のままにしていても、Amazon EKS クラスターのバージョンは 1.25 から 1.26、1.27、1.28 に更新できます。
  - マネージド型ノードと Fargate ノードの kubelet が Kubernetes バージョン 1.24 以前の場合、これは kube-apiserver より 2 マイナーバージョンまでしか古くできません。つまり、kubelet がバージョン 1.24 以前の場合、クラスターを更新できるのは 2 バージョン先までです。例えば、kubelet がバージョン 1.21 の場合、Amazon EKS クラスターのバージョンを 1.21 から 1.22、1.23 に更新できますが、kubelet が 1.21 のままでは、クラスターを 1.24 に更新できません。
- 更新を開始する前のベストプラクティスとして、ノード上の kubelet がコントロールプレーンと同じ Kubernetes バージョンであることを確認してください。
- クラスターが 1.8.0 より前のバージョンの Amazon VPC CNI plugin for Kubernetes で設定されている場合、クラスターを更新する前に、プラグインを最新バージョンに更新することをお勧めします。プラグインのアップデートについては、「Amazon VPC CNI plugin for Kubernetes Amazon EKS アドオンの使用」を参照してください。
- クラスターをバージョン 1.25 以降に更新しようとしており、クラスターに AWS Load Balancer Controller をデプロイしている場合は、クラスターのバージョンを 1.25 に更新する前に、コントローラーをバージョン 2.4.7 以降に更新してください。詳細については、Kubernetes 1.25 のリリースノートを参照してください。

eksctl

この手順には、eksctl バージョン 0.183.0 以降が必要です。お使いのバージョンは、以下のコマンドを使用して確認できます。

```
eksctl version
```

eksctl のインストールと更新の手順については、eksctl ドキュメントの「インストール」を参照してください。

Amazon EKS コントロールプレーンの Kubernetes バージョンを更新します。*my-cluster* をクラスター名に置き換えます。*1.30* を、Amazon EKS がサポートするクラスターの更新

後のバージョン番号に置き換えてください。サポートされているバージョン番号のリストについては、「Amazon EKS Kubernetes のバージョン」を参照してください。

```
eksctl upgrade cluster --name my-cluster --version 1.30 --approve
```

この更新は完了までに数分かかることがあります。

AWS Management Console

a. https://console.aws.amazon.com/eks/home#/clusters で Amazon EKS コンソールを開きます。
b. 更新する Amazon EKS クラスター名を選択し、[クラスターバージョンの更新] を選択します。
c. [Kubernetes バージョン] で、バージョンを選択してクラスターを更新し、[更新] を選択します。
d. [クラスター名] にクラスター名を入力し、[確認] を選択します。

   この更新は完了までに数分かかることがあります。

AWS CLI

a. 次の AWS CLI コマンドを使用して、Amazon EKS クラスターを更新します。*example values* を自分の値に置き換えます。*1.30* を、Amazon EKS がサポートするクラスターの更新後のバージョン番号に置き換えてください。サポートされているバージョン番号のリストについては、「Amazon EKS Kubernetes のバージョン」を参照してください。

   ```
   aws eks update-cluster-version --region region-code --name my-cluster --
   kubernetes-version 1.30
   ```

   出力例は次のとおりです。

   ```
   {
       "update": {
           "id": "b5f0ba18-9a87-4450-b5a0-825e6e84496f",
           "status": "InProgress",
           "type": "VersionUpdate",
           "params": [
               {
                   "type": "Version",
   ```

```
                "value": "1.30"
            },
            {
                "type": "PlatformVersion",
                "value": "eks.1"
            }
        ],
[...]
        "errors": []
    }
}
```

b. クラスター更新のステータスをモニタリングするには、次のコマンドを使用します。前のコマンドから返されたクラスター名と更新 ID を使用します。Successful ステータスが表示されると、更新は完了です。この更新は完了までに数分かかることがあります。

```
aws eks describe-update --region region-code --name my-cluster --update-
id b5f0ba18-9a87-4450-b5a0-825e6e84496f
```

出力例は次のとおりです。

```
{
    "update": {
        "id": "b5f0ba18-9a87-4450-b5a0-825e6e84496f",
        "status": "Successful",
        "type": "VersionUpdate",
        "params": [
            {
                "type": "Version",
                "value": "1.30"
            },
            {
                "type": "PlatformVersion",
                "value": "eks.1"
            }
        ],
[...]
        "errors": []
    }
}
```

5. クラスターの更新が完了したら、更新したクラスターでの Kubernetes と同じマイナーバージョンに、ノードを更新する必要があります。詳細については、セルフマネージド型ノードの更新およびマネージド型ノードグループの更新を参照してください。Fargate で起動される新しい Pods であれば、クラスターのバージョンと一致する kubelet バージョンを持っています。それまでに存在していた Fargate Pods は変更されていません。
6. (オプション) クラスターを更新する前に、そのクラスターに Kubernetes Cluster Autoscaler をデプロイしてある場合は、更新後の Kubernetes のメジャーバージョンとマイナーバージョンに一致するように、Cluster Autoscaler を最新バージョンに更新します。

   a. ウェブブラウザで Cluster Autoscaler リリース ページを開き、クラスターの Kubernetes メジャーバージョンとマイナーバージョンに一致する最新の Cluster Autoscaler バージョンを見つけます。例えば、クラスターの Kubernetes バージョンが `1.30` である場合、`1.30` で始まる最新の Cluster Autoscaler リリースを見つけます。次のステップで使用するため、そのリリースのセマンティックバージョン番号 (例: `1.30.n`) を書き留めておきます。

   b. 次のコマンドを使用して、Cluster Autoscaler イメージタグを、前のステップで書き留めたバージョンに設定します。必要に応じて、*`1.30.n`* を独自の値に置き換えます。

   ```
   kubectl -n kube-system set image deployment.apps/cluster-autoscaler cluster-
   autoscaler=registry.k8s.io/autoscaling/cluster-autoscaler:v1.30.n
   ```

7. (GPU ノードを含むクラスターのみ) クラスターに GPU 対応のノードグループ (`p3.2xlarge` など) がある場合は、クラスターの Kubernetes 向け NVIDIA デバイスプラグイン DaemonSet を更新する必要があります。次のコマンドを実行する前に、*`vX.X.X`* を必要となる NVIDIA/k8s-device-plugin バージョンに置き換えます。

   ```
   kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/
   nvidia-device-plugin.yml
   ```

8. Amazon VPC CNI plugin for Kubernetes、CoreDNS、および `kube-proxy` アドオンを更新するサービスアカウントトークンにリストされている最小バージョンに対してアドオンを更新することをお勧めします。

   - Amazon EKS アドオンを使用している場合は、Amazon EKS コンソールで [クラスター] をクリックし、左のナビゲーションペインで更新したクラスター名を選択します。通知がコンソールに表示されます。更新可能なアドオンごとに、新しいバージョンが利用可能であることが通知されます。アドオンを更新するには、[アドオン] タブを選択します。更新があるアドオンが表示されているボックスで [今すぐ更新] を選択し、使用可能なバージョンを選択してから、[更新] を選択します。

- 別の方法として、AWS CLI または eksctl を使用してアドオンを更新することもできます。詳細については、「アドオンの更新」を参照してください。

9. 必要に応じて、kubectl のバージョンを更新します。Amazon EKS クラスターコントロールプレーンとのマイナーバージョンの相違が 1 つ以内である kubectl バージョンを使用する必要があります。例えば、1.29 kubectl クライアントは Kubernetes、1.28、1.29 および 1.30 クラスターで動作します。現在インストールされているバージョンは、次のコマンドで確認できます。

```
kubectl version --client
```

Amazon EKS クラスターの削除

Amazon EKS クラスターの使用が終了したら、関連付けられたリソースを削除して、不要なコストが発生しないようにする必要があります。

接続されたクラスターを削除するには、「クラスターの登録解除」を参照してください。

> **Important**
>
> - クラスター内にロードバランサーに関連付けられているアクティブなサービスがある場合は、クラスターを削除する前にこれらのサービスを削除して、ロードバランサーが正しく削除されるようにする必要があります。この操作を行わないと、VPC 内のリソースが孤立し、VPC を削除できなくなる可能性があります。
> - クラスター作成者が削除されたために エラーが発生した場合は、この記事を参照して解決してください。
> - Amazon Managed Service for Prometheus リソースはクラスターのライフサイクル外にあるため、クラスターとは別途維持する必要があります。クラスターを削除するときは、対象コストを抑えるために、該当するスクレイパーも削除してください。詳細については、Amazon Managed Service for Prometheus ユーザーガイドの「スクレイパーの検出と作成」を参照してください。

eksctl、AWS Management Console、またはAWS CLI を使用してクラスターを削除できます。

eksctl

`eksctl` を使用して Amazon EKS クラスターとノードを削除するには

この手順には、`eksctl` バージョン `0.183.0` 以降が必要です。お使いのバージョンは、以下のコマンドを使用して確認できます。

```
eksctl version
```

`eksctl` のインストールまたはアップグレードの手順については、`eksctl` ドキュメントの「インストール」を参照してください。

1. クラスターで実行されているすべてのサービスを一覧表示します。

   ```
   kubectl get svc --all-namespaces
   ```

2. 関連付けられた `EXTERNAL-IP` 値のあるサービスすべてを削除します。これらのサービスの前には Elastic Load Balancing ロードバランサーがあり、ロードバランサーと関連リソースを適切に解放するには、Kubernetes でそれらを削除する必要があります。

   ```
   kubectl delete svc service-name
   ```

3. 次のコマンドを使用して、クラスターとそれに関連するノードを削除します。この際、*prod* を実際のクラスター名に置き換えます。

   ```
   eksctl delete cluster --name prod
   ```

   出力:

   ```
   [#]  using region region-code
   [#]  deleting EKS cluster "prod"
   [#]  will delete stack "eksctl-prod-nodegroup-standard-nodes"
   [#]  waiting for stack "eksctl-prod-nodegroup-standard-nodes" to get deleted
   [#]  will delete stack "eksctl-prod-cluster"
   [#]  the following EKS cluster resource(s) for "prod" will be deleted: cluster.
    If in doubt, check CloudFormation console
   ```

AWS Management Console

AWS Management Console を使用して Amazon EKS クラスターを削除するには

1. クラスターで実行されているすべてのサービスを一覧表示します。

```
kubectl get svc --all-namespaces
```

2. 関連付けられた EXTERNAL-IP 値のあるサービスすべてを削除します。これらのサービスの前には Elastic Load Balancing ロードバランサーがあり、ロードバランサーと関連リソースを適切に解放するには、Kubernetes でそれらを削除する必要があります。

```
kubectl delete svc service-name
```

3. すべてのノードグループと Fargate プロファイルを削除します。

   a. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。

   b. 左のナビゲーションペインで Amazon EKS [クラスター] を選択し、クラスターのタブ付きリストから、削除するクラスター名を選択します。

   c. [コンピューティング] タブを選択し、削除するノードグループを選択します。[削除] を選択し、ノードグループの名前を入力し、[削除] を選択します。クラスター内のすべてのノードグループを削除します。

   > **Note**
   >
   > ここでリストに表示されるノードグループは、マネージド型ノードグループのみです。

   d. 削除する [Fargate プロファイル] を選択し、[削除] を選択し、プロファイルの名前を入力した上で [削除] を選択します。クラスター内のすべてのFargate プロファイルを削除します。

4. クラスター内のすべてのセルフマネージド型ノード AWS CloudFormation スタックを削除します。

   a. https://console.aws.amazon.com/cloudformation で AWS CloudFormation コンソール を開きます。

   b. 削除するノードスタックを選択し、[削除] を選択します。

c. [スタックの削除] 確認ダイアログボックスで、[スタックを削除] をクリックします。クラスター内のすべてのセルフマネージド型ノードスタックを削除します。

5. クラスターを削除します。

a. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。

b. 削除するクラスターを選択し、[削除] を選択します。

c. クラスターの削除確認画面で、[削除] を選択します。

6. (オプション) VPC AWS CloudFormation スタックを削除します。

a. https://console.aws.amazon.com/cloudformation で AWS CloudFormation コンソール を開きます。

b. 削除する VPC スタックを選択し、[削除] をクリックします。

c. [スタックの削除] 確認ダイアログボックスで、[スタックを削除] をクリックします。

AWS CLI

AWS CLI を使用して Amazon EKS クラスターを削除するには

1. クラスターで実行されているすべてのサービスを一覧表示します。

```
kubectl get svc --all-namespaces
```

2. 関連付けられた EXTERNAL-IP 値のあるサービスすべてを削除します。これらのサービスの前には Elastic Load Balancing ロードバランサーがあり、ロードバランサーと関連リソースを適切に解放するには、Kubernetes でそれらを削除する必要があります。

```
kubectl delete svc service-name
```

3. すべてのノードグループと Fargate プロファイルを削除します。

a. 次のコマンドを使用して、クラスター内のノードグループを一覧表示します。

```
aws eks list-nodegroups --cluster-name my-cluster
```

> **Note**
>
> ここでリストに表示されるノードグループは、マネージド型ノードグループのみです。

b. 次のコマンドを使用して、各ノードグループを削除します。クラスター内のすべてのノードグループを削除します。

```
aws eks delete-nodegroup --nodegroup-name my-nodegroup --cluster-name my-cluster
```

c. 次のコマンドを使用して、クラスターの Fargate プロファイルを一覧表示します。

```
aws eks list-fargate-profiles --cluster-name my-cluster
```

d. 次のコマンドを使用して、各 Fargate プロファイルを削除します。クラスター内のすべてのFargate プロファイルを削除します。

```
aws eks delete-fargate-profile --fargate-profile-name my-fargate-profile --cluster-name my-cluster
```

4. クラスター内のすべてのセルフマネージド型ノード AWS CloudFormation スタックを削除します。

a. 以下のコマンドで、使用可能な AWS CloudFormation スタックを一覧表示します。出力結果から、ノードテンプレート名を見つけます。

```
aws cloudformation list-stacks --query "StackSummaries[].StackName"
```

b. 次のコマンドを使用して、各ノードスタックを削除します。この際、*node-stack* を実際のノードスタック名に置き換えます。クラスター内のすべてのセルフマネージド型ノードスタックを削除します。

```
aws cloudformation delete-stack --stack-name node-stack
```

5. 以下のコマンドを使用して、クラスターを削除します。*my-cluster* を実際のクラスター名に置き換えます。

```
aws eks delete-cluster --name my-cluster
```

6. (オプション) VPC AWS CloudFormation スタックを削除します。

   a. 以下のコマンドで、使用可能な AWS CloudFormation スタックを一覧表示します。結果の出力で VPC テンプレート名を見つけます。

   ```
   aws cloudformation list-stacks --query "StackSummaries[].StackName"
   ```

   b. 以下のコマンドを使用して、VPC スタックを削除します。*my-vpc-stack* を実際の VPC スタック名に置き換えます。

   ```
   aws cloudformation delete-stack --stack-name my-vpc-stack
   ```

Amazon EKS クラスターエンドポイントアクセスコントロール

このトピックは、Amazon EKS クラスターの Kubernetes API サーバーエンドポイントのプライベートアクセスを有効にし、インターネットからのパブリックアクセスを制限または完全に無効にするのに役立ちます。

新しいクラスターを作成すると、Amazon EKS によってマネージド型の Kubernetes API サーバーのエンドポイントが作成されます。ユーザーはこのエンドポイントを、(`kubectl` などの Kubernetes 管理ツールを通じて) クラスターとの通信に使用します。デフォルトでは、この API サーバーエンドポイントはインターネットに公開され、API サーバーへのアクセスは、AWS Identity and Access Management (IAM) とネイティブの Kubernetes Role Based Access Control (RBAC) の組み合わせを使用して保護されます。

Kubernetes API サーバーへのプライベートアクセスを有効にすると、ノードと API サーバー間のすべての通信が VPC 内で行われるようにできます。インターネットから API サーバーにアクセスできる IP アドレスを制限したり、API サーバーへのインターネットアクセスを完全に無効にしたりできます。

> **Note**
>
> このエンドポイントは Kubernetes API サーバー用であり、AWS API と通信するための従来の AWS PrivateLink エンドポイントではないため、Amazon VPC コンソールにはエンドポイントとして表示されません。

クラスターでエンドポイントへのプライベートアクセスを有効にすると、Amazon EKS によって自動的に Route 53 のプライベートホストゾーンが作成され、クラスターの VPC に関連付けられます。このプライベートホストゾーンは Amazon EKS によって管理され、アカウントの Route 53 リソースには表示されません。プライベートホストゾーンが API サーバーに正しくトラフィックをルーティングするためには、VPC で `enableDnsHostnames` と `enableDnsSupport` が `true` に設定され、VPC 用に設定された DHCP オプションで、ドメイン名サーバーリストに `AmazonProvidedDNS` が含まれている必要があります。詳細については、Amazon VPC ユーザーガイドの「VPC の DNS サポートを表示および更新する」を参照してください。

API サーバーエンドポイントのアクセス要件は、新しいクラスターを作成するときに定義できます。また、クラスターの API サーバーエンドポイントのアクセスは、随時更新できます。

クラスターエンドポイントのアクセスの変更

既存クラスターのエンドポイントのアクセスを変更するには、このセクションの手順に従ってください。次の表は、サポートされている API サーバーエンドポイントのアクセスの組み合わせとそれらに関連付けられている動作を示しています。

API サーバーエンドポイントのアクセスオプション

エンドポイントのパブリックアクセス	エンドポイントのプライベートアクセス	Behavior
有効	Disabled	• これは、新しい Amazon EKS クラスターのデフォルトの動作です。 • クラスタの VPC 内から発信された Kubernetes API リクエスト (ノードからコントロールプレーンへの通信など) は VPC を離れますが、Amazon のネットワークは離れません。 • クラスター API サーバーにはインターネットからアクセスできます。必要に応じて、パブリックエンドポイントにアクセスできる

エンドポイントのパブリックアクセス	エンドポイントのプライベートアクセス	Behavior
		CIDR ブロックを制限できます。特定の CIDR ブロックへのアクセスを制限する場合は、プライベートエンドポイントも有効にするか、指定する CIDR ブロックに、ノードと Fargate Pods (使用している場合) がパブリックエンドポイントにアクセスするアドレスが含まれていることを確認することをお勧めします。
有効	有効	• クラスタの VPC (ノードからコントロールプレーンへの通信など) 内の Kubernetes API リクエストは、プライベートVPCエンドポイントを使用します。 • クラスター API サーバーにはインターネットからアクセスできます。必要に応じて、パブリックエンドポイントにアクセスできる CIDR ブロックを制限できます。

エンドポイントのパブリックアクセス	エンドポイントのプライベートアクセス	Behavior
Disabled	有効	• クラスター API サーバーへのすべてのトラフィックは、クラスターの VPC または接続されたネットワーク内から送信する必要があります。 • インターネットから API サーバーへのパブリックアクセスは存在しません。kubectl コマンドはすべて、VPC または接続されたネットワーク内から実行する必要があります。接続オプションについては、プライベート専用 API サーバーへのアクセスを参照してください。 • クラスターの API サーバーエンドポイントは、パブリック DNS サーバーによって VPC のプライベート IP アドレスに解決されます。これまでは、エンドポイントは VPC 内からしか解決できませんでした。 エンドポイントが既存のクラスターの VPC 内のプライベート IP アドレスに解決されない場合は、次の操作を実行できます。 • パブリックアクセスを有効にし、再度無効にしま

エンドポイントのパブリックアクセス	エンドポイントのプライベートアクセス	Behavior
		す。この操作は、クラスターに対して 1 回行うだけで済みます。それ以降、エンドポイントはプライベート IP アドレスに解決されます。 • クラスターを更新します。

AWS Management Console または AWS CLI を使用して、クラスター API サーバーのエンドポイントアクセスを変更できます。

AWS Management Console

AWS Management Console を使用してクラスター API サーバーエンドポイントのアクセスを変更するには

1. https://console.aws.amazon.com/eks/home#/clusters で Amazon EKS コンソールを開きます。
2. クラスター名を選択すると、そのクラスターの情報を表示されます。
3. [Networking] (ネットワーキング) タブを開き、[Update] (更新) を選択します。
4. [Private access] (プライベートアクセス) の場合は、クラスターの Kubernetes API サーバーエンドポイントに対するプライベートアクセスを有効にするか無効にするかを選択します。プライベートアクセスを有効にした場合、クラスターの VPC 内から送信される Kubernetes API リクエストは、プライベート VPC エンドポイントを使用します。パブリックアクセスを無効にするには、プライベートアクセスを有効にする必要があります。
5. [Public access] (パブリックアクセス) の場合は、クラスターの Kubernetes API サーバーエンドポイントに対するパブリックアクセスを有効にするか無効にするかを選択します。パブリックアクセスを無効にすると、クラスターの Kubernetes API サーバーはクラスター VPC 内からのみリクエストを受信できます。
6. (オプション) [Public access (パブリックアクセス)] で有効化を行うと、インターネットからパブリックエンドポイントと通信するためのアドレスを指定できるようになります。[詳細設定] を選択します。CIDR ブロックを (*203.0.113.5/32* のように) 入力します。ブロッ

クに予約済みアドレスを含めることはできません。[ソースの追加] を選択すると、追加のブロックを入力できます。指定できる CIDR ブロックには最大数があります。詳細については、「Amazon EKS Service Quotas」を参照してください。ブロックをまったく指定しない場合、パブリック API サーバーエンドポイントは、すべて (0.0.0.0/0) の IP アドレスからリクエストを受信します。CIDR ブロックを使用してパブリックエンドポイントへのアクセスを制限する場合は、同時にプライベートエンドポイントアクセスも有効化することをお勧めします。これにより、ノードと (存在している場合は) Fargate Pods がクラスターと通信できるようになります。プライベートエンドポイントが有効になっていない場合は、パブリックアクセスエンドポイントの CIDR ソースに、VPC からの出力ソースを含める必要があります。例えば、プライベートサブネットに NAT ゲートウェイを介してインターネットと通信するノードがある場合、パブリックエンドポイントで許可された CIDR ブロックの一部として、NAT ゲートウェイのアウトバウンド IP アドレスを追加する必要があります。

7. [更新] を選択して終了します。

AWS CLI

AWS CLI を使用してクラスター API サーバーエンドポイントのアクセスを変更するには

AWS CLI バージョン 1.27.160 以降を使用して、次のステップを実行します。現在のバージョンは、aws --version で確認できます。AWS CLI をインストールまたはアップグレードするには、「AWS CLI のインストール」を参照してください。

1. 次の AWS CLI コマンドを使用してクラスター API サーバーエンドポイントのアクセスを更新します。クラスター名と必要なエンドポイントアクセス値を置き換えます。endpointPublicAccess=true を設定した場合は、(オプションで) 1 つの CIDR ブロック、または publicAccessCidrs の CIDR ブロックのカンマ区切りリストを入力できます。ブロックに予約済みアドレスを含めることはできません。CIDR ブロックを指定すると、パブリック API サーバーエンドポイントはリストされたブロックからのリクエストのみを受信します。指定できる CIDR ブロックには最大数があります。詳細については、「Amazon EKS Service Quotas」を参照してください。CIDR ブロックを使用してパブリックエンドポイントへのアクセスを制限する場合は、同時にプライベートエンドポイントアクセスも有効化することをお勧めします。これにより、ノードと (存在している場合は) Fargate Pods がクラスターと通信できるようになります。プライベートエンドポイントが有効になっていない場合は、パブリックアクセスエンドポイントの CIDR ソースに、VPC からの出力ソースを含める必要があります。例えば、プライベートサブネットに NAT ゲートウェイを介してインターネットと通信するノードがある場合、パブリックエンドポイントで許可された CIDR ブロックの一部として、NAT ゲートウェイのアウトバウンド IP アドレス

を追加する必要があります。CIDR ブロックを指定しない場合、パブリック API サーバーエンドポイントはすべての (0.0.0.0/0) IP アドレスからリクエストを受信します。

> **Note**
>
> 次のコマンドは、API サーバーエンドポイントの 1 つの IP アドレスからのプライベートアクセスとパブリックアクセスを有効にします。*203.0.113.5/32* の部分は、単一の CIDR ブロック、またはネットワークアクセスが許可される CIDR ブロックのカンマ区切りリストに置き換えます。

```
aws eks update-cluster-config \
    --region region-code \
    --name my-cluster \
    --resources-vpc-config
 endpointPublicAccess=true,publicAccessCidrs="203.0.113.5/32",endpointPrivateAccess=true
```

出力例は次のとおりです。

```
{
    "update": {
        "id": "e6f0905f-a5d4-4a2a-8c49-EXAMPLE00000",
        "status": "InProgress",
        "type": "EndpointAccessUpdate",
        "params": [
            {
                "type": "EndpointPublicAccess",
                "value": "true"
            },
            {
                "type": "EndpointPrivateAccess",
                "value": "true"
            },
            {
                "type": "publicAccessCidrs",
                "value": "[\203.0.113.5/32\"]"
            }
        ],
        "createdAt": 1576874258.137,
        "errors": []
    }
```

```
}
```

2. 次のコマンドでエンドポイントアクセス更新のステータスをモニタリングします。この際、以前のコマンドで返ったクラスター名と更新 ID を使用します。ステータスが Successful と表示されたら、更新は完了です。

```
aws eks describe-update \
    --region region-code \
    --name my-cluster \
    --update-id e6f0905f-a5d4-4a2a-8c49-EXAMPLE00000
```

出力例は次のとおりです。

```
{
    "update": {
        "id": "e6f0905f-a5d4-4a2a-8c49-EXAMPLE00000",
        "status": "Successful",
        "type": "EndpointAccessUpdate",
        "params": [
            {
                "type": "EndpointPublicAccess",
                "value": "true"
            },
            {
                "type": "EndpointPrivateAccess",
                "value": "true"
            },
            {
                "type": "publicAccessCidrs",
                "value": "[\203.0.113.5/32\"]"
            }
        ],
        "createdAt": 1576874258.137,
        "errors": []
    }
}
```

プライベート専用 API サーバーへのアクセス

クラスターの Kubernetes API サーバーエンドポイントに対するパブリックアクセスを無効にした場合は、VPC または 接続されたネットワーク 内からのみ API サーバーにアクセスできます。Kubernetes API サーバーエンドポイントにアクセスする方法はいくつかあります。

接続されたネットワーク

AWS トランジットゲートウェイまたはその他の接続オプションを使用してネットワークを VPC に接続し、接続されたネットワークのコンピュータを使用します。接続されたネットワークからのポート 443 でのイングレストラフィックを許可するためのルールが、Amazon EKS コントロールプレーンセキュリティグループに含まれていることを確認する必要があります。

Amazon EC2 踏み台ホスト

Amazon EC2 インスタンスをクラスターの VPC のパブリックサブネットで起動し、SSH 経由でそのインスタンスにログインして `kubectl` コマンドを実行できます。詳細については、「Linux の AWS 踏み台ホスト」を参照してください。踏み台ホストからのポート 443 でのイングレストラフィックを許可するためのルールが、Amazon EKS コントロールプレーンセキュリティグループに含まれていることを確認する必要があります。詳細については、「Amazon EKS セキュリティグループの要件および考慮事項」を参照してください。

踏み台ホスト用に `kubectl` を設定するときには、クラスターの RBAC 設定に既にマッピングされている AWS 認証情報を使用するか、踏み台ホストが使用する IAM プリンシパル を RBAC 設定に追加してから、エンドポイントのパブリックアクセスを削除します。詳細については、the section called “Kubernetes API へのアクセスを許可する”および許可されていないか、アクセスが拒否されました (`kubectl`)を参照してください。

AWS Cloud9 IDE

AWS Cloud9 は、ブラウザだけでコードを記述、実行、およびデバッグできるクラウドベースの統合開発環境 (IDE) です。クラスターの VPC に AWS Cloud9 IDE を作成し、その IDE を使用してクラスターと通信できます。詳細については、AWS Cloud9 で環境を作成するを参照してください。Amazon EKS コントロールプレーンセキュリティグループに、IDE セキュリティグループからのポート 443 でのイングレストラフィックを許可するためのルールが、含まれていることを確認する必要があります。詳細については、「Amazon EKS セキュリティグループの要件および考慮事項」を参照してください。

AWS Cloud9 IDE 用に `kubectl` を設定するときには、クラスターの RBAC 設定に既にマッピングされている AWS 認証情報を使用するか、IDE が使用する IAM プリンシパルを RBAC

設定に追加してから、エンドポイントのパブリックアクセスを削除します。詳細については、Kubernetes API へのアクセスを許可する および許可されていないか、アクセスが拒否されました (kubectl)を参照してください。

既存のクラスター上でシークレット暗号化を有効にする

シークレット暗号化 を有効にすると、Kubernetes シークレットは選択した AWS KMS key を使用して暗号化されます。KMS キーは、次の条件を満たす必要があります。

- 対称
- データの暗号化と復号が可能
- クラスターと同じ AWS リージョン に作成
- KMS キーが別のアカウントで作成された場合、IAM プリンシパルには、その KMS キーへのアクセス権が必要となります。

詳細については、「AWS Key Management Service デベロッパーガイド」の「他のアカウントの IAM プリンシパルが KMS キーを使用することを許可する」を参照してください。

> **Warning**
> 一度有効化したシークレット暗号化は無効化できません。このアクションを元に戻すことはできません。

eksctl

暗号化は次の 2 つの方法のいずれかで有効にできます。

- 1 つのコマンドでクラスターに暗号化を追加します。

  シークレットを自動的に再暗号化するには、次のコマンドを実行します。

  ```
  eksctl utils enable-secrets-encryption \
      --cluster my-cluster \
      --key-arn arn:aws:kms:region-code:account:key/key
  ```

  シークレットの自動的な再暗号化をオプトアウトするには、次のコマンドを実行します。

```
eksctl utils enable-secrets-encryption
    --cluster my-cluster \
    --key-arn arn:aws:kms:region-code:account:key/key \
    --encrypt-existing-secrets=false
```

- kms-cluster.yaml ファイルを使用してクラスターに暗号化を追加します。

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code

secretsEncryption:
  keyARN: arn:aws:kms:region-code:account:key/key
```

シークレットに自動的な再暗号化をさせるには、次のコマンドを実行します。

```
eksctl utils enable-secrets-encryption -f kms-cluster.yaml
```

シークレットの自動的な再暗号化をオプトアウトするには、次のコマンドを実行します。

```
eksctl utils enable-secrets-encryption -f kms-cluster.yaml --encrypt-existing-
secrets=false
```

AWS Management Console

1. https://console.aws.amazon.com/eks/home#/clusters で Amazon EKS コンソール を開きます。
2. KMS 暗号化を追加するクラスターを選択します。
3. [Overview] (概要) タブを選択します (これはデフォルトで選択されています)。
4. [Secrets encryption] (シークレットの暗号化) セクションまでスクロールダウンし、[Enable] (有効化) ボタンを選択します。
5. ドロップダウンリストからキーを選択し、[Enable] (有効化) ボタンを選択します。キーが一覧表示されていない場合は、最初にキーを作成する必要があります。詳細については、「キーの作成」を参照してください。

6. [Confirm] (確認) ボタンを選択して、指定したキーを使用します。

AWS CLI

1. 次の AWS CLI コマンドを使用して、シークレット暗号化設定をクラスターに関連付けます。*example values* を自分の値に置き換えます。

```
aws eks associate-encryption-config \
    --cluster-name my-cluster \
    --encryption-config '[{"resources":["secrets"],"provider":
{"keyArn":"arn:aws:kms:region-code:account:key/key"}}]'
```

出力例は次のとおりです。

```
{
  "update": {
    "id": "3141b835-8103-423a-8e68-12c2521ffa4d",
    "status": "InProgress",
    "type": "AssociateEncryptionConfig",
    "params": [
      {
        "type": "EncryptionConfig",
        "value": "[{\"resources\":[\"secrets\"],\"provider\":{\"keyArn\":
\"arn:aws:kms:region-code:account:key/key\"}}]"
      }
    ],
    "createdAt": 1613754188.734,
    "errors": []
  }
}
```

2. 次のコマンドを使用すると、暗号化更新のステータスをモニタリングできます。前の出力で返された特定の cluster name と update ID を使用します。Successful ステータスが表示されると、更新は完了です。

```
aws eks describe-update \
    --region region-code \
    --name my-cluster \
    --update-id 3141b835-8103-423a-8e68-12c2521ffa4d
```

出力例は次のとおりです。

```
{
  "update": {
    "id": "3141b835-8103-423a-8e68-12c2521ffa4d",
    "status": "Successful",
    "type": "AssociateEncryptionConfig",
    "params": [
      {
        "type": "EncryptionConfig",
        "value": "[{\"resources\":[\"secrets\"],\"provider\":{\"keyArn\":
\"arn:aws:kms:region-code:account:key/key\"}}]"
      }
    ],
    "createdAt": 1613754188.734>,
    "errors": []
  }
}
```

3. クラスターで暗号化が有効になっていることを確認するには、describe-cluster コマンドを実行します。レスポンスの内容は、文字列 EncryptionConfig です。

```
aws eks describe-cluster --region region-code --name my-cluster
```

クラスターで暗号化を有効にしたら、既存のすべてのシークレットを新しいキーで暗号化する必要があります。

> **Note**
>
> eksctl を使用する場合、シークレットを自動的に再暗号化することをオプトアウトした場合にのみ、次のコマンドを実行する必要があります。

```
kubectl get secrets --all-namespaces -o json | kubectl annotate --overwrite -f - kms-
encryption-timestamp="time value"
```

> **Warning**
>
> 既存のクラスターでシークレット暗号化を有効にし、使用する KMS キーが削除されている場合は、このクラスターを復旧する手段はありません。KMS キーを削除すると、クラスター

は永続的にパフォーマンスが低下した状態になります。詳細については、「AWSKMS キーを削除する」を参照してください。

> **Note**
>
> create-key コマンドでは、デフォルトで対称暗号化 KMS キーが作成されます。この際には、アカウントのルート管理者に AWS KMS アクションとリソースへのアクセスを許可する、キーポリシーが使用されます。アクセス許可の範囲を絞り込む場合は、create-cluster API を呼び出すプリンシパルのポリシーで、kms:DescribeKey および kms:CreateGrant アクションが許可されていることを確認します。
> KMS エンベロープ暗号化を使用するクラスターの場合、kms:CreateGrant アクセス許可が必要です。条件 kms:GrantIsForAWSResource は、CreateCluster アクションでサポートされていないため、CreateCluster を実行するユーザーの kms:CreateGrant アクセス許可を制御する KMS ポリシーで使用しないでください。

Amazon EKS クラスター の Windows サポートの有効化

Windows ノードをデプロイする前に、以下の考慮事項を確認してください。

考慮事項

- HostProcess ポッドを使用して Windows ノードでホスト ネットワークを使用できます。詳細については、Kubernetes ドキュメントの「Create a Windows HostProcessPod」を参照してください。
- CoreDNS など、Linux でのみ動作するコアシステム Pods を実行するには、Amazon EKS クラスターに 1 つ以上の Linux ノードまたは Fargate ノードが含まれている必要があります。
- kubelet および kube-proxy イベントログは EKS Windows Event Log にリダイレクトされ、200 MB の制限に設定されます。
- Pods のセキュリティグループ を、Windows ノードで実行する Pods で使用することはできません。
- Windows ノードで カスタムネットワーキング を使用することはできません。
- IPv6 を Windows ノードで使用することはできません。
- Windows ノードは、ノードごとに 1 つの Elastic Network Interface をサポートします。デフォルトでは、Windows ノードごとに実行できる Pods の数は、ノードのインスタンスタイプの Elastic

Network Interface ごとに使用できる IP アドレスの数から、1 を引いた数に等しくなります。詳細については、Amazon EC2 ユーザーガイドの「各インスタンスタイプのネットワークインターフェイスごとの IP アドレス」を参照してください。

- Amazon EKS クラスターでは、ロードバランサーを持つ単一のサービスが、最大 1024 個までのバックエンド Pods をサポートできます。各 Pod には固有の IP アドレスがあります。OS ビルド 17763.2746 で始まる Windows Server 更新プログラム以降、以前の 64 個の Pods という上限はなくなりました。
- Windows コンテナは、Fargate の Amazon EKS Pods でサポートされていません。
- `vpc-resource-controller` ポッドからはログを取得できません。コントローラーをデータプレーンにデプロイしていたときには取得できていました。
- IPv4 アドレスが新しいポッドに割り当てられるまでにはクールダウン期間があります。これは、古い `kube-proxy` ルールが原因で、同じ IPv4 アドレスを持つ古いポッドにトラフィックが流れるのを防ぎます。
- コントローラーのソースは GitHub で管理されます。コントローラーに関する投稿や、問題の登録を行うには、GitHub の プロジェクト にアクセスしてください。
- Windows マネージド型ノードグループのカスタム AMI ID を指定するとき、AWS IAM Authenticator 設定マップに `eks:kube-proxy-windows` を追加します。詳細については、「AMI ID を指定する場合の制限と条件」を参照してください。

前提条件

- 既存のクラスター。クラスターは、次の表に示す Kubernetes バージョンとプラットフォームバージョンのいずれかを実行している必要があります。一覧にあるバージョンより後の Kubernetes とプラットフォームのバージョンもサポートされます。クラスターまたはプラットフォームのバージョンが次のいずれかのバージョンより前の場合は、クラスターのデータプレーン上で レガシー Windows サポートを有効にする 必要があります。クラスターが次の Kubernetes とプラットフォームのバージョンのいずれか以降になったら、コントロールプレーン上で レガシー Windows サポートを削除 して Windows サポートの有効化 を実行できます。

Kubernetes バージョン	プラットフォームバージョン
1.30	eks.2
1.29	eks.1

Kubernetes バージョン	プラットフォームバージョン
1.28	eks.1
1.27	eks.1
1.26	eks.1
1.25	eks.1
1.24	eks.2

- CoreDNS を実行するには、クラスターに少なくとも 1 つ (推奨値は少なくとも 2 つ) の Linux ノードまたは Fargate Pod が必要です。レガシー Windows サポートを有効にする場合は、 CoreDNS の実行に Linux ノードを使用する必要があります (Fargate Pod は使用できません)。
- 既存の Amazon EKS クラスター の IAM ロール。

Windows サポートの有効化

クラスターが、前提条件のリストにある Kubernetes およびプラットフォームのバージョンのいずれか以降ではない場合、代わりにレガシー Windows サポートを有効にする必要があります。詳細については、「レガシー Windows サポートの有効化」を参照してください。

クラスターで Windows サポートを有効にしたことがない場合は、次のステップに進みます。

前提条件 の一覧にある Kubernetes またはプラットフォームのバージョンより前のクラスターで Windows サポートを有効にした場合は、まず データプレーンから vpc-resource-controller と vpc-admission-webhook を削除 する必要があります。これらは廃止され、もう必要ありません。

クラスターに対して Windows サポートを有効にするには

1. クラスターに Amazon Linux ノードがなく、Pods のセキュリティグループを使用する場合は、次のステップに進みます。それ以外の場合は、クラスターのロールに AmazonEKSVPCResourceController マネージドポリシーがアタッチされていることを確認します。*eksClusterRole* をクラスターロール名に置き換えます。

   ```
   aws iam list-attached-role-policies --role-name eksClusterRole
   ```

出力例は次のとおりです。

```
{
    "AttachedPolicies": [
        {
            "PolicyName": "AmazonEKSClusterPolicy",
            "PolicyArn": "arn:aws:iam::aws:policy/AmazonEKSClusterPolicy"
        },
        {
            "PolicyName": "AmazonEKSVPCResourceController",
            "PolicyArn": "arn:aws:iam::aws:policy/AmazonEKSVPCResourceController"
        }
    ]
}
```

前の出力のようにポリシーがアタッチされている場合は、次のステップをスキップします。

2. Amazon EKS クラスター の IAM ロール に AmazonEKSVPCResourceController マネージドポリシーを添付します。*eksClusterRole* をクラスターロール名に置き換えます。

```
aws iam attach-role-policy \
  --role-name eksClusterRole \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSVPCResourceController
```

3. 次の内容で、*vpc-resource-controller-configmap.yaml* という名前のファイルを作成します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: amazon-vpc-cni
  namespace: kube-system
data:
  enable-windows-ipam: "true"
```

4. ConfigMap をクラスターに適用する

```
kubectl apply -f vpc-resource-controller-configmap.yaml
```

5. aws-auth ConfigMap に、Windows ノードのインスタンスロールに eks:kube-proxy-windows RBAC 権限グループを含めるマッピングが含まれていることを確認します。次のコマンドを使用してインストールします。

```
kubectl get configmap aws-auth -n kube-system -o yaml
```

出力例は次のとおりです。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: aws-auth
  namespace: kube-system
data:
  mapRoles: |
    - groups:
      - system:bootstrappers
      - system:nodes
      - eks:kube-proxy-windows # This group is required for Windows DNS resolution
 to work
      rolearn: arn:aws:iam::111122223333:role/eksNodeRole
      username: system:node:{{EC2PrivateDNSName}}
[...]
```

グループの下に eks:kube-proxy-windows がリストされているはずです。グループが指定されていない場合は、必要なグループを含むように ConfigMap を更新するか、作成する必要があります。aws-auth ConfigMapの詳細については、「aws-auth ConfigMap をクラスターに適用する」を参照してください。

データプレーンからのレガシー Windows サポートの削除

前提条件 の一覧にある Kubernetes またはプラットフォームのバージョンより前のクラスターで Windows サポートを有効にした場合は、まずデータプレーンから vpc-resource-controller とvpc-admission-webhook を削除する必要があります。これらで提供されていた機能はコントロールプレーンで有効になったため、これらは廃止され、もう必要ありません。

1. 次のコマンドを使用して vpc-resource-controller をアンインストールします。このコマンドは、元々インストールしたツールに関係なく使用します。*region-code* (/manifests/

の後のそのテキストのインスタンスのみ) を、クラスターが含まれている AWS リージョン に置き換えます。

```
kubectl delete -f https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-resource-controller/latest/vpc-resource-controller.yaml
```

2. インストールに使用したツールの手順を使用して、vpc-admission-webhook をアンインストールします。

   eksctl

   以下のコマンドを実行します。

   ```
   kubectl delete deployment -n kube-system vpc-admission-webhook
   kubectl delete service -n kube-system vpc-admission-webhook
   kubectl delete mutatingwebhookconfigurations.admissionregistration.k8s.io vpc-admission-webhook-cfg
   ```

   kubectl on macOS or Windows

   以下のコマンドを実行します。*region-code* (/manifests/ の後のそのテキストのインスタンスのみ) を、クラスターが含まれている AWS リージョン に置き換えます。

   ```
   kubectl delete -f https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-admission-webhook/latest/vpc-admission-webhook-deployment.yaml
   ```

3. コントロールプレーン上で、クラスターに対する Windows サポートを有効 にします。

Windows サポートを無効にする

クラスターで Windows サポートを無効にするには

1. クラスターに Amazon Linux ノードが含まれていて、Pods のセキュリティグループ をそれらとともに使用する場合は、このステップをスキップしてください。

   クラスターのロールから AmazonVPCResourceController マネージド IAM ポリシーを削除します。*eksClusterRole* をクラスタロールの名前と *111122223333* のアカウントIDに置き換えます。

```
aws iam detach-role-policy \
    --role-name eksClusterRole \
    --policy-arn arn:aws:iam::aws:policy/AmazonEKSVPCResourceController
```

2. amazon-vpc-cni ConfigMap で Windows IPAM を無効にします。

```
kubectl patch configmap/amazon-vpc-cni \
                    -n kube-system \
                    --type merge \
                    -p '{"data":{"enable-windows-ipam":"false"}}'
```

ポッドのデプロイ

クラスターにポッドをデプロイするときに、さまざまなノードタイプを混在させて実行する場合に使用するオペレーティングシステムを指定する必要があります。

Linux Pods の場合は、マニフェストで以下のノードセレクタテキストを使用します。

```
nodeSelector:
        kubernetes.io/os: linux
        kubernetes.io/arch: amd64
```

Windows Pods の場合は、マニフェストで以下のノードセレクタテキストを使用します。

```
nodeSelector:
        kubernetes.io/os: windows
        kubernetes.io/arch: amd64
```

サンプルアプリケーションをデプロイして、使用されているノードセレクタを確認できます。

レガシー Windows サポートの有効化

クラスターが、前提条件 の一覧にある Kubernetes とプラットフォームのバージョンのいずれか以降である場合は、代わりにコントロールプレーンで Windows サポートを有効にすることをお勧めします。詳細については、「Windows サポートの有効化」を参照してください。

次のステップは、クラスターまたはプラットフォームのバージョンが 前提条件 の一覧にあるバージョンより前の場合に、Amazon EKS クラスターのデータプレーンに対してレガシー Windows サ

ポートを有効にするのに役立ちます。クラスターとプラットフォームのバージョンが、前提条件 の一覧にあるバージョン以降になったら、レガシー Windows サポートを削除 し、コントロールプレーンに対して有効にする ことをお勧めします。

クラスターに対してレガシー Windows サポートを有効にするには、eksctl、Windows クライアント、または macOS あるいは Linux のクライアントを使用できます。

eksctl

eksctl を使用してクラスターに対してレガシー Windows サポートを有効にするには

前提条件

この手順には、eksctl バージョン 0.183.0 以降が必要です。お使いのバージョンは、以下のコマンドを使用して確認できます。

```
eksctl version
```

eksctl のインストールまたはアップグレードの詳細については、「eksctl ドキュメント」の「インストール」を参照してください。

1. 以下の eksctl コマンドを使用して、Amazon EKS クラスターで Windows サポートを有効にします。*my-cluster* を自分のクラスター名に置き換えます。このコマンドは、Windows ワークロードを実行するために Amazon EKS クラスターが必要とする、VPC リソースコントローラーと VPC アドミッションコントローラーのウェブフックをデプロイします。

   ```
   eksctl utils install-vpc-controllers --cluster my-cluster --approve
   ```

   > **Important**
   >
   > VPC アドミッションコントローラーのウェブフックは、発行日から 1 年後に有効期限が切れる証明書で署名されます。ダウンタイムを回避するには、有効期限が切れる前に証明書を更新してください。詳細については、「VPC アドミッションウェブフック証明書の更新」を参照してください。

2. Windows サポートを有効にした後、Windows ノードグループをクラスターに起動できます。詳細については、「セルフマネージド型 Windows ノードの起動」を参照してください。

Windows

Windows クライアントを使用してクラスターに対してレガシー Windows サポートを有効にするには

次のステップでは、*region-code* をクラスターが存在する AWS リージョン に置き換えてください。

1. VPC リソースコントローラーをクラスターにデプロイします。

```
kubectl apply -f https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-resource-controller/latest/vpc-resource-controller.yaml
```

2. VPC アドミッションコントローラーウェブフックをクラスターにデプロイします。

   a. 必要なスクリプトとデプロイファイルをダウンロードします。

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-admission-webhook/latest/vpc-admission-webhook-deployment.yaml;
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-admission-webhook/latest/Setup-VPCAdmissionWebhook.ps1;
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-admission-webhook/latest/webhook-create-signed-cert.ps1;
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-admission-webhook/latest/webhook-patch-ca-bundle.ps1;
```

   b. OpenSSL と jq をインストールします。

   c. VPC アドミッションウェブフックを設定してデプロイします。

```
./Setup-VPCAdmissionWebhook.ps1 -DeploymentTemplate ".\vpc-admission-webhook-deployment.yaml"
```

> **Important**
>
> VPC アドミッションコントローラーのウェブフックは、発行日から 1 年後に有効期限が切れる証明書で署名されます。ダウンタイムを回避するには、有効期限が切れる前に証明書を更新してください。詳細については、「VPC アドミッションウェブフック証明書の更新」を参照してください。

3. クラスターに必要なクラスターロールバインドがあるかどうかを確認します。

```
kubectl get clusterrolebinding eks:kube-proxy-windows
```

以下の出力例のような出力が返された場合、クラスターには必要なロールバインドがあります。

```
NAME                       AGE
eks:kube-proxy-windows     10d
```

出力に Error from server (NotFound) が含まれている場合、クラスターには必要なクラスターロールバインドがありません。*eks-kube-proxy-windows-crb.yaml* という名前のファイルを以下の内容で作成して、バインドを追加します。

```
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: eks:kube-proxy-windows
  labels:
    k8s-app: kube-proxy
    eks.amazonaws.com/component: kube-proxy
subjects:
  - kind: Group
    name: "eks:kube-proxy-windows"
roleRef:
  kind: ClusterRole
  name: system:node-proxier
  apiGroup: rbac.authorization.k8s.io
```

設定をクラスターに適用します。

```
kubectl apply -f eks-kube-proxy-windows-crb.yaml
```

4. Windows サポートを有効にした後、Windows ノードグループをクラスターに起動できます。詳細については、「セルフマネージド型 Windows ノードの起動」を参照してください。

macOS and Linux

macOS または Linux クライアントでクラスターに対してレガシー Windows サポートを有効にするには

この手順では、クライアントシステムに openssl ライブラリと jq JSON プロセッサがインストールされている必要があります。

次のステップでは、*region-code* を、クラスターが存在する AWS リージョン に置き換えてください。

1. VPC リソースコントローラーをクラスターにデプロイします。

```
kubectl apply -f https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-resource-controller/latest/vpc-resource-controller.yaml
```

2. クラスターの VPC アドミッションコントローラーウェブフックのマニフェストを作成します。

   a. 必要なスクリプトとデプロイファイルをダウンロードします。

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-admission-webhook/latest/webhook-create-signed-cert.sh
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-admission-webhook/latest/webhook-patch-ca-bundle.sh
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-admission-webhook/latest/vpc-admission-webhook-deployment.yaml
```

   b. シェルスクリプトに、実行に必要なアクセス許可を追加します。

```
chmod +x webhook-create-signed-cert.sh webhook-patch-ca-bundle.sh
```

   c. セキュアな通信のためのシークレットを作成します。

```
./webhook-create-signed-cert.sh
```

   d. シークレットを確認します。

```
kubectl get secret -n kube-system vpc-admission-webhook-certs
```

   e. ウェブフックを設定し、デプロイファイルを作成します。

```
cat ./vpc-admission-webhook-deployment.yaml | ./webhook-patch-ca-bundle.sh >
 vpc-admission-webhook.yaml
```

3. VPC アドミッションウェブフックをデプロイします。

```
kubectl apply -f vpc-admission-webhook.yaml
```

> **Important**
>
> VPC アドミッションコントローラーのウェブフックは、発行日から 1 年後に有効期限が切れる証明書で署名されます。ダウンタイムを回避するには、有効期限が切れる前に証明書を更新してください。詳細については、「VPC アドミッションウェブフック証明書の更新」を参照してください。

4. クラスターに必要なクラスターロールバインドがあるかどうかを確認します。

```
kubectl get clusterrolebinding eks:kube-proxy-windows
```

以下の出力例のような出力が返された場合、クラスターには必要なロールバインドがあります。

```
NAME                       ROLE                                AGE
eks:kube-proxy-windows   ClusterRole/system:node-proxier   19h
```

出力に `Error from server (NotFound)` が含まれている場合、クラスターには必要なクラスターロールバインドがありません。*eks-kube-proxy-windows-crb.yaml* という名前のファイルを以下の内容で作成して、バインドを追加します。

```
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: eks:kube-proxy-windows
  labels:
    k8s-app: kube-proxy
    eks.amazonaws.com/component: kube-proxy
subjects:
  - kind: Group
    name: "eks:kube-proxy-windows"
```

```
roleRef:
  kind: ClusterRole
  name: system:node-proxier
  apiGroup: rbac.authorization.k8s.io
```

設定をクラスターに適用します。

```
kubectl apply -f eks-kube-proxy-windows-crb.yaml
```

5. Windows サポートを有効にした後、Windows ノードグループをクラスターに起動できます。詳細については、「セルフマネージド型 Windows ノードの起動」を参照してください。

VPC アドミッションウェブフック証明書の更新

VPC アドミッションウェブフックで使用される証明書は、発行後 1 年で期限切れになります。ダウンタイムを回避するには、有効期限が切れる前に証明書を更新することが重要です。現在の証明書の有効期限は、次のコマンドを使用して確認できます。

```
kubectl get secret \
    -n kube-system \
    vpc-admission-webhook-certs -o json | \
    jq -r '.data."cert.pem"' | \
    base64 -decode | \
    openssl x509 \
    -noout \
    -enddate | \
    cut -d= -f2
```

出力例は次のとおりです。

```
May 28 14:23:00 2022 GMT
```

証明書は、eksctl または Windows または Linux/macOS コンピュータを使用して更新できます。VPC アドミッションウェブフックのインストールに使用した当初のツールの指示に従ってください。たとえば、最初に eksctl で VPC アドミッションウェブフックをインストールした場合、eksctl の指示に従って証明書を更新する必要があります。

eksctl

1. 証明書を再インストールします。*my-cluster* を自分のクラスター名に置き換えます。

```
eksctl utils install-vpc-controllers -cluster my-cluster -approve
```

2. 次のような出力が表示されます。

```
2021/05/28 05:24:59 [INFO] generate received request
2021/05/28 05:24:59 [INFO] received CSR
2021/05/28 05:24:59 [INFO] generating key: rsa-2048
2021/05/28 05:24:59 [INFO] encoded CSR
```

3. ウェブフックデプロイメントを再起動します。

```
kubectl rollout restart deployment -n kube-system vpc-admission-webhook
```

4. 更新した証明書の有効期限が切れていて、Windows Pods が `Container creating` 状態のままの場合は、それらの Pods を削除して再デプロイする必要があります。

Windows

1. 新しい証明書を生成するスクリプトを取得します。

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-admission-webhook/latest/webhook-create-signed-cert.ps1;
```

2. スクリプトのパラメータを準備します。

```
./webhook-create-signed-cert.ps1 -ServiceName vpc-admission-webhook-svc -SecretName vpc-admission-webhook-certs -Namespace kube-system
```

3. ウェブフックデプロイメントを再起動します。

```
kubectl rollout restart deployment -n kube-system vpc-admission-webhook-deployment
```

4. 更新した証明書の有効期限が切れていて、Windows Pods が `Container creating` 状態のままの場合は、それらの Pods を削除して再デプロイする必要があります。

Linux and macOS

前提条件

コンピュータに OpenSSL と jq がインストールされている必要があります。

1. 新しい証明書を生成するスクリプトを取得します。

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-admission-webhook/latest/webhook-create-signed-cert.sh
```

2. 権限を変更します。

```
chmod +x webhook-create-signed-cert.sh
```

3. スクリプトを実行します。

```
./webhook-create-signed-cert.sh
```

4. ウェブフックを再起動します。

```
kubectl rollout restart deployment -n kube-system vpc-admission-webhook-deployment
```

5. 更新した証明書の有効期限が切れていて、Windows Pods が Container creating 状態のままになっている場合は、それらの Pods を削除して再展開する必要があります。

Windows ノードでより高い Pod 密度のサポート

Amazon EKS では、各 Pod に VPC の IPv4 アドレスが割り当てられます。このため、ノード上でさらに多くの Pods を実行するのに十分なリソースがある場合でも、ノードにデプロイできる Pods の数の上限は、使用可能な IP アドレスによって制限されます。Windows ノードでサポートされる Elastic Network Interface は 1 つだけなので、デフォルトでは Windows ノードで使用できる IP アドレスの最大数は次のようになります。

```
Number of private IPv4 addresses for each interface on the node - 1
```

1 つの IP アドレスがネットワークインターフェースのプライマリ IP アドレスとして使用されるため、Pods に割り当てることはできません。

IP プレフィックスの委任を有効にすると、Windows ノードで Pod 密度をより高めることができます。この機能により、セカンダリ IPv4 アドレスを割り当てる代わりに、プライマリネットワークインターフェイスに /28IPv4 プレフィックスを割り当てることができます。IP プレフィックスを割り当てると、ノードで使用できる IPv4 アドレスの最大数が次のように増加します。

```
(Number of private IPv4 addresses assigned to the interface attached to the node - 1) *
 16
```

使用可能な IP アドレス数が大幅に増加したため、使用可能な IP アドレスによってノード上の Pods の数をスケーリングできることが制限されることはありません。詳細については、「Amazon EC2 ノードで使用可能な IP アドレスの量を増やす」を参照してください。

プライベートクラスターの要件

このトピックでは、AWS クラウド にデプロイされているが、アウトバウンドインターネットアクセスがない Amazon EKS クラスターをデプロイする方法について説明します。AWS Outposts にローカルクラスターがある場合、このトピックの代わりに「Outpost 上のセルフマネージド型の Amazon Linux ノードの起動」を参照してください。

Amazon EKS でのネットワークに詳しくない場合は、「De-mystifying cluster networking for Amazon EKS worker nodes (Amazon EKS ワーカーノードのクラスターネットワークを解明する)」を参照してください。クラスターにアウトバウンドインターネットアクセスがない場合、次の要件を満たす必要があります。

- クラスターは VPC 内のコンテナレジストリからイメージを取得する必要があります。VPC 内に Amazon Elastic Container Registry を作成し、そこにコンテナイメージをコピーしてノードの取得元にすることができます。詳細については、「あるリポジトリから別のリポジトリにコンテナイメージをコピーする」を参照してください。
- クラスターでは、エンドポイントのプライベートアクセスが有効になる必要があります。これは、ノードをクラスターエンドポイントに登録するために必要です。エンドポイントのパブリックアクセスはオプションです。詳細については、「Amazon EKS クラスターエンドポイントアクセスコントロール」を参照してください。
- セルフマネージド型 Linux および Windows ノードには、起動する前に次のブートストラップ引数を含める必要があります。これらの引数は Amazon EKS のイントロスペクションをバイパスするため、VPC 内からの Amazon EKS API へのアクセスは不要です。
  1. 次のコマンドを使用して、クラスターのエンドポイントの値を確認します。*my-cluster* の部分は、自分のクラスター名に置き換えます。

```
aws eks describe-cluster --name my-cluster --query cluster.endpoint --output text
```

出力例は次のとおりです。

```
https://EXAMPLE108C897D9B2F1B21D5EXAMPLE.sk1.region-code.eks.amazonaws.com
```

2. 次のコマンドを使用して、クラスターの認証機関の値を確認します。*my-cluster* の部分は、自分のクラスター名に置き換えます。

```
aws eks describe-cluster --name my-cluster --query cluster.certificateAuthority --output text
```

返された出力は長い文字です。

3. 次のコマンドの *cluster-endpoint* および *certificate-authority* を前のコマンドで返された出力の値に置き換えます。セルフマネージド型ノードを起動する際にブートストラップ引数を指定する方法の詳細については、「セルフマネージド型の Amazon Linux ノードの起動」および「セルフマネージド型 Windows ノードの起動」を参照してください。

- Linux ノードを使用する場合。

```
--apiserver-endpoint cluster-endpoint --b64-cluster-ca certificate-authority
```

その他の引数については、「GitHub」の「ブートストラップのスクリプト」を参照してください。

- Windows ノードを使用する場合。

> **Note**
>
> カスタムサービス CIDR を使用している場合は、-ServiceCIDR パラメータを使用して指定する必要があります。そうしなければ、クラスター内の Pods の DNS 解決が失敗します。

```
-APIServerEndpoint cluster-endpoint -Base64ClusterCA certificate-authority
```

その他の引数については、「ブートストラップスクリプトの設定パラメータ」を参照してください。

- クラスターの `aws-auth ConfigMap` は VPC 内から作成する必要があります。エントリの作成と `aws-auth ConfigMap` への追加の詳細については、ターミナルで **`eksctl create iamidentitymapping --help`** と入力してください。サーバに `ConfigMap` が存在しない場合は、コマンドを使用して ID マッピングを追加したときに `eksctl` によって作成されます。
- サービスアカウントの IAM ロールで構成された Pods は、AWS Security Token Service (AWS STS) API コールから認証情報を入手します。アウトバウンドインターネットアクセスがない場合は、VPC 内で AWS STS VPC エンドポイントを作成して使用する必要があります。ほとんどの AWS v1 SDK は、AWS STS VPC エンドポイントを使用しないグローバル AWS STS エンドポイント (`sts.amazonaws.com`) をデフォルトで使用します。AWS STS VPC エンドポイントを使用するには、リージョンの AWS STS エンドポイント (`sts.region-code.amazonaws.com`) を使用するように SDK を構成する必要がある場合があります。詳細については、「サービスアカウントの AWS Security Token Service エンドポイントを設定する」を参照してください。

- クラスターの VPC サブネットには、Pods がアクセスする必要のあるすべての AWS のサービスに対して VPN インターフェイスエンドポイントが必要です。詳細については、「インターフェイス VPC エンドポイントを使用して AWS サービスにアクセスする」を参照してください。下表には、一般的に使用されるサービスとエンドポイントがリスト表示されています。エンドポイントの詳細なリストについては、AWS PrivateLink ガイドの「AWS PrivateLink と連携する AWS サービス」を参照してください。

サービス	エンドポイント
Amazon EC2	com.amazonaws.*region-code* .ec2
Amazon Elastic Container Registry (コンテナイメージの取得用)	com.amazonaws.*region-code* .ecr.api、com.amazonaws.*region-code* .ecr.dkr、および com.amazonaws.*region-code* .s3
Application Load Balancerおよび ネットワークロードバランサー	com.amazonaws.*region-code* .elasticloadbalancing
AWS X-Ray	com.amazonaws.*region-code* .xray
Amazon CloudWatch Logs	com.amazonaws.*region-code* .logs

サービス	エンドポイント
AWS Security Token Service (サービスアカウントに IAM ロールを使用している場合に必要)	com.amazonaws.*region-code* .sts

考慮事項

- セルフマネージド型ノードはすべて、必要な VPC インターフェイスエンドポイントを持つサブネットにデプロイする必要があります。マネージド型ノードグループを作成する場合には、VPC インターフェイスエンドポイントのセキュリティグループでサブネットの CIDR を許可するか、ノードのセキュリティグループを作成し VPC インターフェイスエンドポイントのセキュリティグループに追加する必要があります。
- Pods で Amazon EFSAmazon EFS ボリュームを使用する場合、Amazon EFS CSI ドライバー をデプロイする前に、Amazon EKS クラスターと同じ AWS リージョン を使用するように、ドライバーの kustomization.yaml ファイルを変更して、コンテナイメージを設定する必要があります。
- AWS Load Balancer Controller を使用して、AWS Application Load Balancer (ALB) および Network Load Balancer をプライベートクラスターにデプロイできます。デプロイするときは、コマンドラインフラグを使用して、`enable-shield`、`enable-waf`、および `enable-wafv2` を false に設定する必要があります。Ingress オブジェクトのホスト名を使用した証明書の検出は、サポートされていません。これは、コントローラーが VPC エンドポイントを持たない AWS Certificate Manager に到達する必要があるからです。

  このコントローラーでは、Fargate で必要な IP ターゲットを持つネットワークロードバランサをサポートします。詳細については、Amazon EKS でのアプリケーション負荷分散およびネットワークロードバランサーを作成するを参照してください。
- Cluster Autoscaler がサポートされています。クラスターオートスケーラー Pods をデプロイする場合、コマンドラインに `--aws-use-static-instance-list=true` が含まれていることを確認してください。詳細については、「GitHub」で「静的インスタンスリストを使用」を参照してください。ワーカーノード VPC には、AWS STS VPC エンドポイントと自動スケーリング VPC エンドポイントも含める必要があります。
- 一部のコンテナソフトウェア製品では、AWS Marketplace Metering Service にアクセスする API 呼び出しを使用して、使用状況をモニタリングします。プライベートクラスターではこれらの呼び出しが許可されないため、これらのコンテナタイプはプライベートクラスターには使用できません。

Amazon EKS Kubernetes のバージョン

Kubernetes は、新機能、設計の更新、およびバグ修正とともに急速に進化しています。コミュニティは、平均して 4 か月に一度、新しい Kubernetes のマイナーバージョン (1.30など) をリリースします。Amazon EKS は、マイナーバージョンのアップストリームリリースと非推奨サイクルに従います。新しい Kubernetes バージョンが Amazon EKS で利用可能になったら、利用可能な最新のバージョンが使用できるよう、クラスターをタイムリーに更新することをお勧めします。

マイナーバージョンは、リリース後最初の 14 か月間は Amazon EKS の標準サポート対象となります。標準サポート終了日を過ぎたバージョンは、自動的に次の 12 か月間の延長サポートに入ります。延長サポートでは、クラスター時間あたりの追加料金で、所定の Kubernetes バージョンをより長く使用することができます。延長サポート期間が終了する前にクラスターを更新しなかった場合、クラスターは現在サポートされている最も古い延長バージョンに自動的にアップグレードされます。

Amazon EKS でサポートされている利用可能な最新 Kubernetes バージョンを使用してクラスターを作成することをお勧めします。アプリケーションに特定のバージョンの Kubernetes が必要な場合は、古いバージョンを選択できます。標準サポートまたは延長サポートが提供されているどのバージョンでも、新しい Amazon EKS クラスターを作成できます。

標準サポートで入手可能なバージョン

現在、Amazon EKS 標準サポートでは以下の Kubernetes バージョンをご利用いただけます。

- 1.30
- 1.29
- 1.28
- 1.27
- 1.26

標準サポートの各バージョンで注意すべき重要な変更点については、「標準サポートバージョンのリリースノート」を参照してください。

延長サポートで利用可能なバージョン

現在、Amazon EKS 延長サポートでは、以下の Kubernetes バージョンをご利用いただけます。

- 1.25
- 1.24

- 1.23

延長サポートにおいて各バージョンで注意すべき重要な変更点については、「延長サポートバージョンのリリースノート」を参照してください。

以下の Kubernetes バージョンは、現在 Amazon EKS の延長サポートで利用可能ですが、これらのバージョンでは新しいクラスターを作成できないという追加要件があります。

- 1.22
- 1.21

  これらのバージョンの詳細については、「バージョン 1.21 および 1.22 のリリースノート」を参照してください。

Amazon EKS Kubernetes リリースカレンダー

次の表は、各 Kubernetes バージョンについて検討すべき重要なリリース日とサポート日を示しています。

> **Note**
>
> 月と年のみの日付はおおよその日付であり、確定後に正確な日付で更新されます。

Kubernetes バージョン	アップストリームのリリース	Amazon EKS リリース	標準サポート終了日	延長サポートの終了日
1.30	2024 年 4 月 17 日	2024 年 5 月 23 日	2025 年 7 月 23 日	2026 年 7 月 23 日
1.29	2023 年 12 月 13 日	2024 年 1 月 23 日	2025 年 3 月 23 日	2026 年 3 月 23 日
1.28	2023 年 8 月 15 日	2023 年 9 月 26 日	2024 年 11 月 26 日	2025 年 11 月 26 日
1.27	2023 年 4 月 11 日	2023 年 5 月 24 日	2024 年 7 月 24 日	2025 年 7 月 24 日

Kubernetes バージョン	アップストリームのリリース	Amazon EKS リリース	標準サポート終了日	延長サポートの終了日
1.26	2022 年 12 月 9 日	2023 年 4 月 11 日	2024 年 6 月 11 日	2025 年 6 月 11 日
1.25	2022 年 8 月 23 日	2023 年 2 月 22 日	2024 年 5 月 1 日	2025 年 5 月 1 日
1.24	2022 年 5 月 3 日	2022 年 11 月 15 日	2024 年 1 月 31 日	2025 年 1 月 31 日
1.23	2021 年 12 月 7 日	2022 年 8 月 11 日	2023 年 10 月 11 日	2024 年 10 月 11 日
1.22	2021 年 8 月 4 日	2022 年 4 月 4 日	2023 年 6 月 4 日	2024 年 9 月 1 日
1.21	2021 年 4 月 8 日	2021 年 7 月 19 日	2023 年 2 月 16 日	2024 年 7 月 15 日

Amazon EKS のバージョンのよくある質問

標準サポートで利用できる Kubernetes バージョンはいくつありますか?

Kubernetes の各バージョンに Kubernetes コミュニティが提供するサポートに沿って、Amazon EKS では、Kubernetes の 4 つ以上の実稼働対応バージョンの標準サポートを提供するように努めています。特定の Kubernetes マイナーバージョンの標準サポート終了日については、少なくとも 60 日前に発表します。新しい Kubernetes バージョンの Amazon EKS における認定およびリリースのプロセスにより、Amazon EKS での Kubernetes バージョンの標準サポート終了日は、Kubernetes プロジェクトがそのバージョンのアップストリームのサポートを停止した日以降になります。

Kubernetes は Amazon EKS による標準サポートをどのくらいの期間受けられますか?

Kubernetes バージョンは、Amazon EKS で最初に利用可能になってから 14 か月間標準サポートを受けていました。これは、Amazon EKS で利用可能なバージョンが、Kubernetes アップストリームでサポートされなくなった場合でも当てはまります。当社では、Amazon EKS でサポー

トされている Kubernetes バージョンに適用されるセキュリティパッチをバックポートしています。

Amazon EKS で Kubernetes バージョンの標準サポートが終了する際には通知が届きますか?

はい。アカウント内のいずれかのクラスターでサポートの終了が近い Kubernetes バージョンを実行している場合、そのバージョンが Amazon EKS でリリースされてから約 12 か月後に、Amazon EKS から AWS Health Dashboard を通じて通知が送信されます。通知にはサポート終了日が含まれます。これは、通知の日から 60 日以上後です。

Amazon EKS では、どの Kubernetes 機能がサポートされてますか?

Amazon EKS では、一般提供されている (GA) Kubernetes API のすべての機能がサポートされています。Kubernetes バージョン 1.24 以降、新しいベータ版 API はデフォルトではクラスターで有効になっていません。デフォルトでは、既存のベータ API と既存のベータ API の新しいバージョンは引き続き有効になっています。アルファ機能はサポートされません。

Amazon EKS のマネージド型ノードグループは、クラスターのコントロールプレーンのバージョンとともに自動的に更新されますか?

いいえ。マネージドノード型グループでは、アカウントに Amazon EC2 インスタンスを作成します。これらのインスタンスは、お客様または Amazon EKS がコントロールプレーンを更新しても、自動的にはアップグレードされません。詳細については、「マネージド型ノードグループの更新」を参照してください。コントロールプレーンとノードで、同じ Kubernetes バージョンを保持することをお勧めします。

セルフマネージド型ノードグループは、クラスターのコントロールプレーンのバージョンとともに自動的に更新されますか?

いいえ。セルフマネージド型ノードグループには、アカウント内の Amazon EC2 インスタンスが含まれています。これらのインスタンスは、お客様または代わりに Amazon EKS がコントロールプレーンのバージョンを更新しても、自動的にはアップグレードされません。セルフマネージド型ノードグループは、更新が必要であることをコンソールに表示しません。更新が必要なノードを確認するには、クラスターの [概要] タブにある [ノード] のリストからノードを選択して、そこにインストールされている `kubelet` バージョンを表示します。ノードは手動で更新する必要があります。詳細については、「セルフマネージド型ノードの更新」を参照してください。

Kubernetes プロジェクトでは、最大 3 つのマイナーバージョンに対して、コントロールプレーンとノード間の互換性がテストされます。例えば、`1.27` ノードは、`1.30` コントロールプレーンによってオーケストレーションされた場合も動作し続けます。しかし、コントロールプレーンの背後で 3 つのマイナーバージョンのノードを使用し続けながら、クラスターを実行することはお勧

めしません。詳細については、「Kubernetes ドキュメント」の「Kubernetes のバージョンおよびバージョンスキューのサポートポリシー」を参照してください。コントロールプレーンとノードで、同じ Kubernetes バージョンを保持することをお勧めします。

自動クラスターコントロールプレーンのバージョンに自動アップグレードされた Fargate で実行中の Pods も、アップグレードされますか?

いいえ。Fargate Pods は、Kubernetes デプロイメントなどのレプリケーションコントローラーの一部として実行することを強くお勧めします。その後、すべての Fargate Pods のローリング再起動を実行します。Fargate Pod の新しいバージョンは、更新されたクラスターのコントロールプレーンのバージョンと同じ `kubelet` バージョンでデプロイされます。詳細については、「Kubernetes ドキュメント」の「デプロイメント」を参照してください。

> **Important**
>
> コントロールプレーンを更新する場合は、引き続き Fargate ノードを自分で更新する必要があります。Fargate ノードを更新するには、ノードと対応した Fargate Pod を削除した上で、Pod を再デプロイします。新しい Pod は、クラスターと同じ `kubelet` バージョンでデプロイされます。

Amazon EKS 延長サポートに関するよくある質問

標準サポートと延長サポートという用語を初めて知りました。これらの用語はどういう意味ですか?

Kubernetes のバージョン 向けの Amazon EKS の標準サポートは、Amazon EKS での Kubernetes バージョンがリリースされた時点で開始され、リリース日の 14 か月後に終了します。Kubernetes バージョンの延長サポートは、標準サポートの終了後すぐに開始され、その 12 か月後に終了します。たとえば、バージョン `1.23` 向けの Amazon EKS 標準サポートは 2023 年 10 月 11 日に終了します。バージョン `1.23` の延長サポートは 2023 年 10 月 12 日に開始され、2024 年 10 月 11 日に終了します。

Amazon EKS クラスターの延長サポートを受けるには何をする必要がありますか?

Amazon EKS クラスターの延長サポートを受けるには、何もする必要がありません。標準サポートは、Kubernetes バージョンが Amazon EKS でリリースされた時点で開始され、リリース日の 14 か月後に終了します。Kubernetes バージョンの延長サポートは、標準サポートの終了後すぐに開始され、その 12 か月後に終了します。標準サポートが終了した Kubernetes バージョンで実行されているクラスターは、自動的に延長サポートにオンボードされます。

どの Kubernetes バージョンで延長サポートを受けることができますか?

延長サポートは Kubernetes バージョン 1.23 以降で利用できます。クラスターは、そのバージョンの標準サポートが終了してから最大 12 か月間、どのバージョンでも実行できます。これは、Amazon EKS では各バージョンが 26 か月間サポートされることを意味します (14 か月間標準サポートと 12 か月間延長サポート)。

延長サポートを利用したくない場合はどうすればいいですか?

クラスターを標準の Amazon EKS サポートの Kubernetes バージョンにアップグレードすると、延長サポートには自動的に登録されません。標準サポートの Kubernetes バージョンにアップグレードされていないクラスターは、自動的に延長サポートに入ります。

12 か月間の延長サポートが終了するとどうなりますか?

26 か月のライフサイクル (14 か月の標準サポートと 12 か月の延長サポート) が終了した Kubernetes バージョンで実行されているクラスターは、次のバージョンに自動的にアップグレードされます。

延長サポート終了日には、終了の対象となっているバージョンで新しい Amazon EKS クラスターを作成できなくなります。サポート終了日を過ぎると、既存のコントロールプレーンは、後の段階的なデプロイプロセスを通じて Amazon EKS によりサポートされている最も初期のバージョンに自動的に更新されます。コントロールプレーンの自動更新後は、クラスターアドオンと Amazon EC2 ノードを手動で更新してください。詳細については、「Amazon EKS クラスターに必要な Kubernetes バージョンを更新する」を参照してください。

延長サポート終了後にコントロールプレーンが自動的に更新されるのは、正確にはいつですか?

Amazon EKS での具体的なスケジュールは決まっていません。自動更新は、延長サポート終了日以降に任意のタイミングで実行される可能性があります。更新前には通知は届きません。Amazon EKS の自動更新プロセスに頼ることなく、事前にコントロールプレーンを更新することをお勧めします。詳細については、「Amazon EKS クラスターの Kubernetes バージョンの更新」を参照してください。

コントロールプレーンを 1 つの Kubernetes バージョンで無期限に保持することはできますか?

いいえ。AWS では、クラウドのセキュリティを最優先事項ととらえています。特定の時点 (通常は 1 年) を過ぎると、Kubernetes コミュニティは、一般的な脆弱性と露出 (CVE) パッチのリリースを停止し、サポートされていないバージョンの CVE 提出を非推奨とします。つまり、Kubernetes の古いバージョンに固有の脆弱性がある場合、報告すらされない可能性があります。クラスターは、脆弱性の発生時にも通知および修復オプションなしで公開され続けることに

なります。このため、Amazon EKS では、延長サポートが終了したバージョンでコントロールプレーンを維持することはできません。

延長サポートを受けるには追加料金がかかりますか?

はい。延長サポートで稼働している Amazon EKS クラスターに対して追加料金が発生します。料金の詳細については、AWS ブログの「Kubernetes バージョンの Amazon EKS 延長サポートの料金」を参照してください。

延長サポートには何が含まれますか?

延長サポートの Amazon EKS クラスターは、Kubernetes コントロールプレーンのセキュリティパッチを継続的に受信します。さらに、Amazon EKS は延長サポートバージョンに対する Amazon VPC CNI、`kube-proxy`、および CoreDNS アドオンのパッチをリリースします。また、Amazon EKS は、Amazon Linux、Bottlerocket および Windows 向けに AWS が公開した Amazon EKS に最適化された AMI のパッチと、それらのバージョンの Amazon EKS Fargate ノードのパッチもリリースします。延長サポートに該当するすべてのクラスターは、引き続き AWS からテクニカルサポートを受けることができます。

> **Note**
>
> AWS によって公開されている Amazon EKS 最適化 Windows AMI の延長サポートは、Kubernetes バージョン `1.23` では利用できませんが、Kubernetes バージョン `1.24` 以降では利用できます。

延長サポートの非 Kubernetes コンポーネント用のパッチには制限はありますか?

延長サポートは AWS から Kubernetes 固有のコンポーネントをすべて対象としていますが、すべての場合において、Amazon Linux、Bottlerocket および Windows 向けに AWS が公開した Amazon EKS の最適化された AMI のみがサポートされます。つまり、延長サポートを利用している間は、Amazon EKS に最適化された AMI には、より新しいコンポーネント (OS やカーネルなど) が搭載される場合があります。たとえば、Amazon Linux 2 のライフサイクルが 2025 年に終了すると、Amazon EKS に最適化された Amazon Linux AMI は、より新しい Amazon Linux OS を使用して構築されるようになります。Amazon EKS は、このようなサポートライフサイクルの重要な相違点を Kubernetes バージョンごとに発表し、文書化します。

延長サポートのバージョンを使用して新しいクラスターを作成できますか?

できます。ただし `1.22` と `1.21` を除きます。例えば、`1.23` クラスターは作成できますが、`1.22` クラスターは作成できません。

標準サポートバージョンのリリースノート

このトピックでは、標準サポートの各 Kubernetes バージョンで注意すべき重要な変更点を説明します。アップグレードするときは、クラスターの古いバージョンと新しいバージョン間で発生した変更を注意深く確認してください。

> **Note**
>
> 1.24 以降のクラスターでは、公式に公開された Amazon EKS AMI には、唯一のランタイムとして containerd が含まれています。1.24 よりも前の Kubernetes バージョンは、デフォルトのランタイムとして Docker を使用します。これらのバージョンには、containerd を使用してサポートされているクラスターでワークロードをテストできるブートストラップフラグオプションがあります。詳細については、「Amazon EKS は Dockershim のサポートを終了しました」を参照してください。

Kubernetes 1.30

Kubernetes 1.30 が Amazon EKS で利用可能になりました。Kubernetes 1.30 の詳細については、「公式リリースのお知らせ」を参照してください。

> **Important**
>
> - Amazon EKS バージョン 1.30 以降では、新しく作成されたマネージド型ノードグループは、ノードオペレーティングシステムとして自動的に Amazon Linux 2023 (AL2023) を使用するようデフォルトで設定されます。以前は、新しいノードグループはデフォルトで Amazon Linux 2 (AL2) を使用するよう設定されていました。新しいノードグループを作成するときに AL2 を AMI タイプとして選択すれば、AL2 を引き続き使用できます。
>   - Amazon Linux の詳細については、「Amazon Linux ユーザーガイド」の「Comparing AL2 and AL2023」を参照してください。
>   - マネージドノードグループのオペレーティングシステムの指定の詳細については、「マネージド型ノードグループの作成」を参照してください。

- Amazon EKS 1.30 では、topology.k8s.aws/zone-id ラベルがワーカーノードに追加されます。アベイラビリティーゾーン ID (AZ ID) を使用すると、アカウント間でリソースの場所を区

別できます。詳細については、「AWS RAM ユーザーガイド」の「Availability Zone IDs for your AWS resources」を参照してください。

- 1.30 以降、Amazon EKS は新しく作成されたクラスターに適用された gp2 StorageClass リソースに default 注釈を含めなくなりました。このストレージクラスを名前で参照する場合、これは影響しません。クラスターにデフォルトの StorageClass を設定する場合は、アクションを実行する必要があります。gp2 という名前で StorageClass を参照する必要があります。v1.31.0 または、aws-ebs-csi-driver add-on のインストール時に defaultStorageClass.enabled パラメータを true に設定することで、Amazon EBS が推奨するデフォルトのストレージクラスをデプロイすることもできます。
- Amazon EKS クラスター IAM ロールに必要な最小 IAM ポリシーが変更されました。アクション ec2:DescribeAvailabilityZones は必須です。詳細については、「Amazon EKS クラスターの IAM ロール」を参照してください。

Kubernetes 1.30 変更履歴の全文については、「https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.30.md」を参照してください。

Kubernetes 1.29

Kubernetes 1.29 が Amazon EKS で利用可能になりました。Kubernetes 1.29 の詳細については、「公式リリースのお知らせ」を参照してください。

> **Important**
>
> - 非推奨の FlowSchema の flowcontrol.apiserver.k8s.io/v1beta2 API バージョンと PriorityLevelConfiguration は、Kubernetes v1.29 で現在は提供されていません。非推奨のベータ API グループを使用するマニフェストまたはクライアントソフトウェアがある場合は、v1.29 にアップグレードする前にこれらを変更する必要があります。

- ノードオブジェクトの .status.kubeProxyVersion フィールドは非推奨となり、Kubernetes プロジェクトでは今後のリリースでそのフィールドを削除することが提案されています。非推奨のフィールドは正確ではなく、従来は kubelet によって管理されていましたが、実際には kube-proxy バージョンを認識できず、kube-proxy が実行されているかどうかも判別できませんでした。クライアントソフトウェアでこのフィールドを使用している場合は、停止してください。かかるフィールドの情報に信頼性はなく、現在は非推奨です。

- Kubernetes 1.29 で潜在的なアタックサーフェスを減らすため、LegacyServiceAccountTokenCleanUp 機能は、レガシーの自動生成されたシークレットベースのトークンを長期間 (デフォルトでは 1 年) 使用されていない場合は無効としてラベル付けし、無効としてマークされた後に長期間使用が試みられなかった場合は自動的に削除します (デフォルトではその後 1 年)。このようなトークンを識別するには、以下を実行してください。

```
kubectl get cm kube-apiserver-legacy-service-account-token-tracking -nkube-system
```

Kubernetes 1.29 変更履歴の全文については、「https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.29.md#changelog-since-v1280」を参照してください。

Kubernetes 1.28

Kubernetes 1.28 が Amazon EKS で利用可能になりました。Kubernetes 1.28 の詳細については、「公式リリースのお知らせ」を参照してください。

- Kubernetes v1.28 により、n-2 から n-3 へ、1 つのマイナーバージョンごとのコアノードとコントロールプレーンコンポーネントの間でサポートされるスキューが拡張されました。これにより、サポートされている最も古いマイナーバージョンのノードコンポーネント (kubelet および kube-proxy) が、サポートされている最新のマイナーバージョンのコントロールプレーンコンポーネント (kube-apiserver、kube-scheduler、kube-controller-manager、cloud-controller-manager) と連携できるようになりました。
- Pod GC Controller にあるメトリクス force_delete_pods_total および force_delete_pod_errors_total は、ポッドの強制削除をすべて考慮するよう拡張されました。ポッドが強制的に削除される理由が、ポッドが終了しているのか、孤立しているのか、サービス外テイントで終了しているのか、あるいは終了してスケジュールされていないのかを示す理由がメトリックに追加されます。
- PersistentVolume (PV) コントローラーは、storageClassName が設定されておらずバインドされていない PersistentVolumeClaim にデフォルトの StorageClass を自動的に割り当てるように変更されました。さらに、API サーバー内の PersistentVolumeClaim アドミッション検証メカニズムが調整され、値が未設定の状態から実際の StorageClass 名に変更できるようになりました。

Kubernetes 1.28 変更履歴の全文については、「https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.28.md#changelog-since-v1270」を参照してください。

Kubernetes 1.27

Kubernetes 1.27 が Amazon EKS で利用可能になりました。Kubernetes 1.27 の詳細については、「公式リリースのお知らせ」を参照してください。

Important

- アルファ seccomp アノテーション seccomp.security.alpha.kubernetes.io/pod と container.seccomp.security.alpha.kubernetes.io アノテーションのサポートは削除されました。アルファ seccomp アノテーションは 1.19 で非推奨となり、1.27 で削除されたことで、seccomp アノテーションを持つ Pods に seccomp フィールドが自動的に入力されなくなります。代わりに、Pods またはコンテナの securityContext.seccompProfile フィールドを使用して seccomp プロファイルを構成してください。クラスターで非推奨のアルファ seccomp アノテーションを使用しているかどうかを確認するには、次のコマンドを実行します。

  ```
  kubectl get pods --all-namespaces -o json | grep
   -E 'seccomp.security.alpha.kubernetes.io/pod|
  container.seccomp.security.alpha.kubernetes.io'
  ```

- kubelet の --container-runtime コマンドライン引数が削除されました。1.24 以降、Amazon EKS のデフォルトのコンテナランタイムが containerd となっており、これによりコンテナランタイムを指定する必要がなくなりました。1.27 以降、Amazon EKS は、ブートストラップスクリプトに渡された --container-runtime 引数を無視します。ノードのブートストラッププロセス中のエラーを防ぐために、この引数を --kubelet-extra-args に渡さないことが重要です。すべてのノード作成ワークフローとビルドスクリプトから --container-runtime 引数を削除する必要があります。

- Kubernetes 1.27 の kubelet は、デフォルトの kubeAPIQPS を 50 に、kubeAPIBurst を 100 に増やしました。これらの機能強化により、kubelet がより大量の API クエリを処理できるようになり、応答時間とパフォーマンスが向上します。　スケーリング要件により、Pods に対する需要が増加した場合、修正されたデフォルト値により、kubelet は増加したワークロードを効率的に管理できるようになります。その結果、Pod の起動が速くなり、クラスター操作がより効率的になります。
- よりきめ細かい Pod トポロジを使用して、minDomain などのポリシーを分散できます。このパラメータにより、Pods を分散させる必要のあるドメインの最小数を指定できま

す。nodeAffinityPolicy および nodeTaintPolicy を使用することで、Pod の分散を管理する際の粒度をさらに細かくすることができます。これは、ノードのアフィニティ、テイント、Pod's 仕様の topologySpreadConstraints の matchLabelKeys フィールドによって異なります。これにより、ローリングアップグレード後の分散計算のために Pods を選択できます。

- Kubernetes1.27 は PersistentVolumeClaims (PVCs) の存続期間を制御する StatefulSets の新しいポリシーメカニズムをベータ版に昇格しました。新しい PVC リテンションポリシーでは、StatefulSet が削除されたとき、または StatefulSet 内のレプリカがスケールダウンされたときに、StatefulSet スペックテンプレートから生成された PVCs を自動的に削除するか、保持するかを指定できます。
- Kubernetes API サーバーの goaway-chance オプションを使用すると、接続がランダムに閉じることで、HTTP/2 クライアント接続が単一の API サーバーインスタンスでスタックされることを防ぐことができます。接続が閉じると、クライアントは再接続を試み、負荷分散の結果として別の API サーバーにアクセスする可能性があります。Amazon EKS バージョン 1.27 では、goaway-chance フラグが有効になっています。Amazon EKS クラスターで実行されているワークロードで HTTP GOAWAY と互換性のないクライアントが使用されている場合は、接続終了時に再接続することで GOAWAY を処理できるようにクライアントを更新することをお勧めします。

Kubernetes 1.27 変更履歴の全文については、「https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.27.md#changelog-since-v1260」を参照してください。

Kubernetes 1.26

Kubernetes 1.26 が Amazon EKS で利用可能になりました。Kubernetes 1.26 の詳細については、「公式リリースのお知らせ」を参照してください。

> **Important**
>
> Kubernetes 1.26 では、CRI v1alpha2 のサポートは終了しています。この結果、コンテナランタイムが CRI v1 をサポートしていない場合、kubelet はノードを登録しなくなります。これは、Kubernetes 1.26 は containerd のマイナーバージョン 1.5 以前のバージョンをサポートされていないことも意味します。containerd を使用している場合は、ノードを Kubernetes 1.26 にアップグレードする前に containerd バージョン 1.6.0 以降にアップグレードする必要があります。また、v1alpha2 のみをサポートする他のコンテナランタイムもアップグレードする必要があります。詳細については、コンテナランタイムベンダーにお問い合わせください。デフォルトでは、Amazon Linux および Bottlerocket AMI には containerd バージョン 1.6.6 が含まれます。

- Kubernetes 1.26 にアップグレードする前に、Amazon VPC CNI plugin for Kubernetes をバージョン 1.12 以降にアップグレードしてください。Amazon VPC CNI plugin for Kubernetes バージョン 1.12 またはそれ以降にアップグレードしていない場合、Amazon VPC CNI plugin for Kubernetes はクラッシュします。詳細については、「Amazon VPC CNI plugin for Kubernetes Amazon EKS アドオンの使用」を参照してください。
- Kubernetes API サーバーの goaway-chance オプションを使用すると、接続がランダムに閉じることで、HTTP/2 クライアント接続が単一の API サーバーインスタンスでスタックされることを防ぐことができます。接続が閉じると、クライアントは再接続を試み、負荷分散の結果として別の API サーバーにアクセスする可能性があります。Amazon EKS バージョン 1.26 では、goaway-chance フラグが有効になっています。Amazon EKS クラスターで実行されているワークロードで HTTP GOAWAY と互換性のないクライアントが使用されている場合は、接続終了時に再接続することで GOAWAY を処理できるようにクライアントを更新することをお勧めします。

Kubernetes 1.26 変更履歴の全文については、「https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.26.md#changelog-since-v1250」を参照してください。

延長サポートバージョンのリリースノート

このトピックでは、延長サポートにおいて各 Kubernetes バージョンで注意すべき重要な変更点を説明します。アップグレードするときは、クラスターの古いバージョンと新しいバージョン間で発生した変更を注意深く確認してください。

Kubernetes 1.25

Kubernetes 1.25 が Amazon EKS で利用可能になりました。Kubernetes 1.25 の詳細については、「公式リリースのお知らせ」を参照してください。

> **Important**
>
> - Kubernetes バージョン 1.25 以降、Amazon EC2 P2 インスタンスを Amazon EKS に最適化された高速 Amazon Linux AMI で使用するには、追加設定が必要です。これらの Kubernetes バージョン 1.25 以降向けの AMI は NVIDIA 525 シリーズまたはそれ以降のドライバーに対応しており、P2 インスタンスとの互換性がありません。NVIDIA 525 シリーズおよびそれ以降のドライバーは P3、P4、P5 インスタンスとの互換性があるため、これらのインスタンスを Kubernetes バージョン 1.25 またはそれ以降の AMI で使用することができます。Amazon EKS クラスターをバージョン 1.25 にアップグレードする前に、P2 インスタンスを P3、P4、P5 インスタンスに移行します。また、NVIDIA

525 シリーズまたはそれ以降と連携するように、アプリケーションを自発的にアップグレードする必要があります。新しい NVIDIA 525 シリーズ以降のドライバーは、2024 年 1 月後半に Kubernetes バージョン 1.23 および 1.24 にバックポートする予定です。

- PodSecurityPolicy (PSP) は Kubernetes 1.25 で削除されます。PSPs は、Pod Security Admission (PSA) および Pod Security Standards (PSS) に置き換えられます。PSA は、PSS で概説されているセキュリティコントロールを実装する組み込みのアドミッションコントローラーです。PSA および PSS は Kubernetes 1.25 で安定状態に移行し、Amazon EKS ではデフォルトで有効になります。クラスター内に PSPs がある場合は、クラスターをバージョン 1.25 にアップグレードする前に、PSP から組み込みの Kubernetes PSS または Policy as Code ソリューションに移行してください。PSP から移行しない場合、ワークロードが中断される可能性があります。詳細については、「ポッドセキュリティポリシー (PSP) の削除に関するよくある質問」を参照してください。
- Kubernetes バージョン 1.25 には、API の優先度と公平性 (APF) として知られる、既存の機能の動作が変わる変更が含まれています。APF は、リクエスト量が増加しているときに発生する可能性のある過負荷から API サーバーを保護する役割を果たします。保護は、所定の時点で処理できる同時リクエストの数に制限を設けることで行なわれます。この処理は、さまざまなワークロードやユーザーからのリクエストに明確な優先レベルや制限を適用することで実施されます。このアプローチにより、重要なアプリケーションや優先度の高いリクエストを優先的に扱いながら、優先度の低いリクエストが API サーバーを圧迫するのを防ぐことができます。詳しくは、Kubernetes ドキュメントの「API の優先順位と公平性」または EKS ベストプラクティスガイドの「API の優先順位と公平性」を参照してください。

  これらのアップデートは PR #10352 および PR #118601 で導入されました。以前は、APF はすべてのタイプのリクエストを一様に処理し、各リクエストが同時リクエスト制限の 1 単位を消費していました。APF の動作が変更されたことで、API サーバーに非常に大きな負荷がかかる要因となる LIST リクエストにより多くの同時実行単位が割り当てられるようになりました。API サーバーは、LIST リクエストによって返されるオブジェクトの数を推定します。返されるオブジェクトの数に比例した同時実行単位を割り当てます。

  Amazon EKS バージョン 1.25 以降にアップグレードすると、このアップデートされた動作により、多量の LIST リクエストを持つ (以前は問題なく機能していた) ワークロードに、レート制限が発生する可能性があります。これは HTTP 429 のレスポンスコードとして表示されます。LIST リクエストへのレート制限により、ワークロードの中断が発生する可能性を回避するため、ワークロードを再構築してこれらのリクエストのレートを減らすことを強くお勧めします。または、APF 設定を調整して、重要なリクエストにより多く

の容量を割り当てて、重要でないリクエストに割り当てる容量を減らすことで、この問題を解決することもできます。これらの緩和方法の詳細については、EKS ベストプラクティスガイドの「リクエストがドロップされるのを防ぐには」を参照してください。

- Amazon EKS 1.25 には、更新された YAML ライブラリを含むクラスター認証の機能強化が含まれています。kube-system 名前空間で見つかった aws-auth ConfigMap の YAML の値がマクロで始まり、最初の文字が中括弧である場合、中括弧 ({ }) の前後に引用符 (“ ”) を追加する必要があります。これは、aws-iam-authenticator バージョン v0.6.3 によって Amazon EKS 1.25 で aws-auth ConfigMap が正確に解析されるようにするために必要です。
- EndpointSlice のベータ API バージョン (discovery.k8s.io/v1beta1) は Kubernetes 1.21 で非推奨となっており、Kubernetes 1.25 現在は提供されていません。この API は discovery.k8s.io/v1 に更新されました。詳細については、Kubernetes ドキュメントの「EndpointSlice」を参照してください。AWS Load Balancer Controller v2.4.6 以前は、v1beta1 エンドポイントを使用して EndpointSlices と通信していました。AWS Load Balancer Controller のために EndpointSlices の設定を使用している場合は、Amazon EKS クラスターを 1.25 にアップグレードする前に AWS Load Balancer Controller v2.4.7 にアップグレードする必要があります。AWS Load Balancer Controller のために EndpointSlices の設定を使用しているときに 1.25 にアップグレードすると、コントローラーがクラッシュし、ワークロードが中断されます。コントローラーをアップグレードするには、「AWS Load Balancer Controller とは」を参照してください。

- SeccompDefault は、Kubernetes 1.25 でベータ版に昇格しました。kubelet を設定するときに --seccomp-default フラグを設定すると、コンテナランタイムは unconfined (seccomp disabled) モードではなく、その RuntimeDefaultseccomp プロファイルを使用します。デフォルトプロファイルは、ワークロードの機能を維持しながら、セキュリティデフォルトの強力なセットを提供します。このフラグは利用可能ですが、Amazon EKS はデフォルトでこのフラグを有効にしないため、Amazon EKS の動作は事実上変更されていません。必要に応じて、ノードでこれを有効にすることができます。詳細については、Kubernetes ドキュメントのチュートリアル「seccomp を使用してコンテナの Syscall を制限する」を参照してください。
- Docker (Dockershim とも呼ばれる) のコンテナランタイムインターフェイス (CRI) のサポートは、Kubernetes 1.24 以降から削除されました。Kubernetes 1.24 以降のクラスター用の Amazon EKS の公式 AMIs における唯一のコンテナランタイムは containerd です。Amazon EKS 1.24 以降に更新する前に、サポートされなくなったブートストラップスクリプトフラグへの参照をすべて削除する必要があります。詳細については、「Amazon EKS は Dockershim のサポートを終了しました」を参照してください。

- ワイルドカードクエリのサポートは CoreDNS 1.8.7 で非推奨となり、CoreDNS 1.9 で削除されました。これはセキュリティ対策として行われました。ワイルドカードクエリは機能しなくなり、IP アドレスの代わりに NXDOMAIN を返します。
- Kubernetes API サーバーの goaway-chance オプションを使用すると、接続がランダムに閉じることで、HTTP/2 クライアント接続が単一の API サーバーインスタンスでスタックされることを防ぐことができます。接続が閉じると、クライアントは再接続を試み、負荷分散の結果として別の API サーバーにアクセスする可能性があります。Amazon EKS バージョン 1.25 では、goaway-chance フラグが有効になっています。Amazon EKS クラスターで実行されているワークロードで HTTP GOAWAY と互換性のないクライアントが使用されている場合は、接続終了時に再接続することで GOAWAY を処理できるようにクライアントを更新することをお勧めします。

Kubernetes 1.25 変更履歴の全文については、「https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.25.md#changelog-since-v1240」を参照してください。

Kubernetes 1.24

Kubernetes 1.24 が Amazon EKS で利用可能になりました。Kubernetes 1.24 の詳細については、「公式リリースのお知らせ」を参照してください。

> **Important**
>
> - Kubernetes 1.24 以降、新しいベータ版 API はデフォルトではクラスターで有効になっていません。デフォルトでは、既存のベータ API と既存のベータ API の新しいバージョンは引き続き有効になっています。Amazon EKS はアップストリーム Kubernetes 1.24 と同じ動作をします。新規および既存の両方の API 操作の新機能を制御する機能ゲートは、デフォルトで有効になっています。これはアップストリーム Kubernetes と一致しています。詳細については、「GitHub」の「KEP-3136: ベータ API はデフォルトでオフになっている」を参照してください。
> - Docker (Dockershim とも呼ばれる) のコンテナランタイムインターフェイス (CRI) のサポートは、Kubernetes 1.24 から削除されました。Amazon EKS 公式 AMI には唯一のランタイムとして containerd があります。Amazon EKS 1.24 以降に移行する前に、サポートされなくなったブートストラップスクリプトフラグへの参照をすべて削除する必要があります。また、ワーカーノードで IP 転送が有効になっていることを確認してください。詳細については、「Amazon EKS は Dockershim のサポートを終了しました」を参照してください。

- Fluentd をすでに Container Insights 用に構成している場合、クラスターを更新する前に Fluentd を Fluent Bit に移行する必要があります。Fluentd パーサーは JSON 形式のログメッセージのみを解析するように構成されています。`dockerd` とは異なり、`containerd` コンテナランタイムには JSON 形式ではないログメッセージがあります。Fluent Bit に移行しないと、構成された Fluentd's パーサーの一部が Fluentd コンテナー内で大量のエラーを生成します。移行の詳細については、「CloudWatch Logs へログを送信する DaemonSet として Fluent Bit を設定する」を参照してください。
- Kubernetes `1.23` 以前のバージョンでは、検証不能な IP と DNS サブジェクト代替名 (SAN) を含む `kubelet` サービング証明書は、検証不能な SAN とともに自動的に発行されます。これらの検証不能な SAN は、プロビジョニングされた証明書から除外されます。バージョン `1.24` 以降のクラスターでは、SAN を検証できない場合、`kubelet` サービング証明書は発行されません。これにより、`kubectl` exec コマンドと `kubectl` ログコマンドが機能しなくなります。詳細については、「クラスターをKubernetes 1.24 にアップグレードする前の証明書署名に関する考慮事項」を参照してください。
- Fluent Bit を使用する Amazon EKS `1.23` クラスターをアップグレードする場合は、`k8s/1.3.12` 以降のものが実行中であることを確認する必要があります。これを行うには、GitHub から最新の該当する Fluent Bit YAML ファイルを再適用します。詳細については、「Amazon CloudWatch ユーザーガイド」の「Fluent Bit のセットアップ」を参照してください。

- Topology Aware Hints を使用すると、クラスターワーカーノードが複数のアベイラビリティーゾーンにデプロイされている場合に、トラフィックをゾーン内に保持する設定を指定できます。ゾーン内でトラフィックをルーティングすると、コストを削減し、ネットワークパフォーマンスを向上させることができます。デフォルトでは、Topology Aware Hints は Amazon EKS `1.24` で有効になっています。詳細については、「Kubernetes ドキュメント」の「トポロジー対応のヒント」を参照してください。
- `PodSecurityPolicy` (PSP) は、Kubernetes `1.25` で削除される予定です。PSPs は、ポッドセキュリティアドミッション (PSA) に置き換えられています。PSA は、ポッドセキュリティ標準 (PSS) で概説されているセキュリティコントロールを使用するビルトインのアドミッションコントローラーです。PSA と PSS はどちらもベータ機能であり、Amazon EKS ではデフォルトで有効になっています。バージョン `1.25` での PSP の削除に対処するには、Amazon EKS に PSS を実装することをお勧めします。詳細については、「AWS ブログ」の「Amazon EKS でのポッドセキュリティ標準の実装」を参照してください。

- `client.authentication.k8s.io/v1alpha1` ExecCredential は、Kubernetes `1.24` で削除されます。ExecCredential API は、Kubernetes `1.22` で一般に利用できるようになりました。`v1alpha1` API に依存する client-go 認証情報プラグインを使用する場合は、`v1` API への移行方法について、プラグインのディストリビューターにお問い合わせください。
- Kubernetes `1.24` については、アップストリームの Cluster Autoscaler プロジェクトに機能を提供しました。この機能は、Amazon EKS のマネージド型ノードグループのゼロノードへのスケーリングとノードからのスケーリングを簡素化します。以前は、Cluster Autoscaler がゼロノードにスケーリングされたマネージド型ノードグループのリソース、ラベル、テイントを理解するには、基盤となる Amazon EC2 Auto Scaling グループに、それが担当するノードの詳細情報をタグ付けする必要がありました。現在、マネージド型ノードグループに実行中のノードがない場合、Cluster Autoscaler は Amazon EKS `DescribeNodegroup` API 操作を呼び出します。この API オペレーションは、Cluster Autoscaler がマネージド型ノードグループのリソース、ラベル、テイントについて必要とする情報を提供します。この機能を使用するには、Cluster Autoscaler サービスアカウントの IAM ポリシーに `eks:DescribeNodegroup` アクセス許可を追加する必要があります。Amazon EKS マネージド型ノードグループを強化する Auto Scaling グループの Cluster Autoscaler タグの値がノードグループ自体と競合する場合、Cluster Autoscaler は Auto Scaling グループタグの値を使用します。これは、必要に応じて値をオーバーライドできるようにするためです。詳細については、「Autoscaling」を参照してください。
- Amazon EKS `1.24` で Inferentia または Trainium インスタンスタイプを使用する場合は、AWS Neuron デバイスプラグインバージョン 1.9.3.0 以降にアップグレードする必要があります。詳細については、「AWS Neuron ドキュメント」の「Neuron K8 リリース [1.9.3.0]」を参照してください。
- `Containerd` では、`IPv6` が Pods 用にデフォルトで有効になっています。これは、ノードのカーネル設定を Pod ネットワークの名前空間に適用します。このため、Pod 内のコンテナは `IPv4` (`127.0.0.1`) と `IPv6` (`::1`) の両方のループバックアドレスにバインドされます。`IPv6` はデフォルトの通信用プロトコルです。クラスターをバージョン `1.24` に更新する前に、マルチコンテナ Pods をテストすることをお勧めします。ループバックインターフェイスのすべての IP アドレスにバインドできるようにアプリを変更します。ライブラリの大部分は `IPv6` バインディングを有効にします。これは `IPv4` との下位互換性を備えています。アプリケーションコードを変更できない場合、次の 2 つのオプションがあります。
  - `init` コンテナを実行し、`disable ipv6` を `true` (`sysctl -w net.ipv6.conf.all.disable_ipv6=1`) に設定します。
  - アプリケーション Pods と一緒に `init` コンテナを挿入するように MutatingAdmissionWebhook を設定します。

すべてのノードですべての Pods 用に IPv6 をブロックする必要がある場合、インスタンスで IPv6 を無効にしなければならない場合があります。

- Kubernetes API サーバーの goaway-chance オプションを使用すると、接続がランダムに閉じることで、HTTP/2 クライアント接続が単一の API サーバーインスタンスでスタックされることを防ぐことができます。接続が閉じると、クライアントは再接続を試み、負荷分散の結果として別の API サーバーにアクセスする可能性があります。Amazon EKS バージョン 1.24 では、goaway-chance フラグが有効になっています。Amazon EKS クラスターで実行されているワークロードで HTTP GOAWAY と互換性のないクライアントが使用されている場合は、接続終了時に再接続することで GOAWAY を処理できるようにクライアントを更新することをお勧めします。

Kubernetes 1.24 変更履歴の全文については、「https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.24.md#changelog-since-v1230」を参照してください。

Kubernetes 1.23

Kubernetes 1.23 が Amazon EKS で利用可能になりました。Kubernetes 1.23 の詳細については、「公式リリースのお知らせ」を参照してください。

Important

- Kubernetes ツリー内からコンテナストレージインターフェイス (CSI) ボリュームへの移行機能が有効になっています。この機能により、Amazon EBS の既存の Kubernetes ツリー内ストレージプラグインを、対応する Amazon EBS CSI ドライバーに置き換えることができます。詳細については、Kubernetes ブログの「Kubernetes 1.17 機能: Kubernetes ツリー内から CSI ボリュームへの移行機能がベータ版に移行」を参照してください。

  この機能は、ツリー内 API を同等の CSI API に変換し、代替 CSI ドライバーにオペレーションを委任します。この機能を使用すると、これらのワークロードに属する既存の StorageClass、PersistentVolume、および PersistentVolumeClaim オブジェクトを使用する場合、顕著な変化はほとんどない可能性が高いです。この機能により、Kubernetes は、すべてのストレージ管理オペレーションをツリー内プラグインから CSI ドライバーに委任できます。既存のクラスターで Amazon EBS ボリュームを使用する場合は、クラスターをバージョン 1.23 に更新する前に、クラスターで Amazon EBS CSI ドライバーをインストールします。既存のクラスターを更新する前にドライバーをインストールしないと、ワークロードが中断される可能性があります。Amazon EBS ボリュームを使用するワークロードを新しい 1.23 クラスターでデプロイする場合は、クラスターに

ワークロードをデプロイする前に、クラスターに Amazon EBS CSI ドライバーをインストールします。クラスターに Amazon EBS CSI ドライバーをインストールする方法については、「Amazon EBS CSI ドライバー」を参照してください。移行機能に関するよくある質問については、「Amazon EBS CSI 移行に関するよくある質問」を参照してください。

- AWS によって公開されている Amazon EKS 最適化 Windows AMI の延長サポートは、Kubernetes バージョン `1.23` では利用できませんが、Kubernetes バージョン `1.24` 以降では利用できます。

- Kubernetes はバージョン `1.20` での `dockershim` のサポートを停止し、バージョン `1.24` で `dockershim` を削除しました。詳細については、Kubernetes ブログの「Kubernetes は Dockershim から以降: コミットメントと次のステップ」を参照してください。Amazon EKS は、Amazon EKS バージョン `1.24` 以降、`dockershim` のサポートを終了します。Amazon EKS バージョン `1.24` から、Amazon EKS 公式 AMI は唯一のランタイムとして `containerd` を備えることになります。

  Amazon EKS バージョン `1.23` は引き続き `dockershim` をサポートしますが、今すぐアプリケーションのテストを開始して、Docker 依存関係を特定して削除することをお勧めします。これにより、クラスターをバージョン `1.24` に更新する準備が整います。`dockershim` の削除の詳細については、「Amazon EKS は `Dockershim` のサポートを終了しました」を参照してください。
- Kubernetes は、Pods、サービス、およびノード向けに `IPv4/IPv6` デュアルスタックネットワーキングの一般提供を開始しました。ただし、Amazon EKS と Amazon VPC CNI plugin for Kubernetes では、デュアルスタックネットワークをサポートしていません。クラスターは `IPv4` または `IPv6` アドレスを Pods およびサービスに割り当てることができますが、両方のアドレスタイプを割り当てることはできません。
- Kubernetes は、ポッドセキュリティアドミッション (PSA) 機能のベータ版の提供を開始しました。この機能は、デフォルトで有効になっています。詳細については、Kubernetes のドキュメントの「Pod Security Admission」を参照してください。PSA は、ポッドセキュリティポリシー (PSP) アドミッションコントローラーを置き換えます。PSP アドミッションコントローラーはサポートされておらず、Kubernetes バージョン `1.25` で削除される予定です。

  PSP アドミッションコントローラーは、適用レベルを設定する特定の名前空間ラベルに基づいて、名前空間の Pods に対して Pod セキュリティ標準を適用します。詳細については、「Amazon EKS のベストプラクティスガイド」の「Pod Security Standards (PSS) and Pod Security Admission (PSA)」(Pod Security Standards (PSS) および Pod Security Admission (PSA)) を参照してください。

- クラスターでデプロイされる kube-proxy イメージは、Amazon EKS Distro (EKS-D) によって維持される最小基本イメージとなりました。イメージには最小限のパッケージが含まれており、シェルやパッケージマネージャーは含まれていません。
- Kubernetes は、エフェメラルコンテナのベータ版の提供を開始しました。エフェメラルコンテナは、既存の Pod と同じ名前空間で実行される一時的なコンテナです。これらを使用して、トラブルシューティングやデバッグの目的で Pods およびコンテナの状態を観察できます。これは、コンテナがクラッシュしたか、コンテナイメージにデバッグユーティリティが含まれていないことを理由として kubectl exec が不十分である場合に、インタラクティブなトラブルシューティングに特に役立ちます。デバッグユーティリティを含むコンテナの例は、distroless イメージです。詳細については、Kubernetes ドキュメントの「Debugging with an ephemeral debug container」(エフェメラルデバッグコンテナを使用したデバッグ) を参照してください。
- Kubernetes は、HorizontalPodAutoscaler autoscaling/v2 の安定した API の一般提供を開始しました。HorizontalPodAutoscaler autoscaling/v2beta2 API は非推奨です。1.26 ではご利用いただけなくなります。
- Kubernetes API サーバーの goaway-chance オプションを使用すると、接続がランダムに閉じることで、HTTP/2 クライアント接続が単一の API サーバーインスタンスでスタックされることを防ぐことができます。接続が閉じると、クライアントは再接続を試み、負荷分散の結果として別の API サーバーにアクセスする可能性があります。Amazon EKS バージョン 1.23 では、goaway-chance フラグが有効になっています。Amazon EKS クラスターで実行されているワークロードで HTTP GOAWAY と互換性のないクライアントが使用されている場合は、接続終了時に再接続することで GOAWAY を処理できるようにクライアントを更新することをお勧めします。

Kubernetes 1.23 変更履歴の全文については、「https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.23.md#changelog-since-v1220」を参照してください。

バージョン 1.21 および 1.22 のリリースノート

> **Important**
>
> これらのバージョンでは、新しいクラスターを作成することはできません。

このトピックでは、バージョン 1.22 と 1.21 で注意すべき重要な変更点について解説します。アップグレードするときは、クラスターの古いバージョンと新しいバージョン間で発生した変更を注意深く確認してください。

Kubernetes バージョン 1.22

次のアドミッションコントローラーは、すべての 1.22 プラットフォームバージョンで有効です: DefaultStorageClass、DefaultTolerationSeconds、LimitRanger、MutatingAdmissionWebh および DefaultIngressClass。

Kubernetes バージョン	Amazon EKS プラットフォームバージョン	リリースノート	リリース日
1.22.17	eks.28	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 5 月 16 日
1.22.17	eks.26	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 4 月 1 日
1.22.17	eks.14	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 6 月 30 日
1.22.17	eks.13	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 6 月 9 日
1.22.17	eks.12	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 5 月 5 日
1.22.17	eks.11	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 3 月 24 日
1.22.16	eks.10	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 1 月 27 日

Kubernetes バージョン	Amazon EKS プラットフォームバージョン	リリースノート	リリース日
1.22.15	eks.9	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2022 年 12 月 5 日
1.22.15	eks.8	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2022 年 11 月 18 日
1.22.15	eks.7	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2022 年 11 月 7 日
1.22.13	eks.6	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2022 年 9 月 21 日
1.22.10	eks.5	etcd 耐障害性が改善された新しいプラットフォームバージョン。	2022 年 8 月 15 日
1.22.10	eks.4	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。このプラットフォームバージョンでは、すべてのワーカーノードに aws:eks:cluster-name でタグを付ける新しいタグ付けコントローラーも導入され、これらのワーカーノードにコストを簡単に割り当てることができます。詳細については、「請求用のリソースにタグを付ける」を参照してください。	2022 年 7 月 21 日
1.22.10	eks.3	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2022 年 7 月 7 日

Kubernetes バージョン	Amazon EKS プラットフォームバージョン	リリースノート	リリース日
1.22.9	eks.2	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2022 年 5 月 31 日
1.22.6	eks.1	Amazon EKS 向け Kubernetes バージョン 1.22 の初回リリース。	2022 年 4 月 4 日

Kubernetes バージョン **1.21**

次のアドミッションコントローラーは、すべての 1.21 プラットフォームバージョンで有効です: DefaultStorageClass、DefaultTolerationSeconds、LimitRanger、MutatingAdmissionWebh および DefaultIngressClass。

Kubernetes バージョン	Amazon EKS プラットフォームバージョン	リリースノート	リリース日
1.21.14	eks.33	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 5 月 16 日
1.21.14	eks.31	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 4 月 1 日
1.21.14	eks.18	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 6 月 9 日
1.21.14	eks.17	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 5 月 5 日

Kubernetes バージョン	Amazon EKS プラットフォームバージョン	リリースノート	リリース日
1.21.14	eks.16	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 3 月 24 日
1.21.14	eks.15	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 1 月 27 日
1.21.14	eks.14	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2022 年 12 月 5 日
1.21.14	eks.13	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2022 年 11 月 18 日
1.21.14	eks.12	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2022 年 11 月 7 日
1.21.13	eks.11	etcd 耐障害性が改善された新しいプラットフォームバージョン。	2022 年 10 月 10 日
1.21.13	eks.10	etcd 耐障害性が改善された新しいプラットフォームバージョン。	2022 年 8 月 15 日

Kubernetes バージョン	Amazon EKS プラットフォームバージョン	リリースノート	リリース日
1.21.13	eks.9	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。このプラットフォームバージョンでは、すべてのワーカーノードに aws:eks:cluster-name でタグを付ける新しいタグ付けコントローラーも導入され、これらのワーカーノードにコストを簡単に割り当てることができます。詳細については、「請求用のリソースにタグを付ける」を参照してください。	2022 年 7 月 21 日
1.21.13	eks.8	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2022 年 7 月 7 日
1.21.12	eks.7	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2022 年 5 月 31 日

Kubernetes バージョン	Amazon EKS プラットフォームバージョン	リリースノート	リリース日
1.21.9	eks.6	AWS Security Token Service エンドポイントは、以前のプラットフォームバージョンのグローバルエンドポイントに復元されます。サービスアカウントの IAM ロールを使用する際にリージョンのエンドポイントを使用する場合は、そのエンドポイントを有効にする必要があります。リージョンのエンドポイントを有効にする方法の手順については、「サービスアカウントの AWS Security Token Service エンドポイントを設定する」を参照してください。	2022 年 4 月 8 日
1.21.5	eks.5	サービスアカウントの IAM ロール を使用する際に、グローバルエンドポイントではなく AWS Security Token Service リージョンのエンドポイントがデフォルトで使用されるようになりました。ただし、この変更は、eks.6 のグローバルエンドポイントに復元されます。 更新された Fargate スケジューラは、大規模なデプロイ時にノードを非常に高いレートでプロビジョニングします。	2022 年 3 月 10 日

Kubernetes バージョン	Amazon EKS プラットフォームバージョン	リリースノート	リリース日
1.21.5	eks.4	Amazon VPC CNI セルフマネージドおよび Amazon EKS アドオンのバージョン 1.10.1-eksbuild.1 が、デプロイされるデフォルトのバージョンになりました。	2021 年 12 月 13 日
1.21.2	eks.3	Kubernetes コントロールプレーンで実行されている VPC リソースコントローラーで Windows IPv4 アドレス管理をサポートする、新しいプラットフォームバージョンです。Fargate の Fluent Bit を口グするための Kubernetes フィルターディレクティブを追加しました。	2021 年 11 月 8 日
1.21.2	eks.2	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2021 年 9 月 17 日
1.21.2	eks.1	Amazon EKS 向け Kubernetes バージョン 1.21 の初回リリース。	2021 年 7 月 19 日

Amazon EKS のプラットフォームバージョン

Amazon EKS プラットフォームのバージョンは、Amazon EKS クラスターコントロールプレーンの各機能 (有効な Kubernetes API サーバーのフラグや、現在の Kubernetes パッチバージョンなど) を表しています。Kubernetes マイナーバージョンにはそれぞれ、Amazon EKS プラットフォームのバージョンが 1 つ以上関連付けられています。別の Kubernetes マイナーバージョンのプラットフォームバージョンは独立しています。クラスターの現在のプラットフォームバージョンを取得す

るには、AWS CLI または AWS Management Consoleを使用します。AWS Outposts にローカルクラスターがある場合、このトピックの代わりに「Amazon EKS ローカルクラスタープラットフォームのバージョン」を参照してください。

新しい Kubernetes マイナーバージョン (1.30 など) が Amazon EKS で利用可能になると、その Kubernetes マイナーバージョンの初期 Amazon EKS プラットフォームバージョンは `eks.1` から開始されます。ただし、Amazon EKS では新しいプラットフォームバージョンを定期的にリリースすることで、新しい Kubernetes コントロールプレーン設定を有効化し、セキュリティ修正プログラムを提供します。

Amazon EKS プラットフォームのバージョンで、新しいマイナーバージョンが使用可能になった場合は、以下が行われます。

- Amazon EKS プラットフォームのバージョン番号がインクリメントされます (`eks.`*`n+1`*)。
- Amazon EKS は、既存のすべてのクラスターを、対応する Kubernetes マイナーバージョン用の最新の Amazon EKS プラットフォームバージョンに自動的にアップグレードします。既存の Amazon EKS プラットフォームバージョンの自動アップグレードは、段階的にロールアウトされます。ロールアウトプロセスには時間がかかる場合があります。最新の Amazon EKS プラットフォームバージョンの機能がすぐに必要な場合は、新しい Amazon EKS クラスターを作成する必要があります。

  クラスターが現在のプラットフォームバージョンより 2 バージョン以上遅れている場合、Amazon EKS がクラスターを自動的に更新できなかった可能性があります。この原因の詳細については、「Amazon EKS プラットフォームのバージョンは、現在のプラットフォーム バージョンより 2 つ以上遅れています」を参照してください。
- Amazon EKS は、対応するパッチバージョンを適用して新しいノード AMI を発行する可能性があります。ただし、すべてのパッチバージョンは、特定の Kubernetes マイナーバージョンにおいて、EKS コントロールプレーンとノード AMI との間で互換性を持ちます。

新しい Amazon EKS プラットフォームバージョンにより、重大な変更が発生したり、サービスが中断されたりすることはありません。

クラスターは常に、指定された Kubernetes バージョン用に入手可能な、最新の Amazon EKS プラットフォームバージョン (`eks.`*`n`*) で作成されます。クラスターを新しい Kubernetes マイナーバージョンに更新する場合、そのクラスターは、更新した Kubernetes マイナーバージョンでの、現在の Amazon EKS プラットフォームバージョンを受け取ります。

最新および間近にリリースされた、Amazon EKS プラットフォームのバージョンを以下の表に示します。

Kubernetes バージョン 1.30

次の受付コントローラーは、すべての 1.30 プラットフォームバージョンで有効です: NodeRestriction、ExtendedResourceToleration、NamespaceLifecycle、LimitRanger、Serv

Kubernetes バージョン	EKS プラットフォームのバージョン	リリースノート	リリース日
1.30.1	eks.3	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 6 月 7 日
1.30.0	eks.2	EKS 用 Kubernetes バージョン 1.30 の初回リリース。詳細については、「Kubernetes 1.30」を参照してください。	2024 年 5 月 23 日

Kubernetes バージョン 1.29

次の受付コントローラーは、すべての 1.29 プラットフォームバージョンで有効です: NodeRestriction、ExtendedResourceToleration、NamespaceLifecycle、LimitRanger、Serv

Kubernetes バージョン	EKS プラットフォームのバージョン	リリースノート	リリース日
1.29.5	eks.8	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 6 月 7 日
1.29.4	eks.7	CoreDNS 自動スケーリング、セキュリティの修正および機能強化が行われた、新しいプラットフォームバー	2024 年 5 月 16 日

Kubernetes バージョン	EKS プラットフォームのバージョン	リリースノート	リリース日
		ジョン。CoreDNS 自動スケーリングの詳細については、「CoreDNS の自動スケーリング」を参照してください。	
1.29.3	eks.6	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 4 月 18 日
1.29.1	eks.5	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 3 月 29 日
1.29.1	eks.4	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 3 月 20 日
1.29.1	eks.3	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 3 月 12 日
1.29.0	eks.1	EKS 用 Kubernetes バージョン 1.29 の初回リリース。詳細については、「Kubernetes 1.29」を参照してください。	2024 年 1 月 23 日

Kubernetes バージョン 1.28

次の受付コントローラーは、すべての 1.28 プラットフォームバージョンで有効です: NodeRestriction、ExtendedResourceToleration、NamespaceLifecycle、LimitRanger、Serv

Kubernetes バージョン	EKS プラットフォームのバージョン	リリースノート	リリース日
1.28.10	eks.14	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 6 月 7 日
1.28.9	eks.13	CoreDNS 自動スケーリング、セキュリティの修正および機能強化が行われた、新しいプラットフォームバージョン。CoreDNS 自動スケーリングの詳細については、「CoreDNS の自動スケーリング」を参照してください。	2024 年 5 月 16 日
1.28.8	eks.12	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 4 月 18 日
1.28.7	eks.11	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 3 月 29 日
1.28.7	eks.10	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 3 月 20 日
1.28.6	eks.9	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 3 月 12 日
1.28.5	eks.7	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 1 月 17 日

Kubernetes バージョン	EKS プラットフォームのバージョン	リリースノート	リリース日
1.28.4	eks.6	アクセスエントリ、セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 12 月 14 日
1.28.4	eks.5	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 12 月 12 日
1.28.3	eks.4	EKS Pod Identity、セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 11 月 10 日
1.28.3	eks.3	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 11 月 3 日
1.28.2	eks.2	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 10 月 16 日
1.28.1	eks.1	EKS 用 Kubernetes バージョン 1.28 の初回リリース。詳細については、「Kubernetes 1.28」を参照してください。	2023 年 9 月 26 日

Kubernetes バージョン 1.27

次の受付コントローラーは、すべての 1.27 プラットフォームバージョンで有効です: NodeRestriction、ExtendedResourceToleration、NamespaceLifecycle、LimitRanger、Serv

Kubernetes バージョン	EKS プラットフォームのバージョン	リリースノート	リリース日
1.27.14	eks.18	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 6 月 7 日
1.27.13	eks.17	CoreDNS 自動スケーリング、セキュリティの修正および機能強化が行われた、新しいプラットフォームバージョン。CoreDNS 自動スケーリングの詳細については、「CoreDNS の自動スケーリング」を参照してください。	2024 年 5 月 16 日
1.27.12	eks.16	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 4 月 18 日
1.27.11	eks.15	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 3 月 29 日
1.27.11	eks.14	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 3 月 20 日
1.27.10	eks.13	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 3 月 12 日
1.27.9	eks.11	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 1 月 17 日

Kubernetes バージョン	EKS プラットフォームのバージョン	リリースノート	リリース日
1.27.8	eks.10	アクセスエントリ、セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 12 月 14 日
1.27.8	eks.9	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 12 月 12 日
1.27.7	eks.8	EKS Pod Identity、セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 11 月 10 日
1.27.7	eks.7	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 11 月 3 日
1.27.6	eks.6	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 10 月 16 日
1.27.4	eks.5	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 8 月 30 日
1.27.4	eks.4	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 7 月 30 日
1.27.3	eks.3	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 6 月 30 日
1.27.2	eks.2	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 6 月 9 日

Kubernetes バージョン	EKS プラットフォームのバージョン	リリースノート	リリース日
1.27.1	eks.1	EKS 用 Kubernetes バージョン 1.27 の初回リリース。詳細については、「Kubernetes 1.27」を参照してください。	2023 年 5 月 24 日

Kubernetes バージョン 1.26

次の受付コントローラーは、すべての 1.26 プラットフォームバージョンで有効です: NodeRestriction、ExtendedResourceToleration、NamespaceLifecycle、LimitRanger、Serv

Kubernetes バージョン	EKS プラットフォームのバージョン	リリースノート	リリース日
1.26.15	eks.19	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 6 月 7 日
1.26.15	eks.18	CoreDNS 自動スケーリング、セキュリティの修正および機能強化が行われた、新しいプラットフォームバージョン。CoreDNS 自動スケーリングの詳細については、「CoreDNS の自動スケーリング」を参照してください。	2024 年 5 月 16 日
1.26.15	eks.17	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 4 月 18 日
1.26.14	eks.16	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 3 月 29 日

Kubernetes バージョン	EKS プラットフォームのバージョン	リリースノート	リリース日
1.26.14	eks.15	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 3 月 20 日
1.26.13	eks.14	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 3 月 12 日
1.26.12	eks.12	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 1 月 17 日
1.26.11	eks.11	アクセスエントリ、セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 12 月 14 日
1.26.11	eks.10	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 12 月 12 日
1.26.10	eks.9	EKS Pod Identity、セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 11 月 10 日
1.26.10	eks.8	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 11 月 3 日
1.26.9	eks.7	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 10 月 16 日
1.26.7	eks.6	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 8 月 30 日

Kubernetes バージョン	EKS プラットフォームのバージョン	リリースノート	リリース日
1.26.7	eks.5	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 7 月 30 日
1.26.6	eks.4	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 6 月 30 日
1.26.5	eks.3	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 6 月 9 日
1.26.4	eks.2	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 5 月 5 日
1.26.2	eks.1	EKS 用 Kubernetes バージョン 1.26 の初回リリース。詳細については、「Kubernetes 1.26」を参照してください。	2023 年 4 月 11 日

Kubernetes バージョン 1.25

次の受付コントローラーは、すべての 1.25 プラットフォームバージョンで有効です: NodeRestriction、ExtendedResourceToleration、NamespaceLifecycle、LimitRanger、Serv

Kubernetes バージョン	EKS プラットフォームのバージョン	リリースノート	リリース日
1.25.16	eks.20	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 6 月 7 日

Kubernetes バージョン	EKS プラットフォームのバージョン	リリースノート	リリース日
1.25.16	eks.19	CoreDNS 自動スケーリング、セキュリティの修正および機能強化が行われた、新しいプラットフォームバージョン。CoreDNS 自動スケーリングの詳細については、「CoreDNS の自動スケーリング」を参照してください。	2024 年 5 月 16 日
1.25.16	eks.18	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 4 月 18 日
1.25.16	eks.17	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 3 月 29 日
1.25.16	eks.16	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 3 月 20 日
1.25.16	eks.15	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 3 月 12 日
1.25.16	eks.13	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 1 月 17 日
1.25.16	eks.12	アクセスエントリ、セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 12 月 14 日

Kubernetes バージョン	EKS プラットフォームのバージョン	リリースノート	リリース日
1.25.16	eks.11	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 12 月 12 日
1.25.15	eks.10	EKS Pod Identity、セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 11 月 10 日
1.25.15	eks.9	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 11 月 3 日
1.25.14	eks.8	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 10 月 16 日
1.25.12	eks.7	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 8 月 30 日
1.25.12	eks.6	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 7 月 30 日
1.25.11	eks.5	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 6 月 30 日
1.25.10	eks.4	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 6 月 9 日
1.25.9	eks.3	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 5 月 5 日

Kubernetes バージョン	EKS プラットフォームのバージョン	リリースノート	リリース日
1.25.8	eks.2	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 3 月 24 日
1.25.6	eks.1	EKS 用 Kubernetes バージョン 1.25 の初回リリース。詳細については、「Kubernetes 1.25」を参照してください。	2023 年 2 月 21 日

Kubernetes バージョン 1.24

次のアドミッションコントローラーは、すべての 1.24 プラットフォームバージョンで有効です: CertificateApproval、CertificateSigning、CertificateSubjectRestriction、DefaultIn および ValidatingAdmissionWebhook。

Kubernetes バージョン	EKS プラットフォームのバージョン	リリースノート	リリース日
1.24.17	eks.23	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 6 月 7 日
1.24.17	eks.22	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 5 月 16 日
1.24.17	eks.21	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 4 月 18 日

Kubernetes バージョン	EKS プラットフォームのバージョン	リリースノート	リリース日
1.24.17	eks.20	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 3 月 29 日
1.24.17	eks.19	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 3 月 20 日
1.24.17	eks.18	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 3 月 12 日
1.24.17	eks.16	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 1 月 17 日
1.24.17	eks.15	アクセスエントリ、セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 12 月 14 日
1.24.17	eks.14	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 12 月 12 日
1.24.17	eks.13	EKS Pod Identity、セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 11 月 10 日
1.24.17	eks.12	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 11 月 3 日
1.24.17	eks.11	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 10 月 16 日

Kubernetes バージョン	EKS プラットフォームのバージョン	リリースノート	リリース日
1.24.16	eks.10	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 8 月 30 日
1.24.16	eks.9	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 7 月 30 日
1.24.15	eks.8	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 6 月 30 日
1.24.14	eks.7	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 6 月 9 日
1.24.13	eks.6	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 5 月 5 日
1.24.12	eks.5	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 3 月 24 日
1.24.8	eks.4	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 1 月 27 日
1.24.7	eks.3	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2022 年 12 月 5 日
1.24.7	eks.2	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2022 年 11 月 18 日

Kubernetes バージョン	EKS プラットフォームのバージョン	リリースノート	リリース日
1.24.7	eks.1	EKS 用 Kubernetes バージョン 1.24 の初回リリース。詳細については、「Kubernetes 1.24」を参照してください。	2022 年 11 月 15 日

Kubernetes バージョン 1.23

次のアドミッションコントローラーは、すべての 1.23 プラットフォームバージョンで有効です: CertificateApproval、CertificateSigning、CertificateSubjectRestriction、DefaultIn および ValidatingAdmissionWebhook。

Kubernetes バージョン	EKS プラットフォームのバージョン	リリースノート	リリース日
1.23.17	eks.25	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 6 月 7 日
1.23.17	eks.24	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 5 月 16 日
1.23.17	eks.23	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 4 月 18 日
1.23.17	eks.22	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 3 月 29 日

Kubernetes バージョン	EKS プラットフォームのバージョン	リリースノート	リリース日
1.23.17	eks.21	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 3 月 20 日
1.23.17	eks.20	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 3 月 12 日
1.23.17	eks.18	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 1 月 17 日
1.23.17	eks.17	アクセスエントリ、セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 12 月 14 日
1.23.17	eks.16	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 12 月 12 日
1.23.17	eks.15	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 11 月 10 日
1.23.17	eks.14	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 11 月 3 日
1.23.17	eks.13	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 10 月 16 日
1.23.17	eks.12	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 8 月 30 日

Kubernetes バージョン	EKS プラットフォームのバージョン	リリースノート	リリース日
1.23.17	eks.11	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 7 月 30 日
1.23.17	eks.10	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 6 月 30 日
1.23.17	eks.9	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 6 月 9 日
1.23.17	eks.8	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 5 月 5 日
1.23.17	eks.7	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 3 月 24 日
1.23.14	eks.6	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 1 月 27 日
1.23.13	eks.5	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2022 年 12 月 5 日
1.23.13	eks.4	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2022 年 11 月 18 日
1.23.12	eks.3	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2022 年 11 月 7 日

Kubernetes バージョン	EKS プラットフォームのバージョン	リリースノート	リリース日
1.23.10	eks.2	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2022 年 9 月 21 日
1.23.7	eks.1	EKS 用 Kubernetes バージョン 1.23 の初回リリース。詳細については、「Kubernetes 1.23」を参照してください。	2022 年 8 月 11 日

最新のプラットフォームバージョンを取得する

クラスターの現在のプラットフォームバージョンを取得するには (コンソール)

1. Amazon EKS コンソールを開きます。
2. ナビゲーションペインで [クラスター] を選択します。
3. クラスターのリストで、プラットフォームバージョンを確認する [クラスター名] を選択します。
4. [概要] タブを選択します。
5. [プラットフォームバージョン] は、[詳細] セクションで参照できます。

クラスターの現在のプラットフォームバージョンを取得するには (AWS CLI)

1. プラットフォームバージョンを確認するクラスターの [名前] を確認します。
2. 次のコマンドを実行します。

```
aws eks describe-cluster --name my-cluster --query cluster.platformVersion
```

出力例は次のとおりです。

```
"eks.10"
```

Autoscaling

オートスケーリングとは、需要の変化に合わせてリソースを自動的に増減させる機能です。これは Kubernetes の主要な機能であり、この機能がなければ、手動で実行するために膨大な人的リソースが必要になります。

Amazon EKS は 2 つの自動スケーリング製品をサポートしています。

Karpenter

Karpenter は柔軟で高性能な Kubernetes クラスターオートスケーラーで、アプリケーションの可用性とクラスター効率の向上に役立ちます。Karpenter では、1 分以内にアプリケーション負荷の変化に応じて、適切なサイズのコンピューティングリソース (Amazon EC2 インスタンスなど) を起動します。Kubernetes と AWS を統合することで、Karpenter は、ワークロードの要件を正確に満たすジャストインタイムコンピューティングリソースをプロビジョンできます。Karpenter は、クラスターワークロードの具体的な要件に基づいて、新しいコンピューティングリソースを自動的にプロビジョンします。これには、コンピューティング、ストレージ、高速化、スケジューリングの各要件が含まれます。Amazon EKS は Karpenter を使用したクラスターをサポートしていますが、Karpenter は適合するいずれの Kubernetes クラスターでも動作します。詳細については、Karpenter ドキュメントを参照してください。

Cluster Autoscaler

Kubernetes の Cluster Autoscaler は、ポッドが失敗したり、他のノードに再スケジュールされた場合に、クラスター内のノード数を自動的に調整します。Cluster Autoscaler は Auto Scaling グループを使用します。詳細については、「AWS の Cluster Autoscaler」を参照してください。

アクセスを管理する

Amazon EKS クラスターへのアクセスを管理する方法について説明します。Amazon EKS を使用するには、Kubernetes と AWS Identity and Access Management (AWS IAM) がどのようにアクセスコントロールをするのかについての知識が必要になります。

このセクションには以下が含まれます。

the section called “Kubernetes API へのアクセスを許可する” — アプリケーションまたはユーザーが Kubernetes API を認証できるようにするための方法を、説明します。アクセスエントリ、aws-auth ConfigMap、または外部の OIDC プロバイダーを使用できます。

the section called “kubectl を使用してクラスターにアクセスする” — Amazon EKS クラスターと通信するように kubectl を設定する方法について説明します。AWS CLI を使用して kubeconfig ファイルを作成します。

the section called “AWS へのアクセスを許可する” - Kubernetes サービスアカウントを AWS IAM ロールに関連付ける方法を説明します。サービスアカウント (IRSA) には、Pod Identity または IAM ロールを使用できます。

一般的なタスク

- 開発者に Kubernetes API へのアクセスを許可します。AWS Management Console 内の Kubernetes リソースを表示します。
  - 解決策: アクセスエントリを使用して、Kubernetes RBAC 権限を AWS IAM ユーザーまたはロールに関連付けます。
- AWS 認証情報を使用して、kubectl を Amazon EKS クラスターと通信するように設定します。
  - 解決策: AWS CLI を使用して kubeconfig ファイルを作成します。
- Ping Identity など外部 ID プロバイダーを使用して、Kubernetes API に対してユーザーを認証します。
  - 解決策: 外部の OIDC プロバイダーをリンクします。
- Kubernetes クラスターのワークロードに AWS API を呼び出す機能を付与します。
  - 解決策: Pod Identity を使用して AWS IAM ロールを Kubernetes サービスアカウントに関連付けます。

背景:

- Kubernetes サービスアカウントの仕組みを学びます。
- Kubernetes ロールベースアクセスコントロール (RBAC) モデルを確認します。
- AWS リソースへのアクセスを管理する方法については、「AWS IAM ユーザーガイド」を参照してください。あるいは、無料の AWS IAM の使用に関する入門トレーニングも受講できます。

Kubernetes API へのアクセスを許可する

クラスターには Kubernetes API エンドポイントがあります。Kubectl はこの API を使用します。この API は、次の 2 種類の ID を使用して認証することができます。

- AWS Identity and Access Management (IAM) プリンシパル (ロールまたはユーザー) — このタイプには IAM への認証が必要です。IAM ユーザーとして、または ID ソースから提供された認証情報を使用して、フェデレーテッド ID で AWS にサインインできます。IAM ロールを使用して前もって管理者により ID フェデレーションが設定されている場合、フェデレーティッド ID としてのみサインインできます。フェデレーションを使用して AWS にアクセスする場合、間接的にロールを引き受けることになります。ユーザーがこのタイプの ID を使用すると、次のことが行われます。
  - ユーザーがクラスター上の Kubernetes オブジェクトを操作できるように Kubernetes アクセス許可を割り当てることができます。IAM プリンシパルにアクセス許可を割り当てて、クラスター上の Kubernetes オブジェクトにアクセスできるようにする方法の詳細については、「アクセスエントリを管理する」を参照してください。
  - IAM アクセス許可を割り当てて、ユーザーが Amazon EKS API、AWS CLI、AWS CloudFormation、AWS Management Console、または `eksctl` を使用して Amazon EKS クラスターとそのリソースを操作できるようにすることができます。詳細については、サービス認証リファレンスの「Amazon Elastic Kubernetes Service で定義されるアクション」を参照してください。
  - ノードは IAM ロールを引き受けることでクラスターに参加します。IAM プリンシパルを使用してクラスターにアクセスできます。プリンシパルは、Amazon EKS コントロールプレーンで実行される AWS IAM Authenticator for Kubernetes によって提供されます。
- 独自の OpenID Connect (OIDC) プロバイダーのユーザー — このタイプでは、OIDC プロバイダーへの認証が必要です。Amazon EKS クラスターで独自の OIDC プロバイダーを設定する方法については、「OpenID Connect アイデンティティプロバイダーからクラスターのユーザーを認証する」を参照してください。ユーザーがこのタイプの ID を使用すると、次のことが行われます。
  - ユーザーがクラスター上の Kubernetes オブジェクトを操作できるように Kubernetes アクセス許可を割り当てることができます。

- IAM アクセス許可を割り当てて、ユーザーが Amazon EKS API、AWS CLI、AWS CloudFormation、AWS Management Console、または `eksctl` を使用して Amazon EKS クラスターとそのリソースを操作できるようにすることはできません。

クラスターでは両方のタイプの ID を使用できます。IAM の認証方法を無効にすることはできません。OIDC 認証は任意です。

IAM アイデンティティと Kubernetes のアクセス許可を関連付ける

Kubernetes 用の AWS IAM Authenticator はクラスターのコントロールプレーンにインストールされます。これにより、クラスター上の Kubernetes リソースへのアクセスを許可する AWS Identity and Access Management (IAM) プリンシパル (ロールとユーザー) が有効になります。次のいずれかの方法を使用して、IAM プリンシパルにクラスター上の Kubernetes オブジェクトへのアクセスを許可できます。

- アクセスエントリの作成 — クラスターが、クラスターの Kubernetes バージョンの「前提条件」セクションに記載されているプラットフォームバージョンと同じかそれ以降の場合は、このオプションを使用することをお勧めします。

  アクセスエントリを使用して、クラスター外から IAM プリンシパルの Kubernetes アクセス許可を管理します。クラスターへのアクセスの追加および管理には、EKS API、AWS Command Line Interface、AWS SDK、AWS CloudFormation、および AWS Management Console を使用できます。つまり、クラスターを作成したのと同じツールでユーザーを管理できるということです。

  開始するには、アクセスエントリの設定、次に 既存の `aws-auth` ConfigMap エントリをアクセスエントリに移行する に従います。
- **`aws-auth` ConfigMap** へのエントリの追加 — クラスターのプラットフォームバージョンが「前提条件」セクションに記載されているバージョンよりも前の場合は、このオプションを使用する必要があります。クラスターのプラットフォームバージョンが、クラスターの Kubernetes バージョンの「前提条件」セクションに記載されているプラットフォームバージョンと同じかそれ以降で、`ConfigMap` にエントリを追加した場合は、それらのエントリをアクセスエントリに移行することをお勧めします。ただし、マネージド型ノードグループで使用される IAM ロールのエントリや Fargate プロファイルなど、Amazon EKS が `ConfigMap` に追加したエントリは移行できません。詳細については、「the section called “Kubernetes API へのアクセスを許可する”」を参照してください。

- aws-auth ConfigMap オプションを使用する必要がある場合は、**eksctl create iamidentitymapping** コマンドを使用してエントリを ConfigMap に追加できます。詳細については、eksctlドキュメントの「IAM ユーザーとロールの管理」を参照してください。

クラスター認証モードを設定する

各クラスターには認証モードがあります。認証モードによって、IAM プリンシパルがクラスター上の Kubernetes オブジェクトにアクセスできるようにするために使用できる方法が決まります。認証モードは 3 つあります。

> **Important**
>
> アクセス入力メソッドを一度有効にすると、無効にすることはできません。ConfigMap メソッドがクラスターの作成時に有効になっていない場合、後で有効にすることはできません。アクセスエントリの導入前に作成されたすべてのクラスターでは、ConfigMap メソッドが有効になっています。

クラスター内部の aws-auth ConfigMap

これは Amazon EKS クラスターのオリジナルの認証モードです。クラスターを作成した IAM プリンシパルは、kubectl を使用してクラスターにアクセスできる初期ユーザーです。初期ユーザーは、aws-auth ConfigMap のリストに他のユーザーを追加し、クラスター内の他のユーザーに影響を与えるアクセス許可を割り当てる必要があります。ConfigMap には管理するエントリがないため、これらの他のユーザーは初期ユーザーを管理したり削除したりできません。

ConfigMap とアクセスエントリの両方

この認証モードでは、両方の方法を使用して IAM プリンシパルをクラスターに追加できます。それぞれの方法には別々のエントリが保存されることに注意してください。例えば、AWS CLI からアクセスエントリを追加しても、aws-auth ConfigMap は更新されません。

アクセスエントリのみ

この認証モードでは、EKS API、AWS Command Line Interface、AWS SDK、AWS CloudFormation、および AWS Management Console を使用して IAM プリンシパルのクラスターへのアクセスを管理できます。

各アクセスエントリにはタイプがあり、プリンシパルを特定の名前空間に制限するアクセススコープと、事前設定された再利用可能なアクセス許可ポリシーを設定するアクセスポリシーを組

み合わせて使用できます。また、STANDARD タイプと Kubernetes RBAC グループを使用してカスタムアクセス許可を割り当てることもできます。

[Authentication mode] (認証モード)	方法
`ConfigMap` のみ (`CONFIG_MAP`)	`aws-auth ConfigMap`
EKS API と `ConfigMap` (`API_AND_CONFIG_MAP`)	EKS API、AWS Command Line Interface、AWS SDK、AWS CloudFormation、AWS Management Console、および `aws-auth ConfigMap` のアクセスエントリ
EKS API のみ (`API`)	EKS API、AWS Command Line Interface、AWS SDK、AWS CloudFormation、および AWS Management Console のアクセスエントリ

アクセスエントリを管理する

前提条件

- Amazon EKS クラスターのクラスターアクセスオプションに関する知識。詳細については、「Kubernetes API へのアクセスを許可する」を参照してください。
- 既存の Amazon EKS クラスター。デプロイするには、「Amazon EKS の使用開始」を参照してください。アクセスエントリを使用してクラスターの認証モードを変更するには、クラスターのプラットフォームバージョンが、次の表に記載されているバージョンと同じかそれ以降、または表に記載されているバージョンよりも Kubernetes のバージョンが新しい必要があります。

Kubernetes バージョン	プラットフォームバージョン
`1.30`	`eks.2`
`1.29`	`eks.1`
`1.28`	`eks.6`

Kubernetes バージョン	プラットフォームバージョン
1.27	eks.10
1.26	eks.11
1.25	eks.12
1.24	eks.15
1.23	eks.17

現在の Kubernetes バージョンとプラットフォームバージョンを確認するには、次のコマンドの *my-cluster* をクラスターの名前に置き換えて、変更したコマンドを実行します: **aws eks describe-cluster --name *my-cluster* --query 'cluster.{"Kubernetes Version": version, "Platform Version": platformVersion}'**

> **Important**
>
> Amazon EKS は、クラスターを表に記載されているプラットフォームバージョンに更新したあと、クラスターを最初に作成した IAM プリンシパルに対し、クラスターへの管理者権限を持つアクセスエントリを作成します。その IAM プリンシパルにクラスターへの管理者権限を与えたくない場合は、Amazon EKS が作成したアクセスエントリを削除します。前の表に記載されているものより前のプラットフォームバージョンのクラスターの場合、クラスター作成者が常にクラスター管理者になります。クラスターを作成した IAM ユーザーまたはロールからは、クラスター管理者権限を削除することはできません。

- クラスターに対し `CreateAccessEntry`、`ListAccessEntries`、`DescribeAccessEntry`、`DeleteAccessEntry`、`Up` のアクセス許可を持つ IAM プリンシパル。Amazon EKS アクセス許可の詳細については、サービス認証リファレンスの「Amazon Elastic Kubernetes Service で定義されるアクション」を参照してください。
- アクセスエントリを作成する既存の IAM プリンシパル、または更新または削除する既存のアクセスエントリ。

アクセスエントリの設定

アクセスエントリを使い始めるには、クラスターの認証モードを `API_AND_CONFIG_MAP` または `API` モードに変更する必要があります。これにより、アクセスエントリ用の API が追加されます。

AWS Management Console

アクセスエントリを作成するには

1. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。
2. アクセスエントリを作成するクラスターの名前を選択します。
3. [リモートアクセス] タブを選択します。
4. 認証モードには、クラスターの現在の認証モードが表示されます。モードが EKS API と表示されている場合は、すでにアクセスエントリを追加でき、残りの手順は省略できます。
5. [アクセスの管理] を選択します。
6. クラスター認証モードの場合は、EKS API でモードを選択します。認証モードを、EKS API およびアクセスエントリを削除するモードに戻すことはできないことに注意してください。
7. [Save changes] (変更の保存) をクリックします。Amazon EKS がクラスターの更新を開始し、クラスターのステータスが Updating に変わり、変更が [更新履歴] タブに記録されます。
8. クラスターのステータスが Active に戻るまで待ちます。クラスターが Active になったら、アクセスエントリの作成 の手順に従って IAM プリンシパルのクラスターへのアクセスを追加できます。

AWS CLI

前提条件

AWS CLI v1 の最新バージョンがデバイスまたは AWS CloudShell にインストールされ、設定されています。AWS CLI v2 は数日間新機能をサポートしません。現在のバージョンは、`aws --version | cut -d / -f2 | cut -d ' ' -f1` で確認できます。macOS の `yum`、`apt-get`、または Homebrew などのパッケージマネージャは、AWS CLI の最新バージョンより数バージョン遅れることがあります。最新バージョンをインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI のインストール、更新、およびアンインストール」と「`aws configure` でのクイック設定」を参照してください。AWS CloudShell にインストールされている AWS CLI バージョンは、最新バージョンより数バージョン遅れている可能性

もあります。更新するには、「AWS CloudShell ユーザーガイド」の「ホームディレクトリへの AWS CLI のインストール」を参照してください。

1.

2. 以下のコマンドを実行します。*my-cluster* の部分は、自分のクラスター名に置き換えます。ConfigMap メソッドを完全に無効にする場合は、API_AND_CONFIG_MAP を API に置き換えてください。

   Amazon EKS がクラスターの更新を開始し、クラスターのステータスが UPDATING に変わり、変更が aws eks list-updates に記録されます。

   ```
   aws eks update-cluster-config --name my-cluster --access-config
    authenticationMode=API_AND_CONFIG_MAP
   ```

3. クラスターのステータスが Active に戻るまで待ちます。クラスターが Active になったら、アクセスエントリの作成 の手順に従って IAM プリンシパルのクラスターへのアクセスを追加できます。

アクセスエントリの作成

考慮事項

アクセスエントリを作成する前に、次の点を考慮してください。

- アクセスエントリには、1 つだけの既存の IAM プリンシパルの Amazon リソースネーム (ARN) が含まれます。IAM プリンシパルは複数のアクセスエントリに含めることはできません。指定する ARN に関するその他の考慮事項
  - IAM のベストプラクティスでは、長期認証情報を持つ IAM ユーザーではなく、短期認証情報を持つ IAM ロールを使用してクラスターにアクセスすることを推奨しています。詳細は、IAM ユーザーガイドの「人間のユーザーが一時的な認証情報を使用して AWS にアクセスするには、ID プロバイダーとのフェデレーションの使用が必要です」を参照してください。
  - ARN が IAM ロール用の場合は、パスを含めることができます。aws-auth ConfigMap エントリの ARN にはパスを含めることはできません。例えば、ARN は arn:aws:iam::*111122223333*:role/*development/apps/my-role* または arn:aws:iam::*111122223333*:role/*my-role* にすることができます。
  - アクセスエントリのタイプが STANDARD 以外の場合 (タイプに関する次の考慮事項を参照)、ARN はクラスターと同じ AWS アカウント である必要があります。タイプが STANDARD

の場合、ARN はクラスターが属するアカウントと同じ、または異なる AWS アカウント でもかまいません。

- アクセスエントリの作成後に IAM プリンシパルを変更することはできません。
- この ARN を含む IAM プリンシパルを削除しても、アクセスエントリは自動的に削除されません。削除する IAM プリンシパルの ARN を含むアクセスエントリを削除することをお勧めします。アクセスエントリを削除せず、IAM プリンシパルを再作成すると、同じ ARN があっても、アクセスエントリは機能しません。これは、再作成された IAM プリンシパルの ARN は同じですが、再作成された IAM プリンシパルの `roleID` または `userID` (`aws sts get-caller-identity` AWS CLI コマンドで確認できます) は、元の IAM プリンシパルの ARN とは異なるためです。アクセスエントリの IAM プリンシパルの `roleID` または `userID` が表示されない場合でも、Amazon EKS はそれをアクセスエントリとともに保存します。

- 各アクセスエントリにはタイプがあります。`EC2 Linux` (Linux または Bottlerocket のセルフマネージド型ノードで使用される IAM ロールの場合)、`EC2 Windows` (Windows のセルフマネージド型ノードで使用される IAM ロールの場合)、`FARGATE_LINUX` (AWS Fargate (Fargate) で使用される IAM ロールの場合)、または`STANDARD` をタイプとして指定できます。タイプを指定しない場合、Amazon EKS は自動的にタイプを `STANDARD` に設定します。マネージド型ノードグループや Fargate プロファイルに使用される IAM ロールのアクセスエントリを作成する必要はありません。Amazon EKS は、クラスターのプラットフォームバージョンに関係なくこれらのロールのエントリを `aws-auth` `ConfigMap` に追加するためです。

  アクセスエントリの作成後にタイプを変更することはできません。

- アクセスエントリのタイプが `STANDARD` の場合、アクセスエントリのユーザー名を指定できます。ユーザー名の値を指定しない場合、Amazon EKS はアクセスエントリのタイプと、指定した IAM プリンシパルが IAM ロールか IAM ユーザーかに応じて、以下のいずれかの値を設定します。独自のユーザー名を指定する特別な理由がない限り、ユーザー名を指定せず、Amazon EKS に自動生成させることをお勧めします。独自のユーザー名を指定する場合:
  - `system:`、`eks:`、`aws:`、`amazon:`、または `iam:` で始めることはできません。
  - ユーザー名が IAM ロール用の場合は、ユーザー名の末尾に `{{SessionName}}` を追加することをお勧めします。ユーザー名に `{{SessionName}}` を追加する場合、ユーザー名の {{SessionName}} の前にコロンを含める必要があります。このロールを引き受けると、ロールを引き受けるときに指定されたセッションの名前が自動的にクラスターに渡され、CloudTrail ログに表示されます。例えば、`john{{SessionName}}` というユーザー名にはできません。ユーザー名は `:john{{SessionName}}` または `jo:hn{{SessionName}}` でなければなりません。コロンは `{{SessionName}}` の前になければいけません。次の表の Amazon EKS によって生成されたユーザー名には ARN が含まれています。ARN にはコロンが含まれているため、

この要件を満たしています。ユーザー名に {{SessionName}} を含めなければ、コロンは不要です。

IAM プリンシパルタイプ	タイプ	Amazon EKS が自動的に設定するユーザー名の値
ユーザー	STANDARD	ユーザーの ARN。 例：arn:aws:iam:: *111122223333* :user/*my-user*
ロール	STANDARD	想定されるロールの STS ARN。Amazon EKS では、{{SessionName}} がロールに追加されます。 例：arn:aws:sts:: *111122223333* :assumed-role/ *my-role*/{{SessionName}} 指定したロールの ARN にパスが含まれている場合、Amazon EKS は生成されたユーザー名からそのパスを削除します。
ロール	EC2 Linux、または EC2 Windows	system:node:{{EC2PrivateDNSName}}
ロール	FARGATE_LINUX	system:node:{{SessionName}}

アクセスエントリの作成後にユーザー名を変更することができます。

- アクセスエントリのタイプが STANDARD で、Kubernetes RBAC 認証を使用する場合は、アクセスエントリに 1 つ以上のグループ名を追加できます。アクセスエントリを作成したら、グループ名を追加および削除できます。IAM プリンシパルがクラスター上の Kubernetes オブジェクトにアク

セスできるようにするには、Kubernetes ロールベース認証 (RBAC) オブジェクトを作成して管理する必要があります。クラスター上に Kubernetes `RoleBinding` または `ClusterRoleBinding` を作成し、グループ名を `kind: Group` の `subject` として指定します。Kubernetes は、バインディングの `roleRef` でも指定した Kubernetes `Role` または `ClusterRole` オブジェクトで指定したすべてのクラスターオブジェクトにアクセスすることを IAM プリンシパルに許可します。グループ名を指定する場合は、Kubernetes ロールベース認証 (RBAC) オブジェクトについて理解しておくことをお勧めします。詳細については、「Kubernetes ドキュメント」の「RBAC 認証の使用」を参照してください。

> **Important**
>
> Amazon EKS は、クラスターに存在する Kubernetes RBAC オブジェクトに、指定したグループ名が含まれていることを確認しません。

クラスター上の Kubernetes オブジェクトに IAM プリンシパルがアクセスするのを許可する Kubernetes の代わりに、またはそれに加えて、Amazon EKS アクセスポリシーをアクセスエントリに関連付けることができます。Amazon EKS は、IAM プリンシパルがアクセスポリシーのアクセス許可を使用してクラスター上の Kubernetes オブジェクトにアクセスすることを許可します。アクセスポリシーのアクセス許可の範囲は、指定した Kubernetes 名前空間に限定できます。アクセスポリシーを使用する場合、Kubernetes RBAC オブジェクトを管理する必要はありません。詳細については、「アクセスポリシーとアクセスエントリとの関連付けおよび関連付け解除」を参照してください。

- タイプが `EC2 Linux` または `EC2 Windows` のアクセスエントリを作成する場合、アクセスエントリを作成する IAM プリンシパルには `iam:PassRole` アクセス許可が必要です。詳細については、IAM ユーザーガイドの「AWS のサービス にロールを渡すアクセス許可をユーザーに付与する」を参照してください。
- 標準的な IAM 動作と同様に、アクセスエントリの作成と更新は最終的には一貫性があり、最初の API コールが正常に戻ってから有効になるまでに数秒かかる場合があります。発生する可能性のあるこれらの遅延を考慮して、アプリケーションを設計する必要があります。アプリケーションの重要で高可用性のコードパスには、アクセスエントリの作成や更新を含めないことをお勧めします。代わりに、実行頻度が低い別の初期化またはセットアップルーチンに の変更を加えます。また、本番稼働ワークフローが依存する前に、変更が伝達済みであることを確認します。
- アクセスエントリは、サービスリンクロールをサポートしていません。プリンシパル ARN がサービスリンクロールである場合、アクセスエントリを作成することはできません。サービスリンク

ロールは、arn:aws:iam::*:role/aws-service-role/* 形式の ARN によって識別できます。

AWS Management Console または AWS CLI を使用してアクセスエントリを作成できます。

AWS Management Console

アクセスエントリを作成するには

1. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。
2. アクセスエントリを作成するクラスターの名前を選択します。
3. [リモートアクセス] タブを選択します。
4. [アクセスエントリの作成] を選択します。
5. IAM プリンシパルには、既存の IAM ロールまたはユーザーを選択します。IAM のベストプラクティスでは、長期認証情報を持つ IAM ユーザーではなく、短期認証情報を持つ IAM ロールを使用してクラスターにアクセスすることを推奨しています。詳細は、IAM ユーザーガイドの「人間のユーザーが一時的な認証情報を使用して AWS にアクセスするには、ID プロバイダーとのフェデレーションの使用が必要です」を参照してください。
6. [タイプ] で、アクセスエントリがセルフマネージド型の Amazon EC2 ノードに使用されるノードロール用のものである場合は、[EC2 Linux] または [EC2 Windows] を選択します。それ以外の場合は、デフォルト (標準) をそのまま使用します。
7. 選択した タイプが標準で、ユーザー名を指定する場合は、ユーザー名を入力します。
8. 選択した タイプが標準で、IAM プリンシパルに Kubernetes RBAC 認証を使用したい場合は、グループに 1 つ以上の名前を指定します。グループ名を指定せず、Amazon EKS 認証を使用する場合は、後のステップで、またはアクセスエントリの作成後にアクセスポリシーを関連付けることができます。
9. (オプション) [タグ] では、アクセスエントリにラベルを割り当てます。例えば、同じタグを持つすべてのリソースを検索しやすくするためです。
10. [Next] を選択します。
11. [アクセスポリシーの追加] ページで、選択したタイプが標準で、Amazon EKS が IAM プリンシパルにクラスター上の Kubernetes オブジェクトへのアクセス許可を与えることを許可する場合は、次の手順を実行します。それ以外の場合は、次へ を選択します。

a. [ポリシー名] には、アクセスポリシーを選択します。アクセスポリシーのアクセス許可は表示できませんが、Kubernetes ユーザー向け `ClusterRole` オブジェクトと同様のアクセス許可が含まれています。詳細については、Kubernetes ドキュメントの「ユーザー向けロール」を参照してください。

b. 以下のオプションのいずれかを選択します。

- クラスター — Amazon EKS が IAM プリンシパルに、クラスター上のすべての Kubernetes オブジェクトのアクセスポリシー内のアクセス許可があるように承認する場合は、このオプションを選択します。
- Kubernetes 名前空間 — Amazon EKS が IAM プリンシパルに、クラスター上の特定の Kubernetes 名前空間のすべての Kubernetes オブジェクトのアクセスポリシー内のアクセス許可があるように承認する場合は、このオプションを選択します。[名前空間]には、クラスター上の Kubernetes 名前空間の名前を入力します。名前空間をさらに追加する場合は、[新しい名前空間を追加する] を選択し、名前空間名を入力します。

c. ポリシーを追加するには、[ポリシーを追加] を選択します。ポリシーごとに異なる範囲を設定できますが、各ポリシーは 1 回しか追加できません。

d. [Next] を選択します。

12. アクセスエントリの設定を確認してください。何かが正しくないと思われる場合は、[戻る] を選択して手順に戻り、エラーを修正します。設定が正しければ、[作成] を選択します。

AWS CLI

前提条件

AWS CLI v1 の最新バージョンがデバイスまたは AWS CloudShell にインストールされ、設定されています。AWS CLI v2 は数日間新機能をサポートしません。現在のバージョンは、`aws --version | cut -d / -f2 | cut -d ' ' -f1` で確認できます。macOS の `yum`、`apt-get`、または Homebrew などのパッケージマネージャは、AWS CLI の最新バージョンより数バージョン遅れることがあります。最新バージョンをインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI のインストール、更新、およびアンインストール」と「`aws configure` でのクイック設定」を参照してください。AWS CloudShell にインストールされている AWS CLI バージョンは、最新バージョンより数バージョン遅れている可能性もあります。更新するには、「AWS CloudShell ユーザーガイド」の「ホームディレクトリへの AWS CLI のインストール」を参照してください。

アクセスエントリを作成するには

アクセスエントリを作成するには、以下のいずれかの例を使用できます。

- セルフマネージド型 Amazon EC2 Linux ノードグループのアクセスエントリを作成します。*my-cluster* はクラスターの名前、*111122223333* は自分の AWS アカウント ID に、*eks-my-Cluster-Self-Managed-NG-1* はノード IAM ロールの名前に置き換えます。ノードグループが Windows ノードグループの場合は、*EC2_Linux* を **EC2_Windows** に置き換えてください。

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/EKS-my-cluster-self-managed-ng-1 --type EC2_Linux
```

  STANDARD 以外のタイプを指定する場合、--kubernetes-groups オプションは使用できません。タイプが STANDARD 以外の値であるため、このアクセスエントリにアクセスポリシーを関連付けることはできません。

- Amazon EC2 セルフマネージド型ノードグループには使用されていない IAM ロールを許可するアクセスエントリを作成し、Kubernetes にクラスターへのアクセスを許可してもらいます。*my-cluster* をクラスターの名前に、*111122223333* を AWS アカウント ID に、*my-role* を IAM ロールの名前に置き換えます。*Viewer* は、クラスター上の Kubernetes RoleBinding または ClusterRoleBinding オブジェクトで指定したグループの名前に置き換えます。

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/my-role --type STANDARD --user Viewers --
kubernetes-groups Viewers
```

- IAM ユーザーがクラスターに対して認証できるようにするアクセスエントリを作成します。これも可能であるためこの例が示されていますが、IAM のベストプラクティスでは、長期認証情報を持つ IAM ユーザーではなく、短期認証情報を持つ IAM ロールを使用してクラスターにアクセスすることを推奨しています。詳細は、IAM ユーザーガイドの「人間のユーザーが一時的な認証情報を使用して AWS にアクセスするには、ID プロバイダーとのフェデレーションの使用が必要です」を参照してください。

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:user/my-user --type STANDARD --username my-user
```

このユーザーに Kubernetes API ディスカバリーロールのアクセス許可よりも多くのクラスターへのアクセスを許可したい場合は、`--kubernetes-groups` オプションが使用されていないため、アクセスエントリにアクセスポリシーを関連付ける必要があります。詳細については、「アクセスポリシーとアクセスエントリとの関連付けおよび関連付け解除」と Kubernetes ドキュメントの「API ディスカバリーロール」を参照してください。

アクセスエントリの更新

AWS Management Console または AWS CLI を使用してアクセスエントリを更新できます。

AWS Management Console

アクセスエントリを更新するには

1. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。
2. アクセスエントリを作成するクラスターの名前を選択します。
3. [リモートアクセス] タブを選択します。
4. 更新するアクセスエントリを選択します。
5. [編集] を選択します。
6. [ユーザー名] では、既存の値を変更できます。
7. [グループ]では、既存のグループ名を削除したり、新しいグループ名を追加したりできます。system: nodes または system: bootstrappers というグループ名が存在する場合は、削除しないでください。これらのグループを削除すると、クラスターが正しく機能しなくなる可能性があります。グループ名を指定せず、Amazon EKS 認証を使用する場合は、後のステップで、アクセスポリシーを関連付けてください。
8. [タグ] では、アクセスエントリにラベルを割り当てます。例えば、同じタグを持つすべてのリソースを検索しやすくするためです。また、既存のタグを削除することもできます。
9. [変更の保存] をクリックします。
10. アクセスポリシーをエントリに関連付ける場合は、アクセスポリシーとアクセスエントリとの関連付けおよび関連付け解除 を参照してください。

AWS CLI

前提条件

ご使用のデバイスまたは AWS CloudShell で、バージョン 2.12.3 以降、または AWS Command Line Interface (AWS CLI) のバージョン 1.27.160 以降がインストールおよび設定されていること。現在のバージョンを確認するには、「**aws --version | cut -d / -f2 | cut -d ' ' -f1**」を参照してください。macOS の yum、apt-get、または Homebrew などのパッケージマネージャは、AWS CLI の最新バージョンより数バージョン遅れることがあります。最新バージョンをインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI のインストール、更新、およびアンインストール」と「aws configure でのクイック設定」を参照してください。AWS CloudShell にインストールされている AWS CLI バージョンは、最新バージョンより数バージョン遅れている可能性もあります。更新するには、「AWS CloudShell ユーザーガイド」の「ホームディレクトリへの AWS CLI のインストール」を参照してください。

アクセスエントリを更新するには

my-cluster はクラスターの名前、*111122223333* は AWS アカウント ID に、*EKS-my-cluster-my-namespace-Viewers* は IAM ロールの名前に置き換えます。

```
aws eks update-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/EKS-my-cluster-my-namespace-Viewers --kubernetes-
groups Viewers
```

アクセスエントリのタイプが STANDARD 以外の値の場合、--kubernetes-groups オプションは使用できません。また、アクセスポリシーを STANDARD 以外のタイプのアクセスエントリに関連付けることもできません。

アクセスエントリの削除

アクセスエントリを誤って削除したことが判明した場合は、いつでも再作成できます。削除するアクセスエントリがアクセスポリシーに関連付けられている場合、その関連付けは自動的に削除されます。アクセスエントリを削除する前に、アクセスポリシーとアクセスエントリの関連付けを解除する必要はありません。

AWS Management Console または AWS CLI を使用してアクセスエントリを削除できます。

AWS Management Console

アクセスエントリを削除するには

1. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。
2. アクセスエントリを削除するクラスターの名前を選択します。
3. [リモートアクセス] タブを選択します。
4. [アクセスエントリ] リストで、削除するアクセスエントリを選択します。
5. [Delete] を選択します。
6. 確認ダイアログボックスで、[削除] を選択します。

AWS CLI

前提条件

ご使用のデバイスまたは AWS CloudShell で、バージョン `2.12.3` 以降、または AWS Command Line Interface (AWS CLI) のバージョン `1.27.160` 以降がインストールおよび設定されていること。現在のバージョンを確認するには、「**`aws --version | cut -d / -f2 | cut -d ' ' -f1`**」を参照してください。macOS の `yum`、`apt-get`、または Homebrew などのパッケージマネージャは、AWS CLI の最新バージョンより数バージョン遅れることがあります。最新バージョンをインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI のインストール、更新、およびアンインストール」と「aws configure でのクイック設定」を参照してください。AWS CloudShell にインストールされている AWS CLI バージョンは、最新バージョンより数バージョン遅れている可能性もあります。更新するには、「AWS CloudShell ユーザーガイド」の「ホームディレクトリへの AWS CLI のインストール」を参照してください。

アクセスエントリを削除するには

`my-cluster` をクラスターの名前、*`111122223333`* を AWS アカウント ID に、*`my-role`* をクラスターにアクセスさせたくない IAM ロールの名前に置き換えます。

```
aws eks delete-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/my-role
```

アクセスポリシーとアクセスエントリとの関連付けおよび関連付け解除

1 つ以上のアクセスポリシーを タイプ STANDARD のアクセスエントリに割り当てることができます。Amazon EKS は、クラスターで正しく機能するために必要なアクセス許可を他のタイプのアクセスエントリに自動的に付与します。Amazon EKS アクセスポリシーには、IAM アクセス許可ではなく Kubernetes アクセス許可が含まれます。アクセスポリシーをアクセスエントリに関連付ける前に、各アクセスポリシーに含まれる Kubernetes アクセス許可をすでに理解できていることを確かめてください。詳細については、「アクセスポリシーでの許可」を参照してください。どのアクセスポリシーも要件を満たさない場合は、アクセスポリシーをアクセスエントリに関連付けないでください。代わりに、アクセスエントリに 1 つ以上のグループ名を指定し、Kubernetes ロールベースのアクセス制御オブジェクトを作成して管理します。詳細については、「アクセスエントリの作成」を参照してください。

前提条件

- 既存のアクセスエントリ。作成する場合は「アクセスエントリの作成」を参照してください。
- `ListAccessEntries`、`DescribeAccessEntry`、`UpdateAccessEntry`、`ListAccessPolicies`、A および `DisassociateAccessPolicy` のアクセス許可を持つ AWS Identity and Access Management ロールまたはユーザー。詳細については、サービス認証リファレンスの「Amazon Elastic Kubernetes Service で定義されるアクション」を参照してください。

アクセスポリシーをアクセスエントリに関連付ける前に、以下の要件を考慮してください。

- 各アクセスエントリには複数のアクセスポリシーを関連付けることができますが、各ポリシーを 1 つのアクセスエントリに関連付けることができるのは 1 回だけです。複数のアクセスポリシーを関連付ける場合、アクセスエントリの IAM プリンシパルには、関連するすべてのアクセスポリシーに含まれるすべてのアクセス許可が付与されます。
- アクセスポリシーをクラスター上のすべてのリソースに範囲指定することも、1 つ以上の Kubernetes 名前空間の名前を指定して範囲指定することもできます。名前空間の名前にはワイルドカード文字を使用できます。例えば、`dev-` で始まるすべての名前空間にアクセスポリシーを適用する場合は、名前空間の名前として `dev-*` を指定できます。名前空間がクラスターに存在し、スペルがクラスター上の実際の名前空間の名前と一致していることを確認してください。Amazon EKS は、クラスター上の名前空間のスペルや存在を確認しません。
- アクセスポリシーをアクセスエントリに関連付けた後、アクセスポリシーのアクセス範囲を変更できます。アクセスポリシーの範囲を Kubernetes 名前空間に設定した場合は、必要に応じて関連付けの名前空間を追加および削除できます。

- グループ名も指定されているアクセスエントリにアクセスポリシーを関連付けると、IAM プリンシパルには、関連するすべてのアクセスポリシーのすべてのアクセス許可が付与されます。また、Kubernetes `Role` で指定されている Kubernetes `Role` または `ClusterRole` オブジェクト、およびグループ名を指定する `RoleBinding` オブジェクトのすべてのアクセス許可も保持されます。
- `kubectl auth can-i --list` コマンドを実行しても、コマンドを実行したときに使用している IAM プリンシパルのアクセスエントリに関連付けられているアクセスポリシーによって割り当てられた Kubernetes アクセス許可は表示されません。このコマンドは、アクセスエントリに指定したグループ名またはユーザー名にバインドした Kubernetes `Role` または `ClusterRole` オブジェクトでアクセス許可を付与した場合にのみ、Kubernetes アクセス許可を表示します。
- **`--as` *`username`*** または **`--as-group` *`group-name`*** で `kubectl` コマンドを使用するなど、クラスター上の Kubernetes オブジェクトを操作する際に Kubernetes ユーザーまたはグループになりすますと、Kubernetes RBAC 認証の使用を強制することになります。その結果、IAM プリンシパルには、アクセスエントリに関連付けられたアクセスポリシーによって割り当てられるアクセス許可はありません。IAM プリンシパルを偽装するユーザーまたはグループが持つ Kubernetes アクセス許可は、グループ名またはユーザー名にバインドした Kubernetes `Role` または `ClusterRole` オブジェクトで付与した Kubernetes アクセス許可だけです。IAM プリンシパルが関連するアクセスポリシー内のアクセス許可を持つようにするには、Kubernetes ユーザーまたはグループを偽装しないでください。IAM プリンシパルには、Kubernetes `Role` で付与したアクセス許可や、アクセスエントリに指定したグループ名またはユーザー名にバインドした `ClusterRole` オブジェクトも引き続き保持されます。詳細については、Kubernetes ドキュメントの「ユーザー偽装」を参照してください。

AWS Management Console または AWS CLI を使用してアクセスポリシーをアクセスエントリに関連付けることができます。

AWS Management Console

AWS Management Console を使用してアクセスポリシーをアクセスエントリに関連付けるには

1. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。
2. アクセスポリシーを関連付けるアクセスエントリがあるクラスターの名前を選択します。
3. [リモートアクセス] タブを選択します。

4. アクセスエントリのタイプが標準の場合は、Amazon EKS アクセスポリシーを関連付けたり関連付けを解除したりできます。アクセスエントリのタイプが標準以外の場合、このオプションは使用できません。
5. [アクセスポリシーを関連付ける] を選択します。
6. [ポリシー名] には、IAM プリンシパルに付与したいアクセス許可を持つポリシーを選択します。各ポリシーに含まれるアクセス許可を表示するには、「アクセスポリシーでの許可」を参照してください。
7. [アクセススコープ] では、アクセス範囲を選択します。[Cluster] を選択すると、すべての Kubernetes 名前空間のリソースについて、アクセスポリシー内のアクセス許可が IAM プリンシパルに付与されます。[Kubernetes 名前空間] を選択した場合は、[新規名前空間の追加] を選択できます。表示される [名前空間] フィールドには、クラスター上の Kubernetes 名前空間の名前を入力できます。IAM プリンシパルに複数の名前空間のアクセス許可を持たせたい場合は、複数の名前空間を入力できます。
8. [アクセスポリシーを追加] を選択します。

AWS CLI

前提条件

ご使用のデバイスまたは AWS CloudShell で、バージョン `2.12.3` 以降、または AWS Command Line Interface (AWS CLI) のバージョン `1.27.160` 以降がインストールおよび設定されていること。現在のバージョンを確認するには、「**`aws --version | cut -d / -f2 | cut -d ' ' -f1`**」を参照してください。macOS の `yum`、`apt-get`、または Homebrew などのパッケージマネージャは、AWS CLI の最新バージョンより数バージョン遅れることがあります。最新バージョンをインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI のインストール、更新、およびアンインストール」と「aws configure でのクイック設定」を参照してください。AWS CloudShell にインストールされている AWS CLI バージョンは、最新バージョンより数バージョン遅れている可能性もあります。更新するには、「AWS CloudShell ユーザーガイド」の「ホームディレクトリへの AWS CLI のインストール」を参照してください。

アクセスポリシーをアクセスエントリに関連付けるには

1. 利用可能なアクセスポリシーを表示します。

```
aws eks list-access-policies --output table
```

出力例は次のとおりです。

```
----------------------------------------------------------------------------------
|                                        ListAccessPolicies
                       |
+--------------------------------------------------------------------------------
+
||                                          accessPolicies
                      ||
|+-------------------------------------------------------------------
+-------------------------------+|
||                                arn                                |
     name              ||
|+-------------------------------------------------------------------
+-------------------------------+|
||  arn:aws:eks::aws:cluster-access-policy/AmazonEKSAdminPolicy        |
 AmazonEKSAdminPolicy         ||
||  arn:aws:eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy |
 AmazonEKSClusterAdminPolicy  ||
||  arn:aws:eks::aws:cluster-access-policy/AmazonEKSEditPolicy         |
 AmazonEKSEditPolicy          ||
||  arn:aws:eks::aws:cluster-access-policy/AmazonEKSViewPolicy         |
 AmazonEKSViewPolicy          ||
|+-------------------------------------------------------------------
+-------------------------------+|
```

各ポリシーに含まれるアクセス許可を表示するには、「アクセスポリシーでの許可」を参照してください。

2. 既存のアクセスエントリを表示します。*my-cluster* の部分は、自分のクラスター名に置き換えます。

```
aws eks list-access-entries --cluster-name my-cluster
```

出力例は次のとおりです。

```
{
    "accessEntries": [
        "arn:aws:iam::111122223333:role/my-role",
        "arn:aws:iam::111122223333:user/my-user"
    ]
```

```
}
```

3. アクセスポリシーをアクセスエントリに関連付けます。次の例では、AmazonEKSViewPolicy アクセスポリシーをアクセスエントリに関連付けます。*my-role* IAM ロールがクラスター上の Kubernetes オブジェクトにアクセスしようとするたびに、Amazon EKS は、ポリシー内のアクセス許可を使用して *my-namespace1* と *my-namespace2* の Kubernetes 名前空間のみの Kubernetes オブジェクトにアクセスするアクセス許可をロールに付与します。*my-cluster* をクラスターの名前、*111122223333* を AWS アカウント ID に、*my-role* を Amazon EKS に Kubernetes クラスターへのアクセスを許可してほしい IAM ロールの名前に置き換えます。

```
aws eks associate-access-policy --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/my-role \
    --access-scope type=namespace,namespaces=my-namespace1,my-namespace2 --
policy-arn arn:aws:eks::aws:cluster-access-policy/AmazonEKSViewPolicy
```

IAM プリンシパルにクラスター全体のアクセス許可を持たせたい場合は、**type=namespace,namespaces=*my-namespace1*,*my-namespace2*** を **type=cluster** に置き換えてください。複数のアクセスポリシーをアクセスエントリに関連付ける場合は、それぞれに固有のアクセスポリシーを指定してコマンドを複数回実行します。関連するアクセスポリシーにはそれぞれ独自の範囲があります。

> **Note**
>
> 関連するアクセスポリシーの範囲を後で変更する場合は、新しい範囲で前のコマンドをもう一度実行してください。例えば、*my-namespace2* を削除したい場合は、**type=namespace,namespaces=*my-namespace1*** のみを使用してコマンドを再実行します。範囲を **namespace** から **cluster** に変更する場合は、**type=cluster** を使用して **type=namespace,namespaces=*my-namespace1*,*my-namespace2*** を削除してコマンドを再実行します。

アクセスポリシーとアクセスエントリの関連付けを解除するには

1. どのアクセスポリシーをアクセスエントリに関連付けるかを決定します。

```
aws eks list-associated-access-policies --cluster-name my-cluster --principal-
arn arn:aws:iam::111122223333:role/my-role
```

出力例は次のとおりです。

```
{
    "clusterName": "my-cluster",
    "principalArn": "arn:aws:iam::111122223333",
    "associatedAccessPolicies": [
        {
            "policyArn": "arn:aws:eks::aws:cluster-access-
policy/AmazonEKSViewPolicy",
            "accessScope": {
                "type": "cluster",
                "namespaces": []
            },
            "associatedAt": "2023-04-17T15:25:21.675000-04:00",
            "modifiedAt": "2023-04-17T15:25:21.675000-04:00"
        },
        {
            "policyArn": "arn:aws:eks::aws:cluster-access-
policy/AmazonEKSAdminPolicy",
            "accessScope": {
                "type": "namespace",
                "namespaces": [
                    "my-namespace1",
                    "my-namespace2"
                ]
            },
            "associatedAt": "2023-04-17T15:02:06.511000-04:00",
            "modifiedAt": "2023-04-17T15:02:06.511000-04:00"
        }
    ]
}
```

前の例では、このアクセスエントリの IAM プリンシパルには、クラスター上のすべての名前空間に対する表示アクセス許可と、2 つの Kubernetes 名前空間に対する管理者アクセス許可があります。

2. アクセスポリシーとアクセスエントリの関連付けを解除します。この例では、AmazonEKSAdminPolicy ポリシーとアクセスエントリの関連付けが解除されます。ただし、IAM プリンシパルは *my-namespace1* と *my-namespace2* 名前空間のオブジェクトに対する AmazonEKSViewPolicy アクセスポリシー内のアクセス許可を保持します。これは、そのアクセスポリシーがアクセスエントリとの関連付けを解除されていないためです。

```
aws eks disassociate-access-policy --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/my-role \
    --policy-arn arn:aws:eks::aws:cluster-access-policy/AmazonEKSAdminPolicy
```

アクセスポリシーでの許可

アクセスポリシーには、Kubernetes `verbs` (アクセス許可) と `resources` を含む `rules` が含まれます。アクセスポリシーには IAM のアクセス許可やリソースは含まれません。Kubernetes `Role` および `ClusterRole` オブジェクトと同様に、アクセスポリシーには `allow` `rules` が含まれます。アクセスポリシーの内容は変更できません。独自のアクセスポリシーを作成することはできません。アクセスポリシーのアクセス許可がニーズに合わない場合は、Kubernetes RBAC オブジェクトを作成し、アクセスエントリのグループ名を指定します。詳細については、「アクセスエントリの作成」を参照してください。アクセスポリシーに含まれるアクセス許可は、Kubernetes ユーザー向けのクラスターロールのアクセス許可と似ています。詳細については、Kubernetes ドキュメントの「ユーザー向けロール」を参照してください。

アクセスポリシーを選択し、その内容を確認します。各アクセスポリシーの各テーブルの各行は個別のルールです。

AmazonEKSAdminPolicy

このアクセスポリシーには、リソースに対するほとんどのアクセス許可を IAM プリンシパルに付与するアクセス許可が含まれています。アクセスエントリに関連付けられている場合、そのアクセス範囲は通常 1 つ以上の Kubernetes 名前空間です。IAM プリンシパルにクラスター上のすべてのリソースへの管理者アクセスを許可する場合は、代わりに AmazonEKSClusterAdminPolicy アクセスポリシーをアクセスエントリに関連付けてください。

ARN — `arn:aws:eks::aws:cluster-access-policy/AmazonEKSAdminPolicy`

Kubernetes API グループ	Kubernetes のリソース	Kubernetes 動詞 (アクセス許可)
`apps`	`daemonsets` , `deployments` , `deployments/rollback` , `deployments/scale` , `replicasets` , `replicasets/`	`create`, `delete`, `deletecollection` , `patch`, `update`

Kubernetes API グループ	Kubernetes のリソース	Kubernetes 動詞 (アクセス許可)
	scale, statefulsets , statefulsets/scale	
apps	controllerrevisions , daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , replicasets , replicasets/scale , replicasets/status , statefulsets , statefulsets/scale , statefulsets/status	get, list, watch
authorization.k8s.io	localsubjectaccessreviews	create
autoscaling	horizontalpodautoscalers	create, delete, deletecollection , patch, update
autoscaling	horizontalpodautoscalers , horizontalpodautoscalers/status	get, list, watch
batch	cronjobs, jobs	create, delete, deletecollection , patch, update
batch	cronjobs, cronjobs/status , jobs, jobs/status	get, list, watch
discovery.k8s.io	endpointslices	get, list, watch

Kubernetes API グループ	Kubernetes のリソース	Kubernetes 動詞 (アクセス許可)
extensions	daemonsets , deployments , deployments/rollback , deployments/scale , ingresses , networkpolicies , replicasets , replicasets/scale , replicationcontrollers/scale	create, delete, deletecollection , patch, update
extensions	daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , ingresses , ingresses/status , networkpolicies , replicasets , replicasets/scale , replicasets/status , replicationcontrollers/scale	get, list, watch
networking.k8s.io	ingresses , ingresses/status , networkpolicies	get, list, watch
networking.k8s.io	ingresses , networkpolicies	create, delete, deletecollection , patch, update
policy	poddisruptionbudgets	create, delete, deletecollection , patch, update
policy	poddisruptionbudgets , poddisruptionbudgets/status	get, list, watch

Kubernetes API グループ	Kubernetes のリソース	Kubernetes 動詞 (アクセス許可)
rbac.authorization.k8s.io	rolebindings , roles	create, delete, deletecollection , get, list, patch, update, watch
	configmaps , endpoints , persistentvolumeclaims , persistentvolumeclaims/status , pods, replicationcontrollers , replicationcontrollers/scale , serviceaccounts , services, services/status	get,list, watch
	pods/attach , pods/exec , pods/portforward , pods/proxy , secrets, services/proxy	get, list, watch
	configmaps , events, persistentvolumeclaims , replicationcontrollers , replicationcontrollers/scale , secrets, serviceaccounts , services, services/proxy	create, delete, deletecollection , patch, update
	pods, pods/attach , pods/exec , pods/portforward , pods/proxy	create, delete, deletecollection , patch, update

Kubernetes API グループ	Kubernetes のリソース	Kubernetes 動詞 (アクセス許可)
	serviceaccounts	impersonate
	bindings, events, limitranges , namespaces/status , pods/log, pods/status , replicationcontrollers/status , resourcequotas , resourcequotas/status	get, list, watch
	namespaces	get,list, watch

AmazonEKSClusterAdminPolicy

このアクセスポリシーには、IAM プリンシパル管理者にクラスターへのアクセス権を付与するアクセス許可が含まれています。アクセスエントリに関連付けられている場合、そのアクセス範囲は通常、Kubernetes 名前空間ではなくクラスターになります。IAM プリンシパルの管理範囲をより限定したい場合は、代わりに AmazonEKSAdminPolicy アクセスポリシーをアクセスエントリに関連付けることを検討してください。

ARN — arn:aws:eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy

Kubernetes API グループ	Kubernetes nonResourceURLs	Kubernetes のリソース	Kubernetes 動詞 (アクセス許可)
*		*	*
	*		*

AmazonEKSAdminViewPolicy

このアクセスポリシーには、IAM プリンシパルにクラスターの全リソースをリスト/表示するアクセス権を付与するアクセス許可が含まれています。これには Kubernetes Secrets が含まれることに注意してください。

ARN — `arn:aws:eks::aws:cluster-access-policy/AmazonEKSAdminViewPolicy`

Kubernetes API グループ	Kubernetes のリソース	Kubernetes 動詞 (アクセス許可)
*	*	get, list, watch

AmazonEKSEditPolicy

このアクセスポリシーには、IAM プリンシパルが Kubernetes リソースを編集できるようにするアクセス許可が含まれています。

ARN — `arn:aws:eks::aws:cluster-access-policy/AmazonEKSEditPolicy`

Kubernetes API グループ	Kubernetes のリソース	Kubernetes 動詞 (アクセス許可)
apps	daemonsets , deployments , deployments/rollback , deployments/scale , replicasets , replicasets/scale, statefulsets , statefulsets/scale	create, delete, deletecollection , patch, update
apps	controllerrevisions , daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , replicasets , replicasets/scale , replicase	get, list, watch

Kubernetes API グループ	Kubernetes のリソース	Kubernetes 動詞 (アクセス許可)
	ts/status , statefulsets , statefulsets/scale , statefulsets/status	
autoscaling	horizontalpodautoscalers , horizontalpodautoscalers/status	get, list, watch
autoscaling	horizontalpodautoscalers	create, delete, deletecollection , patch, update
batch	cronjobs, jobs	create, delete, deletecollection , patch, update
batch	cronjobs, cronjobs/status , jobs, jobs/status	get, list, watch
discovery.k8s.io	endpointslices	get, list, watch
extensions	daemonsets , deployments , deployments/rollback , deployments/scale , ingresses , networkpolicies , replicasets , replicasets/scale , replicationcontrollers/scale	create, delete, deletecollection , patch, update

Kubernetes API グループ	Kubernetes のリソース	Kubernetes 動詞 (アクセス許可)
extensions	daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , ingresses , ingresses/status , networkpolicies , replicasets , replicasets/scale , replicasets/status , replicationcontrollers/scale	get, list, watch
networking.k8s.io	ingresses , networkpolicies	create, delete, deletecollection , patch, update
networking.k8s.io	ingresses , ingresses/status , networkpolicies	get, list, watch
policy	poddisruptionbudgets	create, delete, deletecollection , patch, update
policy	poddisruptionbudgets , poddisruptionbudgets/status	get, list, watch
	namespaces	get, list, watch
	pods/attach , pods/exec , pods/portforward , pods/proxy , secrets, services/proxy	get, list, watch
	serviceaccounts	impersonate

Kubernetes API グループ	Kubernetes のリソース	Kubernetes 動詞 (アクセス許可)
	pods, pods/attach , pods/exec , pods/port forward , pods/proxy	create, delete, deletecollection , patch, update
	configmaps , events, persistentvolumeclaims , replicationcontrollers , replicationcontrollers/scale , secrets, serviceaccounts , services, services/proxy	create, delete, deletecollection , patch, update
	configmaps , endpoints , persistentvolumeclaims , persistentvolumeclaims/status , pods, replicationcontrollers , replicationcontrollers/scale , serviceaccounts , services, services/status	get, list, watch
	bindings, events, limitranges , namespaces/status , pods/log, pods/status , replicationcontrollers/status , resourcequotas , resourcequotas/status	get, list, watch

AmazonEKSViewPolicy

このアクセスポリシーには、IAM プリンシパルがほとんどの Kubernetes リソースを表示できるようにするアクセス許可が含まれています。

ARN — arn:aws:eks::aws:cluster-access-policy/AmazonEKSViewPolicy

Kubernetes API グループ	Kubernetes のリソース	Kubernetes 動詞 (アクセス許可)
apps	controllerrevisions , daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , replicasets , replicasets/scale , replicasets/status , statefulsets , statefulsets/scale , statefulsets/status	get, list, watch
autoscaling	horizontalpodautoscalers , horizontalpodautoscalers/status	get, list, watch
batch	cronjobs, cronjobs/status , jobs, jobs/status	get, list, watch
discovery.k8s.io	endpointslices	get, list, watch
extensions	daemonsets 、daemonsets/status 、deployments 、deployments/scale、deployments/status 、ingresses 、ingresses/	get, list, watch

Kubernetes API グループ	Kubernetes のリソース	Kubernetes 動詞 (アクセス許可)
	status、networkpolicies 、replicasets 、replicasets/scale、replicasets/status 、replicationcontrollers/scale	
networking.k8s.io	ingresses , ingresses/status , networkpolicies	get, list, watch
policy	poddisruptionbudgets , poddisruptionbudgets/status	get, list, watch
	configmaps , endpoints , persistentvolumeclaims , persistentvolumeclaims/status , pods, replicationcontrollers , replicationcontrollers/scale , serviceaccounts , services, services/status	get, list, watch
	bindings、events、limitranges 、namespaces/status、pods/log、pods/status 、replicationcontrollers/status 、resourcequotas 、resourcequotas/status	get, list, watch

Kubernetes API グループ	Kubernetes のリソース	Kubernetes 動詞 (アクセス許可)
	namespaces	get, list, watch

アクセスポリシーの更新

アクセスポリシーが導入されてからのアクセスポリシーへの更新に関する詳細を表示します。このページの変更に関する自動通知を入手するには、Amazon EKS ドキュメントの履歴ページから、RSS フィードにサブスクライブしてください。

変更	説明	日付
AmazonEKSAdminViewPolicy を追加します。	Secrets などのリソースを含む、拡張ビューアクセス用の新しいポリシーを追加します。	2024 年 4 月 23 日
アクセスポリシーが導入されました。	Amazon EKS はアクセスポリシーを導入しました。	2023 年 5 月 29 日

既存の aws-auth ConfigMap エントリをアクセスエントリに移行する

クラスター上の aws-auth ConfigMap にエントリを追加した場合は、aws-auth ConfigMap 内の既存のエントリのアクセスエントリを作成することをお勧めします。アクセスエントリを作成したら、そのエントリを ConfigMap から削除できます。aws-authConfigMap のエントリにアクセスポリシーを関連付けることはできません。アクセスポリシーを IAM プリンシパルに関連付けるには、アクセスエントリを作成します。

> **Important**
>
> マネージドノードグループまたは Fargate プロファイルをクラスターに追加したときに Amazon EKS によって作成された既存の aws-auth ConfigMap エントリは削除しないでください。Amazon EKS が ConfigMap で作成したエントリを削除すると、クラスターは正しく機能しなくなります。ただし、セルフマネージド型ノードグループのアクセスエントリを作成した後で、それらのエントリを削除することはできます。

前提条件

- アクセスエントリとアクセスポリシーに関する知識。詳細については、アクセスエントリを管理するおよびアクセスポリシーとアクセスエントリとの関連付けおよび関連付け解除を参照してください。
- 「IAM ロールまたはユーザーに Amazon EKS クラスター上の Kubernetes オブジェクトへのアクセスを許可する」のトピックの前提条件に記載されているバージョンと同じかそれ以降のプラットフォームバージョンの既存のクラスター。
- デバイスまたは AWS CloudShell にインストールされている eksctl コマンドラインツールのバージョン 0.183.0 以降。eksctl をインストールまたはアップグレードするには、eksctl ドキュメントの「インストール」を参照してください。
- kube-system 名前空間内の aws-auth ConfigMap を変更する Kubernetes アクセス許可。
- CreateAccessEntry および ListAccessEntries のアクセス許可を持つ AWS Identity and Access Management ロールまたはユーザー。詳細については、サービス認証リファレンスの「Amazon Elastic Kubernetes Service で定義されるアクション」を参照してください。

エントリを **aws-auth ConfigMap** からアクセスエントリに移行するには

1. aws-auth ConfigMap 内の既存のエントリを表示します。*my-cluster* の部分は、自分のクラスター名に置き換えます。

   ```
   eksctl get iamidentitymapping --cluster my-cluster
   ```

   出力例は次のとおりです。

   ```
   ARN
                USERNAME                                GROUPS
                            ACCOUNT
   arn:aws:iam::111122223333:role/EKS-my-cluster-Admins
                Admins                                  system:masters
   arn:aws:iam::111122223333:role/EKS-my-cluster-my-namespace-Viewers
                my-namespace-Viewers                    Viewers
   arn:aws:iam::111122223333:role/EKS-my-cluster-self-managed-ng-1
                               system:node:{{EC2PrivateDNSName}}
    system:bootstrappers,system:nodes
   arn:aws:iam::111122223333:user/my-user
                my-user
   ```

```
arn:aws:iam::111122223333:role/EKS-my-cluster-fargateprofile1
                              system:node:{{SessionName}}
 system:bootstrappers,system:nodes,system:node-proxier
arn:aws:iam::111122223333:role/EKS-my-cluster-managed-ng
                              system:node:{{EC2PrivateDNSName}}
 system:bootstrappers,system:nodes
```

2. 前の出力で返された作成済み ConfigMap エントリのいずれかにアクセスエントリを作成します。アクセスエントリを作成するときは、出力で返されるARN、USERNAME、GROUPS および ACCOUNT と同じ値を指定してください。出力例では、最後の 2 つのエントリを除くすべてのエントリのアクセスエントリを作成します。これらのエントリは Fargate プロファイルとマネージド型ノードグループ用に Amazon EKS によって作成されたものだからです。
3. 作成したすべてのアクセスエントリのエントリを ConfigMap から削除します。ConfigMap からエントリを削除しない場合、IAM プリンシパル ARN のアクセスエントリの設定によって ConfigMap エントリが上書きされます。*111122223333* は AWS アカウント ID に、*EKS-my-cluster-my-namespace-Viewers* は、ConfigMap のエントリに含まれるロールの名前に置き換えます。削除するエントリが IAM ロールではなく IAM ユーザー用のものである場合は、**role** を **user** に、*EKS-my-cluster-my-namespace-Viewers* をユーザー名に置き換えます。

```
eksctl delete iamidentitymapping --arn arn:aws:iam::111122223333:role/EKS-my-
cluster-my-namespace-Viewers --cluster my-cluster
```

クラスターに対する IAM プリンシパルのアクセスの有効化

> **Important**
>
> aws-auth ConfigMap は廃止されました。Kubernetes API へのアクセスを管理する際は、アクセスエントリを使用することをお勧めします。

IAM プリンシパルを使用してクラスターにアクセスします。プリンシパルは、Amazon EKS コントロールプレーンで実行される AWS IAM Authenticator for Kubernetes によって有効になります。オーセンティケーターは、その設定情報を aws-auth ConfigMap から取得します。すべての aws-auth ConfigMap 設定については、GitHub の「完全な設定形式」を参照してください。

Amazon EKS クラスターに IAM プリンシパルを追加する

Amazon EKS クラスターを作成すると、このクラスターを作成する IAM プリンシパルには、Amazon EKS コントロールプレーンのクラスターロールベースアクセスコントロール (RBAC) 設定で、`system:masters` 許可が自動的に付与されます。このプリンシパルは表示可能な設定の中には表示されません。したがって、どのプリンシパルが最初にクラスターを作成したのかを追跡する必要があります。追加の IAM プリンシパルがクラスターとインタラクションできるようにするには、Kubernetes 内の `aws-auth ConfigMap` を編集し、`aws-auth ConfigMap` で指定した `group` の名前を使用して、Kubernetes `rolebinding` または `clusterrolebinding` を作成します。

> **Note**
>
> Kubernetes ロールベースのアクセスコントロール (RBAC) の設定の詳細については、Kubernetes ドキュメントの「RBAC 承認の使用」を参照してください。

IAM プリンシパルを Amazon EKS クラスターに追加するには

1. クラスターへのアクセスに `kubectl` が使用している認証情報を判別します。使用中のコンピュータ上で、次のコマンドを使用して、`kubectl` が使用する認証情報を判別できます。デフォルトのパスを使用しない場合は、***~/.kube/config*** を `kubeconfig` ファイルへのパスに置き換えます。

   ```
   cat ~/.kube/config
   ```

   出力例は次のとおりです。

   ```
   [...]
   contexts:
   - context:
       cluster: my-cluster.region-code.eksctl.io
       user: admin@my-cluster.region-code.eksctl.io
     name: admin@my-cluster.region-code.eksctl.io
   current-context: admin@my-cluster.region-code.eksctl.io
   [...]
   ```

   前述の出力例では、*admin* という名前のユーザーの認証情報が *my-cluster* という名前のクラスター用に設定されています。このユーザーがクラスターを作成したユーザーである場合は、すでにそのクラスターにアクセスできます。クラスターを作成したユーザーでない場合は、以降の

手順を実行して、他の IAM プリンシパルのクラスターへのアクセスを有効にする必要があります。IAM のベストプラクティスでは、ユーザーではなくロールに許可を付与することが推奨されています。次のコマンドを使用すると、現在クラスターへのアクセスが可能な、他のプリンシパルを確認できます。

```
kubectl describe -n kube-system configmap/aws-auth
```

出力例は次のとおりです。

```
Name:         aws-auth
Namespace:    kube-system
Labels:       <none>
Annotations:  <none>

Data
====
mapRoles:
----
- groups:
  - system:bootstrappers
  - system:nodes
  rolearn: arn:aws:iam::111122223333:role/my-node-role
  username: system:node:{{EC2PrivateDNSName}}


BinaryData
====

Events:  <none>
```

前の例は、デフォルトの aws-auth ConfigMap です。ノードインスタンスのロールだけがクラスターにアクセスできます。

2. IAM プリンシパルをマッピングできる既存の Kubernetes roles と rolebindings または clusterroles と clusterrolebindings があることを確認してください。これらのリソースの詳細については、「Kubernetes ドキュメント」の「RBAC 認証の使用」を参照してください。

   1. 既存の Kubernetes roles または clusterroles を表示します。Roles は namespace にスコープされますが、clusterroles はクラスターにスコープされます。

```
kubectl get roles -A
```

```
kubectl get clusterroles
```

2. 前の出力で返された role または clusterrole の詳細を表示します。クラスター内で IAM プリンシパルに必要となる許可 (rules) があることを確認します。

   role-name を、前のコマンド出力で返された role の名前に置き換えます。*kube-system* を role の名前空間に置き換えます。

   ```
   kubectl describe role role-name -n kube-system
   ```

   cluster-role-name を、前のコマンド出力で返された clusterrole の名前に置き換えます。

   ```
   kubectl describe clusterrole cluster-role-name
   ```

3. 既存の Kubernetes rolebindings または clusterrolebindings を表示します。Rolebindings は namespace にスコープされますが、clusterrolebindings はクラスターにスコープされます。

   ```
   kubectl get rolebindings -A
   ```

   ```
   kubectl get clusterrolebindings
   ```

4. rolebinding または clusterrolebinding の詳細を表示して、前のステップで roleRef としてリストされた role または clusterrole、そして subjects にリストされたグループ名があることを確認します。

   role-binding-name を、前のコマンド出力で返された rolebinding の名前に置き換えます。*kube-system* を rolebinding の namespace に置き換えます。

   ```
   kubectl describe rolebinding role-binding-name -n kube-system
   ```

   出力例は次のとおりです。

   ```
   apiVersion: rbac.authorization.k8s.io/v1
   ```

```
kind: RoleBinding
metadata:
  name: eks-console-dashboard-restricted-access-role-binding
  namespace: default
subjects:
- kind: Group
  name: eks-console-dashboard-restricted-access-group
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: eks-console-dashboard-restricted-access-role
  apiGroup: rbac.authorization.k8s.io
```

cluster-role-binding-name を、前のコマンド出力で返された clusterrolebinding の名前に置き換えます。

```
kubectl describe clusterrolebinding cluster-role-binding-name
```

出力例は次のとおりです。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks-console-dashboard-full-access-binding
subjects:
- kind: Group
  name: eks-console-dashboard-full-access-group
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: eks-console-dashboard-full-access-clusterrole
  apiGroup: rbac.authorization.k8s.io
```

3. aws-auth ConfigMap を編集します。eksctl のようなツールにより ConfigMap を更新することができます。あるいは、手動で編集しての更新も可能です。

> **Important**
>
> ConfigMap の編集には、eksctl や、その他のツールを使用することをお勧めします。使用できる他のツールについては、Amazon EKS ベストプラクティスガイドの「ツールを使用して aws-authConfigMap を変更する」を参照してください。aws-

auth ConfigMap の形式が不適切な場合、クラスターへのアクセスが失われる場合があります。

eksctl

前提条件

デバイスまたは AWS CloudShell にインストールされている eksctl コマンドラインツールのバージョン 0.183.0 以降。eksctl をインストールまたはアップグレードするには、eksctl ドキュメントの「インストール」を参照してください。

1. ConfigMap に現在のマッピングを表示します。*my-cluster* を自分のクラスター名に置き換えます。*region-code* をクラスターのある AWS リージョン に置き換えます。

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

出力例は次のとおりです。

```
ARN
                    USERNAME                                GROUPS
              ACCOUNT
arn:aws:iam::111122223333:role/eksctl-my-cluster-my-nodegroup-
NodeInstanceRole-1XLS7754U3ZPA     system:node:{{EC2PrivateDNSName}}
 system:bootstrappers,system:nodes
```

2. ロールのマッピングを追加します。*my-role* をロール名前で置き換えます。*eks-console-dashboard-full-access-group* を Kubernetes RoleBinding または ClusterRoleBinding オブジェクトで指定されたグループの名前に置き換えます。*111122223333* をアカウント ID に置き換えます。*###*を任意の名前に置き換えることができます。

```
eksctl create iamidentitymapping --cluster my-cluster --region=region-code \
    --arn arn:aws:iam::111122223333:role/my-role --username admin --group eks-
console-dashboard-full-access-group \
    --no-duplicate-arns
```

> **Important**
>
> ロールの ARN には、role/my-team/developers/my-role などのパスを含めることはできません。ロールの ARN は、arn:aws:iam::*111122223333*:role/*my-role* のような形式にする必要があります。この例では、my-team/developers/ を削除する必要があります。

出力例は次のとおりです。

```
[...]
2022-05-09 14:51:20 [#]  adding identity "arn:aws:iam::111122223333:role/my-
role" to auth ConfigMap
```

3. ユーザーのマッピングを追加します。IAM のベストプラクティスでは、ユーザーではなくロールに許可を付与することが推奨されています。*my-user* をユーザー名に置き換えます。*eks-console-dashboard-restricted-access-group* を Kubernetes RoleBinding または ClusterRoleBinding オブジェクトで指定されたグループの名前に置き換えます。*111122223333* をアカウント ID に置き換えます。*my-user* を任意の名前に置き換えることができます。

```
eksctl create iamidentitymapping --cluster my-cluster --region=region-code \
    --arn arn:aws:iam::111122223333:user/my-user --username my-user --
group eks-console-dashboard-restricted-access-group \
    --no-duplicate-arns
```

出力例は次のとおりです。

```
[...]
2022-05-09 14:53:48 [#]  adding identity "arn:aws:iam::111122223333:user/my-
user" to auth ConfigMap
```

4. ConfigMap のマッピングを再度表示します。

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

出力例は次のとおりです。

```
ARN
                    USERNAME                                      GROUPS
                      ACCOUNT
arn:aws:iam::111122223333:role/eksctl-my-cluster-my-nodegroup-
NodeInstanceRole-1XLS7754U3ZPA    system:node:{{EC2PrivateDNSName}}
 system:bootstrappers,system:nodes
arn:aws:iam::111122223333:role/admin
                        my-role                                       eks-console-
dashboard-full-access-group
arn:aws:iam::111122223333:user/my-user
                        my-user                                       eks-console-
dashboard-restricted-access-group
```

Edit ConfigMap manually

1. 編集する ConfigMap を開きます。

```
kubectl edit -n kube-system configmap/aws-auth
```

> **Note**
>
> 「Error from server (NotFound): configmaps "aws-auth" not found」というエラーが表示された場合は、aws-auth ConfigMap をクラスターに適用する の手順を使用してストック ConfigMap を適用します。

2. IAM プリンシパルを ConfigMap に追加します。IAM グループは IAM プリンシパルではないため、ConfigMap に追加できません。
   - IAM ロールを (たとえば、フェデレーティッドユーザーに) 追加するには、ConfigMap の mapRoles セクションの data の下にロールの詳細を追加します。ファイルに存在しない場合は、このセクションを追加します。各エントリは、次のパラメータをサポートしています。
     - rolearn: 追加する IAM ロールの ARN。この値にパスを含めることはできません。例えば、arn:aws:iam::*111122223333*:role/my-team/developers/*role-name* のような ARN を指定することはできません。ARN は、arn:aws:iam::*111122223333*:role/*role-name* のようにする必要があります。

  - ユーザー名: IAM ロールにマッピングする Kubernetes 内のユーザー名。
  - グループ: ロールをマップするグループまたは Kubernetes グループのリスト。グループには、デフォルトグループか、`clusterrolebinding` または `rolebinding` で指定されたグループを使用します。詳細については、Kubernetes ドキュメントの「デフォルトのロールとロールのバインディング」を参照してください。
- IAM ユーザーを追加するには: IAM のベストプラクティスでは、ユーザーではなくロールに許可を付与することが推奨されています。`ConfigMap` の `mapUsers` セクション (`data` の下) にユーザーの詳細を追加します。ファイルに存在しない場合は、このセクションを追加します。各エントリは、次のパラメータをサポートしています。
  - userarn: 追加する IAM ユーザーの ARN。
  - ユーザー名: IAM ユーザーにマッピングする Kubernetes 内のユーザー名。
  - グループ: ユーザーをマップするグループまたは Kubernetes グループのリスト。グループには、デフォルトグループか、`clusterrolebinding` または `rolebinding` で指定されたグループを使用します。詳細については、Kubernetes ドキュメントの「デフォルトのロールとロールのバインディング」を参照してください。

例えば、次の YAML ブロックには次が含まれます。

- ノードがそれ自体をクラスター、およびすべてのクラスターですべての Kubernetes リソースを表示することが可能な Kubernetes グループにマッピングされた `my-console-viewer-role` IAM ロールに登録できるように、IAM ノードインスタンスを Kubernetes グループにマッピングする `mapRoles` セクション。`my-console-viewer-role` IAM ロールに必要な IAM と Kubernetes グループのアクセス許可のリストについては、「必要なアクセス許可」を参照してください。
- `system:masters` Kubernetes グループ、および特定の名前空間で Kubernetes リソースを表示することが可能な、Kubernetes グループにマッピングされた他の AWS アカウントからの `my-user` ユーザーに、デフォルトの AWS アカウントからの `admin` IAM ユーザーをマッピングする `mapUsers` セクション。`my-user` IAM ユーザーに必要な IAM と Kubernetes グループのアクセス許可のリストについては、「必要なアクセス許可」を参照してください。

必要に応じて行を追加または削除し、すべて *`example values`* を、実際の値で置き換えます。

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this
 file will be
```

```
# reopened with the relevant failures.
#
apiVersion: v1
data:
  mapRoles: |
    - groups:
      - system:bootstrappers
      - system:nodes
      rolearn: arn:aws:iam::111122223333:role/my-role
      username: system:node:{{EC2PrivateDNSName}}
    - groups:
      - eks-console-dashboard-full-access-group
      rolearn: arn:aws:iam::111122223333:role/my-console-viewer-role
      username: my-console-viewer-role
  mapUsers: |
    - groups:
      - system:masters
      userarn: arn:aws:iam::111122223333:user/admin
      username: admin
    - groups:
      - eks-console-dashboard-restricted-access-group
      userarn: arn:aws:iam::444455556666:user/my-user
      username: my-user
```

3. ファイルを保存し、テキストエディタを終了します。

aws-auth ConfigMap をクラスターに適用する

マネージド型ノードグループを作成するとき、または aws-auth を使用してノードグループを作成するときに、ConfigMap eksctl が自動的に作成され、クラスターに適用されます。これはノードをクラスターに追加するために当初作成されたものですが、この ConfigMap を使用して、ロールベースアクセスコントロール (RBAC) アクセスを IAM プリンシパルに追加することもできます。セルフマネージド型ノードが起動済みで、クラスターに aws-auth ConfigMap を適用していない場合は、次の手順に従って適用できます。

aws-authConfigMap をクラスターに適用するには

1. aws-auth ConfigMap が適用済みであるかどうかを確認します。

```
kubectl describe configmap -n kube-system aws-auth
```

「Error from server (NotFound): configmaps "aws-auth" not found」というエラーが表示された場合は、以下のステップを実行してストック ConfigMap を適用します。

2. AWS オーセンティケーター設定マップをダウンロード、編集、適用します。

   a. 設定マップをダウンロードします。

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-
   eks/cloudformation/2020-10-29/aws-auth-cm.yaml
   ```

   b. aws-auth-cm.yaml ファイルで、ノードに関連付けられている IAM ロールの Amazon リソースネーム (ARN) に rolearn を設定します。これを行うには、テキストエディタを使用するか、*my-node-instance-role* を置き換えて次のコマンドを実行します。

   ```
   sed -i.bak -e 's|<ARN of instance role (not instance profile)>|my-node-
   instance-role|' aws-auth-cm.yaml
   ```

   このファイルの他の行は変更しないでください。

   > **Important**
   >
   > ロールの ARN には、role/my-team/developers/my-role などのパスを含めることはできません。ロールの ARN は、arn:aws:iam::*111122223333*:role/*my-role* のような形式にする必要があります。この例では、my-team/developers/ を削除する必要があります。

   ノードグループの AWS CloudFormation スタック出力を検査し、次の値を探すことができます。

   - InstanceRoleARN – eksctl で作成されたノードグループ用
   - NodeInstanceRole – AWS Management Console で、Amazon EKS で発行された AWS CloudFormation テンプレートで作成されたノードグループ用

   c. 設定を適用します。このコマンドが完了するまで数分かかることがあります。

   ```
   kubectl apply -f aws-auth-cm.yaml
   ```

> **Note**
>
> 認証またはリソースタイプのエラーが発生した場合は、トラブルシューティングトピックの「許可されていないか、アクセスが拒否されました (kubectl)」を参照してください。

3. ノードのステータスを監視し、Ready ステータスになるまで待機します。

```
kubectl get nodes --watch
```

Ctrl+C を入力して、シェルプロンプトに戻ります。

OpenID Connect アイデンティティプロバイダーからクラスターのユーザーを認証する

Amazon EKS では、クラスターに対してユーザーを認証する方法として OpenID Connect (OIDC) ID プロバイダーの使用がサポートされています。OIDC ID プロバイダーは、AWS Identity and Access Management (IAM) と併用、または代替として使用できます。IAM の使用の詳細については、「the section called “Kubernetes API へのアクセスを許可する”」を参照してください。クラスターに認証を設定した後、Kubernetes roles および clusterroles を作成してロールにアクセス許可を割り当て、それから Kubernetes rolebindings および clusterrolebindings を使用してアイデンティティにロールをバインドできます。詳細については、「Kubernetes ドキュメント」の「RBAC 認証の使用」を参照してください。

考慮事項

- クラスターには OIDC ID プロバイダーを 1 つ関連付けることができます。
- Kubernetes は OIDC ID プロバイダーを提供しません。既存のパブリック OIDC ID プロバイダーを使用することも、独自の ID プロバイダーを実行することもできます。認定プロバイダーのリストについては、OpenID サイトの「OpenID Certification」(OpenID 認定) を参照してください。
- Amazon EKS が署名キーを検出できるように、OIDC ID プロバイダーの発行者 URL は公的にアクセス可能である必要があります。Amazon EKS は、自己署名証明書を持つ OIDC ID プロバイダーをサポートしていません。
- ノードをクラスターに結合する時に必要なため、クラスターへの IAM 認証を無効にすることはできません。

- Amazon EKS クラスターは、OIDC ID プロバイダーユーザーではなく、AWS IAM プリンシパルが作成する必要があります。これは、クラスター作成者が Kubernetes API ではなく Amazon EKS API とやり取りしているためです。
- コントロールプレーンで CloudWatch ログが有効になっている場合、OIDC ID プロバイダー認証ユーザーは、クラスターの監査ログに一覧表示されます。詳細については、「コントロールプレーンログの有効化と無効化」を参照してください。
- OIDC プロバイダーのアカウントで AWS Management Console にサインインすることはできません。Kubernetes リソースの表示 は、AWS Identity and Access Management アカウントで AWS Management Console にサインインすることで、コンソールのみで可能となります。

OIDC ID プロバイダーを関連付ける

OIDC ID プロバイダーをクラスターに関連付けるには、プロバイダーから以下の情報が必要です。

発行者 URL

API サーバーがトークン検証のために公開署名キーを検出できるようにする OIDC ID プロバイダーの URL。URL は `https://` で始まり、プロバイダーの OIDC ID トークンで `iss` クレームに対応している必要があります。OIDC 標準に従って、パスコンポーネントは許可されますが、クエリパラメータは許可されません。通常 URL は `https://server.example.org` または `https://example.com` のように、ホスト名のみで構成されます。この URL は、以下 `.well-known/openid-configuration` のレベルを指し、インターネット経由で公的にアクセス可能である必要があります。

クライアント ID (オーディエンスとも呼ばれる)

OIDC ID プロバイダーに認証リクエストを行うクライアントアプリケーションの ID。

`eksctl` または AWS Management Console を使用して、ID プロバイダーを関連付けることができます。

eksctl

`eksctl` を使用して、クラスターに OIDC ID プロバイダーを関連付けるには

1. 次の内容で、*`associate-identity-provider.yaml`* という名前のファイルを作成します。*`example values`* を自分の値に置き換えます。`identityProviders` セクションの値は、OIDC ID プロバイダーから取得します。値は、`identityProviders` での `name`、`type`、`issuerUrl`、および `clientId` の設定のみで必要です。

```
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: your-region-code

identityProviders:
  - name: my-provider
    type: oidc
    issuerUrl: https://example.com
    clientId: kubernetes
    usernameClaim: email
    usernamePrefix: my-username-prefix
    groupsClaim: my-claim
    groupsPrefix: my-groups-prefix
    requiredClaims:
      string: string
    tags:
      env: dev
```

> **Important**
>
> groupsPrefix または usernamePrefix に対しては、system: またはその文字列の一部でも指定しないでください。

2. プロバイダーを作成します。

```
eksctl associate identityprovider -f associate-identity-provider.yaml
```

3. kubectl を使用してクラスターと OIDC ID プロバイダーを操作する方法については、Kubernetes ドキュメントの「kubectl の使用」を参照してください。

AWS Management Console

AWS Management Console を使用して、クラスターに OIDC ID プロバイダーを関連付けるには

1. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。

2. クラスターを選択し、[アクセス] タブを選択します。
3. [OIDC ID プロバイダー] セクションで、[ID プロバイダーを関連付ける] を選択します。
4. [OIDC ID プロバイダーの関連付け] ページで、以下のオプションを入力または選択した上で、[関連付け] を選択します。

   - [Name] (名前) に、プロバイダーの一意の名前を入力します。
   - [Issuer URL] (発行者 URL) に、プロバイダーの URL を入力します。この URL は、インターネットからアクセス可能である必要があります。
   - [クライアント ID] に、OIDC ID プロバイダーのクライアント ID (別名、対象者) を入力します。
   - [Username claim] (ユーザー名のクレーム) に、ユーザー名として使用するクレームを入力します。
   - [Groups claim] (グループクレーム) に、ユーザーのグループとして使用するクレームを入力します。
   - (オプション) [Advanced options] (詳細オプション) で、以下の情報を入力または選択します。
     - [Username prefix] (ユーザー名のプレフィックス) – ユーザー名クレームの前に付加するプレフィックスを入力します。プレフィックスは、既存の名前との衝突を防ぐために、ユーザー名クレームの前に付加されます。値を入力せず、ユーザーネームが `email` 以外の場合、プレフィックスはデフォルトで [Issuer URL] (発行者 URL) の値になります。値に `-` を使用すると、すべてのプレフィックスを無効にできます `system:` またはその文字列の一部でも指定しないでください。
     - [Groups prefix] (グループプレフィックス) – グループクレームの前に付加するプレフィックスを入力します。プレフィックスは、 `system: groups` など既存の名前との衝突を防ぐために、グループクレームの前に付加されます。例えば、値 `oidc:` は、`oidc:engineering` および `oidc:infra` などのグループ名を作成します。`system:` またはその文字列の一部でも指定しないでください。
     - [Required claims] (必要なクレーム) – [Add claim] (クレームを追加) を選択し、クライアント ID トークンに必要なクレームを記述する、キーと値のペアを 1 つ以上入力します。ペアは、ID トークンで必要なクレームを記述します。設定されている場合、各クレームは、一致する値を持つ ID トークンに存在することが検証されます。
5. `kubectl` を使用してクラスターと OIDC ID プロバイダーを操作する方法については、Kubernetes ドキュメントの「kubectl の使用」を参照してください。

OIDC ID プロバイダーとクラスターの関連付けを解除する

OIDC ID プロバイダーとクラスターの関連付けを解除する場合、プロバイダーに含まれるユーザーはクラスターにアクセスできなくなります。ただし、IAM プリンシパルを使用して、クラスターへ引き続きアクセスすることは可能です。

AWS Management Console を使用して、OIDC ID プロバイダーとクラスターの関連付けを解除するには

1. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。
2. [OIDC ID プロバイダー] セクションで、[関連付けの解除] を選択し ID プロバイダー名を入力した上で、[Disassociate] を選択します。

IAM ポリシーの例

OIDC ID プロバイダーとクラスターの関連付けを防ぐには、以下の IAM ポリシーを作成して、Amazon EKS 管理者の IAM アカウントに関連付けます。詳細については、「IAM ユーザーガイド」の「IAM ポリシーの作成」と「Adding IAM identity permissions」(IAM ID アクセス許可の追加)、「サービス認証リファレンス」の「Amazon Elastic Kubernetes Service のアクション、リソース、条件キー」を参照してください。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "denyOIDC",
            "Effect": "Deny",
            "Action": [
                "eks:AssociateIdentityProviderConfig"
            ],
            "Resource": "arn:aws:eks:us-west-2.amazonaws.com:111122223333:cluster/*"

        },
        {
            "Sid": "eksAdmin",
            "Effect": "Allow",
            "Action": [
                "eks:*"
            ],
            "Resource": "*"
        }
```

```
    ]
}
```

以下のポリシーの例では、clientID が kubernetes で、issuerUrl が https://cognito-idp.us-west-2amazonaws.com/* の場合、OIDC ID プロバイダーの関連付けを許可します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCognitoOnly",
            "Effect": "Deny",
            "Action": "eks:AssociateIdentityProviderConfig",
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-instance",
            "Condition": {
                "StringNotLikeIfExists": {
                    "eks:issuerUrl": "https://cognito-idp.us-west-2.amazonaws.com/*"
                }
            }
        },
        {
            "Sid": "DenyOtherClients",
            "Effect": "Deny",
            "Action": "eks:AssociateIdentityProviderConfig",
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-instance",
            "Condition": {
                "StringNotEquals": {
                    "eks:clientId": "kubernetes"
                }
            }
        },
        {
            "Sid": "AllowOthers",
            "Effect": "Allow",
            "Action": "eks:*",
            "Resource": "*"
        }
    ]
}
```

パートナー検証済み OIDC ID プロバイダー

Amazon EKS は、互換性のある OIDC ID プロバイダーのサポートを提供するパートナーネットワークと連携しています。ID プロバイダーと Amazon EKS との統合の詳細については、次のパートナーのドキュメントを参照してください。

パートナー	製品	ドキュメント
PingIdentity	PingOne for Enterprise	インストール手順

Amazon EKS では、すべてのユースケースをカバーする幅広いオプションの提供を目指しています。ここに記載されていない商用サポート対象の OIDC 互換 ID プロバイダーを開発する場合、パートナーチーム aws-container-partners@amazon.com に詳細をお問い合わせください。

Amazon EKS クラスターの **kubeconfig** ファイルを作成または更新する

このトピックでは、クラスター用の kubeconfig ファイルを作成します (または既存のファイルを更新します)。

kubectl コマンドラインツールは、kubeconfig ファイルの設定情報を使用して、クラスターの API サーバーと通信します。詳細については、Kubernetes ドキュメントの「kubeconfig ファイルを使用したクラスターアクセスの整理」を参照してください。

Amazon EKS はクラスター認証に kubectl で aws eks get-token コマンドを使用します。デフォルトでは、AWS CLI は次のコマンドで返されるものと同じ認証情報を使用します。

```
aws sts get-caller-identity
```

前提条件

- 既存の Amazon EKS クラスター。デプロイするには、「Amazon EKS の使用開始」を参照してください。
- デバイスまたは AWS CloudShell に、kubectl コマンドラインツールがインストールされていること。バージョンは、ご使用のクラスターの Kubernetes バージョンと同じか、1 つ前のマイナーバージョン以前、あるいはそれより新しいバージョンが使用できます。例えば、クラスターのバー

ジョンが 1.29 である場合、kubectl のバージョン 1.28、1.29、または 1.30 が使用できます。kubectl をインストールまたはアップグレードする方法については、「kubectl のインストールまたは更新」を参照してください。

- ご使用のデバイスまたは AWS CloudShell で、バージョン 2.12.3 以降、または AWS Command Line Interface (AWS CLI) のバージョン 1.27.160 以降がインストールおよび設定されていること。現在のバージョンを確認するには、「**aws --version | cut -d / -f2 | cut -d ' ' -f1**」を参照してください。macOS の yum、apt-get、または Homebrew などのパッケージマネージャは、AWS CLI の最新バージョンより数バージョン遅れることがあります。最新バージョンをインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI のインストール、更新、およびアンインストール」と「aws configure でのクイック設定」を参照してください。AWS CloudShell にインストールされている AWS CLI バージョンは、最新バージョンより数バージョン遅れている可能性もあります。更新するには、「AWS CloudShell ユーザーガイド」の「ホームディレクトリへの AWS CLI のインストール」を参照してください。
- 指定したクラスターに対して eks:DescribeCluster API アクションを使用するアクセス許可を持つ IAM ユーザーまたはロール。詳細については、「Amazon EKS でのアイデンティティベースのポリシーの例」を参照してください。独自の OpenID Connect プロバイダーの ID を使用してクラスターにアクセスする場合は、Kubernetes ドキュメントの「kubectl の使用」を参照して、kube config ファイルを作成または更新してください。

kubeconfig ファイルを自動で作成する

前提条件

- ご使用のデバイスまたは AWS CloudShell で、バージョン 2.12.3 以降、または AWS Command Line Interface (AWS CLI) のバージョン 1.27.160 以降がインストールおよび設定されていること。現在のバージョンを確認するには、「**aws --version | cut -d / -f2 | cut -d ' ' -f1**」を参照してください。macOS の yum、apt-get、または Homebrew などのパッケージマネージャは、AWS CLI の最新バージョンより数バージョン遅れることがあります。最新バージョンをインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI のインストール、更新、およびアンインストール」と「aws configure でのクイック設定」を参照してください。AWS CloudShell にインストールされている AWS CLI バージョンは、最新バージョンより数バージョン遅れている可能性もあります。更新するには、「AWS CloudShell ユーザーガイド」の「ホームディレクトリへの AWS CLI のインストール」を参照してください。
- 指定したクラスターに対して eks:DescribeCluster API アクションを使用するアクセス許可。詳細については、「Amazon EKS でのアイデンティティベースのポリシーの例」を参照してください。

kubeconfig ファイルを AWS CLI で作成するには

1. クラスター用の kubeconfig ファイルを作成もしくは更新します。*region-code* をお客様のクラスターが存在する AWS リージョン に置き換え、*my-cluster* をお客様のクラスターの名前に置き換えます。

   ```
   aws eks update-kubeconfig --region region-code --name my-cluster
   ```

   デフォルトでは、最終的な設定ファイルは、ホームディレクトリのデフォルト kubeconfig パス (.kube) に作成されるか、その場所で既存 config ファイルとマージされます。別のパスは **--kubeconfig** オプションを使用して指定できます。

   kubectl コマンドを発行する場合、認証に使用する **--role-arn** オプションで IAM ロール ARN を指定できます。それ以外の場合は、デフォルトの AWS CLI または SDK の認証情報チェーンの中の IAM プリンシパルが使用されます。デフォルトの AWS CLI または SDK の ID を表示するには、 `aws sts get-caller-identity` コマンドを実行します。

   使用可能なすべてのオプションについては、`aws eks update-kubeconfig help` コマンドを実行するか、「AWS CLI コマンドリファレンス」の「update-kubeconfig」を参照してください。
2. 設定をテストします。

   ```
   kubectl get svc
   ```

   出力例は次のとおりです。

   ```
   NAME             TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
   svc/kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   1m
   ```

   認証またはリソースタイプのエラーが発生した場合は、トラブルシューティングトピックの「許可されていないか、アクセスが拒否されました (kubectl)」を参照してください。

Kubernetes ワークロードに Kubernetes サービスアカウントを使用して AWS へのアクセスを許可する

Kubernetes サービスアカウントは、Pod で実行されるプロセスのアイデンティティを提供します。詳細については、「Kubernetes ドキュメント」の「サービスアカウントの管理」を参照してくださ

い。Pod が AWS サービスにアクセスする必要がある場合、サービスアカウントを AWS Identity and Access Management のアイデンティティにマッピングして、アクセス権を付与することができます。詳細については、「サービスアカウントの IAM ロール」を参照してください。

サービスアカウントトークン

バージョン Kubernetes では、`BoundServiceAccountTokenVolume` 機能がデフォルトで有効になっています。この機能により Kubernetes で実行しているワークロードがオーディエンス、時間、およびキーバインド化された JSON Web トークンをリクエストできるようにすることで、サービスアカウントトークンのセキュリティが向上します。サービスアカウントトークンの有効期限は 1 時間です。Kubernetes の以前のバージョンでは、トークンに有効期限はありませんでした。つまり、これらのトークンに依存するクライアントは 1 時間以内にトークンを更新する必要があります。次の Kubernetes クライアント SDK では、必要な時間内にトークンを自動的に更新します。

- Go バージョン `0.15.7` 以降
- Python バージョン `12.0.0` 以降
- Java バージョン `9.0.0` 以降
- JavaScript バージョン `0.10.3` 以降
- Ruby `master` ブランチ
- Haskell バージョン`0.3.0.0`
- C# バージョン `7.0.5` 以降

ワークロードで古いバージョンのクライアントを使用している場合は、更新する必要があります。有効期限付きの新しいサービスアカウントトークンにクライアントがスムーズに移行できるようにするに、Kubernetes ではデフォルトの 1 時間を超えてサービスアカウントトークンの有効期限が延長されます。Amazon EKS クラスターの場合、延長できる有効期限は 90 日です。Amazon EKS クラスターの Kubernetes API サーバーでは、90 日を超えるトークンのリクエストは拒否されます。アプリケーションとその依存関係を確認し、Kubernetes クライアント SDK が前記のバージョンと同じかそれ以降であることを確認することをお勧めします。

API サーバーが 1 時間を超える古いトークンによるリクエストを受信すると、API 監査ログイベントに `annotations.authentication.k8s.io/stale-token` で注釈を付けます。注釈の値は次の例のようになります。

```
subject: system:serviceaccount:common:fluent-bit, seconds after warning threshold:
 4185802.
```

クラスターでコントロールプレーンのログ記録が有効になっている場合、注釈は監査ログに記録されます。以下の CloudWatch Log Insights クエリを使用すると、Amazon EKS クラスター内で古いトークンを使っているすべての Pods を特定できます。

```
fields @timestamp
| filter @logStream like /kube-apiserver-audit/
| filter @message like /seconds after warning threshold/
| parse @message "subject: *, seconds after warning threshold:*\"" as subject,
 elapsedtime
```

`subject` は、Pod が使用したサービスアカウントを示します。`elapsedtime` は、最新のトークンを読み込んでからの経過時間 (秒単位) を示します。`elapsedtime` が 90 日 (7,776,000 秒間) を超えると、API サーバーへのリクエストは拒否されます。トークンの自動更新を行う前記のバージョンのいずれかを使用するよう、アプリケーションの Kubernetes クライアント SDK をプロアクティブに更新する必要があります。使用しているサービスアカウントトークンが 90 日に近く、かつトークンの有効期限が切れる前にクライアント SDK のバージョンを更新するために十分な時間がない場合、既存の Pods を終了して新しいものを作成できます。サービスアカウントトークンが再フェッチされるので、クライアントバージョン SDK を更新するのに 90 日が追加されたことになります。

Pod がデプロイの一部である場合、高可用性を維持しながら Pods を終了する方法として、次のコマンドによるロールアウトの実行をお勧めします。*my-deployment* をデプロイの名前で置き換えます。

```
kubectl rollout restart deployment/my-deployment
```

クラスターアドオン

以下のクラスターアドオンが更新され、サービスアカウントトークンを自動的に再フェッチする Kubernetes クライアント SDK が使用できるようになりました。リストされているバージョン、またはそれ以降のバージョンを、 クラスターにインストールすることをお勧めします。

- Amazon VPC CNI plugin for Kubernetes およびメトリクスヘルパーのプラグインのバージョン `1.8.0` 以降。現在のバージョンを確認したり、更新したりするには、「Amazon VPC CNI plugin for Kubernetes Amazon EKS アドオンの使用」および「cni-metrics-helper」を参照してください。
- CoreDNS バージョン `1.8.4` 以降。現在のバージョンを確認したり、更新したりするには、「CoreDNS Amazon EKS アドオンの使用」を参照してください。
- AWS Load Balancer Controller バージョン `2.0.0` 以降。現在のバージョンを確認したり、更新したりするには、「AWS Load Balancer Controller とは」を参照してください。

- 現在の kube-proxy バージョン 現在のバージョンを確認したり、更新したりするには、「Kubernetes kube-proxy アドオンの使用」を参照してください。
- AWS for Fluent Bit バージョン 2.25.0 以降。現在のバージョンを更新するには、GitHub で「Releases」(リリース) を参照してください。
- Fluentd イメージバージョン 1.14.6-1.2 以降、および Kubernetes メタデータバージョン 2.11.1 以降用の Fluentd フィルタープラグイン。

Amazon Elastic Kubernetes Service スクラスター上のワークロードへの AWS Identity and Access Management 権限の付与

Amazon EKS には、Amazon EKS クラスターで実行されるワークロードに AWS Identity and Access Management 権限を付与する方法が 2 つあります。サービスアカウント用の IAM ロールと EKS Pod Identity です。

サービスアカウントの IAM ロール

サービスアカウント用の IAM ロール (IRSA) は、Amazon S3 バケットや Amazon DynamoDB テーブルなど、他の様々な AWS リソースにアクセスするためのきめ細かい IAM アクセス許可で AWS 上で動作する Kubernetes アプリケーションを構成します。同じ Amazon EKS クラスター内で複数のアプリケーションを同時に実行し、各アプリケーションに必要な最小限のアクセス権限のみを持たせることができます。IRSAは、Amazon EKS、Amazon EKS Anywhere、Red Hat OpenShift Service on AWS、Amazon EC2 インスタンス上のセルフマネージド Kubernetes クラスターなどの、AWS にサポートされるさまざまな Kubernetes デプロイオプションをサポートするように構築されました。そのため、IRSA は IAM のような基本的な AWS サービスを使用して構築されており、Amazon EKS サービスや EKS API に直接依存することはありませんでした。詳細については、「サービスアカウントの IAM ロール」を参照してください。

EKS Pod Identity

EKS Pod Identity は、Amazon S3 バケット、Amazon DynamoDB テーブルなどのさまざまな AWS リソースにアクセスするためのアプリケーションを認証するためのシンプルなワークフローをクラスター管理者に提供します。EKS Pod Identity は EKS 専用であるため、クラスター管理者が Kubernetes アプリケーションを設定して IAM アクセス許可を取得する方法が簡単になります。これらの権限は、AWS Management Console、EKS API、および AWS CLI から直接行うより少ない手順で簡単に設定できるようになり、クラスター内のどの Kubernetes オブジェクトに対しても実行する必要がありません。クラスター管理者は EKS サービスと IAM サービスを切り替えたり、特権的な IAM オペレーションを使用してアプリケーションに必要な権限を設定したりする必要がありません。新しいクラスターを作成するときにロールの信頼ポリシーを更新し

なくても、IAM ロールを複数のクラスターで使用できるようになりました。EKS Pod Identity が提供する IAM 認証情報には、クラスター名、名前空間、サービスアカウント名などの属性を含むロールセッションタグが含まれます。ロールセッションタグを使用すると、管理者は一致するタグに基づいて AWS リソースへのアクセスを許可することで、複数のサービスアカウントで機能する単一のロールを作成できます。詳細については、「EKS Pod Identity」を参照してください。

EKS Pod Identity と IRSA の比較

大まかに言うと、EKS Pod Identity と IRSA はどちらも、Kubernetes クラスター上で実行されているアプリケーションに IAM アクセス許可を付与できるようにします。ただし、設定方法、サポートされる制限、有効になる機能は根本的に異なります。以下では、両ソリューションの重要な側面をいくつか比較します。

	EKS Pod Identity	IRSA
ロール拡張性	新しく導入された Amazon EKS サービスプリンシパル `pods.eks.amazonaws.com` との信頼を確立するには、各ロールを一度設定する必要があります。この 1 回限りのステップを済ませると、そのロールを新しいクラスターで使用するたびにそのロールの信頼ポリシーを更新する必要がなくなります。	新しいクラスターでロールを使用するたびに、IAM ロールの信頼ポリシーを新しい EKS クラスター OIDC プロバイダーエンドポイントで更新する必要があります。
クラスタのスケーラビリティ	EKS Pod Identity では、ユーザーが IAM OIDC プロバイダーをセットアップする必要がないため、この制限は適用されません。	クラスターには、OpenID Connect (OIDC) 発行者の URL が関連付けられています。IRSA を使用するには、IAM の EKS クラスターごとに固有の OpenID Connect プロバイダーを作成する必要があります。IAM のデフォルトのグローバル制限は、AWS ア

	EKS Pod Identity	IRSA
		カウント につき 100 の OIDC プロバイダーです。IRSA で AWS アカウント につき 100 を超える EKS クラスターを使用する予定の場合は、IAM OIDC プロバイダーの制限に達します。
ロールのスケーラビリティ	EKS Pod Identity では、ユーザーが信頼ポリシーで IAM ロールとサービスアカウント間の信頼関係を定義する必要がないため、この制限は適用されません。	IRSA では、IAM ロールとサービスアカウント間の信頼関係を、ロールの信頼ポリシーで定義します。デフォルトでは、信頼ポリシーの長さは 2048 です。つまり、通常、1 つの信頼ポリシーで 4 つの信頼関係を定義できます。信頼ポリシーの長さの上限を増やすことはできますが、通常、1 つの信頼ポリシー内で作成できる信頼関係は最大 8 つに制限されます。

	EKS Pod Identity	IRSA
ロールの再利用性	EKS Pod Identity によって提供される一時的な AWS STS 認証情報には、クラスター名、名前空間、サービスアカウント名などのロールセッションタグが含まれます。ロールセッションタグを使用すると、管理者は、アタッチされたタグに基づいて AWS リソースへのアクセスを許可することで、複数のサービスアカウントで使用できる単一の IAM ロールを作成し、そのロールを有効な権限を変えることができます。これは、属性ベースのアクセス制御 (ABAC) とも呼ばれます。詳細については、「EKS Pod Identity の権限を定義してタグに基づいて役割を引き受ける」を参照してください。	AWS STS セッションタグはサポートされていません。ロールはクラスタ間で再利用できますが、すべてのポッドにそのロールのすべての権限が付与されます。
サポートされている環境	EKS Pod Identity は Amazon EKS でのみ使用できます。	IRSAは、Amazon EKS、Amazon EKS Anywhere、Red Hat OpenShift Service on AWS、Amazon EC2インスタンス上のセルフマネージド Kubernetes クラスターなどに使用できます。

	EKS Pod Identity	IRSA
サポートされている EKS バージョン	EKS Kubernetes のバージョン 1.24 以降 。特定のバージョンについては、「EKS Pod Identity クラスターバージョン」を参照してください。	サポート対象のすべての EKS クラスターバージョン。

EKS Pod Identity

Pod のコンテナ内のアプリケーションは AWS SDK または AWS CLI で、AWS のサービス (IAM) アクセス許可を使用した AWS Identity and Access Management への API リクエストを行うことができます。アプリケーションは AWS 認証情報で AWS API リクエストに署名する必要があります。

EKS Pod Identity は、Amazon EC2 インスタンスプロファイルから Amazon EC2 インスタンスに認証情報を提供する場合と同じような方法で、アプリケーションの認証情報を管理する機能があります。AWS 認証情報を作成してコンテナに配布したり、Amazon EC2 インスタンスのロールを使用したりする必要はありません。IAM ロールを Kubernetes サービスアカウントと関連付けて、サービスアカウントを使用するように Pods を設定できます。

各 EKS Pod Identity の関連付けは、指定されたクラスター内のネームスペース内のサービスアカウントにロールをマップします。複数のクラスターに同じアプリケーションがある場合、ロールの信頼ポリシーを変更することなく、各クラスターで同一の関連付けを行うことができます。

ポッドが関連付けられたサービスアカウントを使用する場合、Amazon EKS はポッドのコンテナに環境変数を設定します。環境変数は、AWS CLI を含む AWS SDK が EKS Pod Identity の認証情報を使用するように設定します。

EKS Pod Identity の利点

EKS Pod Identity には次の利点があります。

- 最小特権 – IAM アクセス許可の範囲をサービスアカウントに設定すると、そのサービスアカウントを使用する Pods のみにそのアクセス許可を付与できます。また、この機能により、`kiam` や `kube2iam` などのサードパーティーのソリューションが不要になります。
- 認証情報の分離 – Pod's のコンテナは、そのコンテナが使用するサービスアカウントに関連付けられた IAM ロールの認証情報のみを取得できます。コンテナは、他の Pods のコンテナで使われている認証情報にアクセスすることはできません。Pod Identity を使用する場合、Pod's コンテナに

は Amazon EKS ノード IAM ロール に割り当てられたアクセス許可も付与されます (Amazon EC2 インスタンスメタデータサービス (IMDS) への Pod のアクセスをブロックしていない場合)。詳細については、「ワーカーノードに割り当てられたインスタンスプロファイルへのアクセスを制限する」を参照してください。

- 監査性 – 遡及的な監査を確実に行うため、AWS CloudTrail を介してアクセスとイベントのロギングを利用できます。

EKS Pod Identity は OIDC ID プロバイダーを使用しないため、サービスアカウントの IAM ロール より簡単な方法です。EKS Pod Identity には次の機能強化があります。

- 独立オペレーション – 多くの組織では、OIDC ID プロバイダーの作成は、Kubernetes クラスターの管理とは別のチームの責任です。EKS Pod Identity には明確な役割分担があり、EKS Pod Identity の関連付けの設定はすべて Amazon EKS で行われ、IAM アクセス許可の設定はすべて IAM で行われます。
- 再利用性 – EKS Pod Identity は、サービスアカウントの IAM ロールが使用するクラスターごとの個別のプリンシパルの代わりに、単一の IAM プリンシパルを使用します。IAM 管理者は以下のプリンシパルを任意のロールの信頼ポリシーに追加して、EKS Pod Identity で使用できるようにします。

```
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            }
```

- スケーラビリティ – 一時的な認証情報の各セットは、各ポッドで実行する各 AWS SDK ではなく、EKS Pod Identity EKS Auth 内のサービスによって引き継がれます。次に、各ノードで実行される Amazon EKS Pod Identity エージェントが SDK に認証情報を発行します。そのため、負荷はノードごとに 1 回に減り、各ポッドで重複することはありません。詳細については、「EKS Pod Identity の仕組み」を参照してください。

この 2 つの選択肢を比較する詳細については、「Kubernetes ワークロードに Kubernetes サービスアカウントを使用して AWS へのアクセスを許可する」を参照してください。

EKS Pod Identity のセットアップの概要

以下の手順を実行して EKS Pod Identity を有効にします。

1. Amazon EKS Pod Identity エージェントのセットアップ — この手順は、クラスタごとに 1 回だけ実行します。

2. IAM ロールを EKS Pod Identity と共に引き受けるための Kubernetes サービスアカウントを設定する — この手順は、アプリケーションに付与する固有の権限セットごとに実行します。

3. Kubernetes サービスアカウントを使用するように Pods を設定するには — この手順は、AWS のサービス へのアクセスが必要な Pod ごとに実行します。

4. サポートされる AWS SDK を使用する – ワークロードがサポートされているバージョンの AWS SDK を使用していること、およびワークロードがデフォルトの認証情報チェーンを使用していることを確認します。

EKS Pod Identity に関する考慮事項

- 各クラスター内の各 Kubernetes サービスアカウントには 1 つの IAM ロールを関連付けることができます。EKS Pod Identity の関連付けを編集することで、サービスアカウントにマップされるロールを変更できます。
- 関連付けることができるのはクラスターと同じ AWS アカウント にあるロールだけです。別のアカウントから、EKS Pod Identity が使用するように設定したこのアカウントのロールに、アクセスを委任できます。アクセス権の委任および `AssumeRole` については、「IAM ユーザーガイド」の「AWS アカウント間の IAM ロールを使用したアクセスの委任」を参照してください。
- EKS Pod Identity エージェントが必要です。ノード上で Kubernetes `DaemonSet` として実行され、実行されているノード上のポッドにのみ認証情報を提供します。EKS Pod Identity エージェントの互換性の詳細については、以下のセクション EKS Pod Identity の制限事項 を参照してください。
- EKS Pod Identity エージェント はノードの `hostNetwork` を使用し、ノード上のリンクローカルアドレス上のポート `80` とポート `2703` を使用します。このアドレスは `169.254.170.23` 対 IPv4、`[fd00:ec2::23]` 対 IPv6 のクラスターです。

  `IPv6` アドレスを無効にするか、または localhost `IPv6` IP アドレスを禁止すると、エージェントは起動できません。`IPv6` を使用できないノードでエージェントを起動するには、「EKS Pod Identity エージェントで `IPv6` を無効にする」の手順に従って `IPv6` 設定を無効にします。

EKS Pod Identity クラスターバージョン

EKS Pod Identity を使用するには、クラスターのプラットフォームバージョンが、次の表に記載されているバージョンと同じかそれ以降、または表に記載されている Kubernetes バージョンよりも新しいバージョンである必要があります。

Kubernetes バージョン	プラットフォームバージョン
1.30	eks.2
1.29	eks.1
1.28	eks.4
1.27	eks.8
1.26	eks.9
1.25	eks.10
1.24	eks.13

EKS Pod Identity と互換性のあるアドオンのバージョン

> **Important**
>
> EKS Pod Identity を EKS アドオンとともに使用するには、EKS Pod Identity の関連付けを手動で作成する必要があります。AWS Management Console のアドオン設定で IAM ロールを選択しないでください。そのロールは IRSA でのみ使用されます。

IAM 認証情報を必要とする Amazon EKS アドオンとセルフマネージド型アドオンは、EKS Pod Identity、IRSA、またはインスタンスロールを使用できます。EKS Pod Identity をサポートする IAM 認証情報を使用するアドオンのリストは次のとおりです。

- Amazon VPC CNI plugin for Kubernetes `1.15.5-eksbuild.1` 以降
- AWS Load Balancer Controller`2.7.0` 以降。AWS Load Balancer Controller は EKS アドオンとしては使用できませんが、セルフマネージド型アドオンとして利用できることに注意してください。

EKS Pod Identity の制限事項

EKS Pod Identity は以下で使用できます。

- 前のトピック EKS Pod Identity クラスターバージョン に記載されている Amazon EKS クラスターバージョン。
- Linux Amazon EC2 インスタンスであるクラスターのワーカーノード。

EKS Pod Identity は以下では使用できません。

- 中国リージョン
- AWS GovCloud (US).
- AWS Outposts.
- Amazon EKS Anywhere
- Amazon EC2 で作成して実行する Kubernetes クラスター。EKS Pod Identity コンポーネントは Amazon EKS でのみ使用できます。

EKS Pod Identity は以下では使用できません。

- Linux Amazon EC2 インスタンス以外の場所で実行されるポッド。AWS Fargate (Fargate) 上で実行される Linux ポッドと Windows ポッドはサポートされていません。Windows Amazon EC2 インスタンスで実行されるポッドはサポートされていません。
- IAM 認証情報を必要とする Amazon EKS アドオン。EKS アドオンは、代わりにサービスアカウントの IAM ロールのみを使用できます。IAM 認証情報を使用する EKS アドオンのリストには以下が含まれます。
  - CSI ストレージドライバー: EBS CSI、EFS CSI、Amazon FSx for Lustre CSI ドライバー、Amazon FSx for NetApp ONTAP CSI ドライバー、Amazon FSx for OpenZFS CSI ドライバー、Amazon File Cache CSI ドライバー、Kubernetes Secrets Store CSI ドライバー用の AWS Secrets and Configuration Provider (ASCP)。

> **Note**
>
> これらのコントローラー、ドライバー、プラグインが EKS アドオンではなく自己管理型アドオンとしてインストールされている場合、最新の AWS SDK を使用するように更新されている限り、EKS Pod Identity がサポートされます。

EKS Pod Identity の仕組み

EKS Pod Identity の関連付けは、Amazon EC2 インスタンスプロファイルから Amazon EC2 インスタンスに認証情報を提供する場合と同じような方法で、アプリケーションの認証情報を管理する機能があります。

Amazon EKS Pod Identity は、追加の EKS Auth API と各ノードで実行されるエージェントポッドを使用して、ワークロードに認証情報を提供します。

Amazon EKS アドオン、自己管理型コントローラ、オペレータ、その他のアドオンなどのアドオンでは、作成者は最新の AWS SDK を使用するようにソフトウェアを更新する必要があります。EKS Pod Identity と Amazon EKS によって作成されたアドオンとの互換性のリストについては、前の EKS Pod Identity の制限事項 セクションを参照してください。

EKS Pod Identity をコードで使用する

コードでは、AWS SDK を使用して AWS サービスにアクセスできます。SDK AWS を使用してサービスのクライアントを作成するコードを記述すると、デフォルトで SDK AWS Identity and Access Management は使用する認証情報を一連の場所で検索します。有効な認証情報が見つかると、検索は停止されます。使用されるデフォルトの場所について詳しくは、「AWS SDK and Tools リファレンスガイド」の「認証情報プロバイダーチェーン」を参照してください。

EKS Pod Identity がコンテナ認証情報プロバイダーに追加されました。このプロバイダーはデフォルトの認証情報チェーンのステップで検索されます。現在、ワークロードが認証情報チェーンの前にある認証情報を使用している場合、同じワークロードに EKS Pod Identity の関連付けを設定しても、それらの認証情報は引き続き使用されます。この方法では、古い認証情報を削除する前に、まず関連付けを作成することで、他の種類の認証情報から安全に移行できます。

コンテナ認証情報プロバイダーは、各ノードで実行されるエージェントからの一時的な認証情報を提供します。Amazon EKS では、エージェントは Amazon EKS Pod Identity エージェントであり、Amazon Elastic Container Service ではエージェントは `amazon-ecs-agent` です。SDK は環境変数を使用して接続するエージェントを検索します。

一方、サービスアカウント用の IAM ロールは、AWS SDK が `AssumeRoleWithWebIdentity` を使用して AWS Security Token Service と交換しなければならないウェブ ID トークンを提供します。

EKS Pod Identity エージェントが Pod とどのように連携するか

1. Amazon EKS が EKS Pod Identity の関連付けを持つサービスアカウントを使用する新しいポッドを起動すると、クラスターは次のコンテンツを Pod マニフェストに追加します。

```
    env:
    - name: AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE
      value: "/var/run/secrets/pods.eks.amazonaws.com/serviceaccount/eks-pod-
identity-token"
    - name: AWS_CONTAINER_CREDENTIALS_FULL_URI
      value: "http://169.254.170.23/v1/credentials"
    volumeMounts:
    - mountPath: "/var/run/secrets/pods.eks.amazonaws.com/serviceaccount/"
      name: eks-pod-identity-token
  volumes:
  - name: eks-pod-identity-token
    projected:
      defaultMode: 420
      sources:
      - serviceAccountToken:
          audience: pods.eks.amazonaws.com
          expirationSeconds: 86400 # 24 hours
          path: eks-pod-identity-token
```

2. Kubernetes はポッドを実行するノードを選択します。次に、ノード上の Amazon EKS Pod Identity エージェントは AssumeRoleForPodIdentity アクションを使用して EKS Auth API から一時的な認証情報を取得します。
3. EKS Pod Identity エージェントは、コンテナ内で実行する AWS SDK でこれらの認証情報を利用できるようにします。
4. デフォルトの認証情報チェーンを使用する認証情報プロバイダーを指定しなくても、アプリケーションで SDK を使用できます。または、コンテナ認証情報プロバイダーを指定します。使用されるデフォルトの場所について詳しくは、「AWS SDK and Tools リファレンスガイド」の「認証情報プロバイダーチェーン」を参照してください。
5. SDK は環境変数を使用して EKS Pod Identity エージェントに接続し、認証情報を取得します。

> **Note**
>
> ワークロードが現在、認証情報チェーンの早い段階にある認証情報を使用している場合、同じワークロードに EKS Pod Identity の関連付けを設定しても、その認証情報は引き続き使用されます。

Amazon EKS Pod Identity エージェントのセットアップ

EKS Pod Identity の関連付けは、Amazon EC2 インスタンスプロファイルから Amazon EC2 インスタンスに認証情報を提供する場合と同じような方法で、アプリケーションの認証情報を管理する機能があります。

Amazon EKS Pod Identity は、追加の EKS Auth API と各ノードで実行されるエージェントポッドを使用して、ワークロードに認証情報を提供します。

考慮事項

- **IPv6**

  EKS Pod Identity エージェントはデフォルトでポッドが認証情報をリクエストするために IPv4 と IPv6 のアドレスをリッスンします。エージェントは、IPv4 のループバック (localhost) IP アドレス 169.254.170.23 と IPv6 の localhost IP アドレス [fd00:ec2::23] を使用します。

  IPv6 アドレスを無効にするか、または localhost IPv6 IP アドレスを禁止すると、エージェントは起動できません。IPv6 を使用できないノードでエージェントを起動するには、「EKS Pod Identity エージェントで IPv6 を無効にする」の手順に従って IPv6 設定を無効にします。

Amazon EKS Pod Identity エージェントの作成

エージェントの前提条件

- 既存の Amazon EKS クラスター。デプロイするには、「Amazon EKS の使用開始」を参照してください。クラスタバージョンとプラットフォームバージョンは、EKS Pod Identity クラスターバージョン に記載されているバージョン以降である必要があります。
- ノードロールには、エージェントが EKS Auth API で AssumeRoleForPodIdentity アクションを実行する権限があります。AWS 管理ポリシー: AmazonEKSWorkerNodePolicy を使用することも、次のようなカスタムポリシーを追加することもできます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks-auth:AssumeRoleForPodIdentity"
            ],
```

```
            "Resource": "*"
        }
    ]
}
```

このアクションをタグで制限して、エージェントを使用するポッドが引き受けることができるロールを制限できます。

- ノードは Amazon ECR にアクセスしてイメージをダウンロードできます。アドオンのコンテナイメージは、Amazon コンテナイメージレジストリ に記載されているレジストリにあります。

  なお、イメージの場所を変更して、AWS Management Console の [オプションの設定] や AWS CLI の `--configuration-values` で EKS アドオンの `imagePullSecrets` を提供することができます。
- ノードは Amazon EKS Auth API にアクセスできます。プライベートクラスターでは、AWS PrivateLink の `eks-auth` エンドポイントが必要です。

AWS Management Console

1. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。
2. 左のナビゲーションペインで、[クラスター] を選択し、次にアドオンを設定するEKS Pod Identity エージェントを選択します。
3. [アドオン] タブを選択します。
4. [その他のアドオンを入手] を選択します。
5. EKS Pod Identity のアドオンボックスの右上にあるボックスを選択し、[次へ] を選択します。
6. [選択したアドオン設定を構成する] ページの [バージョン] ドロップダウンリストで任意のバージョンを選択します。
7. (オプション) [オプションの設定] を展開して追加の設定を入力します。例えば、代替のコンテナイメージの場所と `ImagePullSecrets` を指定できます。許可されたキーのある JSON Schema は、[アドオン設定スキーマ] に表示されます。

   設定キーと値を [設定値] に入力します。
8. [Next] を選択します。
9. EKS Pod Identity がクラスター上で実行されていることを確認してください。

```
kubectl get pods -n kube-system | grep 'eks-pod-identity-agent'
```

出力例は次のとおりです。

```
eks-pod-identity-agent-gmqp7                                          1/1
 Running   1 (24h ago)   24h
eks-pod-identity-agent-prnsh                                          1/1
 Running   1 (24h ago)   24h
```

これで、クラスターで EKS Pod Identity の関連付けを使用できるようになりました。詳細については、「IAM ロールを EKS Pod Identity と共に引き受けるための Kubernetes サービスアカウントを設定する」を参照してください。

AWS CLI

1. 次の AWS CLI コマンドを実行します。`my-cluster` を自分のクラスター名に置き換えます。

```
aws eks create-addon --cluster-name my-cluster --addon-name eks-pod-identity-
agent --addon-version v1.0.0-eksbuild.1
```

> **Note**
>
> EKS Pod Identity エージェントは、サービスアカウントの IAM ロール用の `service-account-role-arn` を使用しません。EKS Pod Identity エージェントにはノードロールの権限を付与する必要があります。

2. EKS Pod Identity がクラスター上で実行されていることを確認してください。

```
kubectl get pods -n kube-system | grep 'eks-pod-identity-agent'
```

出力例は次のとおりです。

```
eks-pod-identity-agent-gmqp7                                          1/1
 Running   1 (24h ago)   24h
eks-pod-identity-agent-prnsh                                          1/1
 Running   1 (24h ago)   24h
```

これで、クラスターで EKS Pod Identity の関連付けを使用できるようになりました。詳細については、「IAM ロールを EKS Pod Identity と共に引き受けるための Kubernetes サービスアカウントを設定する」を参照してください。

Amazon EKS Pod Identity エージェントの作成

Amazon EKS タイプのアドオンを更新します。Amazon EKS タイプのアドオンをクラスターに追加していない場合は、「Amazon EKS Pod Identity エージェントの作成」を参照してください。

AWS Management Console

1. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。
2. 左のナビゲーションペインで、[クラスター] を選択し、次にアドオンを設定するEKS Pod Identity エージェントを選択します。
3. [アドオン] タブを選択します。
4. 新しいバージョンのアドオンが入手可能な場合、EKS Pod Identity エージェントには [バージョンの更新] ボタンが表示されます。[バージョンの更新] を選択します。
5. [Amazon EKS Pod Identity エージェントの設定] ページで、[バージョン] ドロップダウンリストから新しいバージョンを選択します。
6. [変更を保存] を選択します。

   更新が完了するまでに数秒かかる場合があります。次に、[ステータス] をチェックして、アドオンのバージョンが更新されたことを確認します。

AWS CLI

1. クラスターにインストールされているアドオンのバージョンを確認します。*my-cluster* をクラスター名に置き換えます。

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name eks-pod-identity-agent --query "addon.addonVersion" --output text
   ```

   出力例は次のとおりです。

   ```
   v1.0.0-eksbuild.1
   ```

この手順でアドオンを更新する前に、アドオンを作成する必要があります。

2. AWS CLI を使用してアドオンを更新します。AWS Management Console または eksctl を使用してアドオンを更新する場合は、「アドオンの更新」を参照してください。デバイスに沿ったコマンドをコピーします。必要に応じてコマンドに次の変更を加え、変更したコマンドを実行します。

   - *my-cluster* を自分のクラスター名に置き換えます。
   - *v1.0.0-eksbuild.1* は希望のバージョンに置き換えてください。
   - *&ExampleAWSAccountNo1* は、ご自分のアカウント ID に置き換えます。
   - 次のコマンドを実行します。

   ```
   aws eks update-addon --cluster-name my-cluster --addon-name eks-pod-identity-agent --addon-version v1.0.0-eksbuild.1'
   ```

   更新が完了するまでに数秒かかる場合があります。

3. アドオンのバージョンが更新されたことを確認します。*my-cluster* を自分のクラスター名に置き換えます。

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name eks-pod-identity-agent
   ```

   更新が完了するまでに数秒かかる場合があります。

   出力例は次のとおりです。

   ```
   {
       "addon": {
           "addonName": "eks-pod-identity-agent",
           "clusterName": "my-cluster",
           "status": "ACTIVE",
           "addonVersion": "v1.0.0-eksbuild.1",
           "health": {
               "issues": []
           },
           "addonArn": "arn:aws:eks:region:111122223333:addon/my-cluster/eks-pod-identity-agent/74c33d2f-b4dc-8718-56e7-9fdfa65d14a9",
           "createdAt": "2023-04-12T18:25:19.319000+00:00",
           "modifiedAt": "2023-04-12T18:40:28.683000+00:00",
   ```

```
        "tags": {}
    }
}
```

EKS Pod Identity エージェントの設定

EKS Pod Identity エージェントで **IPv6** を無効にする

AWS Management Console

AWS Management Console で **IPv6** を無効にする

1. EKS Pod Identity エージェントで IPv6 を無効にするには、EKS アドオンの [オプション設定] に次の設定を追加します。

   a. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。

   b. 左のナビゲーションペインで、[クラスター] を選択し、次にアドオンを設定するクラスター名を選択します。

   c. [アドオン] タブを選択します。

   d. EKS Pod Identity エージェントのアドオンボックスの右上にあるボックスを選択し、[編集] を選択します。

   e. EKS Pod Identity エージェントの設定ページの場合:

      i. 使用する [バージョン] を選択します。前のステップと同じバージョンを保持し、別のアクションでバージョンと設定を更新することをお勧めします。

      ii. [オプションの構成設定] を展開します。

      iii. [設定値] で JSON キー "agent": とにネストした JSON オブジェクトの値に、キー "additionalArgs": を指定します。結果のテキストは有効な JSON オブジェクトでなければなりません。このキーと値だけがテキストボックス内のデータである場合は、キーと値を中括弧 {} で囲みます。次の例は、ネットワークポリシーが有効になっていることを示しています。

```
{
    "agent": {
        "additionalArgs": {
            "-b": "169.254.170.23"
        }
```

```
    }
}
```

この設定では、エージェントが使用する唯一のアドレスに IPv4 アドレスを設定します。

f. EKS Pod Identity エージェントのポッドを置き換えて新しい設定を適用するには、[変更を保存] を選択します。

Amazon EKS は、EKS Pod Identity エージェントの Kubernetes DaemonSet のロールアウトを使用して EKS アドオンに変更を適用します。ロールアウトのステータスは、AWS Management Console のアドオン [更新履歴] と kubectl rollout status daemonset/eks-pod-identity-agent --namespace kube-system で追跡できます。

kubectl rollout は以下のコマンドを実行します。

```
$ kubectl rollout

history  -- View rollout history
pause    -- Mark the provided resource as paused
restart  -- Restart a resource
resume   -- Resume a paused resource
status   -- Show the status of the rollout
undo     -- Undo a previous rollout
```

ロールアウトに時間がかかりすぎる場合、Amazon EKS はロールアウトを取り消し、[アドオン更新] のタイプと [失敗] のステータスのメッセージがアドオンの [更新履歴] に追加されます。問題を調査するには、ロールアウトの履歴から開始し、EKS Pod Identity エージェントのポッドで kubectl logs を実行して EKS Pod Identity エージェントのログを確認します。

2. [更新履歴] の新しいエントリのステータスが [成功] の場合、ロールアウトが完了し、アドオンはすべての EKS Pod Identity エージェントのポッドで新しい設定を使用していることを意味します。

AWS CLI

AWS CLI で **IPv6** を無効にする

- EKS Pod Identity エージェントで IPv6 を無効にするには、EKS アドオンの [オプション設定] に次の設定を追加します。

  次の AWS CLI コマンドを実行します。my-cluster をクラスターの名前に置き換え、IAM ロール ARN を使用するロールに置き換えます。

```
aws eks update-addon --cluster-name my-cluster --addon-name eks-pod-identity-
agent \
    --resolve-conflicts PRESERVE --configuration-values '{"agent":
{"additionalArgs": { "-b": "169.254.170.23"}}}'
```

  この設定では、エージェントが使用する唯一のアドレスに IPv4 アドレスを設定します。

  Amazon EKS は、EKS Pod Identity エージェントの Kubernetes DaemonSet のロールアウトを使用して EKS アドオンに変更を適用します。ロールアウトのステータスは、AWS Management Console のアドオン [更新履歴] と kubectl rollout status daemonset/eks-pod-identity-agent --namespace kube-system で追跡できます。

  kubectl rollout は以下のコマンドを実行します。

```
kubectl rollout

history  -- View rollout history
pause    -- Mark the provided resource as paused
restart  -- Restart a resource
resume   -- Resume a paused resource
status   -- Show the status of the rollout
undo     -- Undo a previous rollout
```

  ロールアウトに時間がかかりすぎる場合、Amazon EKS はロールアウトを取り消し、[アドオン更新] のタイプと [失敗] のステータスのメッセージがアドオンの [更新履歴] に追加されます。問題を調査するには、ロールアウトの履歴から開始し、EKS Pod Identity エージェントのポッドで kubectl logs を実行して EKS Pod Identity エージェントのログを確認します。

IAM ロールを EKS Pod Identity と共に引き受けるための Kubernetes サービスアカウントを設定する

このトピックでは、AWS Identity and Access Management (IAM) ロール を EKS Pod Identity と共に引き受けるように Kubernetes サービスアカウントを設定する方法について説明します。任意の Pods はサービスアカウントを使用するように設定すると、ロールにアクセス許可がある AWS のサービス すべてにアクセスできます。

EKS Pod Identity の関連付けを作成するには、1 つの手順しかありません。AWS Management Console、AWS CLI、AWS SDK、AWS CloudFormation などのツールを使用して EKS で関連付けを作成します。クラスター内の関連付けに関するデータやメタデータはどの Kubernetes オブジェクトにもなく、サービスアカウントに注釈を追加することもありません。

前提条件

- 既存のクラスター。まだ所有していない場合は、Amazon EKS の使用開始 でのガイドのいずれかに従って作成します。
- 関連付けを作成している IAM プリンシパルには `iam:PassRole` が必要です。
- ご使用のデバイスまたは で、 のバージョン AWS CLI 以降または AWS CloudShell 以降がインストールおよび設定されていること。現在のバージョンは、`aws --version | cut -d / -f2 | cut -d ' ' -f1` で確認できます。macOS の `yum`、`apt-get`、または Homebrew などのパッケージマネージャは、AWS CLI の最新バージョンより数バージョン遅れることがあります。最新バージョンをインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI のインストール、更新、およびアンインストール」と「`aws configure` でのクイック設定」を参照してください。AWS CloudShell にインストールされている AWS CLI バージョンは、最新バージョンより数バージョン遅れている可能性もあります。更新するには、「AWS CloudShell ユーザーガイド」の「ホームディレクトリへの AWS CLI のインストール」を参照してください。
- デバイスまたは AWS CloudShell に、`kubectl` コマンドラインツールがインストールされていること。バージョンは、ご使用のクラスターの Kubernetes バージョンと同じか、1 つ前のマイナーバージョン以前、あるいはそれより新しいバージョンが使用できます。例えば、クラスターのバージョンが `1.29` である場合、`kubectl` のバージョン `1.28`、`1.29`、または `1.30` が使用できます。`kubectl` をインストールまたはアップグレードする方法については、「`kubectl` のインストールまたは更新」を参照してください。
- クラスター構成を含む既存の `kubectl config` ファイル。`kubectl config` ファイルの作成については、「Amazon EKS クラスターの `kubeconfig` ファイルを作成または更新する」を参照してください。

EKS Pod Identity の関連付けの作成

AWS Management Console

1. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。
2. 左のナビゲーションペインで、[クラスター] を選択し、次にアドオンを設定するEKS Pod Identity エージェントを選択します。
3. [アクセス] タブを選択します。
4. [Pod Identity の関連付け] で [作成] を選択します。
5. [IAM ロール] には、ワークロードに付与する権限を持つ IAM ロールを選択します。

> **Note**
>
> リストには、EKS Pod Identity による使用を許可する以下の信頼ポリシーを持つロールのみが含まれています。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
```

`sts:AssumeRole`

EKS Pod Identity は一時的な認証情報をポッドに渡す前に、`AssumeRole` を使用して IAM ロールを引き継ぎます。

`sts:TagSession`

EKS Pod Identity は、AWS STS へのリクエストにセッションタグを含めるために `TagSession` を使用します。

これらのタグを信頼ポリシーの condition keys で使用して、このロールを使用できるサービスアカウント、名前空間、およびクラスターを制限できます。

Amazon EKS 条件キーのリストについては、「サービス認証リファレンス」の「Amazon Elastic Kubernetes Service によって定義された条件」を参照してください。条件キーを使用できるアクションとリソースについては、「Amazon Elastic Kubernetes Service で定義されるアクション」を参照してください。

6. [Kubernetes 名前空間] には、サービスアカウントとワークロードを含む Kubernetes 名前空間を選択します。オプションで、クラスターに存在しない名前空間を名前空間名で指定できます。
7. [Kubernetes サービスアカウント] で、使用する Kubernetes サービスアカウントを選択します。Kubernetes ワークロードのマニフェストには、このサービスアカウントを指定する必要があります。オプションで、クラスターに存在しないサービスアカウントを名前で指定できます。
8. (オプション) [タグ] で [タグを追加] を選択し、キーと値のペアにメタデータを追加します。これらのタグは関連付けに適用され、IAM ポリシーで使用できます。

   このステップでは、複数のリージョンを追加できます。
9. [Create] (作成) を選択します。

AWS CLI

1. 既存の IAM ポリシーを IAM ロールに関連付ける場合は、次のステップにスキップします。

   IAM ポリシーを作成します。ポリシーを自作することも、必要となるアクセス権限のいくつかが既に付与されている AWS 管理ポリシーをコピーし、特定の要件に応じてカスタマイズすることもできます。詳細については、『IAM ユーザーガイド』の「IAM ポリシーの作成」を参照してください。

   a. Pods にアクセスさせる AWS のサービス の権限を含むファイルを作成します。すべての AWS のサービス に対するアクションの全リストについては、「サービス認証リファレンス」を参照してください。

次のコマンドを実行して、Amazon S3 バケットへの読み取り専用アクセスを許可するサンプルポリシーファイルを作成できます。必要に応じて、このバケットに設定情報またはブートストラップスクリプトを格納すると、Pod 内のコンテナがバケットからファイルを読み取り、アプリケーションにロードできます。このサンプルポリシーを作成する場合は、次のコンテンツをデバイスにコピーします。*my-pod-secrets-bucket* をバケット名に置き換え、コマンドを実行します。

```
cat >my-policy.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::my-pod-secrets-bucket"
        }
    ]
}
EOF
```

b. IAM ポリシーを作成します。

```
aws iam create-policy --policy-name my-policy --policy-document file://my-policy.json
```

2. IAM ロールを作成し、Kubernetes サービスアカウントに関連付けます。

1. IAM ロールを引き受ける既存の Kubernetes サービスアカウントがある場合は、この手順を省略できます。

Kubernetes サービスアカウントを作成します。次のコンテンツをデバイスにコピーします。*my-service-account* を目的の名前に置き換え、必要に応じて *default* を別の名前空間に置き換えます。*default* を変更する場合、名前空間は既に存在している必要があります。

```
cat >my-service-account.yaml <<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
  name: my-service-account
```

```
  namespace: default
EOF
kubectl apply -f my-service-account.yaml
```

以下のコマンドを実行します。

```
kubectl apply -f my-service-account.yaml
```

2. IAM ロール用の信頼ポリシーファイルを作成するには、次のコマンドを実行します。

```
cat >trust-relationship.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
EOF
```

3. ロールを作成します。*my-role* を IAM ロールの名前に、*my-role-description* をユーザーロールの表記に置き換えます。

```
aws iam create-role --role-name my-role --assume-role-policy-document
 file://trust-relationship.json --description "my-role-description"
```

4. IAM ポリシーをロールにアタッチします。*my-role* を IAM ロールの名前に置き換え、作成した既存のポリシーの名前に *my-policy* を置き換えます。

```
aws iam attach-role-policy --role-name my-role --policy-
arn=arn:aws:iam::111122223333:policy/my-policy
```

> **Note**
>
> サービスアカウントの IAM ロールとは異なり、EKS Pod Identity はサービスアカウントの注釈を使用しません。

5. 次のコマンドを実行して、関連付けを作成します。my-cluster をクラスターの名前で置き換え、*my-service-account* を目的の名前に置き換え、必要に応じて *default* を別の名前空間に置き換えます。

```
aws eks create-pod-identity-association --cluster-name my-cluster --role-
arn arn:aws:iam::111122223333:role/my-role --namespace default --service-
account my-service-account
```

出力例は次のとおりです。

```
{
    "association": {
        "clusterName": "my-cluster",
        "namespace": "default",
        "serviceAccount": "my-service-account",
        "roleArn": "arn:aws:iam::111122223333:role/my-role",
        "associationArn": "arn:aws::111122223333:podidentityassociation/my-
cluster/a-abcdefghijklmnop1",
        "associationId": "a-abcdefghijklmnop1",
        "tags": {},
        "createdAt": 1700862734.922,
        "modifiedAt": 1700862734.922
    }
}
```

> **Note**
>
> 名前空間とサービスアカウントは、クラスターには存在しない名前で指定できます。EKS Pod Identity の関連付けを機能させるには、名前空間、サービスアカウント、およびサービスアカウントを使用するワークロードを作成する必要があります。

3. ロールとサービスアカウントが正しく設定されていることを確認します。

a. IAM ロールの信頼ポリシーが正しく設定されていることを確認します。

```
aws iam get-role --role-name my-role --query Role.AssumeRolePolicyDocument
```

出力例は次のとおりです。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow EKS Auth service to assume this role for Pod
 Identities",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
```

b. 前の手順でロールにアタッチしたポリシーが、そのロールにアタッチされていることを確認します。

```
aws iam list-attached-role-policies --role-name my-role --query
 AttachedPolicies[].PolicyArn --output text
```

出力例は次のとおりです。

```
arn:aws:iam::111122223333:policy/my-policy
```

c. 使用するポリシーの Amazon リソースネーム (ARN) を保存する変数を設定します。*my-policy* を、アクセス許可を確認するポリシーの名前に置き換えます。

```
export policy_arn=arn:aws:iam::111122223333:policy/my-policy
```

d. ポリシーのデフォルトバージョンを確認します。

```
aws iam get-policy --policy-arn $policy_arn
```

出力例は次のとおりです。

```
{
    "Policy": {
        "PolicyName": "my-policy",
        "PolicyId": "EXAMPLEBIOWGLDEXAMPLE",
        "Arn": "arn:aws:iam::111122223333:policy/my-policy",
        "Path": "/",
        "DefaultVersionId": "v1",
        [...]
    }
}
```

e. ポリシーの内容を表示して、Pod で必要な権限がすべて含まれていることを確認します。必要であれば、次のコマンドの *1* を、前の出力で返されたバージョンに置き換えます。

```
aws iam get-policy-version --policy-arn $policy_arn --version-id v1
```

出力例は次のとおりです。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::my-pod-secrets-bucket"
        }
    ]
}
```

前の手順でサンプルポリシーを作成した場合、出力は同じになります。別のポリシーを作成した場合、*####*の内容は異なります。

次のステップ

Kubernetes サービスアカウントを使用するように Pods を設定するには

Kubernetes サービスアカウントを使用するように Pods を設定するには

Pod が AWS のサービスにアクセスする必要がある場合、Kubernetes サービスアカウントを使用するように設定する必要があります。サービスアカウントは、AWS のサービスにアクセスする権限がある AWS Identity and Access Management (IAM) ロールに関連付ける必要があります。

前提条件

- 既存のクラスター。まだ所有していない場合は、Amazon EKS の使用開始 でのガイドのいずれかを参照しながら作成します。
- 既存の Kubernetes サービスアカウントと、サービスアカウントを IAM ロールに関連付ける EKS Pod Identity の関連付け。ロールには、Pods が AWS のサービス を使用するアクセス許可を含む IAM ポリシーが関連付けられている必要があります。サービスアカウントとロールの作成および設定方法については、「IAM ロールを EKS Pod Identity と共に引き受けるための Kubernetes サービスアカウントを設定する」を参照してください。
- ご使用のデバイスまたは で、 のバージョン AWS CLI 以降または AWS CloudShell 以降がインストールおよび設定されていること。現在のバージョンは、`aws --version | cut -d / -f2 | cut -d ' ' -f1` で確認できます。macOS の `yum`、`apt-get`、または Homebrew などのパッケージマネージャは、AWS CLI の最新バージョンより数バージョン遅れることがあります。最新バージョンをインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI のインストール、更新、およびアンインストール」と「`aws configure` でのクイック設定」を参照してください。AWS CloudShell にインストールされている AWS CLI バージョンは、最新バージョンより数バージョン遅れている可能性もあります。更新するには、「AWS CloudShell ユーザーガイド」の「ホームディレクトリへの AWS CLI のインストール」を参照してください。
- デバイスまたは AWS CloudShell に、`kubectl` コマンドラインツールがインストールされていること。バージョンは、ご使用のクラスターの Kubernetes バージョンと同じか、1 つ前のマイナーバージョン以前、あるいはそれより新しいバージョンが使用できます。例えば、クラスターのバージョンが `1.29` である場合、`kubectl` のバージョン `1.28`、`1.29`、または `1.30` が使用できます。`kubectl` をインストールまたはアップグレードする方法については、「`kubectl` のインストールまたは更新」を参照してください。
- クラスター構成を含む既存の `kubectl config` ファイル。`kubectl config` ファイルの作成については、「Amazon EKS クラスターの `kubeconfig` ファイルを作成または更新する」を参照してください。

Pod がサービスアカウントを使用するように設定するには

1. 次のコマンドを使用し、Pod をデプロイして設定を確認できるデプロイマニフェストを作成します。*example values* を自分の値に置き換えてください。

```
cat >my-deployment.yaml <<EOF
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      serviceAccountName: my-service-account
      containers:
      - name: my-app
        image: public.ecr.aws/nginx/nginx:X.XX
EOF
```

2. マニフェストをクラスターにデプロイします。

```
kubectl apply -f my-deployment.yaml
```

3. Pod に必要な環境変数が存在することを確認してください。

   a. 前の手順のデプロイ時にデプロイされた Pods を確認します。

   ```
   kubectl get pods | grep my-app
   ```

   出力例は次のとおりです。

   ```
   my-app-6f4dfff6cb-76cv9   1/1     Running   0          3m28s
   ```

   b. Pod にサービスアカウントトークンのファイルがマウントされていることを確認します。

```
kubectl describe pod my-app-6f4dfff6cb-76cv9 | grep
 AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE:
```

出力例は次のとおりです。

```
AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE:  /var/run/secrets/
pods.eks.amazonaws.com/serviceaccount/eks-pod-identity-token
```

4. ロールにアタッチされた IAM ポリシーで割り当てた権限を使用して、Pods が AWS のサービスと対話できることを確認します。

> **Note**
>
> Pod がサービスアカウントに関連付けられた IAM ロールの AWS 認証情報を使用する場合、AWS CLI またはその Pod のコンテナ内の他の SDK は、そのロールによって提供される認証情報を使用します。Amazon EKS ノード IAM ロール に提供された認証情報へのアクセスを制限しない場合、Pod は引き続きこれらの認証情報にアクセスできます。詳細については、「ワーカーノードに割り当てられたインスタンスプロファイルへのアクセスを制限する」を参照してください。

Pods が期待どおりにサービスとやり取りできない場合は、次の手順を実行して、すべてが正しく構成されていることを確認してください。

a. Pods が、EKS Pod Identity の関連付けを介した IAM ロールの引き受けをサポートする AWS SDK バージョンを使用していることを確認します。詳細については、「サポートされる AWS SDK を使用する」を参照してください。

b. デプロイがサービスアカウントを使用していることを確認します。

```
kubectl describe deployment my-app | grep "Service Account"
```

出力例は次のとおりです。

```
Service Account:  my-service-account
```

EKS Pod Identity の権限を定義してタグに基づいて役割を引き受ける

EKS Pod Identity は、クラスター名、名前空間、サービスアカウント名などの属性を含むタグを各ポッドの一時認証情報に添付します。これらのロールセッションタグを使用すると、管理者は一致するタグに基づいて AWS リソースへのアクセスを許可することで、複数のサービスアカウントで機能する単一のロールを作成できます。ロールセッションタグのサポートを追加することで、お客様は同じ IAM ロールと IAM ポリシーを再利用しながら、クラスター間およびクラスター内のワークロード間のセキュリティ境界を厳しくすることができます。

例えば、次のポリシーはオブジェクトが EKS クラスターの名前でタグ付けされた場合に `s3:GetObject` アクションを許可します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectTagging"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/eks-cluster-name": "${aws:PrincipalTag/eks-
cluster-name}"
                }
            }
        }
    ]
}
```

EKS Pod Identity によって追加されたセッションタグのリスト

次のリストには、Amazon EKS による `AssumeRole` リクエストに追加されたタグのすべてのキーが含まれています。これらのタグをポリシーで使用するには、`${aws:PrincipalTag/` の後に、例えば `${aws:PrincipalTag/kubernetes-namespace}` などのキーを続けます。

- `eks-cluster-arn`
- `eks-cluster-name`
- `kubernetes-namespace`
- `kubernetes-service-account`
- `kubernetes-pod-name`
- `kubernetes-pod-uid`

アカウント間のコピー

EKS Pod Identity によって追加されるセッションタグはすべて推移的です。タグのキーと値は、ワークロードがロールを別のアカウントに切り替えるために使用するすべての `AssumeRole` アクションに渡されます。これらのタグを他のアカウントのポリシーで使用して、クロスアカウントシナリオでのアクセスを制限できます。詳細については、IAM ユーザーガイドの「セッションタグを使用したロールの連鎖」を参照してください。

カスタムタグ

EKS Pod Identity は、実行する `AssumeRole` アクションにカスタムタグを追加できません。ただし、IAM ロールに適用するタグは、常に同じ形式 (`${aws:PrincipalTag/MyCustomTag}` など、キーが続く `${aws:PrincipalTag/`) で使用できます。

> **Note**
>
> `sts:AssumeRole` リクエストによってセッションに追加されたタグは、競合が発生した場合に優先されます。例えば、EKS が顧客ロールを引き継ぐときに Amazon EKS がセッションにキー `eks-cluster-name` と値 `my-cluster` を追加すると仮定します。また、値 `my-own-cluster` を含む IAM ロールに `eks-cluster-name` タグも追加しました。この場合は前者が優先され、`eks-cluster-name` タグの値は `my-cluster` になります。

サポートされる AWS SDK を使用する

> **Important**
>
> ドキュメントの以前のバージョンには間違いがありました。AWS SDK for Java v1 は EKS Pod Identity をサポートしていません。

EKS Pod Identity を使用すると、Pods 内のコンテナは、 ウェブ ID トークンファイルを介した IAM ロールの引き受けをサポートする AWS EKS Pod Identity エージェントを使用する必要があります。お使いの AWS SDK には、必ず次のバージョン以降を使用してください。

- Java (バージョン 2) – 2.21.30
- バージョン1に移動 – v1.47.11
- バージョン 2 に移動 – 2023-11-14 リリース
- Python (Boto3) – 1.34.41
- Python (botocore) – 1.34.41
- AWS CLI – 1.30.0

  AWS CLI — 2.15.0
- JavaScript v2 – 2.1550.0
- JavaScript v3 – v3.458.0
- Kotlin – v1.0.1
- Ruby – 3.188.0
- Rust – release-2024-03-13
- C++ – 1.11.263
- .NET – 3.7.734.0
- PowerShell – 4.1.502
- PHP – 3.287.1

サポートされている SDK を使用していることを確認するには、コンテナを構築する際に、「AWS での構築ツール」で、希望する SDK のインストール手順に従ってください。

EKS Pod Identity をサポートするアドオンのリストについては、「EKS Pod Identity と互換性のあるアドオンのバージョン」を参照してください。

EKS Pod Identity 認証情報の使用

EKS Pod Identity の関連付けからの認証情報を使用するには、コードで任意の AWS SDK を使用して SDK AWS を使用するサービスのクライアントを作成できます。デフォルトでは、SDK AWS Identity and Access Management は使用する認証情報を一連の場所で検索します。クライアントの作成時や SDK の初期化時に認証情報プロバイダーを指定しなかった場合は、EKS Pod Identity 認証情報が使用されます。

これがうまくいくのは、EKS Pod Identity がデフォルトの認証情報チェーンのステップで検索されるコンテナ認証情報プロバイダーに追加されているからです。現在、ワークロードが認証情報チェーンの前にある認証情報を使用している場合は、同じワークロードに EKS Pod Identity の関連付けを設定しても、その認証情報は引き続き使用されます。

EKS Pod Identity の仕組みの詳細については、「EKS Pod Identity の仕組み」を参照してください。

EKS Pod Identity ロール

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
```

`sts:AssumeRole`

EKS Pod Identity は一時的な認証情報をポッドに渡す前に、`AssumeRole` を使用して IAM ロールを引き継ぎます。

`sts:TagSession`

EKS Pod Identity は、AWS STS へのリクエストにセッションタグを含めるために `TagSession` を使用します。

これらのタグを信頼ポリシーの condition keys で使用して、このロールを使用できるサービスアカウント、名前空間、およびクラスターを制限できます。

Amazon EKS 条件キーのリストについては、「サービス認証リファレンス」の「Amazon Elastic Kubernetes Service によって定義された条件」を参照してください。条件キーを使用できるアクションとリソースについては、「Amazon Elastic Kubernetes Service で定義されるアクション」を参照してください。

サービスアカウントの IAM ロール

Pod のコンテナ内のアプリケーションは AWS SDK または AWS CLI で、AWS のサービス (IAM) アクセス許可を使用した AWS Identity and Access Management への API リクエストを行うことができます。アプリケーションは AWS 認証情報で AWS API リクエストに署名する必要があります。サービスアカウントの IAM ロールには、Amazon EC2 インスタンスプロファイルから Amazon EC2 インスタンスに認証情報を提供する場合と同じような方法で、アプリケーションの認証情報を管理する機能があります。AWS 認証情報を作成してコンテナに配布したり、Amazon EC2 インスタンスのロールを使用したりする必要はありません。IAM ロールを Kubernetes サービスアカウントと関連付けて、サービスアカウントを使用するように Pods を設定できます。AWS Outposts の Amazon EKS 用ローカルクラスターで、サービスアカウントに IAM ロールを使用することはできません。

サービスアカウントの IAM ロールには、次の利点があります。

- 最小特権 – IAM アクセス許可の範囲をサービスアカウントに設定すると、そのサービスアカウントを使用する Pods のみにそのアクセス許可を付与できます。また、この機能により、`kiam` や `kube2iam` などのサードパーティーのソリューションが不要になります。
- 認証情報の分離 – Pod's のコンテナは、そのコンテナが使用するサービスアカウントに関連付けられた IAM ロールの認証情報のみを取得できます。コンテナは、他の Pods のコンテナで使われている認証情報にアクセスすることはできません。サービスアカウントの IAM ロールを使用する場合、Pod's コンテナには Amazon EKS ノード IAM ロール に割り当てられたアクセス許可も付与されます (Amazon EC2 インスタンスメタデータサービス (IMDS) への Pod のアクセスをブロックしていない場合)。詳細については、「ワーカーノードに割り当てられたインスタンスプロファイルへのアクセスを制限する」を参照してください。
- 監査性 - 遡及的な監査を確実に行うため、AWS CloudTrail を介してアクセスとイベントのロギングを利用できます。

以下の手順を実行してサービスアカウントの IAM ロールを有効にします。

1. クラスターの IAM OIDC プロバイダーを作成する — この手順は、クラスタごとに 1 回だけ実行します。

> **Note**
>
> EKS VPC エンドポイントを有効にすると、その VPC 内から EKS OIDC サービスエンドポイントにアクセスできなくなります。そのため、VPC で `eksctl` を使用して OIDC プロバイダーを作成するなどの操作は機能せず、`https://oidc.eks.region.amazonaws.com` をリクエストしようとするとタイムアウトになります。エラーメッセージの例は次のとおりです。
>
> ```
> ** server can't find oidc.eks.region.amazonaws.com: NXDOMAIN
> ```
>
> このステップを完了するには、VPC の外部、たとえばインターネットに接続された AWS CloudShell 内またはコンピューター上でコマンドを実行します。

2. IAM ロールを引き受けるように Kubernetes サービスアカウントを設定する — この手順は、アプリケーションに付与する固有の権限セットごとに実行します。
3. Kubernetes サービスアカウントを使用するように Pods を設定するには — この手順は、AWS のサービス へのアクセスが必要な Pod ごとに実行します。
4. サポートされる AWS SDK の使用 — ワークロードがサポートされているバージョンの AWS SDK を使用していること、およびワークロードがデフォルトの認証情報チェーンを使用していることを確認します。

IAM、Kubernetes、および OpenID Connect (OIDC) の背景情報

2014 年に、AWS Identity and Access Management は OpenID Connect (OIDC) を使用したフェデレーティッドアイデンティティのサポートを追加しました。この機能により、サポートされている ID プロバイダーで AWS API コールを認証し、有効な OIDC JSON ウェブトークン (JWT) を受け取ることができます。このトークンを AWS STS `AssumeRoleWithWebIdentity` の API オペレーションに渡し、一時的な IAM ロールの認証情報を受け取ることができます。これらの認証情報を使用して、Amazon S3 や DynamoDB などの任意の AWS のサービスとやり取りできます。

各 JWT トークンは署名キーペアによって署名されます。キーは Amazon EKS が管理する OIDC プロバイダーで提供され、プライベートキーは 7 日ごとにローテーションされます。Amazon EKS はパブリックキーの有効期限が切れるまで保持します。外部 OIDC クライアントに接続する場合は、

パブリックキーが期限切れになる前に、キーを更新する必要があることに注意してください。「the section called “署名キーを取得する”」ではその方法を説明しています。

Kubernetes は、独自の内部アイデンティティシステムとして長い間、サービスアカウントを使用してきました。Pods は、Kubernetes API サーバーのみが検証できる自動マウントトークン (OIDC JWT ではない) を使用して Kubernetes API サーバーで認証できます。これらのレガシーサービスアカウントトークンは期限切れにならず、署名キーの更新は難しいプロセスです。Kubernetes バージョン `1.12` では、新しい `ProjectedServiceAccountToken` 機能のサポートが追加されました。この機能は、サービスアカウントアイデンティティを含む OIDC JSON ウェブトークンで、設定可能なオーディエンスをサポートします。

Amazon EKS は、`ProjectedServiceAccountToken` JSON ウェブトークンの署名キーを含むクラスターごとにパブリック OIDC 検出エンドポイントをホストするため、IAM などの外部システムで、Kubernetes によって発行された OIDC トークンを検証して受け入れることができます。

クラスターの IAM OIDC プロバイダーを作成する

クラスターには、OpenID Connect (OIDC) 発行者の URL が関連付けられています。サービスアカウントで AWS Identity and Access Management (IAM) ロールを使用するには、クラスターの OIDC 発行者 URL に対して IAM OIDC プロバイダーが存在している必要があります。

前提条件

- 既存の Amazon EKS クラスター。デプロイするには、「Amazon EKS の使用開始」を参照してください。
- ご使用のデバイスまたは AWS CloudShell で、バージョン `2.12.3` 以降、または AWS Command Line Interface (AWS CLI) のバージョン `1.27.160` 以降がインストールおよび設定されていること。現在のバージョンを確認するには、「**`aws --version | cut -d / -f2 | cut -d ' ' -f1`**」を参照してください。macOS の `yum`、`apt-get`、または Homebrew などのパッケージマネージャは、AWS CLI の最新バージョンより数バージョン遅れることがあります。最新バージョンをインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI のインストール、更新、およびアンインストール」と「aws configure でのクイック設定」を参照してください。AWS CloudShell にインストールされている AWS CLI バージョンは、最新バージョンより数バージョン遅れている可能性もあります。更新するには、「AWS CloudShell ユーザーガイド」の「ホームディレクトリへの AWS CLI のインストール」を参照してください。
- デバイスまたは AWS CloudShell に、`kubectl` コマンドラインツールがインストールされていること。バージョンは、ご使用のクラスターの Kubernetes バージョンと同じか、1 つ前のマイナーバージョン以前、あるいはそれより新しいバージョンが使用できます。例えば、クラスターのバー

ジョンが 1.29 である場合、kubectl のバージョン 1.28、1.29、または 1.30 が使用できます。kubectl をインストールまたはアップグレードする方法については、「kubectl のインストールまたは更新」を参照してください。

- クラスター構成を含む既存の kubectl config ファイル。kubectl config ファイルの作成については、「Amazon EKS クラスターの kubeconfig ファイルを作成または更新する」を参照してください。

eksctl または AWS Management Console を使用して、クラスターの IAM OIDC プロバイダーを作成できます。

eksctl

前提条件

デバイスまたは AWS CloudShell にインストールされている eksctl コマンドラインツールのバージョン 0.183.0 以降。eksctl をインストールまたはアップグレードするには、eksctl ドキュメントの「インストール」を参照してください。

eksctl を使用してクラスターの IAM OIDC ID プロバイダーを作成するには

1. クラスターの OIDC 発行者 ID を決定します。

   クラスターの OIDC 発行者 ID を取得し、変数に格納します。*my-cluster* を独自の値に置き換えます。

   ```
   cluster_name=my-cluster
   ```

   ```
   oidc_id=$(aws eks describe-cluster --name $cluster_name --query
    "cluster.identity.oidc.issuer" --output text | cut -d '/' -f 5)
   ```

   ```
   echo $oidc_id
   ```

2. クラスターの発行者 ID を持つ IAM OIDC プロバイダーが既にアカウントにあるかどうかを確認します。

   ```
   aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
   ```

出力が返された場合は、既にクラスター用の IAM OIDC プロバイダーが存在するため、次の手順をスキップできます。出力が返されない場合は、クラスター用の IAM OIDC プロバイダーを作成する必要があります。

3. 次のコマンドを使用して、クラスターの IAM OIDC ID プロバイダーを作成します。

```
eksctl utils associate-iam-oidc-provider --cluster $cluster_name --approve
```

> **Note**
>
> EKS VPC エンドポイントを有効にすると、その VPC 内から EKS OIDC サービスエンドポイントにアクセスできなくなります。そのため、VPC で `eksctl` を使用して OIDC プロバイダーを作成するなどの操作は機能せず、`https://oidc.eks.region.amazonaws.com` をリクエストしようとするとタイムアウトになります。エラーメッセージの例は次のとおりです。
>
> ```
> ** server can't find oidc.eks.region.amazonaws.com: NXDOMAIN
> ```
>
> このステップを完了するには、VPC の外部、たとえばインターネットに接続された AWS CloudShell 内またはコンピューター上でコマンドを実行します。

AWS Management Console

AWS Management Console を使用してクラスターの IAM OIDC ID プロバイダーを作成するには

1. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。
2. 左側ペインで、[クラスター] を選択し、[クラスター] ページでご自身のクラスターの名前を選択します。
3. [概要] タブの [詳細] セクションで、[OpenID Connect プロバイダーの URL] の値を書き留めます。
4. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
5. 左のナビゲーションペインの [アクセス管理] で [ID プロバイダー] を選択します。[プロバイダー] がクラスターの URL と一致するリストに表示されている場合は、クラスターのプロバイダーがすでに存在します。クラスターの URL に一致するプロバイダーがリストにない場合、プロバイダーを作成する必要があります。

6. プロバイダーを作成するには、[プロバイダーを追加] を選択します。
7. [プロバイダーのタイプ] には [OpenID Connect] を選択します。
8. [プロバイダー URL] に、クラスター用の OIDC プロバイダー URL を貼り付け、[サムプリントを取得] を選択します。
9. [対象者] に **sts.amazonaws.com** を入力し、[プロバイダーを追加] を選択します。

次のステップ

IAM ロールを引き受けるように Kubernetes サービスアカウントを設定する

IAM ロールを引き受けるように Kubernetes サービスアカウントを設定する

このトピックでは、AWS Identity and Access Management (IAM) ロールを引き受けるように Kubernetes サービスアカウントを設定する方法について説明します。任意の Pods はサービスアカウントを使用するように設定すると、ロールにアクセス許可がある AWS のサービス すべてにアクセスできます。

前提条件

- 既存のクラスター。まだ所有していない場合は、Amazon EKS の使用開始 でのガイドのいずれかに従って作成します。
- クラスター用の既存 IAM OpenID Connect (OIDC) プロバイダー。既に所有中かどうかの確認、または作成方法については「クラスターの IAM OIDC プロバイダーを作成する」を参照してください。
- ご使用のデバイスまたは AWS CloudShell で、バージョン `2.12.3` 以降、または AWS Command Line Interface (AWS CLI) のバージョン `1.27.160` 以降がインストールおよび設定されていること。現在のバージョンを確認するには、「**`aws --version | cut -d / -f2 | cut -d ' ' -f1`**」を参照してください。macOS の `yum`、`apt-get`、または Homebrew などのパッケージマネージャは、AWS CLI の最新バージョンより数バージョン遅れることがあります。最新バージョンをインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI のインストール、更新、およびアンインストール」と「aws configure でのクイック設定」を参照してください。AWS CloudShell にインストールされている AWS CLI バージョンは、最新バージョンより数バージョン遅れている可能性もあります。更新するには、「AWS CloudShell ユーザーガイド」の「ホームディレクトリへの AWS CLI のインストール」を参照してください。
- デバイスまたは AWS CloudShell に、`kubectl` コマンドラインツールがインストールされていること。バージョンは、ご使用のクラスターの Kubernetes バージョンと同じか、1 つ前のマイナーバージョン以前、あるいはそれより新しいバージョンが使用できます。例えば、クラスターのバー

ジョンが 1.29 である場合、kubectl のバージョン 1.28、1.29、または 1.30 が使用できます。kubectl をインストールまたはアップグレードする方法については、「kubectl のインストールまたは更新」を参照してください。

- クラスター構成を含む既存の kubectl config ファイル。kubectl config ファイルの作成については、「Amazon EKS クラスターの kubeconfig ファイルを作成または更新する」を参照してください。

IAM ロールを Kubernetes サービスアカウントに関連付けるには

1. 既存の IAM ポリシーを IAM ロールに関連付ける場合は、次のステップにスキップします。

   IAM ポリシーを作成します。ポリシーを自作することも、必要となるアクセス権限のいくつかが既に付与されている AWS 管理ポリシーをコピーし、特定の要件に応じてカスタマイズすることもできます。詳細については、『IAM ユーザーガイド』の「IAM ポリシーの作成」を参照してください。

   a. Pods にアクセスさせる AWS のサービス の権限を含むファイルを作成します。すべての AWS のサービス に対するアクションの全リストについては、「サービス認証リファレンス」を参照してください。

      次のコマンドを実行して、Amazon S3 バケットへの読み取り専用アクセスを許可するサンプルポリシーファイルを作成できます。必要に応じて、このバケットに設定情報またはブートストラップスクリプトを格納すると、Pod 内のコンテナがバケットからファイルを読み取り、アプリケーションにロードできます。このサンプルポリシーを作成する場合は、次のコンテンツをデバイスにコピーします。*my-pod-secrets-bucket* をバケット名に置き換え、コマンドを実行します。

```
cat >my-policy.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::my-pod-secrets-bucket"
        }
    ]
}
EOF
```

b. IAM ポリシーを作成します。

```
aws iam create-policy --policy-name my-policy --policy-document file://my-policy.json
```

2. IAM ロールを作成し、Kubernetes サービスアカウントに関連付けます。eksctl または AWS CLI を使用できます。

eksctl

前提条件

デバイスまたは AWS CloudShell にインストールされている eksctl コマンドラインツールのバージョン 0.183.0 以降。eksctl をインストールまたはアップグレードするには、eksctl ドキュメントの「インストール」を参照してください。

my-service-account を、eksctl で作成する Kubernetes サービスアカウントの名前に置き換え、IAM ロールに関連付けます。*default* は、eksctl でサービスアカウントを作成する名前空間に置き換えます。*my-cluster* の部分は、自分のクラスター名に置き換えます。*my-role* は、サービスアカウントを関連付けるロールの名前に置き換えます。まだ存在しない場合は、eksctl によって作成されます。*111122223333* をアカウント ID に置き換え、*my-policy* を既存のポリシー名に置き換えます。

```
eksctl create iamserviceaccount --name my-service-account --namespace default --cluster my-cluster --role-name my-role \
    --attach-policy-arn arn:aws:iam::111122223333:policy/my-policy --approve
```

> **Important**
>
> ロールまたはサービスアカウントが既に存在する場合、前のコマンドは失敗する可能性があります。eksctl には、そのような状況で使用できるさまざまなオプションがあります。詳細については、**eksctl create iamserviceaccount --help** を実行してください。

AWS CLI

1. IAM ロールを引き受ける既存の Kubernetes サービスアカウントがある場合は、この手順を省略できます。

Kubernetes サービスアカウントを作成します。次のコンテンツをデバイスにコピーします。*my-service-account* を目的の名前に置き換え、必要に応じて *default* を別の名前空間に置き換えます。*default* を変更する場合、名前空間は既に存在している必要があります。

```
cat >my-service-account.yaml <<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
  name: my-service-account
  namespace: default
EOF
kubectl apply -f my-service-account.yaml
```

2. 次のコマンドを使用して、AWS アカウント ID を環境変数に設定します。

```
account_id=$(aws sts get-caller-identity --query "Account" --output text)
```

3. 次のコマンドを使用して、クラスターの OIDC ID プロバイダーを環境変数に設定します。*my-cluster* を自分のクラスター名に置き換えます。

```
oidc_provider=$(aws eks describe-cluster --name my-cluster --region
 $AWS_REGION --query "cluster.identity.oidc.issuer" --output text | sed -e "s/
^https:\/\///")
```

4. サービスアカウントの名前空間と名前の変数を設定します。*my-service-account* を、ロールを引き受けさせる Kubernetes サービスアカウントに置き換えます。*default* は、サービスアカウントの名前空間に置き換えます。

```
export namespace=default
export service_account=my-service-account
```

5. IAM ロール用の信頼ポリシーファイルを作成するには、次のコマンドを実行します。名前空間内のすべてのサービスアカウントにロールの使用を許可する場合は、次の内容をデバイスにコピーします。*StringEquals* を **StringLike** に置き換え、*$service_account* を * に置き換えます。以下の StringEquals または StringLike 条件に複数のエントリを追加して、複数のサービスアカウントまたは名前空間がロールを引き受けられるようにできます。クラスターが属するアカウントとは異なる

AWS アカウントのロールがロールを引き受けられるようにする方法については、「クロスアカウントの IAM アクセス許可」を参照してください。

```
cat >trust-relationship.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::$account_id:oidc-provider/$oidc_provider"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "$oidc_provider:aud": "sts.amazonaws.com",
          "$oidc_provider:sub": "system:serviceaccount:
$namespace:$service_account"
        }
      }
    }
  ]
}
EOF
```

6. ロールを作成します。*my-role* を IAM ロールの名前に、*my-role-description* をユーザーロールの表記に置き換えます。

```
aws iam create-role --role-name my-role --assume-role-policy-document
 file://trust-relationship.json --description "my-role-description"
```

7. IAM ポリシーをロールにアタッチします。*my-role* を IAM ロールの名前に置き換え、作成した既存のポリシーの名前に *my-policy* を置き換えます。

```
aws iam attach-role-policy --role-name my-role --policy-arn=arn:aws:iam::
$account_id:policy/my-policy
```

8. サービスアカウントに、サービスアカウントで引き受ける IAM ロールの Amazon リソースネーム (ARN) の注釈を付けます。*my-role* を既存の IAM ロールの名前に置き換えます。前の手順で、クラスターが属するアカウントとは異なる AWS アカウントのロールに、ロールの引き受けを許可したと仮定します。その場合、AWS アカウントおよび他の

アカウントからのロールを必ず指定してください。詳細については、「クロスアカウントの IAM アクセス許可」を参照してください。

```
kubectl annotate serviceaccount -n $namespace $service_account
 eks.amazonaws.com/role-arn=arn:aws:iam::$account_id:role/my-role
```

3. ロールとサービスアカウントが正しく設定されていることを確認します。

   a. IAM ロールの信頼ポリシーが正しく設定されていることを確認します。

   ```
   aws iam get-role --role-name my-role --query Role.AssumeRolePolicyDocument
   ```

   出力例は次のとおりです。

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Principal": {
                   "Federated": "arn:aws:iam::111122223333:oidc-provider/
   oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
               },
               "Action": "sts:AssumeRoleWithWebIdentity",
               "Condition": {
                   "StringEquals": {
                       "oidc.eks.region-code.amazonaws.com/
   id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:default:my-
   service-account",
                       "oidc.eks.region-code.amazonaws.com/
   id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com"
                   }
               }
           }
       ]
   }
   ```

   b. 前の手順でロールにアタッチしたポリシーが、そのロールにアタッチされていることを確認します。

```
aws iam list-attached-role-policies --role-name my-role --query
 AttachedPolicies[].PolicyArn --output text
```

出力例は次のとおりです。

```
arn:aws:iam::111122223333:policy/my-policy
```

c. 使用するポリシーの Amazon リソースネーム (ARN) を保存する変数を設定します。*my-policy* を、アクセス許可を確認するポリシーの名前に置き換えます。

```
export policy_arn=arn:aws:iam::111122223333:policy/my-policy
```

d. ポリシーのデフォルトバージョンを確認します。

```
aws iam get-policy --policy-arn $policy_arn
```

出力例は次のとおりです。

```
{
    "Policy": {
        "PolicyName": "my-policy",
        "PolicyId": "EXAMPLEBIOWGLDEXAMPLE",
        "Arn": "arn:aws:iam::111122223333:policy/my-policy",
        "Path": "/",
        "DefaultVersionId": "v1",
        [...]
    }
}
```

e. ポリシーの内容を表示して、Pod で必要な権限がすべて含まれていることを確認します。必要であれば、次のコマンドの *1* を、前の出力で返されたバージョンに置き換えます。

```
aws iam get-policy-version --policy-arn $policy_arn --version-id v1
```

出力例は次のとおりです。

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::my-pod-secrets-bucket"
        }
    ]
}
```

前の手順でサンプルポリシーを作成した場合、出力は同じになります。別のポリシーを作成した場合、*####*の内容は異なります。

f. Kubernetes サービスアカウントにロールが注釈されていることを確認します。

```
kubectl describe serviceaccount my-service-account -n default
```

出力例は次のとおりです。

```
Name:                my-service-account
Namespace:           default
Annotations:         eks.amazonaws.com/role-arn:
 arn:aws:iam::111122223333:role/my-role
Image pull secrets:  <none>
Mountable secrets:   my-service-account-token-qqjfl
Tokens:              my-service-account-token-qqjfl
[...]
```

4. (オプション) サービスアカウントの AWS Security Token Service エンドポイントを設定する。AWS では、グローバルエンドポイントの代わりに地域の AWS STS エンドポイントを使用することを推奨しています。これにより、レイテンシーが減少し、組み込みの冗長性が提供され、セッショントークンの有効性が向上します。

次のステップ

Kubernetes サービスアカウントを使用するように Pods を設定するには

Kubernetes サービスアカウントを使用するように Pods を設定するには

Pod が AWS のサービスにアクセスする必要がある場合、Kubernetes サービスアカウントを使用するように設定する必要があります。サービスアカウントは、AWS のサービスにアクセスする権限がある AWS Identity and Access Management (IAM) ロールに関連付ける必要があります。

前提条件

- 既存のクラスター。まだ所有していない場合は、Amazon EKS の使用開始 でのガイドのいずれかを参照しながら作成します。
- クラスター用の既存 IAM OpenID Connect (OIDC) プロバイダー。既に所有中かどうかの確認、または作成方法については「クラスターの IAM OIDC プロバイダーを作成する」を参照してください。
- IAM ロールに関連付けられている既存の Kubernetes サービスアカウント。サービスアカウントには、IAM ロールの Amazon リソースネーム (ARN) の注釈を付ける必要があります。ロールには、Pods が AWS のサービス を使用するアクセス許可を含む IAM ポリシーが関連付けられている必要があります。サービスアカウントとロールの作成および設定方法については、「IAM ロールを引き受けるように Kubernetes サービスアカウントを設定する」を参照してください。
- ご使用のデバイスまたは AWS CloudShell で、AWS Command Line Interface (AWS CLI) のバージョン `2.12.3` 以降または `1.27.160` 以降がインストールおよび設定されていること。現在のバージョンを確認するには、「**`aws --version | cut -d / -f2 | cut -d ' ' -f1`**」を参照してください。macOS の `yum`、`apt-get`、または Homebrew などのパッケージマネージャは、AWS CLI の最新バージョンより数バージョン遅れることがあります。最新バージョンをインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI のインストール、更新、およびアンインストール」と「aws configure でのクイック設定」を参照してください。AWS CloudShell にインストールされている AWS CLI バージョンは、最新バージョンより数バージョン遅れている可能性もあります。更新するには、「AWS CloudShell ユーザーガイド」の「ホームディレクトリへの AWS CLI のインストール」を参照してください。
- デバイスまたは AWS CloudShell に、`kubectl` コマンドラインツールがインストールされていること。バージョンは、ご使用のクラスターの Kubernetes バージョンと同じか、1 つ前のマイナーバージョン以前、あるいはそれより新しいバージョンが使用できます。例えば、クラスターのバージョンが `1.29` である場合、`kubectl` のバージョン `1.28`、`1.29`、または `1.30` が使用できます。`kubectl` をインストールまたはアップグレードする方法については、「`kubectl` のインストールまたは更新」を参照してください。
- クラスター構成を含む既存の `kubectl config` ファイル。`kubectl config` ファイルの作成については、「Amazon EKS クラスターの `kubeconfig` ファイルを作成または更新する」を参照してください。

Pod がサービスアカウントを使用するように設定するには

1. 次のコマンドを使用し、Pod をデプロイして設定を確認できるデプロイマニフェストを作成します。*`example values`* を自分の値に置き換えてください。

```
cat >my-deployment.yaml <<EOF
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      serviceAccountName: my-service-account
      containers:
      - name: my-app
        image: public.ecr.aws/nginx/nginx:X.XX
EOF
```

2. マニフェストをクラスターにデプロイします。

```
kubectl apply -f my-deployment.yaml
```

3. Pod に必要な環境変数が存在することを確認してください。

   a. 前の手順のデプロイ時にデプロイされた Pods を確認します。

   ```
   kubectl get pods | grep my-app
   ```

   出力例は次のとおりです。

   ```
   my-app-6f4dfff6cb-76cv9   1/1     Running   0          3m28s
   ```

   b. Pod が使用している IAM ロールの ARN を確認します。

   ```
   kubectl describe pod my-app-6f4dfff6cb-76cv9 | grep AWS_ROLE_ARN:
   ```

   出力例は次のとおりです。

```
AWS_ROLE_ARN:                 arn:aws:iam::111122223333:role/my-role
```

ロール ARN は、既存のサービスアカウントに注釈を付けたロール ARN と一致する必要があります。サービスアカウントへの注釈付けの詳細については、「IAM ロールを引き受けるように Kubernetes サービスアカウントを設定する」を参照してください。

c. Pod にウェブ ID トークンファイルのマウントがあることを確認します。

```
kubectl describe pod my-app-6f4dfff6cb-76cv9 | grep
 AWS_WEB_IDENTITY_TOKEN_FILE:
```

出力例は次のとおりです。

```
AWS_WEB_IDENTITY_TOKEN_FILE:  /var/run/secrets/eks.amazonaws.com/
serviceaccount/token
```

kubelet が、Pod に代わってトークンをリクエストし、格納します。デフォルトで kubelet は、トークンが合計有効期限の 80% を超えている場合、またはトークンが 24 時間を超えている場合、そのトークンを更新します。Pod 仕様の設定を使用して、デフォルトのサービスアカウントを除くすべてのアカウントの有効期限を変更できます。詳細については、「Kubernetes ドキュメント」の「Service Account Token Volume Projection」(サービスアカウントトークンボリュームのプロジェクション) を参照してください。

クラスターの Amazon EKS Pod Identity ウェブフックは、次の注釈が付いたサービスアカウントを使用している Pods をモニタリングします。

```
eks.amazonaws.com/role-arn: arn:aws:iam::111122223333:role/my-role
```

ウェブフックは、これらの Pods に以前の環境変数を適用します。クラスターでは、環境変数とトークンファイルのマウントを設定するために、ウェブフックを使用する必要はありません。これらの環境変数を指定するため、Pods を手動で設定できます。AWS SDK のサポートされているバージョンは、最初に認証情報チェーンプロバイダーでこれらの環境変数を探します。ロールの認証情報は、この条件を満たす Pods に使用されます。

4. ロールにアタッチされた IAM ポリシーで割り当てた権限を使用して、Pods が AWS のサービスと対話できることを確認します。

> **Note**
>
> Pod がサービスアカウントに関連付けられた IAM ロールの AWS 認証情報を使用する場合、AWS CLI またはその Pod のコンテナ内の他の SDK は、そのロールによって提供される認証情報を使用します。Amazon EKS ノード IAM ロール に提供された認証情報へのアクセスを制限しない場合、Pod は引き続きこれらの認証情報にアクセスできます。詳細については、「ワーカーノードに割り当てられたインスタンスプロファイルへのアクセスを制限する」を参照してください。

Pods が期待どおりにサービスとやり取りできない場合は、次の手順を実行して、すべてが正しく構成されていることを確認してください。

a. Pods が、OpenID Connect ウェブアイデンティティトークンファイルを介した IAM ロールの引き受けをサポートする AWS SDK バージョンを使用していることを確認します。詳細については、「サポートされる AWS SDK の使用」を参照してください。

b. デプロイがサービスアカウントを使用していることを確認します。

```
kubectl describe deployment my-app | grep "Service Account"
```

出力例は次のとおりです。

```
Service Account:  my-service-account
```

c. それでも Pods がサービスにアクセスできない場合は、IAM ロールを引き受けるように Kubernetes サービスアカウントを設定する で説明されている手順を参照して、ロールとサービスアカウントが正しく設定されていることを確認します。

サービスアカウントの AWS Security Token Service エンドポイントを設定する

サービスアカウントの IAM ロール で Kubernetes サービスアカウントを使用している場合、クラスターとプラットフォームのバージョンが同じであるか、次の表にリストされているものより後であれば、サービスアカウントで使用される AWS Security Token Service エンドポイントのタイプを設定できます。Kubernetes またはプラットフォームのバージョンが表に記載されているバージョンよりも前の場合、サービスアカウントはグローバルエンドポイントのみを使用できます。

Kubernetes バージョン	プラットフォームバージョン	デフォルトのエンドポイントタイプ
1.30	eks.2	リージョン別
1.29	eks.1	リージョン別
1.28	eks.1	リージョン別
1.27	eks.1	リージョン別
1.26	eks.1	リージョン別
1.25	eks.1	リージョン別
1.24	eks.2	リージョン別
1.23	eks.1	リージョン別

AWS では、グローバルエンドポイントの代わりに地域の AWS STS エンドポイントを使用することを推奨しています。これにより、レイテンシーが減少し、組み込みの冗長性が提供され、セッショントークンの有効性が向上します。Pod が実行中の AWS リージョンで、AWS Security Token Service がアクティブであること。さらに、AWS リージョン内のサービスに障害が発生した場合に別の AWS リージョンを使用できるよう、アプリケーションに冗長性が組み込まれている必要があります。詳細については、IAM ユーザーガイドの 「AWS リージョン での AWS STS の管理」を参照してください。

前提条件

- 既存のクラスター。まだ所有していない場合は、Amazon EKS の使用開始 でのガイドのいずれかを参照しながら作成します。
- クラスター用の既存の IAM OIDC プロバイダー。詳細については、「クラスターの IAM OIDC プロバイダーを作成する」を参照してください。
- サービスアカウント用の Amazon EKS IAM 機能を使用するように設定された、既存の Kubernetes サービスアカウント。

Kubernetes サービスアカウントで使用するエンドポイントタイプを設定するには

以下に示す例ではすべて、Amazon VPC CNI プラグイン で使用される aws-node Kubernetes サービスアカウントを使用しています。*example values* は、ユーザー独自のサービスアカウント、Pods、名前空間、およびその他のリソースに置き換えることができます。

1. エンドポイントを変更したいサービスアカウントを使用する Pod を選択します。Pod を実行する AWS リージョン を決定します。*aws-node-6mfgv* を ユーザーの Pod 名に、*kube-system* を Pod の名前空間に置き換えます。

   ```
   kubectl describe pod aws-node-6mfgv -n kube-system |grep Node:
   ```

   出力例は次のとおりです。

   ```
   ip-192-168-79-166.us-west-2/192.168.79.166
   ```

   前の出力では、Pod は us-west-2 の AWS リージョン にあるノードで実行されています。
2. Pod's のサービスアカウントが使用しているエンドポイントタイプを確認します。

   ```
   kubectl describe pod aws-node-6mfgv -n kube-system |grep AWS_STS_REGIONAL_ENDPOINTS
   ```

   出力例は次のとおりです。

   ```
   AWS_STS_REGIONAL_ENDPOINTS: regional
   ```

   現在のエンドポイントがグローバルの場合、この出力には global が返されます。出力が返されない場合、デフォルトのエンドポイントタイプが、上書きされないまま使用されています。
3. クラスターまたはプラットフォームのバージョンが表に示されているバージョンと同じかそれ以降の場合は、次のコマンドのいずれかを使用して、サービスアカウントで使用されるエンドポイントタイプをデフォルトタイプから別のタイプに変更できます。*aws-node* をサービスアカウントの名前に置き換え、*kube-system* を、サービスアカウントの名前空間に置き換えます。

   - デフォルトまたは現在のエンドポイントタイプがグローバルであり、それをリージョン別に変更する場合は、以下の手順を実行します。

     ```
     kubectl annotate serviceaccount -n kube-system aws-node eks.amazonaws.com/sts-regional-endpoints=true
     ```

サービスアカウントの IAM ロール を使用して、Pods のコンテナで実行されているアプリケーションで事前署名された S3 URL を生成している場合、リージョナルのエンドポイントの URL の形式は次の例のようになります。

```
https://bucket.s3.us-west-2.amazonaws.com/path?...&X-Amz-Credential=your-access-key-id/date/us-west-2/s3/aws4_request&...
```

- デフォルトまたは現在のエンドポイントタイプがリージョン別であり、それをグローバルに変更する場合は、次の手順を実行します。

```
kubectl annotate serviceaccount -n kube-system aws-node eks.amazonaws.com/sts-regional-endpoints=false
```

アプリケーションが、AWS STS のグローバルエンドポイントに対し明示的にリクエストを行っており、Amazon EKS クラスターにおいて、リージョン別エンドポイントを使用する際のデフォルトの動作がオーバーライドされていない場合は、リクエストが失敗しエラーが返されます。詳細については、「ポッドコンテナは次のエラーを受け取ります: An error occurred (SignatureDoesNotMatch) when calling the GetCallerIdentity operation: Credential should be scoped to a valid region」を参照してください。

Pods のコンテナで実行されているアプリケーションで、事前署名された S3 URL を生成するために サービスアカウントの IAM ロール を使用している場合、グローバルエンドポイントの URL の形式は、次の例のようになります。

```
https://bucket.s3.amazonaws.com/path?...&X-Amz-Credential=your-access-key-id/date/us-west-2/s3/aws4_request&...
```

使用している自動化処理で、事前署名付き URL に特定の形式を想定している場合、または事前署名付き URL を使用するアプリケーションやダウンストリームの依存関係に、ターゲットとして想定する AWS リージョン がある場合、適切な AWS STS エンドポイントを使用するために必要な変更を加えます。

4. 認証情報環境変数を適用するために、サービスアカウントに関連付けられている既存の Pods を削除して再作成します。変更するウェブフックは、既に実行されている Pods には適用されません。*Pods*、*kube-system*、および *-l k8s-app=aws-node* は、注釈を設定した Pods の情報に置き換えることができます。

```
kubectl delete Pods -n kube-system -l k8s-app=aws-node
```

5. Pods がすべて再起動したことを確認します。

```
kubectl get Pods -n kube-system -l k8s-app=aws-node
```

6. Pods のいずれかの環境変数を表示します。AWS_STS_REGIONAL_ENDPOINTS の値が、以前のステップで設定した値であることを確認します。

```
kubectl describe pod aws-node-kzbtr -n kube-system |grep AWS_STS_REGIONAL_ENDPOINTS
```

出力例は次のとおりです。

```
AWS_STS_REGIONAL_ENDPOINTS=regional
```

クロスアカウントの IAM アクセス許可

別のアカウントのクラスターから ID プロバイダーを作成するか、連鎖した AssumeRole オペレーションを使用することで、クロスアカウントの IAM アクセス許可を設定できます。以下の例では、アカウント A がサービスアカウントの IAM ロールをサポートする Amazon EKS クラスターを所有しています。そのクラスターで実行されている Pods は、アカウント B から IAM アクセス許可を引き受ける必要があります。

Example 別のアカウントのクラスターから ID プロバイダーを作成する

Example

この例では、アカウント A はアカウント B にクラスターからの OpenID Connect (OIDC) 発行者 URL を提供します。アカウント B は、アカウント A のクラスターからの OIDC 発行者 URL を使用して、クラスターの IAM OIDC プロバイダーを作成する および IAM ロールを引き受けるように Kubernetes サービスアカウントを設定する の手順に従います。次に、クラスター管理者は、アカウント A のクラスターのサービスアカウントに、アカウント B (*444455556666*) のロールを使用するように注釈を付けます。

```
apiVersion: v1
kind: ServiceAccount
metadata:
  annotations:
```

```
    eks.amazonaws.com/role-arn: arn:aws:iam::444455556666:role/account-b-role
```

Example 連鎖された **AssumeRole** オペレーションを使用する

Example

この例では、アカウント B は、アカウント A のクラスター内の Pods に付与するアクセス許可を持つ IAM ポリシーを作成します。アカウント B (*444455556666*) は、アカウント A (*111122223333*) への AssumeRole 許可を付与する信頼関係を持つ IAM ロールにそのポリシーをアタッチします。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "sts:AssumeRole",
      "Condition": {}
    }
  ]
}
```

アカウント A は、クラスターの OIDC 発行者アドレスで作成された ID プロバイダーから認証情報を取得する信頼ポリシーを持つロールを作成します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:oidc-provider/oidc.eks.region-
code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity"
    }
  ]
}
```

アカウント A は、アカウント B が作成したロールを引き受けるための以下のアクセス許可を持つポリシーをそのロールにアタッチします。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::444455556666:role/account-b-role"
        }
    ]
}
```

アカウント B のロールを引き受ける Pods のアプリケーションコードは、account_b_role と account_a_role の 2 つのプロファイルを使用します。account_b_role プロファイルでは、ソースとして account_a_role プロファイルを使用します。AWS CLI では、~/.aws/config ファイルは以下のようになります。

```
[profile account_b_role]
source_profile = account_a_role
role_arn=arn:aws:iam::444455556666:role/account-b-role

[profile account_a_role]
web_identity_token_file = /var/run/secrets/eks.amazonaws.com/serviceaccount/token
role_arn=arn:aws:iam::111122223333:role/account-a-role
```

他の AWS SDK の連鎖されたプロファイルを指定するには、使用する SDK のドキュメントを参照してください。詳細については、「AWS で構築するツール」を参照してください。

サポートされる AWS SDK の使用

サービスアカウントの IAM ロール を使用すると、Pods 内のコンテナは、OpenID Connect ウェブ ID トークンファイルを介した IAM ロールの引き受けをサポートする AWS SDK バージョンを使用する必要があります。お使いの AWS SDK には、必ず次のバージョン以降を使用してください。

- Java (バージョン 2) – 2.10.11
- Java – 1.11.704
- Go – 1.23.13
- Python (Boto3) – 1.9.220

- Python (botocore) – 1.12.200
- AWS CLI — 1.16.232
- ノード - 2.525.0 と 3.27.0
- Ruby – 3.58.0
- C++ – 1.7.174
- .NET - 3.3.659.1 - `AWSSDK.SecurityToken` も含めることを確認します。
- PHP – 3.110.7

Cluster Autoscaler、AWS Load Balancer Controller とは および Amazon VPC CNI plugin for Kubernetes などの一般的な Kubernetes アドオンの多くは、サービスアカウントの IAM ロールをサポートしています。

サポートされている SDK を使用していることを確認するには、コンテナを構築する際に、「AWS での構築ツール」で、希望する SDK のインストール手順に従ってください。

認証情報の使用

サービスアカウントの IAM ロールの認証情報を使用するには、コードで任意の AWS SDK を使用して SDK を含む AWS サービスのクライアントを作成できます。デフォルトでは、SDK は使用する AWS Identity and Access Management 認証情報を一連の場所で検索します。クライアントの作成時や SDK の初期化時に認証情報プロバイダーを指定しなかった場合は、サービスアカウントの認証情報用の IAM ロールが使用されます。

これがうまくいくのは、サービスアカウント用の IAM ロールがデフォルトの認証情報チェーンのステップとして追加されたからです。現在、ワークロードが認証情報チェーンの前にある認証情報を使用している場合は、同じワークロードのサービスアカウントに IAM ロールを設定しても、その認証情報は引き続き使用されます。

SDK は、`AssumeRoleWithWebIdentity` アクションを使用してサービスアカウント OIDC トークンを AWS Security Token Service からの一時的な認証情報と自動的に交換します。Amazon EKS とこの SDK アクションは、有効期限が切れる前に一時認証情報を更新することで、引き続きローテーションを行います。

署名キーを取得する

Kubernetes は、各 Kubernetes Service Account に `ProjectedServiceAccountToken` を発行します。このトークンは OIDC トークンであり、さらに JSON web token (JWT) の一種です。Amazon

EKS は、外部システムでトークンを検証できるように、トークンの署名キーを含むクラスターごとにパブリック OIDC エンドポイントをホストします。

`ProjectedServiceAccountToken` を検証するには、JSON Web Key Set (JWKS) とも呼ばれる OIDC パブリック署名キーを取得する必要があります。アプリケーションでこれらのキーを使用してトークンを検証します。例えば、PyJWT Python ライブラリを使用して、これらのキーを使用してトークンを検証できます。`ProjectedServiceAccountToken` の詳細については、「the section called “IAM、Kubernetes、および OpenID Connect (OIDC) の背景情報”」を参照してください。

前提条件

- クラスター用の既存 AWS Identity and Access Management IAM OpenID Connect (OIDC) プロバイダー。既に存在しているかどうかを確認する、または作成するには「クラスターの IAM OIDC プロバイダーを作成する」を参照してください。
- AWS CLI – Amazon EKS など AWS のサービスを操作するためのコマンドラインツールです。詳細については、ユーザーガイドの、AWS CLI のインストール、更新、およびアンインストール AWS Command Line Interfaceを参照してください。AWS CLI のインストール後は、設定も行っておくことをお勧めします。詳細については、ユーザーガイドの`aws configure` でクイック設定 AWS Command Line Interfaceを参照してください。

OIDC パブリック署名キーを取得する (AWS CLI)

1. AWS CLI を使用して Amazon EKS クラスターの OIDC URL を取得します。

```
$ aws eks describe-cluster --name my-cluster --query 'cluster.identity.oidc.issuer'
"https://oidc.eks.us-west-2.amazonaws.com/id/8EBDXXXX00BAE"
```

2. curl、または同様のツールを使用して、パブリック署名キーを取得します。結果は、JSON Web Key Set (JWKS) です。

> **Important**
>
> Amazon EKS は、OIDC エンドポイントへの呼び出しをスロットリングします。パブリック署名キーをキャッシュする必要があります。レスポンスに含まれる `cache-control` ヘッダーを考慮します。

> **Important**
>
> Amazon EKS は OIDC 署名キーを 7 日ごとにローテーションします。

```
$ curl https://oidc.eks.us-west-2.amazonaws.com/id/8EBDXXXX00BAE/keys
{"keys":
[{"kty":"RSA","kid":"2284XXXX4a40","use":"sig","alg":"RS256","n":"wklbXXXXMVfQ","e":"AQAB"}
```

Amazon EKS ノード

Kubernetes ノードは、コンテナ化されたアプリケーションを実行するマシンです。各ノードには、以下のコンポーネントがあります。

- コンテナランタイム - コンテナの実行を担当するソフトウェア。
- **`kubelet`** - コンテナが正常で、関連する Pod 内で実行されていることを確認します。
- **`kube-proxy`** - Pods との通信を許可するネットワークルールを維持します。

詳細については、「Kubernetes ドキュメント」の「Nodes」(ノード) を参照してください。

Amazon EKS クラスターは、セルフマネージド型ノード、Amazon EKS マネージド型ノードグループ、AWS Fargate をさまざまに組み合わせて Pods をスケジュールできます。クラスターにデプロイされているノードの詳細については、Kubernetes リソースを表示するを参照してください。

> **Important**
>
> Amazon EKS を使用した AWS Fargate は、AWS GovCloud (米国東部) および AWS GovCloud (米国西部) ではご利用いただけません。

> **Note**
>
> ノードは、クラスターの作成時に選択したサブネットと同じ VPC 内にある必要があります。ただし、ノードは同じサブネット内にある必要はありません。

次の表に、要件を満たすための最適なオプションを決定する際に、評価すべき基準をいくつか示します。このテーブルには、Amazon EKS の外部で作成された接続されたノードは含まれず、閲覧のみ可能です。

> **Note**
>
> Bottlerocket には、この表の一般的な情報と特に異なる点がいくつかあります。詳細については、「GitHub」の「Bottlerocket ドキュメント」を参照してください。

条件	EKS マネージド型ノードグループ	セルフマネージド型ノード	AWS Fargate
AWS Outposts にデプロイできます。	いいえ	はい	いいえ
AWS Local Zone にデプロイ可能	いいえ	はい — 詳細については、「Amazon EKS と AWS Local Zones」を参照してください。	いいえ
Windows を必要とするコンテナを実行できます	はい	はい - ただし、クラスターには少なくとも 1 つ (可用性を考慮して 2 つを推奨) の Linux ノードが必要です。	いいえ
Linux を必要とするコンテナを実行できます	はい	はい	はい
Inferentia チップを必要とするワークロードを実行可能	はい – Amazon Linux ノードのみ	はい – Amazon Linux のみ	いいえ
GPU を必要とするワークロードを実行可能	はい – Amazon Linux ノードのみ	はい – Amazon Linux のみ	いいえ
Arm プロセッサを必要とするワークロードを実行可能	はい	はい	いいえ
AWS Bottlerocket を実行可能	はい	はい	いいえ
ポッドは、カーネルランタイム環境を他の Pods ポッドと共有します。	はい - 各ノード上のすべての Pods	はい - 各ノード上のすべての Pods	いいえ - 各 Pod には専用のカー

条件	EKS マネージド型ノードグループ	セルフマネージド型ノード	AWS Fargate
			ネルがあります。
ポッドは、CPU、メモリ、ストレージ、およびネットワークリソースを他の Pods のポッドと共有します。	Yes: 各ノードでリソースが未使用の状態になります。	はい - 各ノードでリソースが未使用の状態になります。	いいえ - 各 Pod には専用のリソースがあり、リソースの使用率を最大化するために個別にサイズ設定できます。
ポッドは、Pod 仕様でリクエストされているよりも多くのハードウェアとメモリを使用できます	はい - Pod がリクエストよりも多くのリソースを必要とし、リソースがノードで使用可能である場合、Pod は追加のリソースを使用できます。	はい - Pod がリクエストよりも多くのリソースを必要とし、リソースがノードで使用可能である場合、Pod は追加のリソースを使用できます。	いいえ - Pod は、より大きな vCPU とメモリ設定を使用して再展開できます。

条件	EKS マネージド型ノードグループ	セルフマネージド型ノード	AWS Fargate
Amazon EC2 インスタンスをデプロイおよび管理する必要がある	はい – Amazon EKS 最適化 AMI をデプロイした場合、Amazon EKS によって自動化されます。カスタム AMI をデプロイした場合は、インスタンスを手動で更新する必要があります。	はい - 手動で設定するか、Amazon EKS が提供する AWS CloudFormation テンプレートを使用して、Linux (x86)、Linux (Arm)、または Windows ノードをデプロイします。	いいえ
Amazon EC2 インスタンスのオペレーティングシステムの保護、保守、パッチ適用を行う必要がある	はい	はい	いいえ
ノードをデプロイする時に、追加の `kubelet` 引数 などのブートストラップ引数を提供できます。	はい – `eksctl`、またはカスタム AMI を含む起動テンプレートを使用	はい - 詳細については、GitHub の「ブートストラップスクリプトの使用に関する情報」を参照してください。	いいえ

条件	EKS マネージド型ノードグループ	セルフマネージド型ノード	AWS Fargate
ノードに割り当てられた IP アドレスとは異なる CIDR ブロックから、IP アドレスを Pods に割り当てることができます。	はい - カスタム AMI の起動テンプレートを使用します。詳細については、「起動テンプレートを使用したマネージドノードのカスタマイズ」を参照してください。	はい — 詳細については、「ポッド用のカスタムネットワーク」を参照してください。	いいえ
ノードに SSH 接続可能	はい	はい	No – SSH 接続を行うノードホストオペレーティングシステムはありません。
独自のカスタム AMI をノードにデプロイ可能	はい – 起動テンプレートを使用します	はい	いいえ
独自のカスタム CNI をノードにデプロイ可能	はい – カスタム AMI の起動テンプレートを使用します	はい	いいえ

条件	EKS マネージド型ノードグループ	セルフマネージド型ノード	AWS Fargate
ノード AMI を自分で更新する必要がある	はい - Amazon EKS 最適化 AMI をデプロイした場合、更新が利用可能になると Amazon EKS コンソールで通知されます。コンソールでは、ワンクリックで更新を実行できます。カスタム AMI をデプロイした場合、更新が利用可能になっても Amazon EKS コンソールに通知されません。更新は自分で実行する必要があります。	はい – Amazon EKS コンソール以外のツールを使用する。これは、セルフマネージド型ノードは Amazon EKS コンソールでは管理できないためです。	いいえ

条件	EKS マネージド型ノードグループ	セルフマネージド型ノード	AWS Fargate
ノードの Kubernetes バージョンを自分で更新する必要があります	はい - Amazon EKS 最適化 AMI をデプロイした場合、更新が利用可能になると Amazon EKS コンソールで通知されます。コンソールでは、ワンクリックで更新を実行できます。カスタム AMI をデプロイした場合、更新が利用可能になっても Amazon EKS コンソールに通知されません。更新は自分で実行する必要があります。	はい – Amazon EKS コンソール以外のツールを使用する。これは、セルフマネージド型ノードは Amazon EKS コンソールでは管理できないためです。	いいえ: ノードを管理しません。
Pods で Amazon EBS ストレージが使用可能	はい	はい	いいえ
Pods で Amazon EFS ストレージが使用可能	はい	はい	はい
Pods で Amazon FSx for Lustre ストレージが使用可能	はい	はい	いいえ

条件	EKS マネージド型ノードグループ	セルフマネージド型ノード	AWS Fargate
Network Load Balancer をサービスに使用可能	はい	はい	はい (ネットワークロードバランサーを作成するを使用可能な場合)
ポッドはパブリックサブネットで実行可能	はい	はい	いいえ
それぞれの Pods に、異なる VPC セキュリティグループが割り当て可能	はい - Linux ノードのみ	はい - Linux ノードのみ	はい
Kubernetes DaemonSets を実行可能	はい	はい	いいえ
Pod マニフェストで `HostPort` および `HostNetwork` をサポートします	はい	はい	いいえ
AWS リージョン の入手可能性	すべての Amazon EKS 対応リージョン	すべての Amazon EKS 対応リージョン	一部の Amazon EKS 対応リージョン
Amazon EC2 専有ホストでコンテナを実行できます	はい	はい	いいえ

条件	EKS マネージド型ノードグループ	セルフマネージド型ノード	AWS Fargate
料金	複数の Pods を実行する Amazon EC2 インスタンスのコスト。詳細については、「Amazon EC2 料金」を参照してください。	複数の Pods を実行する Amazon EC2 インスタンスのコスト。詳細については、「Amazon EC2 料金表」を参照してください。	個々のFargate メモリとCPU 構成のコスト。各 Pod には、独自のコストがあります。詳細については、「AWS Fargate 料金表」を参照してください。

マネージド型ノードグループ

Amazon EKS マネージド型ノードグループは、Amazon EKS Kubernetes クラスターのノード (Amazon EC2 インスタンス) のプロビジョニングとライフサイクル管理を自動化します。

Amazon EKS マネージド型ノードグループでは、Kubernetes アプリケーションを実行するためのコンピューティング性能を提供する Amazon EC2 インスタンスを個別にプロビジョニングまたは登録する必要はありません。1 回の操作で、クラスターのノードを作成、自動的に更新、または終了できます。ノードを更新し、終了させることで、ノードを自動的にドレーンし、アプリケーションを利用できる状態にしておきます。

すべてのマネージド型ノードは、Amazon EKS によって管理される Amazon EC2 Auto Scaling グループの一部としてプロビジョニングされます。インスタンスや Auto Scaling グループを含むすべてのリソースは、AWS アカウント内で実行されます。各ノードグループは、定義された複数のアベイラビリティーゾーンで実行できます。

Amazon EKS コンソール、`eksctl`、AWS CLI、AWS API、または AWS CloudFormation を含む Infrastructure as Code ツールを使用し、新規または既存のクラスターにマネージド型ノードグループを追加できます。マネージド型ノードグループの一部として起動されたノードは、自動的に Kubernetes Cluster Autoscaler によって自動検出用にタグ付けされます。ノードグループを使用すると、Kubernetes ラベルをノードに適用し、いつでも更新することができます。

Amazon EKS マネージド型ノードグループの使用に追加料金はかかりません。お支払いいただくのはプロビジョニングした AWS リソースの分だけです。これには、Amazon EC2 インスタンス、Amazon EBS ボリューム、Amazon EKS クラスター時間、その他の AWS インフラストラクチャが含まれます。最低料金や前払いの義務は発生しません。

新しい Amazon EKS クラスターおよびマネージド型ノードグループの使用を開始するには、「Amazon EKS の開始方法 – AWS Management Console と AWS CLI」を参照してください。

既存のクラスターにマネージド型ノードグループを追加するには、「マネージド型ノードグループの作成」を参照してください。

マネージド型ノードグループの概念

- Amazon EKS マネージド型ノードグループは、Amazon EC2 インスタンスを作成し、管理します。
- すべてのマネージド型ノードは、Amazon EKS によって管理される Amazon EC2 Auto Scaling グループの一部としてプロビジョニングされます。さらに、Amazon EC2 インスタンスや Auto Scaling グループを含むすべてのリソースは、AWS アカウント内で実行されます。
- マネージド型ノードグループの Auto Scaling グループは、グループの作成時に指定するすべてのサブネットにまたがります。
- Amazon EKS は、Kubernetes Cluster Autoscaler の使用を設定するようにマネージド型ノードグループリソースをタグ付けします。

> **Important**
>
> Amazon EBS ボリュームによってバックアップされ、Kubernetes Autoscaling を使用する複数のアベイラビリティーゾーンにわたってステートフルアプリケーションを実行している場合、それぞれが単一のアベイラビリティーゾーンにスコープされる複数のノードグループを設定する必要があります。また、`--balance-similar-node-groups` 機能を有効にする必要があります。

- カスタム起動テンプレートを使用すると、マネージド型ノードをデプロイするときに、柔軟性とカスタマイズのレベルを向上させることができます。例えば、追加の `kubelet` 引数を指定して、カスタム AMI を使用できます。詳細については、「起動テンプレートを使用したマネージドノードのカスタマイズ」を参照してください。マネージド型ノードグループを最初に作成するときにカスタム起動テンプレートを使用しない場合は、自動生成された起動テンプレートを使用できます。自動生成されたテンプレートを手動で変更しないでください。エラーが発生します。

- Amazon EKS は、マネージド型ノードグループで CVE およびセキュリティパッチの責任共有モデルに従います。マネージド型ノードが Amazon EKS 最適化 AMI を実行する場合、バグや問題が報告されると、Amazon EKS が AMI のパッチ適用バージョンの構築を担当します。修正を公開できます。ただし、これらのパッチが適用された AMI バージョンのマネージド型ノードグループへのデプロイはユーザーが担当します。マネージド型ノードでカスタム AMI を実行する場合、バグや問題が報告され、AMI をデプロイする場合は、ユーザーがパッチ適用バージョンの構築を担当します。詳細については、「マネージド型ノードグループの更新」を参照してください。
- Amazon EKS マネージド型ノードグループは、パブリックサブネットとプライベートサブネットの両方で起動できます。2020 年 4月 22 日 以降にパブリックサブネットでマネージド型ノードグループを起動した場合、インスタンスがクラスターに正常に参加するには、サブネットで `MapPublicIpOnLaunch` を true に設定する必要があります。パブリックサブネットが `eksctl`、または 2020 年 3 月 26 日以降に Amazon EKS が販売した AWS CloudFormation テンプレートを使用して作成された場合、この設定はすでに true に設定されています。パブリックサブネットが 2020 年 3 月 26 日 より前に作成されている場合は、設定を手動で変更する必要があります。詳細については、「サブネットのパブリック IPv4 アドレッシング属性の変更」を参照してください。
- マネージド型ノードグループをプライベートサブネットにデプロイする場合、Amazon ECR にアクセスしてコンテナイメージをプルできることを確認します。これを行うには、NAT ゲートウェイをサブネットのルートテーブルに接続するか、次の AWS PrivateLink VPC エンドポイントを追加します。
  - Amazon ECR API のエンドポイントインターフェイス — `com.amazonaws.region-code.ecr.api`
  - Amazon ECR Docker レジストリ API のエンドポイントインターフェイス — `com.amazonaws.region-code.ecr.dkr`
  - Amazon S3 ゲートウェイのエンドポイント - `com.amazonaws.region-code.s3`

  その他の一般的に使用されるサービスとエンドポイントについては、「プライベートクラスターの要件」を参照してください。
- マネージド型ノードグループは、AWS Outposts、AWS Wavelength または AWS Local Zones にはデプロイできません。
- 1 つのクラスター内に複数のマネージド型ノードグループを作成できます。例えば、一部のワークロード用に、標準 Amazon EKS 最適化 Amazon Linux AMI を使用するノードグループを作成し、GPU サポートを必要とするワークロード用に GPU バリアントを使用する別のノードグループを作成できます。

- マネージド型ノードグループで Amazon EC2 インスタンスのステータスチェックに障害が発生した場合、Amazon EKS は問題の診断に役立つエラーコードを返します。詳細については、「マネージド型ノードグループのエラー」を参照してください。
- Amazon EKS は、マネージド型ノードグループインスタンスに Kubernetes ラベルを追加します。これらの Amazon EKS によって提供されたラベルには、プレフィックス `eks.amazonaws.com` が付与されます。
- Amazon EKS では、終了または更新時に Kubernetes API を使用して、ノードを自動的にドレーンします。
- `AZRebalance` を使用してノードを終了、または目的のノード数を減らす際には、ポッドの停止状態の予算は考慮されません。これらのアクションはノードの Pods を削除しようとします。しかし、15 分を超過する場合、ノード上のすべての Pods が終了しているかどうかにかかわらず、ノードは終了します。ノードが終了するまでの期間を延長するには、Auto Scaling グループにライフサイクルフックを追加します。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「ライフサイクルフックの追加」を参照してください。
- スポット中断通知またはキャパシティリバランス通知を受け取った後、ドレインプロセスを正しく実行するには、`CapacityRebalance` を `true` に設定する必要があります。
- マネージドノードグループを更新すると、Pods 用に設定した Pod の停止状態の予算が考慮されます。詳細については、「マネージド型ノードの更新動作」を参照してください。
- Amazon EKS マネージド型ノードグループの使用に、追加料金はかかりません。プロビジョニングした AWS リソースに対してのみ料金が発生します。
- ノードの Amazon EBS ボリュームを暗号化する場合は、起動テンプレートを使用してノードをデプロイできます。起動テンプレートを使用せずに、暗号化された Amazon EBS ボリュームを持つマネージド型ノードをデプロイするには、アカウントに作成された新しい Amazon EBS ボリュームをすべて暗号化してください。詳細については、「Amazon EC2 ユーザーガイド」の「デフォルトで暗号化」を参照してください。

マネージド型ノードグループのキャパシティータイプ

マネージド型ノードグループを作成する場合、キャパシティータイプをオンデマンドまたはスポットのいずれかから選択できます。Amazon EKS は、Amazon EC2 Auto Scaling グループを使用して、マネージド型ノードグループをデプロイします。これにはオンデマンドのみか、Amazon EC2 スポットインスタンスのみが含まれます。単一の Kubernetes クラスター内で、耐障害性のあるアプリケーションの Pods をスポットのマネージド型ノードグループに、耐障害性のないアプリケーションのポッドをオンデマンドのノードグループに、それぞれスケジュールできます。デフォルトでは、マネージド型ノードグループはオンデマンド Amazon EC2 インスタンスをデプロイします。

オンデマンド

オンデマンドインスタンスでは、長期契約なしで、コンピューティング性能に対して秒単位で支払います。

仕組み

デフォルトでは、[容量タイプ] を指定しない場合、マネージド型ノードグループはオンデマンドインスタンスでプロビジョニングされます。マネージド型ノードグループは、ユーザーの代わりに Amazon EC2 Auto Scaling グループを次のように設定します。

- オンデマンドキャパシティーをプロビジョニングするための配分戦略は、`prioritized` に設定されます。マネージド型ノードグループは、API 内部で渡されたインスタンスタイプの優先度を使用して、オンデマンドキャパシティーを満たすときに最初に使用するインスタンスタイプを決定します。例えば、3 つのインスタンスタイプを `c5.large`、`c4.large`、`c3.large` の順序で指定するとします。オンデマンドインスタンスが起動されると、マネージド型ノードグループは `c5.large`、`c4.large`、`c3.large` の順でオンデマンドキャパシティーを満たします。詳しくは、Amazon EC2 Auto Scaling ユーザーガイドの Amazon EC2 Auto Scaling グループを参照してください。
- Amazon EKS は、キャパシティータイプを `eks.amazonaws.com/capacityType: ON_DEMAND` に指定しているマネージド型ノードグループ内のすべてのノードに、次の Kubernetes ラベルを追加します。このラベルを使用すると、オンデマンドノードで、ステートフルまたは非フォールトトレラントなアプリケーションをスケジュールできます。

スポット

Amazon EC2 スポットインスタンスは、オンデマンドの料金から大きな割引で提供される、Amazon EC2 の予備容量です。Amazon EC2 スポットインスタンスは、EC2 から容量の返却を求められると 2 分間の中断通知をもって中断されることがあります。詳細については、「Amazon EC2 ユーザーガイド」の「スポットインスタンス」を参照してください。Amazon EC2 スポットインスタンスを使用してマネージド型ノードグループを構成し、Amazon EKS クラスターで実行されているコンピューティングノードのコストを最適化できます。

仕組み

マネージド型ノードグループ内でスポットインスタンスを使用するには、キャパシティータイプを `spot` に設定してマネージド型ノードグループを作成してください。マネージド型ノードグループは、ユーザーの代わりに Amazon EC2 Auto Scaling グループを設定し、次のようなスポットベストプラクティスを適用します。

- Spot ノードが最適な Spot キャパシティプールにプロビジョニングされるように、割り当て戦略を以下のいずれかに設定します。
  - `price-capacity-optimized` (PCO) – Kubernetes バージョン `1.28` 以上のクラスタに新しいノードグループを作成する場合、割り当てストラテジーは `price-capacity-optimized` に設定されます。ただし、Amazon EKS マネージドノードグループが PCO をサポートし始める前に `capacity-optimized` で作成済みのノードグループの割り当て戦略は変更されません。
  - `capacity-optimized` (CO) – Kubernetes バージョン `1.27` 以下のクラスターに新しいノードグループを作成すると、割り当て戦略は `capacity-optimized` に設定されます。

  容量を割り当てるために利用可能なスポットキャパシティープールの数を増やすには、複数のインスタンスタイプを使用するようにマネージド型ノードグループを設定します。
- スポットノードが中断するリスクが高い場合に、Amazon EKS がスポットノードを適切にドレーンおよび再調整して、アプリケーションの中断を最小限に抑えられるように、Amazon EC2 Spot Capacity Rebalancing が有効化されています。詳細については、Amazon EC2 Auto Scaling ユーザーガイドの Amazon EC2 Auto Scaling 容量の再調整を参照してください。
  - スポットノードが再調整の推奨事項を受け取ると、Amazon EKS は自動的に新しい代替スポットノードの起動を試みます。
  - 代替スポットノードが `Ready` 状態になる前にスポットの 2 分間の中断通知が到着すると、Amazon EKS は再調整に関する推奨事項を受け取ったスポットノードのドレーンを開始します。Amazon EKS はベストエフォートベースでノードをドレインします。そのため、Amazon EKS が既存のノードをドレインする前に、代替ノードがクラスターに参加するのを待機するという保証はありません。
  - 代替スポットノードがブートストラップされ、Kubernetes 上で `Ready` 状態になると、Amazon EKS は再調整に関する推奨事項を受信したスポットノードを遮断し、ドレーンします。スポットノードを遮断すると、サービスコントローラーがこのスポットノードに新しいリクエストを送信しないようにします。正常でアクティブなスポットノードのリストからも削除されます。スポットノードをドレインすると、実行中の Pods が正常に削除されます。
- Amazon EKS は、キャパシティータイプを `eks.amazonaws.com/capacityType: SPOT` に指定しているマネージド型ノードグループ内のすべてのノードに、次の Kubernetes ラベルを追加します。このラベルを使用して、スポットノードで耐障害性のあるアプリケーションをスケジュールできます。

キャパシティータイプを選択する際の考慮事項

オンデマンドキャパシティーとスポットキャパシティーのどちらでノードグループをデプロイするかを決定するときは、次の条件を考慮する必要があります。

- スポットインスタンスは、ステートレスかつフォールトトレラントで、柔軟性の高いアプリケーションに適しています。これには、バッチトレーニングワークロード、機械学習トレーニングワークロード、Apache Spark などのビッグデータ ETL、キュー処理アプリケーション、ステートレス API エンドポイントなどがあります。スポットは Amazon EC2 の予備容量であり、時間の経過とともに変化する可能性があるため、中断されても問題ないワークロードには、スポットキャパシティーを使用することをお勧めします。具体的には、スポット容量は、必要な容量が使用できない期間を許容できるワークロードに適しています。
- 非フォールトトレラントなアプリケーションには、オンデマンドを使用することをお勧めします。これには、モニタリングツールやオペレーションツールなどのクラスター管理ツール、`StatefulSets` を必要とするデプロイ、データベースなどのステートフルなアプリケーションなどが含まれます。
- スポットインスタンスの使用中にアプリケーションの可用性を最大化するには、スポットのマネージド型ノードグループが複数のインスタンスタイプを使用するように構成することをお勧めします。複数のインスタンスタイプを使用する場合は、次のルールを適用することをお勧めします。
  - マネージド型ノードグループ内で Cluster Autoscaler を使用している場合、同じサイズの vCPU とメモリリソースを持つインスタンスタイプを柔軟に組み合わせて使用することをお勧めします。これは、クラスター内のノードが期待どおりにスケールされるようにするためです。例えば、4 つの vCPU と 8 GiB のメモリが必要な場合は、`c3.xlarge`、`c4.xlarge`、`c5.xlarge`、`c5d.xlarge`、`c5a.xlarge`、`c5n.xlarge`、またはその他の同等なインスタンスタイプを使用してください。
  - アプリケーションの可用性を高めるために、複数のスポットマネージド型ノードグループをデプロイすることをお勧めします。これを行うには、各グループが、同じ vCPU とメモリリソースを持つ、インスタンスタイプの柔軟な組み合わせを使用する必要があります。例えば、4 つの vCPU および 8 GiB のメモリが必要な場合は、`c3.xlarge`、`c4.xlarge`、`c5.xlarge`、`c5d.xlarge`、`c5a.xlarge`、`c5n.xlarge`、または他の同等なインスタンスタイプのマネージド型ノードグループを 1 つ作成し、`m3.xlarge`、`m4.xlarge`、`m5.xlarge`、`m5d.xlarge`、`m5a.xlarge`、`m5n.xlarge`、または他の同等なインスタンスタイプで 2 つ目のマネージド型ノードグループを作成することをお勧めします。
  - カスタム起動テンプレートを使用しているスポットキャパシティータイプでノードグループをデプロイする場合、API を使用して複数のインスタンスタイプを渡します。起動テンプレートを

使って 1 つのインスタンスタイプを渡さないでください。起動テンプレートを使用したノードグループのデプロイについての詳細は、「起動テンプレートを使用したマネージドノードのカスタマイズ」をご覧ください。

マネージド型ノードグループの作成

このトピックでは、Amazon EKS クラスターに登録しているノードの、Amazon EKS マネージド型ノードグループを起動する方法を説明します。ノードがクラスターに参加したら、それらのノードに Kubernetes アプリケーションをデプロイ可能になります。

Amazon EKS マネージド型ノードグループを初めて起動する場合は、このガイドではなく Amazon EKS の使用開始 ガイドのいずれかに従うことをお勧めします。このガイドでは、ノードを使用して Amazon EKS クラスターを作成するためのウォークスルーを説明します。

Important

- Amazon EKS ノードは、標準の Amazon EC2 インスタンスです。通常の Amazon EC2 料金に基づいて請求されます。詳細については、「Amazon EC2 の料金」を参照してください。
- AWS Outposts、AWS Wavelength または AWS の Local Zones が有効になっている AWS リージョン では、マネージド型ノードを作成できません。AWS Outposts、 AWS Wavelength、あるいは AWS Local Zones が有効になっている AWS リージョン では、セルフマネージド型ノードを作成できます。詳細については、「セルフマネージド型の Amazon Linux ノードの起動」、「 セルフマネージド型 Windows ノードの起動」および「セルフマネージド型 Bottlerocket ノードの起動」を参照してください。また、Outpost に自己管理型 Amazon Linux ノードグループを作成することもできます。詳細については、「Outpost 上のセルフマネージド型の Amazon Linux ノードの起動」を参照してください。
- Amazon EKS 最適化 Linux または Bottlerocket に含まれる `bootstrap.sh` ファイル用に AMI ID を指定しない場合、マネージドノードグループは `maxPods` の値に最大数を適用します。vCPU が 30 未満のインスタンスの場合、最大数は `110` です。30 を超える vCPU を持つインスタンスの場合、最大数は `250` に跳ね上がります。これらの数値は、内部の Amazon EKS スケーラビリティチームのテストによる Kubernetes スケーラビリティのしきい値と推奨設定に基づいています。詳細については、「Amazon VPC CNI プラグインがノードあたりのポッド数の制限を引き上げ」のブログ記事を参照してください。

前提条件

- 既存の Amazon EKS クラスター。デプロイするには、「Amazon EKS クラスターの作成」を参照してください。
- ノードが使用する既存の IAM ロール。作成する場合は「Amazon EKS ノードの IAM ロール」を参照してください。このロールに VPC CNI のポリシーがどちらも含まれていない場合、VPC CNI ポッドには以下の別のロールが必要です。
- (オプションですが推奨) 必要な IAM ポリシーがアタッチされた独自の IAM ロールで設定された Amazon VPC CNI plugin for Kubernetes アドオン。詳細については、「サービスアカウントの IAM ロールを使用する Amazon VPC CNI plugin for Kubernetes の設定」を参照してください。
- 「Amazon EC2 インスタンスタイプを選択する」に記載されている考慮事項に関する知識。選択したインスタンスタイプによっては、クラスターと VPC に関する追加の前提条件がある場合もあります。
- Windows マネージドノードグループを追加するには、まずクラスターの Windows サポートを有効にする必要があります。詳細については、「Amazon EKS クラスター の Windows サポートの有効化」を参照してください。

マネージド型ノードグループを作成するには、`eksctl` または AWS Management Console を使用します。

eksctl

`eksctl` を使用してマネージド型ノードグループを作成する

この手順には、`eksctl` バージョン `0.183.0` 以降が必要です。お使いのバージョンは、以下のコマンドを使用して確認できます。

```
eksctl version
```

`eksctl` のインストールまたはアップグレードの手順については、`eksctl` ドキュメントの「インストール」を参照してください。

1. (オプション) [AmazonEKS_CNI_Policy] が管理する IAM ポリシーが Amazon EKS ノードの IAM ロール にアタッチされている場合は、代わりに Kubernetes `aws-node` サービスアカウントに関連付けた IAM ロールに割り当てることをお勧めします。詳細については、「サービスアカウントの IAM ロールを使用する Amazon VPC CNI plugin for Kubernetes の設定」を参照してください。

2. カスタム起動テンプレートの有無にかかわらず、マネージド型ノードグループを作成します。起動テンプレートを手動で指定すると、ノードグループをより詳細にカスタマイズできます。たとえば、カスタム AMI をデプロイしたり、Amazon EKS 最適化 AMI の boostrap.sh スクリプトの引数を指定したりできます。すべての使用可能なオプションとデフォルト値の一覧を表示するには、次のコマンドを入力します。

```
eksctl create nodegroup --help
```

次のコマンドで、*my-cluster* をクラスターの名前に置き換え、*my-mng* をノードグループの名前に置き換えます。ノードグループ名は 63 文字以下である必要があります。先頭は文字または数字でなければなりませんが、残りの文字にはハイフンおよびアンダースコアを含めることもできます。

> **Important**
>
> マネージド型ノードグループを最初に作成する際にカスタム起動テンプレートを使用しない場合は、後でノードグループに使用しないでください。カスタム起動テンプレートを指定しなかった場合、システムにより起動テンプレートが自動生成されます。これを手動で変更することはお勧めしません。自動生成された起動テンプレートを手動で変更すると、エラーが発生する場合があります。

起動テンプレートなし

eksctl は、デフォルトの Amazon EC2 起動テンプレートをアカウント内に作成し、指定したオプションに基づいて作成した起動テンプレートを使用してノードグループをデプロイします。--node-type の値を指定する前に「Amazon EC2 インスタンスタイプを選択する」を参照してください。

ami-family を許可されているキーワードに置き換えます。詳細については、「eksctl ドキュメント」の「Setting the node AMI Family」(ノード AMI ファミリーの設定) を参照してください。*my-key* を Amazon EC2 キーペアまたはパブリックキーの名前に置き換えます。このキーは、起動後のノードに SSH 接続するために使用されます。

> **Note**
>
> Windows の場合、このコマンドは SSH を有効にしません。代わりに Amazon EC2 キーペアをインスタンスに関連付け、インスタンスに RDP できるようにします。

Amazon EC2 キーペアをまだ持っていない場合は、AWS Management Console で作成できます。Linux の詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EC2 のキーペアと Linux インスタンス」を参照してください。Windows の詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EC2 のキーペアと Windows インスタンス」を参照してください。

次の条件が true の場合、IMDS への Pod アクセスをブロックすることをお勧めします。

- すべての Kubernetes サービスアカウントに IAM ロールを割り当てて、Pods が必要最小限のアクセス許可のみを持つように計画しています。
- クラスター内の Pods が、現在の AWS リージョン の取得といったその他の理由で Amazon EC2 インスタンスメタデータサービス (IMDS) へのアクセスを必要としていません。

詳細については、「ワーカーノードに割り当てられたインスタンスプロファイルへのアクセスを制限する」を参照してください。

IMDS への Pod アクセスをブロックする場合は、次のコマンドに **--disable-pod-imds** オプションを追加します。

```
eksctl create nodegroup \
  --cluster my-cluster \
  --region region-code \
  --name my-mng \
  --node-ami-family ami-family \
  --node-type m5.large \
  --nodes 3 \
  --nodes-min 2 \
  --nodes-max 4 \
  --ssh-access \
  --ssh-public-key my-key
```

インスタンスは、オプションで、Pods に非常に多くの IP アドレスを割り当て、インスタンスとは異なる CIDR ブロックから Pods に IP アドレスを割り当て、インターネットにアクセスせずにクラスターにデプロイできます。詳細については、追加オプションのAmazon EC2 ノードで使用可能な IP アドレスの量を増やす、ポッド用のカスタムネットワークおよび プライベートクラスターの要件 を参照して、前のコマンドに追加します。

マネージドノードグループは、インスタンスタイプに基づいて、ノードグループの各ノードで実行できる Pods の最大数に対して 1 つの値を計算して適用します。異なるインスタンスタイプを持つノードグループを作成する場合、すべてのインスタンスタイプで計算された最小値が、ノードグループ内のすべてのインスタンスタイプで実行できる Pods の最大数として適用されます。マネージド型ノードグループは、各 Amazon EC2 インスタンスタイプの Amazon EKS 推奨最大 Pods 数 で参照されているスクリプトを使用して値を計算します。

起動テンプレートの使用

起動テンプレートが既に存在しており、「起動テンプレート設定の基本」で指定した要件を満たしている必要があります。

次の条件が true の場合、IMDS への Pod アクセスをブロックすることをお勧めします。

- すべての Kubernetes サービスアカウントに IAM ロールを割り当てて、Pods が必要最小限のアクセス許可のみを持つように計画しています。
- クラスター内の Pods が、現在の AWS リージョン の取得といったその他の理由で Amazon EC2 インスタンスメタデータサービス (IMDS) へのアクセスを必要としていません。

詳細については、「ワーカーノードに割り当てられたインスタンスプロファイルへのアクセスを制限する」を参照してください。

IMDS への Pod のアクセスをブロックするには、起動テンプレートで必要な設定を行います。

a. 次のコンテンツをデバイスにコピーします。*example values* を置き換えたら、変更したコマンドを実行して `eks-nodegroup.yaml` ファイルを作成します。起動テンプレートなしでデプロイしたときに指定したいくつかの設定は、起動テンプレートに移動されます。`version` を指定しない場合は、テンプレートのデフォルトバージョンが使用されます。

```
cat >eks-nodegroup.yaml <<EOF
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: my-cluster
  region: region-code
managedNodeGroups:
- name: my-mng
  launchTemplate:
    id: lt-id
    version: "1"
EOF
```

eksctl 設定ファイル使用の詳細については、eksctl ドキュメントの「Config file schema」を参照してください。インスタンスは、オプションで大量の IP アドレスを Pods に割り当てたり、インスタンスのブロックとは異なる CIDR ブロックから Pods に IP アドレスを割り当てたり、containerd ランタイムを使用することができます。またアウトバウンドのインターネットアクセスがないクラスターにデプロイすることも可能です。設定ファイルへの追加に関する他のオプションについては、「Amazon EC2 ノードで使用可能な IP アドレスの量を増やす」、「ポッド用のカスタムネットワーク」、「Docker から containerd への移行テスト」、および「プライベートクラスターの要件」を参照してください。

起動テンプレートで AMI ID を指定しなかった場合、マネージドノードグループは、インスタンスタイプに基づいて、ノードグループの各ノードで実行できる Pods の最大数に対して 1 つの値を計算して適用します。異なるインスタンスタイプを持つノードグループを作成する場合、すべてのインスタンスタイプで計算された最小値が、ノードグループ内のすべてのインスタンスタイプで実行できる Pods の最大数として適用されます。マネージド型ノードグループは、各 Amazon EC2 インスタンスタイプの Amazon EKS 推奨最大 Pods 数 で参照されているスクリプトを使用して値を計算します。

起動テンプレートで AMI ID を指定した場合で、カスタムネットワーク を使用しているか、インスタンスに割り当てられている IP アドレスの数を増やす 場合には、ノードグループの各ノードで実行できる Pods の最大数を指定します。詳細については、「各 Amazon EC2 インスタンスタイプの Amazon EKS 推奨最大 Pods 数」を参照してください。

b. 次のコマンドでノードグループをデプロイします。

```
eksctl create nodegroup --config-file eks-nodegroup.yaml
```

AWS Management Console

AWS Management Console を使用してマネージド型ノードグループを作成するには

1. クラスターステータスが ACTIVE と表示されるまで待ちます。まだ ACTIVE ではないクラスターにはマネージド型ノードグループを作成できません。
2. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。
3. マネージド型ノードグループを作成するクラスターの名前を選択します。
4. [コンピューティング] タブを選択します。
5. [ノードグループを追加] を選択します。
6. [ノードグループの設定] ページで、必要に応じてパラメータを指定し、[次へ] を選択します。

   - [名前] – マネージド型ノードグループの一意の名前を入力します。ノードグループ名は 63 文字以下である必要があります。先頭は文字または数字でなければなりませんが、残りの文字にはハイフンおよびアンダースコアを含めることもできます。
   - [ノード IAM ロール] – ノードグループで使用するノードインスタンスロールを選択します。詳細については、「Amazon EKS ノードの IAM ロール」を参照してください。

     > ⚠ Important
     >
     > - クラスターの作成に使用したロールは使用できません。
     > - セルフマネージド型ノードグループによって現在使用されていないロールを使用することをお勧めします。それ以外の場合は、新しいセルフマネージド型ノードグループで使用します。詳細については、「マネージド型ノードグループの削除」を参照してください。

   - 起動テンプレートを使用する — (オプション) 既存の起動テンプレートを使用するかどうかを選択します。[起動テンプレート名] を選択します。次に、[起動テンプレートのバージョン] を選択します。バージョンを選択しない場合、Amazon EKS はテンプレートのデフォルトのバージョンを使用します。起動テンプレートを使用すると、ノードグループを詳細にカスタマイズできます。これにより、カスタム AMI のデプロイや、Pods への大量の IP

アドレスの割り当て、インスタンスのブロックとは異なる CIDR ブロックからの Pods への IP アドレスの割り当て、インスタンスでの containerd ランタイムの有効化、アウトバウンドのインターネットアクセスのないクラスターに対するノードのデプロイなどが可能になります。詳細については、「Amazon EC2 ノードで使用可能な IP アドレスの量を増やす」、「ポッド用のカスタムネットワーク」、「Docker から containerd への移行テスト」、および「プライベートクラスターの要件」を参照してください。

起動テンプレートは、起動テンプレートを使用したマネージドノードのカスタマイズ の要件を満たしている必要があります。独自の起動テンプレートを使用しない場合、Amazon EKS API はデフォルトの Amazon EC2 起動テンプレートをアカウントに作成し、デフォルトの起動テンプレートを使用してノードグループをデプロイします。

サービスアカウントの IAM ロール を実装する場合は、AWS サービスへのアクセス許可を必要とするすべての Pod に、必要なアクセス許可を直接割り当て、クラスター内の Pods が、現在の AWS リージョン を取得するなどの理由で IMDS にアクセスしないようにします。また、起動テンプレートでホストネットワークを使用しない Pods の、IMDS へのアクセスを無効にすることもできます。詳細については、「ワーカーノードに割り当てられたインスタンスプロファイルへのアクセスを制限する」を参照してください。

- Kubernetes ラベル - (オプション) 管理対象ノードグループ内のノードに Kubernetes ラベルを適用することを選択できます。
- Kubernetesテイント - (オプション) 管理対象ノードグループ内のノードに Kubernetes 汚染を適用することを選択できます。[効果] メニューでの利用可能なオプションは **NoSchedule**、**NoExecute**、および **PreferNoSchedule** です。詳細については、「マネージド型ノードグループでのノードテイント」を参照してください。
- [タグ] – (オプション) Amazon EKS マネージド型ノードグループにタグを付けることを選択できます。これらのタグは、Auto Scaling グループやインスタンスなど、ノードグループ内の他のリソースには伝達されません。詳細については、「Amazon EKS リソースのタグ付け」を参照してください。

7. [コンピューティング構成とスケーリングの設定] ページで、必要に応じてパラメータを指定し、[次へ] を選択します。

   - [AMI タイプ] – AMI タイプを選択します。Arm インスタンスをデプロイする場合は、デプロイする前に Amazon EKS 最適化 Arm Amazon Linux AMI の考慮事項を確認してください。

前のページで起動テンプレートを指定し、起動テンプレートで AMI を指定した場合は、値を選択できません。テンプレートの値が表示されます。テンプレートで指定された AMI は、AMI を指定する の要件を満たしている必要があります。

- [キャパシティータイプ] – キャパシティタイプを選択します。キャパシティータイプの選択の詳細については、「マネージド型ノードグループのキャパシティータイプ」を参照してください。同じノードグループ内で、異なるキャパシティータイプを混在させることはできません。両方のキャパシティータイプを使用したい場合は、キャパシティータイプとインスタンスタイプをそれぞれに持つ、別々のノードグループを作成します。
- [インスタンスタイプ] – デフォルトで 1 つまたは複数のインスタンスタイプが指定されています。デフォルトのインスタンスタイプを削除するには、インスタンスタイプの右側にある X を選択します。マネージド型ノードグループで使用するインスタンスタイプを選択します。詳細については、「Amazon EC2 インスタンスタイプを選択する」を参照してください。

  コンソールには、一般的に使用されるインスタンスタイプのセットが表示されます。表示されてないインスタンスタイプを持つマネージド型ノードグループの作成が必要な場合は、`eksctl`、AWS CLI、AWS CloudFormation、または SDK を使用して、ノードグループを作成します。前のページで起動テンプレートを指定した場合、起動テンプレートでインスタンスタイプを指定する必要があるため、値を選択できません。起動テンプレートの値が表示されます。[キャパシティータイプ] で [スポット] を選択した場合は、可用性を高めるために、複数のインスタンスタイプを指定することをお勧めします。
- [ディスクサイズ] – ノードのルートボリュームに使用するディスクサイズ (GiB 単位) を入力します。

  前のページで起動テンプレートを指定した場合は、値を起動テンプレートで指定する必要があるため、値を選択できません。
- [必要なサイズ] – マネージド型ノードグループが起動時に保持する必要があるノードの現在の数を指定します。

> **Note**
>
> Amazon EKS は、ノードグループを自動的にスケールインまたはスケールアウトしません。ただし、これを行うように Kubernetes Cluster Autoscaler を設定することはできます。

- [最小サイズ] – マネージド型ノードグループがスケールインできるノードの最小数を指定します。
- [最大サイズ] – マネージド型ノードグループがスケールアウトできるノードの最大数を指定します。
- ノードグループの更新設定 – (オプション) 並行して更新するノードの数または割合を選択できます。これらのノードは、更新中は使用できません。[使用できない最大値] で、次のいずれかのオプションを選択し、その [値] を指定します:
  - [数値] – 並行して更新できるノードグループ内のノード数を選択して指定します。
  - [パーセンテージ] – 並行して更新できるノードグループ内のノードの割合を選択して指定します。ノードグループに多数のノードがある場合に便利です。

8. [ネットワーキングを指定] ページで、必要に応じてパラメータを指定し、[次へ] を選択します。

- [サブネット]: マネージド型ノードを起動するサブネットを選択します。

> **Important**
>
> Amazon EBS ボリュームによってバックアップされ、Kubernetes Autoscaling を使用する複数のアベイラビリティーゾーンにわたってステートフルアプリケーションを実行している場合、それぞれが単一のアベイラビリティーゾーンにスコープされる複数のノードグループを設定する必要があります。また、`--balance-similar-node-groups` 機能を有効にする必要があります。

> **Important**
>
> - パブリックサブネットを選択し、クラスターでパブリック API サーバーのエンドポイントのみが有効になっている場合は、サブネットの `MapPublicIPOnLaunch` に `true` をセットして、インスタンスがクラスターに正常に参加できるようにします。サブネットが `eksctl`、または 2020 年 3 月 26 日以降に Amazon EKS が販売した AWS CloudFormation テンプレートを使用して作成された場合、この設定はすでに `true` に設定されています。サブネットが `eksctl` または AWS CloudFormation テンプレートで 2020 年 3 月 26 日より前に作成されている場合は、設定を手動で変更する必要があります。詳細については、「サブネットのパブリック IPv4 アドレッシング属性の変更」を参照してください。

- 起動テンプレートを使用しており、複数のネットワークインターフェイスを指定している場合には、たとえ MapPublicIpOnLaunch が true に設定されていても、Amazon EC2 はパブリック IPv4 アドレスの自動的な割り当てを行いません。このシナリオでノードがクラスターに参加するには、クラスターのプライベート API サーバーエンドポイントを有効にするか、NAT ゲートウェイなどの別の方法によってアウトバウンドインターネットアクセスを提供する、プライベートサブネットでノードを起動する必要があります。詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EC2 インスタンスの IP アドレス指定」を参照してください。

- ノードへの SSH アクセスの設定 (オプション)。SSH を有効にすることにより、インスタンスに接続し、問題がある場合に診断情報を収集できます。ノードグループを作成するときは、リモートアクセスを有効にすることを強くお勧めします。ノードグループの作成後にリモートアクセスを有効にすることはできません。

  起動テンプレートの使用を選択した場合、このオプションは表示されません。ノードへのリモートアクセスを有効にするには、起動テンプレートでキーペアを指定し、起動テンプレートで指定したセキュリティグループのノードに対して適切なポートが開いていることを確認します。詳細については、「カスタムセキュリティグループを使用する」を参照してください。

  > **Note**
  >
  > Windows の場合、このコマンドは SSH を有効にしません。代わりに Amazon EC2 キーペアをインスタンスに関連付け、インスタンスに RDP できるようにします。

- [SSH キーペア] (オプション) の場合は、使用する Amazon EC2 SSH キーを選択します。Linux の詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EC2 のキーペアと Linux インスタンス」を参照してください。Windows の詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EC2 のキーペアと Windows インスタンス」を参照してください。起動テンプレートを使用することを選択した場合、選択することはできません。Bottlerocket AMI を使用するノードグループに Amazon EC2 SSH キーが提供されると、管理コンテナも有効になります。詳細については、GitHub の「管理コンテナ」を参照してください。

- [次からの SSH リモートアクセスを許可する] の場合、特定のインスタンスへのアクセスを制限するには、それらのインスタンスに関連付けられているセキュリティグループを選

択します。特定のセキュリティグループを選択しないと、インターネット上のどの場所 (0.0.0.0/0) からでも SSH アクセスが許可されます。

9. [確認と作成] ページで、マネージド型ノードグループの設定を確認し、[作成] を選択します。

   ノードがクラスターに参加できない場合は、「トラブルシューティングガイド」の「ノードをクラスターに結合できません」を参照してください。

10. ノードのステータスを監視し、Ready ステータスになるまで待機します。

```
kubectl get nodes --watch
```

11. (GPU ノードのみ) GPU インスタンスタイプと Amazon EKS 最適化アクセラレーション AMI を選択した場合は、クラスター上の DaemonSet として Kubernetes 用の NVIDIA デバイスプラグインを適用する必要があります。次のコマンドを実行する前に、*vX.X.X* を必要となる NVIDIA/k8s-device-plugin バージョンに置き換えます。

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/nvidia-device-plugin.yml
```

ノードが関連付けられた Amazon EKS クラスターが実行中になったところで、Kubernetes アドオンのインストールとクラスターへのアプリケーションのデプロイを開始できます。以下のトピックは、クラスターの機能を拡張するのに役立ちます。

- クラスターを作成した IAM プリンシパルは、kubectl または AWS Management Console を使用して Kubernetes API サーバーを呼び出すことができる唯一のプリンシパルです。他の IAM プリンシパルがクラスターにアクセスできるようにする場合は、それらを追加する必要があります。詳細については、Kubernetes API へのアクセスを許可する および必要なアクセス許可を参照してください。
- 次の条件が true の場合、IMDS への Pod アクセスをブロックすることをお勧めします。
  - すべての Kubernetes サービスアカウントに IAM ロールを割り当てて、Pods が必要最小限のアクセス許可のみを持つように計画しています。
  - クラスター内の Pods が、現在の AWS リージョン の取得といったその他の理由で Amazon EC2 インスタンスメタデータサービス (IMDS) へのアクセスを必要としていません。

  詳細については、「ワーカーノードに割り当てられたインスタンスプロファイルへのアクセスを制限する」を参照してください。

- Autoscaling – ノードグループ内のノード数を自動的に調整するように、KubernetesCluster Autoscaler を設定します。
- サンプルアプリケーションをクラスターにデプロイします。
- クラスターの管理 – クラスター管理のための重要なツールを使用する方法を解説します。

マネージド型ノードグループの更新

マネージド型ノードグループの更新を開始すると、Amazon EKS はノードを自動的に更新し、マネージド型ノードの更新動作 でリストされた手順を完了します。Amazon EKS 最適化 AMI を使用している場合、Amazon EKS は最新のセキュリティパッチとオペレーティングシステムの更新を、最新の AMI リリースバージョンの一部として自動的にノードに適用します。

いくつかのシナリオでは、Amazon EKS マネージド型ノードグループのバージョンや設定を更新すると便利です。

- Amazon EKS クラスターの Kubernetes バージョンを更新し、同じ Kubernetes バージョンを使用するようにノードを更新したい場合。
- マネージド型ノードグループでは、新しい AMI リリースバージョンを使用できます。AMI バージョンの詳細については、次のセクションを参照してください。
  - Amazon EKS 最適化 Amazon Linux AMI のバージョン
  - Amazon EKS 最適化 Bottlerocket AMI
  - Amazon ECS に最適化された Windows AMI バージョン
- マネージド型ノードグループ内のインスタンスの最小数、最大数、または必要な数を調整する場合。
- マネージド型ノードグループのインスタンスで Kubernetes ラベルを追加または削除したい場合。
- マネージド型ノードグループに AWS タグを追加または削除する場合。
- カスタム AMI の更新などの設定の変更を伴う、新しいバージョンの起動テンプレートをデプロイする必要があります。
- Amazon VPC CNI アドオンのバージョン `1.9.0` 以降をデプロイし、プレフィックス委任用のアドオンを有効にし、ノードグループ内の新しい AWS Nitro System インスタンスで大幅に増加した Pods の数をサポートしたい場合。詳細については、「Amazon EC2 ノードで使用可能な IP アドレスの量を増やす」を参照してください。
- Windows ノードの IP プレフィックスの委任を有効にしており、ノードグループ内の新しい AWS Nitro System インスタンスで、大幅に増加した Pods をサポートしたいと考えています。詳細については、「Amazon EC2 ノードで使用可能な IP アドレスの量を増やす」を参照してください。

マネージド型ノードグループの Kubernetes バージョンに対して、新しい AMI リリースバージョンがある場合は、ノードグループのバージョンを更新して、新しい AMI バージョンを使用することができます。同様に、クラスターがノードグループより新しい Kubernetes バージョンを実行している場合、クラスターの Kubernetes バージョンに一致する最新の AMI リリースバージョンを使用するようにノードグループを更新できます。

スケーリングオペレーションまたは更新によってマネージド型ノードグループ内のノードが終了すると、そのノードの Pods が最初にドレーンされます。詳細については、「マネージド型ノードの更新動作」を参照してください。

ノードグループバージョンの更新

`eksctl` または AWS Management Console を使用して、ノードグループのバージョンを更新できます。更新するバージョンは、コントロールプレーンバージョンよりも新しいバージョンにすることはできません。

eksctl

`eksctl` でノードグループバージョンを更新するには

- 次のコマンドを使用して、マネージド型ノードグループを、ノードに現在デプロイされているのと同じ Kubernetes バージョンの最新 AMI リリースに更新します。*`example value`* をすべて自分の値に置き換えてください。

```
eksctl upgrade nodegroup \
  --name=node-group-name \
  --cluster=my-cluster \
  --region=region-code
```

> **Note**
>
> 起動テンプレートでデプロイされたノードグループを、新しい起動テンプレートのバージョンにアップグレードする場合は、上記のコマンドに `--launch-template-version` *`version-number`* を追加します。起動テンプレートは、起動テンプレートを使用したマネージドノードのカスタマイズ の要件を満たしている必要があります。起動テンプレートにカスタム AMI が含まれている場合、AMI は AMI を指定する の要件を満たしている必要があります。ノードグループを新しいバージョンの起動テンプレートにアップグレードすると、指定した起動テンプレートバージョンの新しい設定と一致するように、すべてのノードがリサイクルされます。

起動テンプレートなしでデプロイしたノードグループを、新しい起動テンプレートバージョンに直接アップグレードすることはできません。代わりに、起動テンプレートを使用して新しいノードグループをデプロイし、新しい起動テンプレートバージョンにノードグループを更新する必要があります。

コントロールプレーンの Kubernetes バージョンと同じバージョンにノードグループをアップグレードできます。たとえば、Kubernetes `1.29` を実行しているクラスターの場合であれば、次のコマンドを使用して、現在 Kubernetes Kubernetes `1.28` を実行しているノードをバージョン `1.29` にアップグレードできます。

```
eksctl upgrade nodegroup \
  --name=node-group-name \
  --cluster=my-cluster \
  --region=region-code \
  --kubernetes-version=1.29
```

AWS Management Console

AWS Management Console を使用して、ノードグループバージョンを更新する

1. https://console.aws.amazon.com/eks/home#/clusters で Amazon EKS コンソールを開きます。
2. 更新するノードグループを含むクラスターを選択します。
3. 少なくとも 1 つのノードグループに利用可能な更新がある場合、ページの上部に利用可能な更新について通知するボックスが表示されます。[Compute] (コンピューティング) タブを選択すると、利用可能な更新があるノードグループの [Node Groups] (ノードグループ) 表の、[AMI release version] (AMI リリースバージョン) 列に、[Update now] (今すぐ更新) が表示されます。ノードグループを更新するには、[Update now] (今すぐ更新) を選択します。

   カスタム AMI でデプロイされたノードグループの通知は表示されません。ノードがカスタム AMI でデプロイされている場合は、次の手順を実行して、新しく更新されたカスタム AMI をデプロイします。

   a. AMI の新しいバージョンを作成します。

   b. 新しい AMI ID を使用して、新しい起動テンプレートのバージョンを作成します。

   c. 起動テンプレートの新しいバージョンにノードを更新する

4. [Update node group version] (ノードグループバージョンの更新) ダイアログボックスで、次のオプションを有効または無効にします。

   - [Update node group version] (ノードグループのバージョンの更新) - カスタム AMI をデプロイした場合、あるいは Amazon EKS に最適化された AMI が現在のクラスターで最新のバージョンである場合には、このオプションは利用できません。
   - [Change launch template version] (起動テンプレートバージョンの変更) - ノードグループがカスタムの起動テンプレートを使用せずにデプロイされている場合、このオプションは利用できません。カスタム起動テンプレートを使用してデプロイされたノードグループの、起動テンプレートのバージョンのみ更新できます。ノードグループを更新する [Launch template version] (起動テンプレートバージョン) を選択します。ノードグループがカスタム AMI で設定されている場合は、選択するバージョンも AMI を指定する必要があります。新しいバージョンの起動テンプレートにアップグレードすると、指定した起動テンプレートバージョンの新しい設定と一致するように、すべてのノードがリサイクルされます。

5. [Update strategy] (更新戦略) で、次のいずれかのオプションを選択します。

   - [Rolling update] (ローリング更新) - このオプションは、クラスターの Pod の中断予算を尊重します。Pod 中断予算の問題により、Amazon EKS がこのノードグループで実行されている Pods を正常にドレーンできない場合、更新が失敗します。
   - [Force update] (強制更新) - このオプションは Pod の中断予算を尊重しません。ノードの再起動を強制的に実行することにより、Pod の中断予算の問題に関係なく更新が行われます。

6. [Update] (更新) を選択します。

ノードグループ設定の編集

マネージド型ノードグループの設定の一部を変更できます。

ノードグループ設定を編集するには

1. https://console.aws.amazon.com/eks/home#/clusters で Amazon EKS コンソールを開きます。
2. 編集するノードグループを含むクラスターを選択します。
3. [Compute] (コンピューティング) タブを選択します。
4. 編集するノードグループを選択し、次に [Edit] (編集) を選択します。
5. (オプション) [Edit Node Group] (ノードグループの編集) ページで以下を実行します。

a. ノードグループのスケーリング設定を編集します。

- [Desired size] (必要なサイズ) - マネージド型ノードグループが保持する必要があるノードの現在の数を指定します。
- [最小サイズ] – マネージド型ノードグループがスケールインできるノードの最小数を指定します。
- [最大サイズ]: マネージド型ノードグループがスケールアウトできるノードの最大数を指定します。ノードグループでサポートされるノードの最大数については、「Amazon EKS Service Quotas」を参照してください。

b. (オプション) ノードグループ内のノードに [Kubernetes ラベル] を追加または削除します。ここに示すラベルは、Amazon EKS で適用したラベルのみです。ここには表示されていない他のラベルがノードに存在する可能性があります。

c. (オプション) ノードグループ内のノードに [Kubernetes テイント] を追加または削除します。追加されたテイントは、**NoSchedule**、**NoExecute**、または **PreferNoSchedule** の影響があります。詳細については、「マネージド型ノードグループでのノードテイント」を参照してください。

d. (オプション) ノードグループリソースに[Tags] (タグ) を追加または削除します。これらのタグは、Amazon EKS ノードグループにのみ適用されます。これらは、Amazon EC2 インスタンスやサブネットなど、ノードグループの他のリソースには伝達されません。

e. (オプション) ノードグループの更新設定を編集します。[Number] (数値) または [Percentage] (パーセンテージ) のいずれかを選択します。

- [数値]: 並行して更新できるノードグループ内のノード数を選択して指定します。これらのノードは、更新中は使用できません。
- [パーセンテージ]: 並行して更新できるノードグループ内のノードの割合を選択して指定します。これらのノードは、更新中は使用できません。ノードグループに多数のノードがある場合に便利です。

f. 編集が終了したら、[変更の保存] を選択します。

マネージド型ノードの更新動作

Amazon EKS マネージド型ワーカーノードのアップグレード戦略には、次のセクションで説明する 4 つの異なるフェーズがあります。

セットアップフェーズ

セットアップフェーズには、次の手順があります。

1. ノードグループに関連付けられた Auto Scaling グループの新しい Amazon EC2 起動テンプレートバージョンを作成します。新しい起動テンプレートバージョンでは、ターゲットの AMI またはカスタム起動テンプレートバージョンを更新に使用します。
2. 最新の起動テンプレートバージョンを使用するように、Auto Scaling グループを更新します。
3. ノードグループの `updateConfig` プロパティを使用して、並列でアップグレードするノードの最大数を決定します。使用不可の最大値には、クォータとして 100 ノードがあります。デフォルト値は 1 ノードです。詳細については、Amazon EKS API リファレンス内の `updateConfig` プロパティを参照してください。

スケールアップフェーズ

マネージド型ノードグループ内のノードをアップグレードするとき、アップグレードされたノードは、アップグレードされているノードと同じアベイラビリティーゾーンで起動されます。この配置を保証するために、Amazon EC2 のアベイラビリティーゾーンの再調整を使用します。詳細については、Amazon EC2 Auto Scaling ユーザーガイドのアベイラビリティーゾーンの再調整を参照してください。この要件を満たすために、マネージド型ノードグループのアベイラビリティーゾーンごとに最大 2 つのインスタンスを起動できます。

スケールアップフェーズには、次の手順があります。

1. Auto Scaling グループの最大サイズと希望のサイズを、次のうちいずれか大きい方だけ増加させます。
   - Auto Scaling グループがデプロイされているアベイラビリティーゾーン数の最大 2 倍。
   - アップグレードができない最大値。

     たとえば、ノードグループのアベイラビリティーゾーンが 5 つで、`maxUnavailable` が 1 である場合、アップグレードプロセスで最大 10 個のノードを起動できます。しかし、`maxUnavailable` が 20 (または 10 より大きいいずれかの値) である場合、プロセスにより起動される新しいノードは 20 個です。
2. Auto Scaling グループのスケーリング後、最新の設定を使用するノードがノードグループに存在するかどうかをチェックします。この手順は、次の基準を満たす場合にのみ成功します。
   - ノードが存在するすべてのアベイラビリティーゾーンで、少なくとも 1 つの新しいノードが起動されます。

- 新しいノードはすべて、Ready 状態である必要があります。
- 新しいノードには Amazon EKS 適用ラベルが必要です。

  これは、通常のノードグループのワーカーノード上の Amazon EKS 適用ラベルです。

  - eks.amazonaws.com/nodegroup-image=*$amiName*
  - eks.amazonaws.com/nodegroup=*$nodeGroupName*

  これは、カスタム起動テンプレートまたは AMI ノードグループのワーカーノード上の Amazon EKS 適用ラベルです。

  - eks.amazonaws.com/nodegroup-image=*$amiName*
  - eks.amazonaws.com/nodegroup=*$nodeGroupName*
  - eks.amazonaws.com/sourceLaunchTemplateId=*$launchTemplateId*
  - eks.amazonaws.com/sourceLaunchTemplateVersion=*$launchTemplateVersion*

3. 新しい Pods のスケジュールを回避するために、ノードをスケジュール不可としてマークします。また、ノードを終了する前にロードバランサーからノードを削除するために、ノードに node.kubernetes.io/exclude-from-external-load-balancers=true のラベルを付けます。

このフェーズで NodeCreationFailure エラーが発生する既知の原因を次に示します。

アベイラビリティーゾーンの容量が不十分です

アベイラビリティーゾーンに、リクエストされたインスタンスタイプの容量がない可能性があります。マネージド型ノードグループを作成するときは、複数のインスタンスタイプを設定することをお勧めします。

アカウント内の EC2 インスタンス制限

Service Quotas を使用してアカウントが同時に実行できる Amazon EC2 インスタンスの数を増やす必要がある場合があります。詳細については、「Linux インスタンス向け Amazon Elastic Compute Cloud ユーザーガイド」の「EC2 の Service Quotas」を参照してください。

カスタムユーザーデータ

カスタムユーザーデータは、ブートストラッププロセスを中断することがあります。このシナリオでは、ノードで kubelet が起動しないか、ノードで想定される Amazon EKS ラベルが取得されない可能性があります。詳細については、「AMI を指定する」を参照してください。

ノードを異常な状態または準備ができていない状態にする変更

ノードのディスク負荷、メモリ負荷、および同様の条件により、ノードが Ready 状態にならない可能性があります。

アップグレードフェーズ

アップグレードフェーズには、次の手順があります。

1. ノードグループのために設定されている使用不可の最大数を上限として、アップグレードが必要なノードをランダムに選択します。
2. ノードから Pods をドレインします。Pods が 15 分以内にノードを離れず、強制フラグがない場合、PodEvictionFailure というエラーが表示され、アップグレードフェーズは失敗します。このシナリオでは、update-nodegroup-version リクエストで強制フラグを適用して、Pods を削除できます。
3. すべての Pod が削除された後にノードを遮断し、60 秒間待ちます。これは、サービスコントローラーがこのノードに新しくリクエストを送信しないようにするためと、アクティブなノードのリストからこのノードを削除するために行われます。
4. 遮断されたノードの Auto Scaling グループに終了リクエストを送信します。
5. 以前のバージョンの起動テンプレートでデプロイされたノードグループ内にノードが存在しなくなるまで、以前のアップグレード手順を繰り返します。

このフェーズで PodEvictionFailure エラーが発生する既知の原因を次に示します。

アグレッシブ PDB

アグレッシブ PDB が Pod で定義されているか、同じ Pod を指す複数の PDB が存在します。

すべてのテイントを許容するデプロイ

すべての Pod が削除されると、前のステップでノードがテイントされているため、ノードは空になると予想されます。ただし、デプロイがすべての汚染を許容する場合、ノードは空ではない可能性が高く、Pod エビクションの失敗につながります。

スケールダウンフェーズ

スケールダウンフェーズでは、Auto Scaling グループの最大サイズと希望するサイズが 1 ずつ減り、更新が開始される前の値に戻ります。

ワークフローのスケールダウンフェーズ中に Cluster Autoscaler がノードグループをスケールアップしているとアップグレードワークフローが判断した場合、ノードグループを元のサイズに戻すことなく、すぐに終了します。

マネージド型ノードグループでのノードテイント

Amazon EKS は、マネージド型ノードグループによる Kubernetes テイントの設定をサポートしています。テイントと許容範囲が連携して、Pods が不適切なノードにスケジュールされないようにします。1 つ以上のテイントをノードに適用できます。これは、テイントを容認しない Pods を、ノードが受け入れるべきではないことを示します。容認は Pods に適用され、テイントが一致するノードに Pods がスケジュールされることを許可しますが、必須ではありません。詳細については、Kubernetes ドキュメントの「テイントと容認」を参照してください。

AWS Management Console または Amazon EKS API を使用して、新規または既存のマネージド型ノードグループに Kubernetes ノードのテイントが適用されます。

- を使用してテイントが適用されたノードグループを作成するを使用してテイントが適用されたノードグループを作成するを使用してテイントが適用されたノードグループを作成する情報はAWS Management Consoleマネージド型ノードグループの作成、、、、、、、、、
- AWS CLI を使用してテイントが適用されたノードグループを作成する例は、次の通りです。

```
aws eks create-nodegroup \
 --cli-input-json '
{
  "clusterName": "my-cluster",
  "nodegroupName": "node-taints-example",
  "subnets": [
     "subnet-1234567890abcdef0",
     "subnet-abcdef01234567890",
     "subnet-021345abcdef67890"
   ],
  "nodeRole": "arn:aws:iam::111122223333:role/AmazonEKSNodeRole",
  "taints": [
     {
         "key": "dedicated",
         "value": "gpuGroup",
         "effect": "NO_SCHEDULE"
     }
   ]
}'
```

詳細な情報と使用例については、Kubernetes リファレンスドキュメントの「テイント」を参照してください。

> **Note**
>
> - テイントは、UpdateNodegroupConfig API を使用してノードグループを作成した後に更新できます。
> - テイントのキーは、文字または数字で始まる必要があります。英字、数字、ハイフン (-)、ピリオド (.)、およびアンダースコア (_) を使用できます。最大 63 文字です
> - オプションで、Taint キーは DNS サブドメインプレフィックスと単一の / で始めることができます。このキーを DNS サブドメインプレフィックスで始める場合は、最大 253 文字まで使用できます。
> - 値は省略可能で、文字または数字で始める必要があります。英字、数字、ハイフン (-)、ピリオド (.)、およびアンダースコア (_) を使用できます。最大 63 文字です
> - Kubernetes直接使用するかAWS Management Console、またはを使用する場合の染み効果は**NoSchedule**、**PreferNoSchedule**、またはでなければなりません**NoExecute**。ただし、AWS CLIまたは API を使用する場合、汚染効果は**NO_SCHEDULE**、**PREFER_NO_SCHEDULE**、またはでなければなりません**NO_EXECUTE**。
> - 1 つのノードグループには、最大 50 個のテイントを使用できます。
> - マネージドノードグループを使用して作成されたテイントをノードから手動で削除した場合、Amazon EKS はテイントをノードに戻しません。これは、管理対象ノードグループ設定でテイントが指定されている場合にも当てはまります。

`aws eks update-nodegroup-config`AWS CLI コマンドを使用して、管理対象ノードグループのテイントを追加、削除、または交換できます。

起動テンプレートを使用したマネージドノードのカスタマイズ

自分の起動テンプレートを使用することで、マネージド型ノードをデプロイでき、最高レベルのカスタマイズを実現できます。起動テンプレートを使用すると、次のような機能が可能となります。

- ノードをデプロイする時に、追加の `kubelet` 引数などのブートストラップ引数を提供します。
- ノードに割り当てられた IP アドレスとは異なる CIDR ブロックから、IP アドレスを Pods に割り当てます。
- 独自のカスタム AMI をノードにデプロイします。

- 独自のカスタム CNI をノードにデプロイします。

最初にマネージドノードグループを作成するときに独自の起動テンプレートを指定する場合にも、後で高い柔軟性を得ることができます。自分の起動テンプレートを使用してマネージド型ノードグループをデプロイする限り、同じ起動テンプレートの異なるバージョンとして繰り返し更新できます。ノードグループを別のバージョンの起動テンプレートに更新すると、指定した起動テンプレートバージョンの新しい設定と一致するように、グループ内のすべてのノードがリサイクルされます。

マネージド型ノードグループは、常に Amazon EC2 Auto Scaling グループで使用する起動テンプレートでデプロイされます。起動テンプレートを指定しない場合、Amazon EKS API はアカウントのデフォルト値を使用して起動テンプレートを自動的に作成します。ただし、自動生成された起動テンプレートを変更することは推奨しません。さらに、カスタム起動テンプレートを使用していない既存のノードグループは、直接更新できません。代わりに、カスタム起動テンプレートを使用して、新しいノードグループを作成する必要があります。

起動テンプレート設定の基本

AWS Management Console、AWS CLI、または AWS を使用して、Amazon EC2 Auto Scaling 起動テンプレートを作成できます。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Auto Scaling グループの起動テンプレートを作成する」を参照してください。起動テンプレートの一部の設定は、マネージド型ノードで使われている設定と似ています。起動テンプレートを使用してノードグループをデプロイまたは更新する場合、一部の設定はノードグループ設定または起動テンプレートのいずれかで指定する必要があります。両方の場所で設定を指定しないでください。存在してはいけない場所に設定が存在する場合、ノードグループの作成や更新などの操作は失敗します。

次の表に、起動テンプレートで禁止されている設定を示します。また、マネージド型ノードグループの設定に必要な同様の設定がある場合は、その設定も示します。リスト化されている設定は、コンソールに表示される設定です。AWS CLI と SDK では、類似するが異なる名前が場合があります。

起動テンプレート — 禁止	Amazon EKS ノードグループ設定
[ネットワークインターフェイス] ([ネットワークインターフェイスを追加]) の下の [サブネット]	[Specify networking] (ネットワーキングを指定) ページの [Node group network configuration] (ノードグループのネットワーク設定) の下の [Subnets] (サブネット)
[高度な詳細] の下の [IAM インスタンスプロファイル]	[Configure Node group] (ノードグループを設定) ページの [Node group configuration] (ノー

起動テンプレート — 禁止	Amazon EKS ノードグループ設定
	ドグループ設定) の下の [Node IAM role] (ノード IAM ロール)
[高度な詳細] の下の [シャットダウン動作] および [停止 – 休止動作]。起動テンプレートのどちらの設定でも、[起動テンプレート設定に含めない] のデフォルトを保持します。	同等のものはありません。Amazon EKS は、Auto Scaling グループではなく、インスタンスのライフサイクルを管理する必要があります。

次の表に、マネージド型ノードグループ構成で禁止される設定を示します。また、起動テンプレートで必要な同様の設定がある場合は、その設定も示します。リスト化されている設定は、コンソールに表示される設定です。AWS CLI と SDK とで名前が似ている場合があります。

Amazon EKS ノードグループ設定 — 禁止	起動テンプレート
(起動テンプレートでカスタム AMI を指定した場合のみ) [Set compute and scaling configuration] (コンピューティングとスケーリングの設定を実行) ページの [Node Group compute configuration] (ノードグループのコンピューティング設定) の下の [AMI type] (AMI タイプ) — [Specified in launch template] (起動テンプレートで指定) と、指定した AMI ID がコンソールに表示されます。 起動テンプレートで [Application and OS Images (Amazon Machine Image)] (アプリケーションと OS のイメージ (Amazon マシンイメージ)) が指定されていない場合は、ノードグループ設定で AMI を選択できます。	[Launch template contents] (起動テンプレートのコンテンツ) での [Application and OS Images (Amazon Machine Image)] (アプリケーションと OS のイメージ (Amazon マシンイメージ)) - 次のいずれかの要件がある場合は、ID を指定する必要があります。 • カスタム AMI の使用。AMI を指定する に記載されている要件を満たさない AMI を指定すると、ノードグループのデプロイが失敗します。 • Amazon EKS に最適化された AMI に含まれる `bootstrap.sh` ファイルに引数を提供するためのユーザーデータを提供する必要がある。インスタンスでは、Pods に対して非常に多くの IP アドレスを割り当てたり、インスタンスとは異なる CIDR ブロックから Pods に対して IP アドレスを割り当てたりできます。さらに、アウトバウンドのインターネットアクセスのないプライベートクラスターをデプロイすることもできます。詳細

Amazon EKS ノードグループ設定 — 禁止	起動テンプレート
	については、次のトピックを参照してください。 • Amazon EC2 ノードで使用可能な IP アドレスの量を増やす • ポッド用のカスタムネットワーク • プライベートクラスターの要件 • AMI を指定する
[Set compute and scaling configuration] (コンピューティングとスケーリングの設定を実行) ページの [Node Group compute configuration] (ノードグループのコンピューティング設定) の下の [Disk size] (ディスクサイズ) — [Specified in launch template] (起動テンプレートで指定) がコンソールに表示されます。	[ストレージ (ボリューム)] ([新しいボリュームの追加]) の下の [サイズ]。これを起動テンプレートで指定する必要があります。
[Specify Networking] (ネットワーキングを指定) ページの [Node group configuration] (ノードグループ設定) の下の [SSH key pair] (SSH キーペア) — コンソールに起動テンプレートで指定したキーまたは [Not specified in launch template] (起動テンプレートで指定されていません) が表示されます。	[キーペア (ログイン)] の下の [キーペア名]。
起動テンプレートを使用する場合は、リモートアクセスを許可するソースセキュリティグループを指定できません。	インスタンスの場合、[ネットワーク設定] の下の [セキュリティグループ]、または [ネットワークインターフェイス] ([ネットワークインターフェイスを追加])の下の [セキュリティグループ]。両方設定することはできません。詳細については、「カスタムセキュリティグループを使用する」を参照してください。

Note

- 起動テンプレートを使用してノードグループをデプロイする場合は、起動テンプレート内の [起動テンプレートのコンテンツ] で 0 または 1 の [インスタンスタイプ] を指定します。または、コンソールの [コンピューティングとスケーリングの構成を設定する] ページにある [インスタンスタイプ] で 0 から 20 までのインスタンスタイプを指定できます。または、Amazon EKS API を使用する他のツールを使用して行うこともできます。起動テンプレートでインスタンスタイプを指定し、その起動テンプレートを使用してノードグループをデプロイする場合、コンソールや、Amazon EKS API を使用する他のツールを使用してインスタンスタイプを指定することはできません。起動テンプレートやコンソール、または Amazon EKS API を使用する他のツールを使用してインスタンスタイプを指定しない場合は、`t3.medium` インスタンスタイプが使用されます。ノードグループがスポットキャパシティータイプを使用している場合は、コンソールを使用して複数のインスタンスタイプを指定することをお勧めします。詳細については、「マネージド型ノードグループのキャパシティータイプ」を参照してください。
- ノードグループにデプロイするコンテナが、インスタンスメタデータサービスバージョン 2 を使用している場合は、起動テンプレートの [メタデータレスポンスのホップ制限] を 2 に設定してください。詳細については、「Amazon EC2 ユーザーガイド」の「Instance metadata and user data」(インスタンスメタデータとユーザーデータ) を参照してください。カスタム起動テンプレートを使用せずにマネージド型ノードグループをデプロイする場合、この値はデフォルトの起動テンプレートのノードグループに対して自動的に設定されます。

Amazon EC2 インスタンスへのタグ付け

起動テンプレートの `TagSpecification` パラメータを使用して、ノードグループの Amazon EC2 インスタンスに適用するタグを指定します。`CreateNodegroup` または `UpdateNodegroupVersion` API を呼び出す IAM エンティティには、`ec2:RunInstances` および `ec2:CreateTags` へのアクセス許可が必要です。また、起動テンプレートにタグが追加される必要があります。

カスタムセキュリティグループを使用する

起動テンプレートでカスタムの Amazon EC2 セキュリティグループを指定し、ノードグループ内のインスタンスに適用できます。これは、インスタンスレベルのセキュリティグループのパラメータ内か、ネットワークインターフェイス設定のパラメータの一部として指定できます。しかし、インス

タンスレベルとネットワークインタフェイス、両方のセキュリティグループを指定して、起動テンプレートを作成することはできません。マネージドノード型グループでカスタムセキュリティグループを使用する際に適用される、次の条件を考慮してください。

- Amazon EKS では、単一のネットワークインターフェイス仕様の起動テンプレートのみ使用できます。
- デフォルトでは、Amazon EKS はクラスターセキュリティグループをノードグループ内のインスタンスに追加して、ノードとコントロールプレーンとの間の通信を容易にします。前述のいずれかのオプションを使用して、起動テンプレートでカスタムセキュリティグループを指定した場合、Amazon EKS はクラスターセキュリティグループを追加しません。したがって、セキュリティグループのインバウンドルールとアウトバウンドルールで、クラスターのエンドポイントとの通信が有効になっていることを確認する必要があります。セキュリティグループのルールが正しくない場合、ワーカーノードはクラスターに参加できません。セキュリティグループルールの詳細については、Amazon EKS セキュリティグループの要件および考慮事項を参照してください。
- ノードグループ内のインスタンスへの SSH アクセスが必要な場合は、そのアクセスを許可するセキュリティグループを含めてください。

Amazon EC2 ユーザーデータ

起動テンプレートには、カスタムユーザーデータのセクションが含まれています。このセクションでは、個々のカスタム AMI を手動で作成しなくても、ノードグループの構成設定を指定できます。Bottlerocket で使用できる設定の詳細については、「GitHub」の「Using user data」(ユーザーデータの使用) を参照してください。

`cloud-init` を使用すると、インスタンス起動時に起動テンプレート内の Amazon EC2 ユーザーデータを提供できます。詳細については、「cloud-init ドキュメント」を参照してください。ユーザーデータを使用すると、一般的な設定操作を実行できます。これには、次の操作が含まれます。

- ユーザーまたはグループを含む
- パッケージのインストール

マネージド型ノードグループで使用される起動テンプレートの Amazon EC2 ユーザーデータは、Amazon Linux AMI の場合 MIME マルチパートアーカイブ、Bottlerocket AMI の場合 TOML の形式である必要があります。これは、ユーザーデータが、ノードがクラスターに参加するために必要な Amazon EKS ユーザーデータにマージされるためです。`kubelet` を起動または変更するコマンドをユーザーデータに指定しないでください。これは Amazon EKS によってマージされたユーザーデー

タの一部として実行されます。ノードへのラベル設定などの一部の kubelet パラメータは、マネージド型ノードグループ API 経由で直接設定できます。

> **Note**
>
> 手動での起動や、カスタムの設定パラメータの受け渡しなど、高度な kubelet のカスタマイズについては、AMI を指定するを参照してください。起動テンプレートでカスタム AMI ID が指定されている場合、Amazon EKS はユーザーデータをマージしません。

次の詳細は、ユーザーデータセクションについて説明しています。

Amazon Linux 2 user data

複数のユーザーデータブロックと単一の MIME マルチパートファイルを組み合わせることができます。例えば、カスタムパッケージをインストールするユーザーデータのシェルスクリプトを、Docker デーモンを設定するクラウドブートフックに組み合わせることができます。MIME マルチパートファイルには次のコンポーネントが含まれます。

- コンテンツタイプとパートバウンダリの宣言: `Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="`
- MIME バージョンの宣言: `MIME-Version: 1.0`
- 次のコンポーネントを含む 1 つ以上のユーザーデータブロック:
  - ユーザーデータブロックの始まりを示す開始境界:`--==MYBOUNDARY==`
  - ブロックのコンテンツの種類の宣言: `Content-Type: text/cloud-config; charset="us-ascii"`。コンテンツタイプの詳細については、cloud-init のドキュメントを参照してください。
  - ユーザーデータのコンテンツ (例えば、シェルコマンドや cloud-init ディレクティブのリスト)。
  - MIME マルチパートファイルの終わりを示す、終了境界: `--==MYBOUNDARY==--`

自分で MIME マルチパートファイルを作成するときに使用できる例は、次の通りです。

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"
```

```
#!/bin/bash
echo "Running custom user data script"

--==MYBOUNDARY==--
```

Amazon Linux 2023 user data

Amazon Linux 2023 (AL2023) では、YAML 設定スキーマを使用する新しいノード初期化プロセス nodeadm が導入されています。セルフマネージド型ノードグループまたは起動テンプレートを持つ AMI を使用している場合は、新しいノードグループの作成時に追加のクラスターメタデータを明示的に指定する必要があります。最低限必要なパラメータの例を以下に示します。ここで、apiServerEndpoint、certificateAuthority、サービスの cidr が必要になります。

```
---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: my-cluster
    apiServerEndpoint: https://example.com
    certificateAuthority: Y2VydGlmaWNhdGVBdXRob3JpdHk=
    cidr: 10.100.0.0/16
```

通常、この設定はユーザーデータにそのまま設定するか、MIME マルチパートドキュメントに埋め込まれます。

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="BOUNDARY"

--BOUNDARY
Content-Type: application/node.eks.aws

---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig spec: [...]

--BOUNDARY--
```

AL2 では、これらのパラメータからのメタデータは Amazon EKS `DescribeCluster` API コールから検出されていました。AL2023 では、ノードの大規模なスケールアップ中に API コールによってスロットリングが発生するリスクがあるため、この動作が変更されました。この変更は、起動テンプレートのないマネージド型ノードグループを使用している場合や、Karpenter を使用している場合には影響しません。詳細については、「Amazon EKS API リファレンス」の `certificateAuthority`、サービスの `cidr`、`DescribeCluster` を参照してください。

Bottlerocket user data

Bottlerocket では、ユーザーデータは TOML 形式で構造化されています。Amazon EKS が提供するユーザーデータとマージするユーザーデータを提供できます。たとえば、追加の `kubelet` 設定を指定できます。

```
[settings.kubernetes.system-reserved]
cpu = "10m"
memory = "100Mi"
ephemeral-storage= "1Gi"
```

サポートされる設定について詳しくは、「Bottlerocket ドキュメント」を参照してください。ユーザーデータにノードラベルとテイントを設定できます。ただし、代わりにノードグループ内にこれらを設定することをお勧めします。この場合、Amazon EKS によりこれらの設定が適用されます。

ユーザーデータをマージしても、書式設定は保持されませんが、コンテンツは同じままです。ユーザーデータに指定した設定は、Amazon EKS によって構成された設定よりも優先されます。したがって、`settings.kubernetes.max-pods` または `settings.kubernetes.cluster-dns-ip` を設定した場合、ユーザーデータのこれらの値がノードに適用されます。

Amazon EKS で、有効な TOML がすべてサポートされるわけではありません。以下は、サポートされていない既知の形式の一覧です。

- 引用符で囲まれたキー内の引用符: `'quoted "value"' = "value"`
- 値内のエスケープされた引用符: `str = "I'm a string. \"You can quote me\""`
- 浮動小数点と整数の混在: `numbers = [ 0.1, 0.2, 0.5, 1, 2, 5 ]`
- 配列内の混合型: `contributors = ["foo@example.com", { name = "Baz", email = "baz@example.com" }]`
- 引用符付きキーを含む括弧で囲まれたヘッダー: `[foo."bar.baz"]`

Windows user data

Windows ユーザーデータは PowerShell コマンドを使用します。マネージド型ノードグループを作成すると、カスタムユーザーデータが Amazon EKS マネージドユーザーデータと結合されます。PowerShell コマンドが最初に表示され、その後にマネージドユーザーデータコマンドが続きます。これらはすべて 1 つの **<powershell></powershell>** タグ内にあります。

> **Note**
>
> 起動テンプレートに AMI ID が指定されていない場合は、ユーザーデータで Windows Amazon EKS ブートストラップスクリプトを使用して Amazon EKS を設定しないでください。

ユーザーデータの例は次のとおりです。

```
<powershell>
Write-Host "Running custom user data script"
</powershell>
```

AMI を指定する

以下のいずれかの要件がある場合は、起動テンプレートの `ImageId` フィールドで AMI ID を指定します。追加情報については、要件を選択してください。

Amazon EKS に最適化された Linux/Bottlerocket AMI に含まれる **bootstrap.sh** ファイルに引数を渡すためのユーザーデータを提供する

ブートストラップとは、インスタンスの起動時に実行できるコマンドの追加を表す用語です。例えば、ブートストラップでは、追加の kubelet 引数を使用できます。起動テンプレートを指定せずに、`eksctl` を使用して引数を `bootstrap.sh` スクリプトに渡すことができます。または、起動テンプレートのユーザーデータセクションに情報を指定することでこれを行うことができます。

eksctl without specifying a launch template

次の内容で、*my-nodegroup.yaml* という名前のファイルを作成します。*example value* をすべて自分の値に置き換えてください。`--apiserver-endpoint`、`--b64-cluster-ca`、および `--dns-cluster-ip` 引数はオプションです。しかし、これらを定義すると、`bootstrap.sh` スクリプトによる `describeCluster` 呼び出しを避けることができま

す。これは、プライベートクラスターのセットアップや、ノードを頻繁にスケールインおよびスケールアウトするクラスターで役立ちます。bootstrap.sh スクリプトの詳細については、「GitHub」で「bootstrap.sh」ファイルを参照してください。

- 必須となる引数は、クラスター名 (*my-cluster*) のみです。
- ami-*1234567890abcdef0* に最適化された AMI ID を取得するには、次のセクションの表を使用できます。
  - Amazon EKS 最適化 Amazon Linux AMI ID の取得
  - Amazon EKS 最適化 Bottlerocket AMI ID の取得
  - Amazon EKS 最適化 Windows AMI ID の取得
- クラスターの *certificate-authority* を取得するには、次のコマンドを実行します。

```
aws eks describe-cluster --query "cluster.certificateAuthority.data" --output text
 --name my-cluster --region region-code
```

- クラスターの *api-server-endpoint* を取得するには、次のコマンドを実行します。

```
aws eks describe-cluster --query "cluster.endpoint" --output text --name my-
cluster --region region-code
```

- 最終的に、--dns-cluster-ip の値はサービス CIDR を示す .10 となります。クラスターの *service-cidr* を取得するには、次のコマンドを実行します。例えば、戻り値が ipv4 10.100.0.0/16 であれば、自分の値は *10.100.0.10* です。

```
aws eks describe-cluster --query "cluster.kubernetesNetworkConfig.serviceIpv4Cidr"
 --output text --name my-cluster --region region-code
```

- この例では、カスタムの max-pods 値を設定するために kubelet 引数を指定します。その際、Amazon EKS 最適化 AMI に含まれている bootstrap.sh スクリプトを使用します。ノードグループ名は 63 文字以下である必要があります。先頭は文字または数字でなければなりませんが、残りの文字にはハイフンおよびアンダースコアを含めることもできます。*my-max-pods-value* の選択についての詳細は、「各 Amazon EC2 インスタンスタイプの Amazon EKS 推奨最大 Pods 数」を参照してください。

```
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
```

```
metadata:
  name: my-cluster
  region: region-code

managedNodeGroups:
  - name: my-nodegroup
    ami: ami-1234567890abcdef0
    instanceType: m5.large
    privateNetworking: true
    disableIMDSv1: true
    labels: { x86-al2-specified-mng }
    overrideBootstrapCommand: |
      #!/bin/bash
      /etc/eks/bootstrap.sh my-cluster \
        --b64-cluster-ca certificate-authority \
        --apiserver-endpoint api-server-endpoint \
        --dns-cluster-ip service-cidr.10 \
        --kubelet-extra-args '--max-pods=my-max-pods-value' \
        --use-max-pods false
```

利用可能な eksctl config ファイルオプションについては、「eksctl ドキュメント」の「Config file schema」(Config ファイルスキーマ) を参照してください。eksctl ユーティリティは引き続き起動テンプレートを作成し、config ファイルに提供したデータを使用してユーザーデータを設定します。

次のコマンドを使用して、ノードグループを作成します。

```
eksctl create nodegroup --config-file=my-nodegroup.yaml
```

User data in a launch template

起動テンプレートのユーザーデータセクションで次の情報を指定します。*example value* をすべて自分の値に置き換えてください。--apiserver-endpoint、--b64-cluster-ca、および --dns-cluster-ip 引数はオプションです。しかし、これらを定義すると、bootstrap.sh スクリプトによる describeCluster 呼び出しを避けることができます。これは、プライベートクラスターのセットアップや、ノードを頻繁にスケールインおよびスケールアウトするクラスターで役立ちます。bootstrap.sh スクリプトの詳細については、「GitHub」で「bootstrap.sh」ファイルを参照してください。

- 必須となる引数は、クラスター名 (*my-cluster*) のみです。
- クラスターの *certificate-authority* を取得するには、次のコマンドを実行します。

```
aws eks describe-cluster --query "cluster.certificateAuthority.data" --output text
 --name my-cluster --region region-code
```

- クラスターの *api-server-endpoint* を取得するには、次のコマンドを実行します。

```
aws eks describe-cluster --query "cluster.endpoint" --output text --name my-
cluster --region region-code
```

- 最終的に、--dns-cluster-ip の値はサービス CIDR を示す .10 となります。クラスターの *service-cidr* を取得するには、次のコマンドを実行します。例えば、戻り値が ipv4 10.100.0.0/16 であれば、自分の値は *10.100.0.10* です。

```
aws eks describe-cluster --query "cluster.kubernetesNetworkConfig.serviceIpv4Cidr"
 --output text --name my-cluster --region region-code
```

- この例では、カスタムの max-pods 値を設定するために kubelet 引数を指定します。その際、Amazon EKS 最適化 AMI に含まれている bootstrap.sh スクリプトを使用します。*my-max-pods-value* の選択についての詳細は、「各 Amazon EC2 インスタンスタイプの Amazon EKS 推奨最大 Pods 数」を参照してください。

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
set -ex
/etc/eks/bootstrap.sh my-cluster \
  --b64-cluster-ca certificate-authority \
  --apiserver-endpoint api-server-endpoint \
  --dns-cluster-ip service-cidr.10 \
  --kubelet-extra-args '--max-pods=my-max-pods-value' \
  --use-max-pods false

--==MYBOUNDARY==--
```

Amazon EKS に最適化された Windows AMI に含まれる **Start-EKSBootstrap.ps1** ファイルに引数を渡すためのユーザーデータを提供する

ブートストラップとは、インスタンスの起動時に実行できるコマンドの追加を表す用語です。起動テンプレートを指定せずに、eksctl を使用して引数を Start-EKSBootstrap.ps1 スクリプトに渡すことができます。または、起動テンプレートのユーザーデータセクションに情報を指定することでこれを行うことができます。

カスタム Windows AMI ID を指定する場合、次の考慮事項に留意してください。

- 起動テンプレートを使用し、必要なブートストラップコマンドをユーザーデータセクションに入力する必要があります。希望の Windows IDを取得するには、Amazon EKS 最適化 Windows AMI の表を使用できます。
- いくつかの制限と条件があります。例えば、eks:kube-proxy-windows を AWS IAM Authenticator 設定マップに追加する必要があります。詳細については、「AMI ID を指定する場合の制限と条件」を参照してください。

起動テンプレートのユーザーデータセクションで次の情報を指定します。*example value* をすべて自分の値に置き換えてください。-APIServerEndpoint、-Base64ClusterCA、および -DNSClusterIP 引数はオプションです。しかし、これらを定義すると、Start-EKSBootstrap.ps1 スクリプトによる describeCluster 呼び出しを避けることができます。

- 必須となる引数は、クラスター名 (*my-cluster*) のみです。
- クラスターの *certificate-authority* を取得するには、次のコマンドを実行します。

```
aws eks describe-cluster --query "cluster.certificateAuthority.data" --output text --
name my-cluster --region region-code
```

- クラスターの *api-server-endpoint* を取得するには、次のコマンドを実行します。

```
aws eks describe-cluster --query "cluster.endpoint" --output text --name my-cluster
 --region region-code
```

- 最終的に、--dns-cluster-ip の値はサービス CIDR を示す .10 となります。クラスターの *service-cidr* を取得するには、次のコマンドを実行します。例えば、戻り値が ipv4 10.100.0.0/16 であれば、自分の値は *10.100.0.10* です。

```
aws eks describe-cluster --query "cluster.kubernetesNetworkConfig.serviceIpv4Cidr" --
output text --name my-cluster --region region-code
```

- その他の引数については、「ブートストラップスクリプトの設定パラメータ」を参照してください。

> **Note**
>
> カスタムサービス CIDR を使用している場合は、-ServiceCIDR パラメータを使用して指定する必要があります。そうしなければ、クラスター内の Pods の DNS 解決が失敗します。

```
<powershell>
[string]$EKSBootstrapScriptFile = "$env:ProgramFiles\Amazon\EKS\Start-EKSBootstrap.ps1"
& $EKSBootstrapScriptFile -EKSClusterName my-cluster `
  -Base64ClusterCA certificate-authority `
  -APIServerEndpoint api-server-endpoint `
  -DNSClusterIP service-cidr.10
</powershell>
```

特定のセキュリティ、コンプライアンス、または社内的なポリシー要件のために、カスタム AMI を実行する

詳細については、「Amazon EC2 ユーザーガイド」の「Amazon マシンイメージ (AMI)」を参照してください。Amazon EKS AMI ビルド仕様には、Amazon Linux をベースにしたカスタム Amazon EKS AMI を構築するためのリソースと設定スクリプトが含まれています。詳細については、「GitHub」の「Amazon EKS AMI ビルド仕様」を参照してください。他のオペレーティングシステムがインストールされたカスタム AMI を作成するには、「GitHub」の「Amazon EKS Sample Custom AMIs」を参照してください。

> **Important**
>
> AMI を指定する際、Amazon EKS ではユーザーデータをマージしません。代わりに、ノードのクラスターへの参加に必要な bootstrap コマンドを、ユーザーが提供する必要があります。ノードがクラスターに参加できない場合、Amazon EKS CreateNodegroup および UpdateNodegroupVersion アクションも失敗します。

AMI ID を指定する場合の制限と条件

以下に、マネージド型ノードグループで AMI ID を指定する際の制限と条件を示します。

- 起動テンプレートで AMI ID を指定する場合と、AMI ID を指定しない場合は、新しいノードグループを作成して切り替える必要があります。
- 新しい AMI バージョンが利用可能になっても、コンソールでは通知を表示しません。ノードグループを新しい AMI バージョンに更新するには、更新された AMI ID で新しいバージョンの起動テンプレートを作成する必要があります。次に、新しい起動テンプレートバージョンでノードグループを更新する必要があります。
- AMI ID を指定した場合は、API の次のフィールドを設定することはできません。
  - `amiType`
  - `releaseVersion`
  - `version`
- AMI ID を指定する場合、API で設定されたすべての `taints` は非同期に適用されます。ノードがクラスターに参加する前にテイントを適用するには、`--register-with-taints` コマンドラインフラグを使用してユーザーデータ内の `kubelet` にテイントを渡す必要があります。詳細については、Kubernetes ドキュメントの「kubelet」を参照してください。
- Windows マネージド型ノードグループのカスタム AMI ID を指定するとき、AWS IAM Authenticator 設定マップに `eks:kube-proxy-windows` を追加します。これは、DNS が正しく機能するために必要です。

  1. AWS IAM Authenticator 設定マップを編集用に開きます。

     ```
     kubectl edit -n kube-system cm aws-auth
     ```

  2. このエントリを、Windows ノードに関連付けられた各 `rolearn` の下の `groups` リストに追加します。設定マップは aws-auth-cm-windows.yaml のようになるはずです。

     ```
     - eks:kube-proxy-windows
     ```

  3. ファイルを保存し、テキストエディタを終了します。

マネージド型ノードグループの削除

このトピックでは、Amazon EKS マネージド型ノードグループを削除する方法について説明します。マネージド型ノードグループを削除すると、まず Amazon EKS が Auto Scaling グループの最小

サイズ、最大サイズ、および必要なサイズをゼロに設定します。これにより、ノードグループがスケールダウンされます。

各インスタンスが終了する前に、Amazon EKS はそのノードから Pods を排出するシグナルを送信します。数分経っても Pods が排出されない場合、Amazon EKS はオートスケーリングにインスタンスの終了を続行させます。すべてのインスタンスが終了すると、Auto Scaling グループは削除されます。

> **Important**
>
> クラスター内の、他のマネージド型ノードグループで使用されていないノード IAM ロールを使用している、マネージド型ノードグループを削除すると、そのロールは `aws-auth ConfigMap` から削除されます。クラスター内のいずれかのセルフマネージド型ノードグループが同じノード IAM ロールを使用している場合、セルフマネージド型ノードは `NotReady` ステータスに移行します。さらに、クラスター操作も中断されます。クラスターのプラットフォームバージョンが アクセスエントリを管理する の前提条件セクションに記載されている最低バージョン以上である場合、セルフマネージドノードグループにのみ使用しているロールのマッピングを追加するには、「アクセスエントリの作成」を参照してください。プラットフォームバージョンがアクセスエントリに必要な最小バージョンより前の場合は、エントリを `aws-auth ConfigMap` に追加し直すことができます。詳細については、ターミナルに `eksctl create iamidentitymapping --help` を入力してください。

マネージド型ノードグループを削除するには、`eksctl` または AWS Management Console を使用します。

eksctl

マネージド型ノードグループを **`eksctl`** を使用して削除するには

次のコマンドを入力します。*`example value`* をすべて自分の値に置き換えてください。

```
eksctl delete nodegroup \
  --cluster my-cluster \
  --name my-mng \
  --region region-code
```

その他のオプションについては、`eksctl` ドキュメントの「ノードグループの削除と排出」を参照してください。

AWS Management Console

AWS Management Console を使用してマネージド型ノードグループを削除するには

1. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。
2. [クラスター] ページで、削除するノードグループを含むクラスターを選択します。
3. 選択したクラスターページで、[コンピューティング] タブを選択します。
4. [Node groups] (ノードグループ) セクションで、削除するノードグループを選択します。その後、[削除] をクリックします。
5. [ノードグループの削除] 確認ダイアログボックスで、ノードグループの名前を入力します。その後、[削除] をクリックします。

AWS CLI

AWS CLI を使用してマネージド型ノードグループを削除するには

1. 次のコマンドを入力します。*example value* をすべて自分の値に置き換えてください。

```
aws eks delete-nodegroup \
  --cluster-name my-cluster \
  --nodegroup-name my-mng \
  --region region-code
```

2. キーボードの矢印キーを使用して、応答出力をスクロールします。終了したら **q** キーを押します。

その他のオプションについては、「AWS CLI コマンドリファレンス」の delete-nodegroup コマンドを参照してください。

セルフマネージド型ノード

クラスターには、Pods がスケジュールされている Amazon EC2 ノードが 1 つ以上含まれています。Amazon EKS ノードは AWS アカウントで実行され、クラスター API サーバーエンドポイントを介してクラスターのコントロールプレーンに接続します。Amazon EC2 料金に基づいて請求されます。詳細については、「Amazon EC2 料金」を参照してください。

クラスターには、複数のノードグループを含めることができます。各ノードグループには、Amazon EC2 Auto Scaling グループにデプロイされる 1 つ以上のノードが含まれます。グループ内のノードのインスタンスタイプは、Karpenter による 属性ベースのインスタンスタイプの選択を使用するときなどに、異なる場合があります。ノードグループ内のすべてのインスタンスは、Amazon EKS ノードの IAM ロールを使用する必要があります。

Amazon EKS は、Amazon EKS 最適化 AMI と呼ばれる特殊な Amazon マシンイメージ (AMI) を提供します。AMI は Amazon EKS と連携するように設定されています。そのコンポーネントには `containerd`、`kubelet`、および AWS IAM Authenticator が含まれます。また、AMI にはクラスターのコントロールプレーンを自動的に検出して接続を許可する、特別なブートストラップスクリプトも含まれます。

CIDR ブロックを使用してクラスターのパブリックエンドポイントへのアクセスを制限する場合は、プライベートエンドポイントアクセスも有効にすることをお勧めします。これは、ノードがクラスターと通信できるようにするためです。プライベートエンドポイントが有効になっていない場合、パブリックアクセスに指定する CIDR ブロックに、VPC からの出力ソースを含める必要があります。詳細については、「Amazon EKS クラスターエンドポイントアクセスコントロール」を参照してください。

Amazon EKS クラスターにセルフマネージド型ノードを追加するには、次のトピックを参照してください。セルフマネージド型ノードを手動で起動する場合は、各ノードに次のタグを追加します。詳細については、「個々のリソースでのタグの追加と削除」を参照してください。このガイドの手順に従うと、必要なタグがノードに自動的に追加されます。

キー	Value
`kubernetes.io/cluster/` *`my-cluster`*	`owned`

一般的な Kubernetes の観点からのノードの詳細については、Kubernetes ドキュメントで「ノード」について参照してください。

トピック

- セルフマネージド型の Amazon Linux ノードの起動
- セルフマネージド型 Bottlerocket ノードの起動
- セルフマネージド型 Windows ノードの起動
- セルフマネージド型 Ubuntu ノードの起動
- セルフマネージド型ノードの更新

セルフマネージド型の Amazon Linux ノードの起動

このトピックでは、Amazon EKS クラスターに登録する Linux ノードの Auto Scaling グループを起動する方法について説明します。ノードがクラスターに参加したら、それらのノードに Kubernetes アプリケーションをデプロイ可能になります。eksctl または AWS Management Consoleを使用して、セルフマネージド型の Amazon Linux ノードを起動することもできます。AWS Outposts でノードを起動する必要がある場合は、「Outpost 上のセルフマネージド型の Amazon Linux ノードの起動」を参照してください。

前提条件

- 既存の Amazon EKS クラスター。デプロイするには、「Amazon EKS クラスターの作成」を参照してください。AWS Outposts、AWS Wavelength、または AWS Local Zones が有効になっている AWS リージョン にサブネットがある場合、それらのサブネットはクラスターの作成時に渡さないようにします。
- ノードが使用する既存の IAM ロール。作成する場合は「Amazon EKS ノードの IAM ロール」を参照してください。このロールに VPC CNI のポリシーがどちらも含まれていない場合、VPC CNI ポッドには以下の別のロールが必要です。
- (オプションですが推奨) 必要な IAM ポリシーがアタッチされた独自の IAM ロールで設定された Amazon VPC CNI plugin for Kubernetes アドオン。詳細については、「サービスアカウントの IAM ロールを使用する Amazon VPC CNI plugin for Kubernetes の設定」を参照してください。
- 「Amazon EC2 インスタンスタイプを選択する」に記載されている考慮事項に関する知識。選択したインスタンスタイプによっては、クラスターと VPC に関する追加の前提条件がある場合もあります。

eksctl

> **Note**
>
> 現時点では、eksctl は Amazon Linux 2023 をサポートしていません。

前提条件

デバイスまたは AWS CloudShell にインストールされている eksctl コマンドラインツールのバージョン 0.183.0 以降。eksctl をインストールまたはアップグレードするには、eksctl ドキュメントの「インストール」を参照してください。

eksctl を使用してセルフマネージド型の Linux ノードを起動する

1. (オプション) [AmazonEKS_CNI_Policy] が管理する IAM ポリシーが Amazon EKS ノードの IAM ロール にアタッチされている場合は、代わりに Kubernetes aws-node サービスアカウントに関連付けた IAM ロールに割り当てることをお勧めします。詳細については、「サービスアカウントの IAM ロールを使用する Amazon VPC CNI plugin for Kubernetes の設定」を参照してください。
2. 次のコマンドは、既存のクラスターにノードグループを作成します。*al-nodes* をノードグループの名前に置き換えます。ノードグループ名は 63 文字以下である必要があります。先頭は文字または数字でなければなりませんが、残りの文字にはハイフンおよびアンダースコアを含めることもできます。*my-cluster* を自分のクラスター名に置き換えます。この名前には、英数字 (大文字と小文字が区別されます) とハイフンのみを使用できます。先頭の文字は英数字である必要があります。また、100 文字より長くすることはできません。名前は、クラスターを作成する AWS リージョン および AWS アカウント 内で一意である必要があります。残りの *example value* を独自の値に置き換えてください。デフォルトでは、ノードはコントロールプレーンと同じ Kubernetes バージョンで作成されます。

   --node-type の値を選択する前に、「Amazon EC2 インスタンスタイプを選択する」を確認してください。

   my-key を Amazon EC2 キーペアまたはパブリックキーの名前に置き換えます。このキーは、起動後のノードに SSH 接続するために使用されます。Amazon EC2 キーペアをまだ持っていない場合は、AWS Management Console で作成できます。詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EC2 キーペア」を参照してください。

   次のコマンドを使用して、ノードグループを作成します。

   > **Important**
   >
   > ノードグループを AWS Outposts、Wavelength、または Local Zones サブネットにデプロイする場合、追加の考慮事項があります。
   >
   > - クラスターを作成したときに、サブネットが渡されていない必要があります。
   > - ノードグループは、サブネットと volumeType: gp2 を指定する設定ファイルを使用して作成する必要があります。詳細については、「eksctl ドキュメント」の「設定ファイルからノードグループを作成する」と「設定ファイルのスキーマ」を参照してください。

```
eksctl create nodegroup \
  --cluster my-cluster \
  --name al-nodes \
  --node-type t3.medium \
  --nodes 3 \
  --nodes-min 1 \
  --nodes-max 4 \
  --ssh-access \
  --managed=false \
  --ssh-public-key my-key
```

次のノードグループをデプロイするには

- デフォルト設定よりもはるかに多くの IP アドレスを Pods に割り当てることができるノードグループについては、「Amazon EC2 ノードで使用可能な IP アドレスの量を増やす」を参照してください。
- インスタンスのブロックとは異なる CIDR ブロックから IPv4 アドレスを Pods に割り当てることができるノードグループについては、「ポッド用のカスタムネットワーク」を参照してください。
- Pods とサービスに IPv6 アドレスを割り当てることができるノードグループについては、「クラスター、Pods、services 用の IPv6 アドレス」を参照してください。
- containerd ランタイムを使用するノードグループについては、config ファイルを使用してノードグループをデプロイする必要があります。詳細については、「Docker から containerd への移行テスト」を参照してください。
- アウトバウンドインターネットアクセスがないノードグループについては、「プライベートクラスターの要件」を参照してください。

すべての使用可能なオプションとデフォルト値の一覧を表示するには、次のコマンドを入力します。

```
eksctl create nodegroup --help
```

ノードがクラスターに参加できない場合は、トラブルシューティングガイドの「ノードをクラスターに結合できません」を参照してください。

出力例は次のとおりです。ノードの作成中に、複数の行が出力されます。出力の最後の行は、次のサンプル行が表示されます。

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. (オプション) サンプルアプリケーション をデプロイして、クラスターと Linux ノードをテストします。
4. 次の条件が true の場合、IMDS への Pod アクセスをブロックすることをお勧めします。
   - すべての Kubernetes サービスアカウントに IAM ロールを割り当てて、Pods が必要最小限のアクセス許可のみを持つように計画しています。
   - クラスター内の Pods が、現在の AWS リージョン の取得といったその他の理由で Amazon EC2 インスタンスメタデータサービス (IMDS) へのアクセスを必要としていません。

   詳細については、「ワーカーノードに割り当てられたインスタンスプロファイルへのアクセスを制限する」を参照してください。

AWS Management Console

ステップ 1: AWS Management Console を使用してセルフマネージド型の Linux ノードを起動する

1. AWS CloudFormation テンプレートの最新バージョンをダウンロードする

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2022-12-23/
amazon-eks-nodegroup.yaml
```

2. クラスターステータスが ACTIVE と表示されるまで待ちます。クラスターがアクティブになる前にノードを起動した場合、ノードはクラスターへの登録に失敗し、再起動する必要があります。
3. https://console.aws.amazon.com/cloudformation で AWS CloudFormation コンソールを開きます。
4. [スタックの作成] を選択し、[新しいリソースを使用 (標準)] を選択します。
5. [テンプレートの指定] で、[テンプレートファイルのアップロード] を選択し、[ファイルを選択] を選択します。

6. ダウンロードした amazon-eks-nodegroup.yaml ファイルを選択します。
7. [次へ] を選択します。
8. [スタックの詳細の指定] ページで、必要に応じて次のパラメータを入力し、[次へ] を選択します。

   - [スタック名]: AWS CloudFormation スタックのスタック名を選択します。例えば、***my-cluster-nodes*** という名前にすることができます。この名前には、英数字 (大文字と小文字が区別されます) とハイフンのみを使用できます。先頭の文字は英数字である必要があります。また、100 文字より長くすることはできません。名前は、クラスターを作成する AWS リージョン および AWS アカウント 内で一意である必要があります。
   - [ClusterName]: Amazon EKS クラスターの作成時に使用した名前を入力します。この名前は、クラスター名と同じにする必要があります。同じでない場合、ノードはクラスターに参加できません。
   - [ClusterControlPlaneSecurityGroup]: VPC の作成時に生成した AWS CloudFormation 出力の [SecurityGroups] 値を選択します。

     次のステップでは、該当するグループを取得する 1 つのオペレーションを説明します。

     1. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。
     2. クラスターの名前を選択します。
     3. [ネットワーキング] タブを選択します。
     4. [ClusterControlPlaneSecurityGroup] ドロップダウンリストから選択する場合は、[追加のセキュリティグループ] の値をリファレンスとして使用します。
   - [NodeGroupName]: ノードグループの名前を入力します。この名前は、ノードに対して作成される Auto Scaling ノードグループを識別するために後で使用できます。ノードグループ名は 63 文字以下である必要があります。先頭は文字または数字でなければなりませんが、残りの文字にはハイフンおよびアンダースコアを含めることもできます。
   - [NodeAutoScalingGroupMinSize]: ノードの Auto Scaling グループがスケールインできる最小ノード数を入力します。
   - NodeAutoScalingGroupDesiredCapacity: スタック作成時にスケーリングする必要のあるノード数を入力します。
   - [NodeAutoScalingGroupMaxSize]: ノードの Auto Scaling グループがスケールアウトできる最大ノード数を入力します。
   - [NodeInstanceType]: ノードのインスタンスタイプを選択します。詳細については、「Amazon EC2 インスタンスタイプを選択する」を参照してください。

- [NodeImageIdSSMParam]: 最新の Amazon EKS 最適化 AMI の Amazon EC2 Systems Manager のパラメータが、可変 Kubernetes バージョン用に事前設定されています。Amazon EKS でサポートされている別の Kubernetes マイナーバージョンを使用するには、*1.XX* を別のサポートされているバージョンに置き換えます。クラスターと同じ Kubernetes バージョンを指定することをお勧めします。

  amazon-linux-2 を別の AMI タイプに置き換えることもできます。詳細については、「Amazon EKS 最適化 Amazon Linux AMI ID の取得」を参照してください。

  > **Note**
  >
  > Amazon EKS ノード AMI は Amazon Linux をベースとしています。Amazon Linux セキュリティセンターで Amazon Linux 2 のセキュリティもしくはプライバシーに関するイベントを追跡したり、関連する RSS フィードをサブスクライブできます。セキュリティおよびプライバシーイベントには、問題の概要、影響を受けるパッケージ、および問題を修正するためにインスタンスを更新する方法などがあります。

- [NodeImageId]: (オプション) (Amazon EKS 最適化 AMI の代わりに) 独自のカスタム AMI を使用している場合は、AWS リージョン のノード AMI ID を入力します。ここで値を指定すると、[NodeImageIdSSMParam] フィールドの値はすべて上書きされます。
- [NodeVolumeSize]: ノードのルートボリュームのサイズを GiB 単位で指定します。
- [NodeVolumeType]: ノードのルートボリュームタイプを指定します。
- [KeyName]: 起動後に、SSH を使用してノードに接続するときに使用できる Amazon EC2 SSH キーペアの名前を入力します。Amazon EC2 キーペアをまだ持っていない場合は、AWS Management Console で作成できます。詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EC2 キーペア」を参照してください。

  > **Note**
  >
  > ここでキーペアを指定しないと、AWS CloudFormation スタックの作成は失敗します。

- [BootstrapArguments]: `kubelet` の追加引数など、ノードブートストラップスクリプトに渡すオプションの引数を指定します。詳細については、「GitHub」の「ブートストラップスクリプトの使用状況」を参照してください。

次のノードグループをデプロイするには

- デフォルト設定よりもはるかに多くの IP アドレスを Pods に割り当てることができるノードグループについては、「Amazon EC2 ノードで使用可能な IP アドレスの量を増やす」を参照してください。
- インスタンスのブロックとは異なる CIDR ブロックから IPv4 アドレスを Pods に割り当てることができるノードグループについては、「ポッド用のカスタムネットワーク」を参照してください。
- Pods とサービスに IPv6 アドレスを割り当てることができるノードグループについては、「クラスター、Pods、services 用の IPv6 アドレス」を参照してください。
- containerd ランタイムを使用するノードグループについては、config ファイルを使用してノードグループをデプロイする必要があります。詳細については、「Docker から containerd への移行テスト」を参照してください。
- アウトバウンドインターネットアクセスがないノードグループについては、「プライベートクラスターの要件」を参照してください。

- [DisableIMDSv1]: 各ノードは、デフォルトでインスタンスメタデータサービスバージョン 1 (IMDSv1) および IMDSv2 をサポートします。IMDSv1 は無効にできます。以後、ノードグループのノードおよび Pods が MDSv1 を使用しないようにするには、DisableIMDsv1 を true に設定します。IDMS の詳細については、「インスタンスメタデータサービスの設定」を参照してください。ノードでのそれへのアクセス制限について詳しくは、ワーカーノードに割り当てられたインスタンスプロファイルへのアクセスを制限するを参照してください。
- [VpcId]: 作成した VPC の ID を入力します。
- [Subnets]: VPC 用に作成したサブネットを選択します。Amazon EKS クラスター VPC の作成 で説明されている手順で VPC を作成した場合は、起動するノードの VPC 内のプライベートサブネットのみを指定します。クラスターの [ネットワーキング] タブから、各サブネットリンクを開き、プライベートのサブネットを確認できます。

> **Important**
>
> - いずれかのサブネットがパブリックサブネットである場合は、パブリック IP アドレスの自動割り当て設定を有効にする必要があります。この設定がパブリックサブネットに対して有効になっていない場合、そのパブリックサブネットにデプロイするノードにはパブリック IP アドレスが割り当てられず、クラスターやその他の AWS のサービスと通信できなくなります。2020 年 3 月 26 日以前に、Amazon EKS AWS CloudFormation VPC テンプレートを使用して、または

eksctl を使用してサブネットがデプロイされた場合、パブリックサブネットではパブリック IP アドレスの自動割り当てが無効になります。サブネットのパブリック IP アドレスの割り当てを有効にする方法については、「サブネットのパブリック IPv4 アドレス属性の変更」を参照してください。ノードがプライベートサブネットにデプロイされている場合、NAT ゲートウェイを介してクラスターや他の AWS のサービスと通信できます。

- サブネットにインターネットアクセスがない場合は、プライベートクラスターの要件の考慮事項と追加の手順を確認してください
- AWS Outposts、Wavelength、または Local Zones サブネット を選択した場合、クラスターの作成時にサブネットが渡されないようにします。

9. [スタックオプションの設定] ページで、希望する設定を選択し、[次へ] を選択します。
10. [AWS CloudFormation が IAM リソースを作成する可能性を認識しています] の左にあるチェックボックスを選択して、[スタックの作成] を選択します。
11. スタックの作成が完了したら、コンソールで選択し、[出力] を選択します。
12. 作成されたノードグループの [NodeInstanceRole] を記録します。これは、Amazon EKS ノードを設定する際、必要になります。

ステップ 2: ノードを有効にしてクラスターに参加させるには

> **Note**
>
> アウトバウンドのインターネットアクセスのないプライベート VPC でノードを起動する場合は、ノードが VPC 内からクラスターに参加できるようにしてください。

1. aws-auth ConfigMap がすでにあるかどうかを確認します。

```
kubectl describe configmap -n kube-system aws-auth
```

2. aws-auth ConfigMap が表示されている場合は、必要に応じて更新してください。

   a. 編集する ConfigMap を開きます。

```
kubectl edit -n kube-system configmap/aws-auth
```

b. 必要に応じて新しい mapRoles エントリを追加します。rolearn 値を、前の手順で記録した [NodeInstanceRole] 値に設定します。

```
[...]
data:
  mapRoles: |
    - rolearn: <ARN of instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
[...]
```

c. ファイルを保存し、テキストエディタを終了します。

3. 「Error from server (NotFound): configmaps "aws-auth" not found」というエラーが表示されたら、ストック ConfigMap を適用してください。

a. 設定マップをダウンロードします。

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-
eks/cloudformation/2020-10-29/aws-auth-cm.yaml
```

b. aws-auth-cm.yaml ファイルで、rolearn 値を前の手順で記録した [NodeInstanceRole] 値に設定します。これを行うには、テキストエディタを使用するか、*my-node-instance-role* を置き換えて次のコマンドを実行します。

```
sed -i.bak -e 's|<ARN of instance role (not instance profile)>|my-node-
instance-role|' aws-auth-cm.yaml
```

c. 設定を適用します。このコマンドが完了するまで数分かかることがあります。

```
kubectl apply -f aws-auth-cm.yaml
```

4. ノードのステータスを監視し、Ready ステータスになるまで待機します。

```
kubectl get nodes --watch
```

Ctrl+C を入力して、シェルプロンプトに戻ります。

> **Note**
>
> 認証またはリソースタイプのエラーが発生した場合は、トラブルシューティングトピックの「許可されていないか、アクセスが拒否されました (kubectl)」を参照してください。

ノードがクラスターに参加できない場合は、「トラブルシューティングガイド」の「ノードをクラスターに結合できません」を参照してください。

5. (GPU ノードのみ) GPU インスタンスタイプと Amazon EKS 最適化アクセラレーション AMI を選択した場合は、クラスター上の DaemonSet として Kubernetes 用の NVIDIA デバイスプラグインを適用する必要があります。次のコマンドを実行する前に、*vX.X.X* を必要となる NVIDIA/k8s-device-plugin バージョンに置き換えます。

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/nvidia-device-plugin.yml
```

ステップ 3: その他のアクション

1. (オプション) サンプルアプリケーションをデプロイして、クラスターと Linux ノードをテストします。
2. (オプション) AmazonEKS_CNI_Policy マネージド IAM ポリシー (IPv4 クラスターがある場合)、または(IPv6 クラスターがある場合に、ユーザー自身が作成した) *AmazonEKS_CNI_IPv6_Policy* が the section called “ノード の IAM ロール” に添付されている場合、代わりに Kubernetes aws-node サービスアカウントに関連付けている IAM ロールに割り当てることをお勧めします。詳細については、「サービスアカウントの IAM ロールを使用する Amazon VPC CNI plugin for Kubernetes の設定」を参照してください。
3. 次の条件が true の場合、IMDS への Pod アクセスをブロックすることをお勧めします。
   - すべての Kubernetes サービスアカウントに IAM ロールを割り当てて、Pods が必要最小限のアクセス許可のみを持つように計画しています。
   - クラスター内の Pods が、現在の AWS リージョン の取得といったその他の理由で Amazon EC2 インスタンスメタデータサービス (IMDS) へのアクセスを必要としていません。

詳細については、「ワーカーノードに割り当てられたインスタンスプロファイルへのアクセスを制限する」を参照してください。

機械学習用のキャパシティブロック

> **Important**
>
> - キャパシティブロックは、特定の Amazon EC2 インスタンスタイプおよび AWS リージョン でのみ使用できます。互換性情報については、「Linux インスタンス用 Amazon EC2 ユーザーガイド」の「キャパシティブロックの操作」を参照してください。
> - キャパシティブロックは現在、Amazon EKS マネージドノードグループまたは Karpenter では使用できません。

機械学習 (ML) のキャパシティブロックを使用すると、短期間の機械学習ワークロードをサポートするために、GPU インスタンスを将来の日付で予約できます。キャパシティブロック内で実行されるインスタンスは、Amazon EC2 UltraClusters 内に自動的に近接して配置されるため、クラスタープレイスメントグループを使用する必要はありません。詳細については、「Linux インスタンス用 Amazon EC2 ユーザーガイド」の「機械学習用のキャパシティブロック」を参照してください。

Amazon EKS でキャパシティブロックを使用して、セルフマネージド型ノードのプロビジョニングとスケーリングを行うことができます。以下のステップは、一般的な概要の例を示しています。

1. AWS Management Consoleで起動テンプレートを作成します。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Use Capacity Blocks for machine learning workloads」を参照してください。

   インスタンスタイプと Amazon マシンイメージ (AMI) の設定を必ず含めてください。
2. キャパシティ予約 ID を使用して、キャパシティブロックを起動テンプレートにリンクします。

   キャパシティブロックをターゲットとする起動テンプレートを作成するための AWS CloudFormation テンプレートの例を以下に示します。

```
NodeLaunchTemplate:
    Type: "AWS::EC2::LaunchTemplate"
    Properties:
```

```
      LaunchTemplateData:
        InstanceMarketOptions:
          MarketType: "capacity-block"
        CapacityReservationSpecification:
          CapacityReservationTarget:
            CapacityReservationId: "cr-02168da1478b509e0"
        IamInstanceProfile:
          Arn: iam-instance-profile-arn
        ImageId: image-id
        InstanceType: p5.48xlarge
        KeyName: key-name
        SecurityGroupIds:
        - sg-05b1d815d1EXAMPLE
        UserData: user-data
```

キャパシティブロックはゾーン型であるため、予約が行われたアベイラビリティーゾーンのサブネットを渡す必要があります。

3. キャパシティ予約が有効になる前にセルフマネージド型ノードグループを作成する場合は、希望キャパシティを 0 に設定します。ノードグループを作成する際には、キャパシティが予約されているアベイラビリティーゾーンの当該サブネットのみを指定するようにしてください。

   使用できる CloudFormation テンプレートのサンプルを以下に示します。この例では、前の例で示した AWS::Amazon EC2::LaunchTemplate リソースの LaunchTemplateId と Version を取得します。また、同じテンプレートの他の場所で宣言されている DesiredCapacity、MaxSize、MinSize、VPCZoneIdentifier の値も取得します。

```
NodeGroup:
  Type: "AWS::AutoScaling::AutoScalingGroup"
  Properties:
    DesiredCapacity: !Ref NodeAutoScalingGroupDesiredCapacity
    LaunchTemplate:
      LaunchTemplateId: !Ref NodeLaunchTemplate
      Version: !GetAtt NodeLaunchTemplate.LatestVersionNumber
    MaxSize: !Ref NodeAutoScalingGroupMaxSize
    MinSize: !Ref NodeAutoScalingGroupMinSize
    VPCZoneIdentifier: !Ref Subnets
    Tags:
      - Key: Name
        PropagateAtLaunch: true
        Value: !Sub ${ClusterName}-${NodeGroupName}-Node
      - Key: !Sub kubernetes.io/cluster/${ClusterName}
```

```
            PropagateAtLaunch: true
            Value: owned
```

4. ノードグループが正常に作成されたら、作成されたノードグループの NodeInstanceRole を必ず記録してください。これは、ノードグループがスケーリングされたときに、新しいノードがクラスターに参加し、Kubernetes がノードを認識できるようにするために必要です。詳細については、「セルフマネージド型の Amazon Linux ノードの起動」に記載されている AWS Management Consoleの手順を参照してください。
5. Auto Scaling グループに対して、キャパシティブロックの予約時間に合わせて、スケジュールされたスケーリングポリシーを作成することをお勧めします。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon EC2 Auto Scaling のスケジュールされたスケーリング」を参照してください。

   予約したすべてのインスタンスを使用できるのは、キャパシティブロックの終了時刻の 30 分前までです。その時点でまだ稼働しているインスタンスでは、終了処理が開始されます。ノードを正常にドレインさせるのに十分な時間を確保するために、キャパシティブロックの予約終了時刻の 30 分以上前にスケーリングがゼロになるようにスケーリングをスケジュールすることをお勧めします。

   代わりに、キャパシティ予約が Active になるたびに手動でスケールアップを行う場合は、キャパシティブロック予約の開始時に Auto Scaling グループの希望キャパシティを更新する必要があります。その場合は、キャパシティブロック予約の終了時刻の 30 分以上前に手動でスケールダウンを行う必要もあります。
6. これで、ノードグループではワークロードと Pods をスケジュールする準備が整いました。
7. Pods が正常にドレインされるように、AWS ノード終了ハンドラーを設定することをお勧めします。このハンドラーは、EventBridge を使用して Amazon EC2 Auto Scaling からの「ASG スケールイン」ライフサイクルイベントを監視し、インスタンスが使用できなくなる前に Kubernetes コントロールプレーンが必要なアクションを実行できるようにします。そうしないと、Pods と Kubernetes のオブジェクトが保留状態でスタックします。詳細については、GitHub の「AWS ノード終了ハンドラー」を参照してください。

   ノード終了ハンドラーを設定しない場合は、正常にドレインさせる時間を十分に確保できるように、30 分のウィンドウに入る前に Pods のドレインを手動で開始することをお勧めします。

セルフマネージド型 Bottlerocket ノードの起動

> **Note**
>
> マネージド型ノードグループでは、ユースケースにいくつかの利点がある場合があります。詳細については、「マネージド型ノードグループ」を参照してください。

このトピックでは、Amazon EKS クラスターに登録している Bottlerocket ノードの Auto Scaling グループを起動する方法について説明します。Bottlerocket は、仮想マシンやベアメタルホスト上でコンテナの実行に使用できる、AWS による Linux ベースのオープンソースオペレーティングシステムです。ノードがクラスターに参加したら、それらのノードに Kubernetes アプリケーションをデプロイ可能になります。Bottlerocket の詳細については、「GitHub」の「Amazon EKS での Bottlerocket AMI の使用」および「`eksctl` ドキュメント」の「カスタム AMI のサポート」を参照してください。

インプレース アップグレードの詳細については、「GitHub」の「Bottlerocket Update Operator」を参照してください。

> **Important**
>
> - Amazon EKS ノードは標準の Amazon EC2 インスタンスであり、通常の Amazon EC2 インスタンス価格に基づいて請求されます。詳細については、「Amazon EC2 料金」を参照してください。
> - AWS Outposts 上の Amazon EKS 拡張クラスターで Bottlerocket ノードを起動できますが、AWS Outposts 上のローカルクラスターでは起動できません。詳細については、「AWS Outposts における Amazon EKS」を参照してください。
> - x86 または Arm プロセッサを使用して Amazon EC2 インスタンスにデプロイできます。ただし、Inferentia チップがあるインスタンスにはデプロイできません。
> - Bottlerocket は AWS CloudFormation と互換性があります。ただし、Amazon EKS の Bottlerocket ノードをデプロイするためにコピーできる公式の CloudFormation テンプレートはありません。
> - Bottlerocket のイメージには、SSH サーバーやシェルは付属していません。SSH で管理者コンテナを有効にし、ユーザーデータとともにブートストラップの設定ステップを渡すために、帯域外のアクセス方式を使用できます。詳細については、「GitHub」の「bottlerocket README.md」のセクションを参照してください。

- 探査
- 管理者コンテナ
- Kubernetes 設定

`eksctl` を使用して Bottlerocket ノードを起動するには

この手順には、`eksctl` バージョン `0.183.0` 以降が必要です。お使いのバージョンは、以下のコマンドを使用して確認できます。

```
eksctl version
```

`eksctl` のインストールまたはアップグレードの手順については、`eksctl` ドキュメントの「インストール」を参照してください。

> **Note**
>
> この手順は、`eksctl` で作成されたクラスターに対してのみ機能します。

1. 次のコンテンツをデバイスにコピーします。*`my-cluster`* を自分のクラスター名に置き換えます。この名前には、英数字 (大文字と小文字が区別されます) とハイフンのみを使用できます。先頭の文字は英数字である必要があります。また、100 文字より長くすることはできません。名前は、クラスターを作成する AWS リージョン および AWS アカウント 内で一意である必要があります。*`ng-bottlerocket`* をノードグループの名前に置き換えます。ノードグループ名は 63 文字以下である必要があります。先頭は文字または数字でなければなりませんが、残りの文字にはハイフンおよびアンダースコアを含めることもできます。Arm インスタンスにデプロイするには、*`m5.large`* を Arm インスタンスタイプに置き換えます。*`my-ec2-keypair-name`* を、起動後に、SSH を使用してノードに接続するときに使用できる Amazon EC2 SSH キーペアの名前に置き換えます。Amazon EC2 キーペアをまだ持っていない場合は、AWS Management Console で作成できます。詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EC2 キーペア」を参照してください。残りすべての *`example values`* を独自の値に置き換えてください。置き換えが完了したら、変更したコマンドを実行して `bottlerocket.yaml` ファイルを作成します。

   Arm Amazon EC2 インスタンスタイプを指定する場合は、デプロイする前に Amazon EKS 最適化 Arm Amazon Linux AMI の考慮事項を確認してください。カスタム AMI を使用したデプロイの手順については、「GitHub」の「Bottlerocket の構築」と、「`eksctl` ドキュメント」の「カ

スタム AMI のサポート」を参照してください。マネージド型ノードグループをデプロイするには、起動テンプレートを使用してカスタム AMI をデプロイします。詳細については、「起動テンプレートを使用したマネージドノードのカスタマイズ」を参照してください。

> **Important**
>
> ノードグループを AWS Outposts、AWS Wavelength、または AWS Local Zones サブネットにデプロイする場合、クラスターの作成時に AWS Outposts、AWS Wavelength、または AWS Local Zone のサブネットを渡さないでください。次の例では、サブネットを指定する必要があります。詳細については、`eksctl` ドキュメントの「設定ファイルからノードグループを作成する」と「Config ファイルのスキーマ」を参照してください。*`region-code`* をクラスターのある AWS リージョン に置き換えます。

```
cat >bottlerocket.yaml <<EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
  version: '1.30'

iam:
  withOIDC: true

nodeGroups:
  - name: ng-bottlerocket
    instanceType: m5.large
    desiredCapacity: 3
    amiFamily: Bottlerocket
    ami: auto-ssm
    iam:
       attachPolicyARNs:
          - arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
          - arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
          - arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
          - arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
    ssh:
```

```
        allow: true
        publicKeyName: my-ec2-keypair-name
EOF
```

2. 次のコマンドでノードをデプロイします。

```
eksctl create nodegroup --config-file=bottlerocket.yaml
```

出力例は次のとおりです。

ノードの作成中に、複数の行が出力されます。出力の最後の行は、次のサンプル行が表示されます。

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. (オプション) Amazon EBS CSI プラグインを使用して、Bottlerocket ノード上に Kubernetes の永続ボリュームを作成します。デフォルトの Amazon EBS ドライバーは、Bottlerocket に含まれていないファイルシステムツールに依存します。ドライバーを使用してストレージクラスを作成する方法の詳細については、「Amazon EBS CSI ドライバー」を参照してください。
4. (オプション) kube-proxy では、デフォルトで、最初に Bottlerocket がブート時に設定するものとは異なるデフォルト値に nf_conntrack_max カーネルパラメータが設定されています。Bottlerocket のデフォルト設定を保持するには、次のコマンドを使用して kube-proxy 設定を編集します。

```
kubectl edit -n kube-system daemonset kube-proxy
```

次の例にある kube-proxy 引数に、--conntrack-max-per-core と --conntrack-min を追加します。0 の設定は、変更がないことを意味します。

```
      containers:
      - command:
        - kube-proxy
        - --v=2
        - --config=/var/lib/kube-proxy-config/config
        - --conntrack-max-per-core=0
        - --conntrack-min=0
```

5. (オプション) サンプルアプリケーションをデプロイして Bottlerocket ノードをテストします。
6. 次の条件が true の場合、IMDS への Pod アクセスをブロックすることをお勧めします。

- すべての Kubernetes サービスアカウントに IAM ロールを割り当てて、Pods が必要最小限のアクセス許可のみを持つように計画しています。
- クラスター内の Pods が、現在の AWS リージョン の取得といったその他の理由で Amazon EC2 インスタンスメタデータサービス (IMDS) へのアクセスを必要としていません。

詳細については、「ワーカーノードに割り当てられたインスタンスプロファイルへのアクセスを制限する」を参照してください。

セルフマネージド型 Windows ノードの起動

このトピックでは、Amazon EKS クラスターに登録する Windows ノードの Auto Scaling グループを起動する方法について説明します。ノードがクラスターに参加したら、それらのノードに Kubernetes アプリケーションをデプロイ可能になります。

> **Important**
>
> - Amazon EKS ノードは標準の Amazon EC2 インスタンスであり、通常の Amazon EC2 インスタンス価格に基づいて請求されます。詳細については、「Amazon EC2 料金」を参照してください。
> - AWS Outposts 上の Amazon EKS 拡張クラスターで Windows ノードを起動できますが、AWS Outposts 上のローカルクラスターでは起動できません。詳細については、「AWS Outposts における Amazon EKS」を参照してください。

クラスターの Windows サポートを有効にします。Windows ノードグループを起動する前に、重要な考慮事項を確認することをお勧めします。詳細については、「Windows サポートの有効化」を参照してください。

`eksctl` または AWS Management Console を使用して、セルフマネージド型の Windows ノードを起動できます。

eksctl

`eksctl` を使用してセルフマネージド型の Windows ノードを起動する

この手順では、`eksctl` をインストール済みで、お使いの `eksctl` バージョンが `0.183.0` 以上であることが必要です。お使いのバージョンは、以下のコマンドを使用して確認できます。

```
eksctl version
```

eksctl のインストールまたはアップグレードの手順については、eksctl ドキュメントの「インストール」を参照してください。

> **Note**
>
> この手順は、eksctl で作成されたクラスターに対してのみ機能します。

1. (オプション) [AmazonEKS_CNI_Policy] マネージド IAM ポリシー (IPv4 クラスターがある場合)、または (IPv6 クラスターがある場合に、ユーザー自身が作成した) *AmazonEKS_CNI_IPv6_Policy* が the section called “ノード の IAM ロール” に添付されている場合、代わりに Kubernetes aws-node サービスアカウントに関連付けている IAM ロールに割り当てることをお勧めします。詳細については、「サービスアカウントの IAM ロールを使用する Amazon VPC CNI plugin for Kubernetes の設定」を参照してください。
2. この手順では、既存のクラスターがあることを前提としています。Windows ノードグループを追加する先の Amazon EKS クラスターと Linux ノードグループがまだない場合は、ガイド「Amazon EKS の開始方法 – eksctl」に従うことをお勧めします。このガイドは、Amazon Linux ノードで Amazon EKS クラスターを作成する方法についての完全なチュートリアルを提供します。

   次のコマンドを使用して、ノードグループを作成します。*region-code* をクラスターのある AWS リージョン に置き換えます。*my-cluster* をクラスター名に置き換えます。この名前には、英数字 (大文字と小文字が区別されます) とハイフンのみを使用できます。先頭の文字は英数字である必要があります。また、100 文字より長くすることはできません。名前は、クラスターを作成する AWS リージョン および AWS アカウント 内で一意である必要があります。*ng-windows* をノードグループの名前に置き換えます。ノードグループ名は 63 文字以下である必要があります。先頭は文字または数字でなければなりませんが、残りの文字にはハイフンおよびアンダースコアを含めることもできます。Kubernetes バージョン 1.24 以降の場合は、*2019* Server 2022 で使用する場合、2022 を Windows に置き換えることができます。残りの *example values* を、独自の値に置き換えます。

   > **Important**
   >
   > ノードグループを AWS Outposts、AWS Wavelength、または AWS Local Zones サブネットにデプロイする場合、クラスターの作成時に AWS

Outposts、Wavelength、または Local Zones サブネットは渡さないでください。AWS Outposts、Wavelength、または Local Zones サブネットを指定した設定ファイルを使用して、ノードグループを作成します。詳細については、eksctl ドキュメントの「設定ファイルからノードグループを作成する」と「設定ファイルのスキーマ」を参照してください。

```
eksctl create nodegroup \
    --region region-code \
    --cluster my-cluster \
    --name ng-windows \
    --node-type t2.large \
    --nodes 3 \
    --nodes-min 1 \
    --nodes-max 4 \
    --managed=false \
    --node-ami-family WindowsServer2019FullContainer
```

Note

- ノードがクラスターに参加できない場合は、トラブルシューティングガイドの「ノードをクラスターに結合できません」を参照してください。
- eksctl コマンドで使用可能なオプションを表示するには、次のコマンドを入力します。

```
eksctl command -help
```

出力例は次のとおりです。ノードの作成中に、複数の行が出力されます。出力の最後の行は、次のサンプル行が表示されます。

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. (オプション) サンプルアプリケーション をデプロイして、クラスターと Windows ノードをテストします。
4. 次の条件が true の場合、IMDS への Pod アクセスをブロックすることをお勧めします。

- すべての Kubernetes サービスアカウントに IAM ロールを割り当てて、Pods が必要最小限のアクセス許可のみを持つように計画しています。
- クラスター内の Pods が、現在の AWS リージョン の取得といったその他の理由で Amazon EC2 インスタンスメタデータサービス (IMDS) へのアクセスを必要としていません。

詳細については、「ワーカーノードに割り当てられたインスタンスプロファイルへのアクセスを制限する」を参照してください。

AWS Management Console

前提条件

- 既存の Amazon EKS クラスターと Linux ノードグループ。これらのリソースがない場合は、Amazon EKS の使用開始 ガイドのいずれかに従って作成することをお勧めします。このガイドでは、Linux ノードで Amazon EKS クラスターを作成する方法について説明します。
- Amazon EKS クラスターの要件を満たす、既存の VPC およびセキュリティグループ。詳細については、Amazon EKS VPC およびサブネットの要件と考慮事項およびAmazon EKS セキュリティグループの要件および考慮事項を参照してください。Amazon EKS の使用開始 ガイドでは、要件を満たす VPC を作成します。または、Amazon EKS クラスター VPC の作成 に従って手動で作成することもできます。
- Amazon EKS クラスターの要件を満たす VPC およびセキュリティグループを使用している、既存の Amazon EKS クラスター。詳細については、「Amazon EKS クラスターの作成」を参照してください。AWS Outposts、AWS Wavelength、または AWS Local Zones が有効になっている AWS リージョン にサブネットがある場合は、クラスターの作成時にそれらのサブネットが渡されるべきではありません。

ステップ 1: AWS Management Console を使用してセルフマネージド型の Windows ノードを起動する

1. クラスターステータスが `ACTIVE` と表示されるまで待ちます。クラスターがアクティブになる前にノードを起動した場合、ノードはクラスターへの登録に失敗し、再起動が必要になります。
2. https://console.aws.amazon.com/cloudformation で AWS CloudFormation コンソール を開きます。

3. [スタックの作成] を選択します。
4. [テンプレートを指定] で、[Amazon S3 URL] を選択します。
5. 次の URL をコピーして、[Amazon S3 URL] に貼り付けます。

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2023-02-09/amazon-eks-windows-nodegroup.yaml
```

6. [次へ] を 2 回選択します。
7. [スタックのクイック作成] ページで、必要に応じて以下のパラメータを入力します。

- [スタック名]: AWS CloudFormation スタックのスタック名を選択します。例えば、***my-cluster*-nodes** という名前にすることができます。
- [ClusterName]: Amazon EKS クラスターの作成時に使用した名前を入力します。

> **Important**
>
> この名前は、ステップ 1: Amazon EKS クラスターを作成する で使用した名前と正確に一致する必要があります。そうしないと、ノードはクラスターに参加できません。

- ClusterControlPlaneSecurityGroup: VPC を作成する際に生成した AWS CloudFormation 出力からセキュリティグループを選択します。

  次のステップでは、該当するグループを取得する 1 つの方法を説明します。

  1. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。
  2. クラスターの名前を選択します。
  3. [ネットワーキング] タブを選択します。
  4. [ClusterControlPlaneSecurityGroup] ドロップダウンリストから選択する場合は、[追加のセキュリティグループ] の値をリファレンスとして使用します。

- [NodeGroupName]: ノードグループの名前を入力します。この名前は、ノードに対して作成される Auto Scaling ノードグループを識別するために後で使用できます。ノードグループ名は 63 文字以下である必要があります。先頭は文字または数字でなければなりませんが、残りの文字にはハイフンおよびアンダースコアを含めることもできます。
- [NodeAutoScalingGroupMinSize]: ノードの Auto Scaling グループがスケールインできる最小ノード数を入力します。

- NodeAutoScalingGroupDesiredCapacity: スタック作成時にスケーリングする必要のあるノード数を入力します。
- [NodeAutoScalingGroupMaxSize]: ノードの Auto Scaling グループがスケールアウトできる最大ノード数を入力します。
- [NodeInstanceType]: ノードのインスタンスタイプを選択します。詳細については、「Amazon EC2 インスタンスタイプを選択する」を参照してください。

> **Note**
>
> Amazon VPC CNI plugin for Kubernetes の最新バージョン用のサポートされているインスタンスタイプは、GitHub の vpc_ip_resource_limit.go にリストされています。サポートされている最新のインスタンスタイプを利用するには、CNI のバージョンを更新することが必要になる場合があります。詳細については、「Amazon VPC CNI plugin for Kubernetes Amazon EKS アドオンの使用」を参照してください。

- [NodeImageIdSSMParam]: 現在推奨されている Amazon EKS 最適化 Windows Core AMI ID の Amazon EC2 Systems Manager パラメータが事前設定されています。Windows のフルバージョンを使用する場合、*Core* は Full に置き換えます。
- [NodeImageId]: (オプション) (Amazon EKS 最適化 AMI の代わりに) 独自のカスタム AMI を使用している場合は、AWS リージョン のノード AMI ID を入力します。このフィールドに値を指定すると、[NodeImageIdSSMParam] フィールドの値はすべて上書きされます。
- [NodeVolumeSize]: ノードのルートボリュームのサイズを GiB 単位で指定します。
- [KeyName]: 起動後に、SSH を使用してノードに接続するときに使用できる Amazon EC2 SSH キーペアの名前を入力します。Amazon EC2 キーペアをまだ持っていない場合は、AWS Management Console で作成できます。詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EC2 キーペア」を参照してください。

> **Note**
>
> ここでキーペアを指定しないと、AWS CloudFormation スタックの作成は失敗します。

- [BootstrapArguments]: kubelet を使用して、-KubeletExtraArgs の追加引数など、ノードブートストラップスクリプトに渡すオプションの引数を指定します。

- [DisableIMDSv1]: 各ノードは、デフォルトでインスタンスメタデータサービスバージョン 1 (IMDSv1) および IMDSv2 をサポートします。IMDSv1 は無効にできます。以後、ノードグループのノードおよび Pods が MDSv1 を使用しないようにするには、DisableIMDsv1 を true に設定します。IDMS の詳細については、「インスタンスメタデータサービスの設定」を参照してください。
- [VpcId]: 作成した VPC の ID を選択します。
- [NodeSecurityGroups]: VPC の作成時に Linux ノードグループ用に作成したセキュリティグループを選択します。Linux ノードに複数のセキュリティグループが添付されている場合、それらのすべてを指定します。これは、例えば、Linux ノードグループが `eksctl` を使用して作成された場合に行います。
- [Subnets]: 作成したサブネットを選択します。Amazon EKS クラスター VPC の作成 で説明されている手順で VPC を作成した場合は、起動するノードの VPC 内のプライベートサブネットのみを指定します。

> **Important**
>
> - いずれかのサブネットがパブリックサブネットである場合は、パブリック IP アドレスの自動割り当て設定を有効にする必要があります。この設定がパブリックサブネットに対して有効になっていない場合、そのパブリックサブネットにデプロイするノードにはパブリック IP アドレスが割り当てられず、クラスターやその他の AWS のサービスと通信できなくなります。2020 年 3 月 26 日以前に、Amazon EKS AWS CloudFormation VPC テンプレートを使用して、または `eksctl` を使用してサブネットがデプロイされた場合、パブリックサブネットではパブリック IP アドレスの自動割り当てが無効になります。サブネットのパブリック IP アドレスの割り当てを有効にする方法については、「サブネットのパブリック IPv4 アドレス属性の変更」を参照してください。ノードがプライベートサブネットにデプロイされている場合、NAT ゲートウェイを介してクラスターや他の AWS のサービスと通信できます。
> - サブネットにインターネットアクセスがない場合は、プライベートクラスターの要件の考慮事項と追加の手順を確認してください
> - AWS Outposts、Wavelength、または Local Zones サブネット を選択する場合は、クラスターの作成時にサブネットを渡さないようにします。

8. スタックが IAM リソースを作成する可能性があることを確認し、[スタックの作成] を選択します。
9. スタックの作成が完了したら、コンソールで選択し、[出力] を選択します。

10. 作成されたノードグループの [NodeInstanceRole] を記録します。これは、Amazon EKS Windows ノードを設定する際、必要になります。

ステップ 2: ノードを有効にしてクラスターに参加させるには

1. aws-auth ConfigMap がすでにあるかどうかを確認します。

```
kubectl describe configmap -n kube-system aws-auth
```

2. aws-auth ConfigMap が表示されている場合は、必要に応じて更新してください。

   a. 編集する ConfigMap を開きます。

```
kubectl edit -n kube-system configmap/aws-auth
```

   b. 必要に応じて新しい mapRoles エントリを追加します。rolearn 値を、前の手順で記録した [NodeInstanceRole] の値に設定します。

```
[...]
data:
  mapRoles: |
- rolearn: <ARN of linux instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
    - rolearn: <ARN of windows instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
        - eks:kube-proxy-windows
[...]
```

   c. ファイルを保存し、テキストエディタを終了します。

3. 「Error from server (NotFound): configmaps "aws-auth" not found」というエラーが表示されたら、ストック ConfigMap を適用してください。

   a. 設定マップをダウンロードします。

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-
eks/cloudformation/2020-10-29/aws-auth-cm-windows.yaml
```

b. aws-auth-cm-windows.yaml ファイルで、rolearn 値を、前の手順で記録した該当する [NodeInstanceRole] 値に設定します。これを行うには、テキストエディタを使用するか、この *example values* を置き換えて次のコマンドを実行します。

```
sed -i.bak -e 's|<ARN of linux instance role (not instance profile)>|my-
node-linux-instance-role|' \
    -e 's|<ARN of windows instance role (not instance profile)>|my-node-
windows-instance-role|' aws-auth-cm-windows.yaml
```

> **Important**
>
> - このファイルの他の行は変更しないでください。
> - Windows ノードと Linux ノードの両方に同じ IAM ロールを使用しないでください。

c. 設定を適用します。このコマンドが完了するまで数分かかることがあります。

```
kubectl apply -f aws-auth-cm-windows.yaml
```

4. ノードのステータスを監視し、Ready ステータスになるまで待機します。

```
kubectl get nodes --watch
```

Ctrl+C を入力して、シェルプロンプトに戻ります。

> **Note**
>
> 認証またはリソースタイプのエラーが発生した場合は、トラブルシューティングトピックの「許可されていないか、アクセスが拒否されました (kubectl)」を参照してください。

ノードがクラスターに参加できない場合は、「トラブルシューティングガイド」の「ノードをクラスターに結合できません」を参照してください。

ステップ 3: その他のアクション

1. (オプション) サンプルアプリケーションをデプロイして、クラスターと Windows ノードをテストします。
2. (オプション) AmazonEKS_CNI_Policy マネージド IAM ポリシー (IPv4 クラスターがある場合)、または(IPv6 クラスターがある場合に、ユーザー自身が作成した) *AmazonEKS_CNI_IPv6_Policy* が the section called “ノード の IAM ロール” に添付されている場合、代わりに Kubernetes aws-node サービスアカウントに関連付けている IAM ロールに割り当てることをお勧めします。詳細については、「サービスアカウントの IAM ロールを使用する Amazon VPC CNI plugin for Kubernetes の設定」を参照してください。
3. 次の条件が true の場合、IMDS への Pod アクセスをブロックすることをお勧めします。

   - すべての Kubernetes サービスアカウントに IAM ロールを割り当てて、Pods が必要最小限のアクセス許可のみを持つように計画しています。
   - クラスター内の Pods が、現在の AWS リージョン の取得といったその他の理由で Amazon EC2 インスタンスメタデータサービス (IMDS) へのアクセスを必要としていません。

   詳細については、「ワーカーノードに割り当てられたインスタンスプロファイルへのアクセスを制限する」を参照してください。

セルフマネージド型 Ubuntu ノードの起動

> **Note**
>
> マネージド型ノードグループでは、ユースケースにいくつかの利点がある場合があります。詳細については、「マネージド型ノードグループ」を参照してください。

このトピックでは、Ubuntu on Amazon Elastic Kubernetes Service (EKS) または Amazon EKS クラスターに登録されている Ubuntu Pro on Amazon Elastic Kubernetes Service (EKS) ノードで Auto Scaling グループを起動する方法について説明します。Ubuntu および Ubuntu Pro for EKS は、AWS と共同で開発されたカスタム AWS カーネルを含め公式の Ubuntu Minimal LTS をもとにし、EKS 専用に構築されています。Ubuntu Pro は、EKS サポート期間の延長、カーネル livepatch、FIPS コンプライアンスと無制限の Pro コンテナを実行する機能をサポートすることで、セキュリティカバレッジを追加します。

ノードがクラスターに参加したら、それらのノードにコンテナ化したアプリケーションをデプロイできるようになります。詳細については、「Ubuntu on AWS」のドキュメントおよび eksctl ドキュメントの「Custom AMI support」を参照してください。

> **Important**
>
> - Amazon EKS ノードは標準の Amazon EC2 インスタンスであり、通常の Amazon EC2 インスタンス価格に基づいて請求されます。詳細については、「Amazon EC2 料金」を参照してください。
> - AWS Outposts 上の Amazon EKS 拡張クラスターで Ubuntu ノードを起動できますが、AWS Outposts 上のローカルクラスターでは起動できません。詳細については、「AWS Outposts における Amazon EKS」を参照してください。
> - x86 または Arm プロセッサを使用して Amazon EC2 インスタンスにデプロイできます。ただし、Inferentia チップがあるインスタンスは、最初に Neuron SDK をインストールする必要がある場合があります。

eksctl を使用して Ubuntu for EKS または Ubuntu Pro for EKS ノードを起動する

この手順には、eksctl バージョン 0.183.0 以降が必要です。お使いのバージョンは、以下のコマンドを使用して確認できます。

```
eksctl version
```

eksctl のインストールまたはアップグレードの手順については、eksctl ドキュメントの「インストール」を参照してください。

> **Note**
>
> この手順は、eksctl で作成されたクラスターに対してのみ機能します。

1. 次のコンテンツをデバイスにコピーします。my-cluster を自分のクラスター名に置き換えます。この名前には、英数字 (大文字と小文字が区別されます) とハイフンのみを使用できます。先頭の文字はアルファベット文字である必要があります。また、100 文字より長くすることはできません。ng-ubuntu をノードグループの名前に置き換えます。ノードグループ名は 63 文字以下である必要があります。先頭は文字または数字でなければなりませんが、残りの文字に

はハイフンおよびアンダースコアを含めることもできます。Arm インスタンスにデプロイするには、`m5.large` を Arm インスタンスタイプに置き換えます。`my-ec2-keypair-name` を、起動後に、SSH を使用してノードに接続するときに使用できる Amazon EC2 SSH キーペアの名前に置き換えます。Amazon EC2 キーペアをまだ持っていない場合は、AWS Management Console で作成できます。詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EC2 キーペア」を参照してください。残りすべての *example values* を独自の値に置き換えてください。置き換えが完了したら、変更したコマンドを実行して `ubuntu.yaml` ファイルを作成します。

> **Important**
>
> ノードグループを AWS Outposts、AWS Wavelength、または AWS Local Zones サブネットにデプロイする場合、クラスターの作成時に AWS Outposts、AWS Wavelength、または AWS Local Zone のサブネットを渡さないでください。次の例では、サブネットを指定する必要があります。詳細については、`eksctl` ドキュメントの「設定ファイルからノードグループを作成する」と「Config ファイルのスキーマ」を参照してください。*region-code* をクラスターのある AWS リージョン に置き換えます。

```
cat >ubuntu.yaml <<EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
  version: '1.30'

iam:
  withOIDC: true

nodeGroups:
  - name: ng-ubuntu
    instanceType: m5.large
    desiredCapacity: 3
    amiFamily: Ubuntu22.04
    ami: auto-ssm
    iam:
```

```
        attachPolicyARNs:
           - arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
           - arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
           - arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
           - arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
    ssh:
        allow: true
        publicKeyName: my-ec2-keypair-name
EOF
```

Ubuntu Pro ノードグループを作成するには、`amiFamily` 値を `UbuntuPro2204` に変更します。

2. 次のコマンドでノードをデプロイします。

```
eksctl create nodegroup --config-file=ubuntu.yaml
```

出力例は次のとおりです。

ノードの作成中に、複数の行が出力されます。出力の最後の行は、次のサンプル行が表示されます。

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. (オプション) サンプルアプリケーションをデプロイして Ubuntu ノードをテストします。
4. 次の条件が true の場合、IMDS への Pod アクセスをブロックすることをお勧めします。

   - すべての Kubernetes サービスアカウントに IAM ロールを割り当てて、Pods が必要最小限のアクセス許可のみを持つように計画しています。
   - クラスター内の Pods が、現在の AWS リージョン の取得といったその他の理由で Amazon EC2 インスタンスメタデータサービス (IMDS) へのアクセスを必要としていません。

   詳細については、「ワーカーノードに割り当てられたインスタンスプロファイルへのアクセスを制限する」を参照してください。

セルフマネージド型ノードの更新

Amazon EKS に最適化された最新の AMI がリリースされたら、セフルマネージド型ノードグループのノードを新しい AMI に置き換えることを検討してください。同様に、Amazon EKS クラスターの

Kubernetes バージョンを更新した場合は、ノードを更新して、同じ Kubernetes バージョンのノードを使用します。

> **Important**
>
> このトピックでは、セルフマネージド型ノードのノードの更新について説明します。マネージド型ノードグループ を使用している場合は、「マネージド型ノードグループの更新」を参照してください。

クラスター内のセルフマネージド型ノードグループを更新して新しい AMI を使用するには、2 つの基本的な方法があります。

新しいノードグループへの移行

新しいノードグループを作成し、Pods をそのグループに移行します。新しいノードグループへの移行は、既存の AWS CloudFormation スタックで AMI ID を単に更新するよりも適切です。これは、移行プロセスが古いノードグループを `NoSchedule` として汚染し、新しいスタックが既存の Pod ワークロードを受け入れる準備ができた後にノードをドレインするためです。

既存のセルフマネージド型ノードグループの更新

既存のノードグループの AWS CloudFormation スタックを更新して、新しい AMI が使用されるようにします。この方法は、`eksctl` を使用して作成されたノードグループではサポートされていません。

新しいノードグループへの移行

このトピックは、新しいノードグループを作成し、既存のアプリケーションを新しいグループに適切に移行し、クラスターから古いノードグループを削除する方法について説明します。新しいノードグループに移行するには、`eksctl` または AWS Management Console を使用します。

eksctl

`eksctl` を使用してアプリケーションを新しいノードグループに移行するには

eksctl を移行に使用する方法の詳細については、`eksctl` ドキュメントの「管理対象外のノードグループ」を参照してください。

この手順には、`eksctl` バージョン `0.183.0` 以降が必要です。お使いのバージョンは、以下のコマンドを使用して確認できます。

```
eksctl version
```

eksctl のインストールまたはアップグレードの手順については、eksctl ドキュメントの「インストール」を参照してください。

> **Note**
>
> この手順は、eksctl で作成されたクラスターとノードグループに対してのみ機能します。

1. *my-cluster* をクラスター名に置き換えて、既存のノードグループの名前を取得します。

   ```
   eksctl get nodegroups --cluster=my-cluster
   ```

   出力例は次のとおりです。

   ```
   CLUSTER      NODEGROUP          CREATED               MIN SIZE      MAX SIZE
    DESIRED CAPACITY     INSTANCE TYPE     IMAGE ID
   default      standard-nodes   2019-05-01T22:26:58Z  1             4             3
                        t3.medium         ami-05a71d034119ffc12
   ```

2. 次のコマンドを使用して、eksctl で新しいノードグループを起動します。コマンドのすべての *example value* を独自の値に置き換えます。バージョン番号は、お使いのコントロールプレーンの Kubernetes バージョンよりも新しいものにすることはできません。また、コントロールプレーンの Kubernetes バージョンより 3 つ以上前のマイナーなバージョンにすることもできません。コントロールプレーンと同じバージョンを使用することをお勧めします。

   次の条件が true の場合、IMDS への Pod アクセスをブロックすることをお勧めします。

   - すべての Kubernetes サービスアカウントに IAM ロールを割り当てて、Pods が必要最小限のアクセス許可のみを持つように計画しています。
   - クラスター内の Pods が、現在の AWS リージョン の取得といったその他の理由で Amazon EC2 インスタンスメタデータサービス (IMDS) へのアクセスを必要としていません。

詳細については、「ワーカーノードに割り当てられたインスタンスプロファイルへのアクセスを制限する」を参照してください。

IMDS への Pod アクセスをブロックするには、次のコマンドに `--disable-pod-imds` オプションを追加します。

> **Note**
>
> 使用可能なフラグとその説明については、「https://eksctl.io/」を参照してください。

```
eksctl create nodegroup \
  --cluster my-cluster \
  --version 1.30 \
  --name standard-nodes-new \
  --node-type t3.medium \
  --nodes 3 \
  --nodes-min 1 \
  --nodes-max 4 \
  --managed=false
```

3. 前のコマンドが完了したら、次のコマンドを使用して、すべてのノードが Ready 状態になったことを確認します。

```
kubectl get nodes
```

4. 次のコマンドを使用して、元のノードグループを削除します。このコマンドで、すべての *example value* を自分のクラスター名とノードグループ名に置き換えます。

```
eksctl delete nodegroup --cluster my-cluster --name standard-nodes-old
```

AWS Management Console and AWS CLI

AWS Management Console と AWS CLI を使用してアプリケーションを新しいノードグループに移行するには

1. 新しいノードグループを起動するには、セルフマネージド型の Amazon Linux ノードの起動の手順に従います。

2. スタックの作成が完了したら、コンソールで選択し、[出力] を選択します。
3. 作成されたノードグループの [NodeInstanceRole] を記録します。Amazon EKS ノードをクラスターに追加するには、これが必要です。

> **Note**
>
> 追加の IAM ポリシーを古いノードグループの IAM ロールにアタッチした場合は、この同じポリシーを新しいノードグループの IAM ロールにアタッチして、新しいグループでその機能を管理します。これは、例えば、Kubernetes Cluster Autoscaler のアクセス許可を追加した場合に適用されます。

4. 相互通信できるように、両方のノードグループのセキュリティグループを更新します。詳細については、「」を参照してくださいAmazon EKS セキュリティグループの要件および考慮事項

   a. 両方のノードグループのセキュリティグループ ID を記録します。これは、AWS CloudFormation スタックからの出力に、NodeSecurityGroup の値として表示されます。

      次の AWS CLI コマンドを使用して、スタック名からセキュリティグループ ID を取得します。これらのコマンドで、`oldNodes` は、古いノードスタックの AWS CloudFormation スタック名、`newNodes` は、移行先のスタックの名前を表します。*`example value`* をすべて自分の値に置き換えてください。

      ```
      oldNodes="old_node_CFN_stack_name"
      newNodes="new_node_CFN_stack_name"

      oldSecGroup=$(aws cloudformation describe-stack-resources --stack-name
       $oldNodes \
      --query 'StackResources[?
      ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
      --output text)
      newSecGroup=$(aws cloudformation describe-stack-resources --stack-name
       $newNodes \
      --query 'StackResources[?
      ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
      --output text)
      ```

   b. 互いにトラフィックを受け入れるように、各ノードのセキュリティグループに進入ルールを追加します。

以下の AWS CLI コマンドでは、他のセキュリティグループからのすべてのプロトコルのトラフィックをすべて許可するインバウンドルールを各セキュリティグループに追加します。この設定により、ワークロードを新しいグループに移行している間に、各ノードグループ内の Pods は相互に通信できるようになります。

```
aws ec2 authorize-security-group-ingress --group-id $oldSecGroup \
--source-group $newSecGroup --protocol -1
aws ec2 authorize-security-group-ingress --group-id $newSecGroup \
--source-group $oldSecGroup --protocol -1
```

5. aws-auth configmap を編集して、新しいノードインスタンスロールを RBAC にマッピングします。

```
kubectl edit configmap -n kube-system aws-auth
```

新しいノードグループの新しい mapRoles エントリを追加します。クラスターが AWS GovCloud (米国東部) または AWS GovCloud (米国東部) の AWS リージョン にある場合は、arn:aws: を arn:aws-us-gov: に置き換えます。

```
apiVersion: v1
data:
  mapRoles: |
    - rolearn: ARN of instance role (not instance profile)
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes>
    - rolearn: arn:aws:iam::111122223333:role/nodes-1-16-NodeInstanceRole-
U11V27W93CX5
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
```

ARN of instance role (not instance profile) スニペットを、前の手順で記録した [NodeInstanceRole] 値に置き換えます。次に、更新された configmap を適用するには、ファイルを保存して閉じます。

6. ノードのステータスを監視し、新しいノードがクラスターに結合され、Ready ステータスになるまで待機します。

```
kubectl get nodes --watch
```

7. (オプション) Kubernetes Cluster Autoscaler を使用している場合は、スケーリングアクションの競合を回避するために、デプロイメントを 0 (ゼロ) レプリカにスケールダウンします。

```
kubectl scale deployments/cluster-autoscaler --replicas=0 -n kube-system
```

8. 以下のコマンドを使用して、NoSchedule で削除する各ノードをテイントに設定します。これは、置き換えるノード上で新しい Pods がスケジュールまたは再スケジュールされないようにするためです。詳細については、Kubernetes ドキュメントの「テイントと容認」を参照してください。

```
kubectl taint nodes node_name key=value:NoSchedule
```

ノードを新しい Kubernetes バージョンにアップグレードする場合は、次のコードスニペットを使用して、特定の Kubernetes バージョン (この場合は 1.28) のすべてのノードを識別してテイントに設定できます。バージョン番号は、お使いのコントロールプレーンの Kubernetes バージョンよりも新しいものにすることはできません。また、コントロールプレーンの Kubernetes バージョンより 3 つ以上前のマイナーなバージョンにすることもできません。コントロールプレーンと同じバージョンを使用することをお勧めします。

```
K8S_VERSION=1.28
nodes=$(kubectl get nodes -o jsonpath="{.items[?
(@.status.nodeInfo.kubeletVersion==\"v$K8S_VERSION\")].metadata.name}")
for node in ${nodes[@]}
do
    echo "Tainting $node"
    kubectl taint nodes $node key=value:NoSchedule
done
```

9. クラスターの DNS プロバイダーを決定します。

```
kubectl get deployments -l k8s-app=kube-dns -n kube-system
```

出力例は次のとおりです。このクラスターでは、DNS 解決に CoreDNS を使用していますが、クラスターからは kube-dns が返される場合があります。

```
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
```

```
coredns   1         1         1            1           31m
```

10. 現在のデプロイメントで実行しているレプリカが 2 つ未満の場合は、そのデプロイメントを 2 つのレプリカにスケールアウトします。前のコマンドで、その出力が返された場合は、*coredns* を **kubedns** に置き換えます。

```
kubectl scale deployments/coredns --replicas=2 -n kube-system
```

11. 次のコマンドを使用して、クラスターから削除する各ノードをドレーンします。

```
kubectl drain node_name --ignore-daemonsets --delete-local-data
```

ノードを新しい Kubernetes バージョンにアップグレードする場合は、次のコードスニペットを使用して、特定の Kubernetes バージョン (この場合は *1.28*) のすべてのノードを識別してドレーンします。

```
K8S_VERSION=1.28
nodes=$(kubectl get nodes -o jsonpath="{.items[?
(@.status.nodeInfo.kubeletVersion==\"v$K8S_VERSION\")].metadata.name}")
for node in ${nodes[@]}
do
    echo "Draining $node"
    kubectl drain $node --ignore-daemonsets --delete-local-data
done
```

12. 古いノードのドレーンが完了したら、以前承認したセキュリティグループのインバウンドルールを取り消します。次に、AWS CloudFormation スタックを削除してインスタンスを終了してください。

> **Note**
>
> さらに IAM ポリシーを古いノードグループの IAM ロールにアタッチした場合 (Kubernetes Cluster Autoscaler に対するアクセス許可を追加する場合など) は、ロールからその追加ポリシーをデタッチしてから、AWS CloudFormation スタックを削除します。

a. 先ほどノードのセキュリティグループ用に作成したインバウンドルールを取り消します。これらのコマンドで、oldNodes は、古いノードスタックの AWS CloudFormation スタック名、newNodes は、移行先のスタックの名前を表します。

```
oldNodes="old_node_CFN_stack_name"
newNodes="new_node_CFN_stack_name"

oldSecGroup=$(aws cloudformation describe-stack-resources --stack-name
 $oldNodes \
--query 'StackResources[?
ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
--output text)
newSecGroup=$(aws cloudformation describe-stack-resources --stack-name
 $newNodes \
--query 'StackResources[?
ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
--output text)
aws ec2 revoke-security-group-ingress --group-id $oldSecGroup \
--source-group $newSecGroup --protocol -1
aws ec2 revoke-security-group-ingress --group-id $newSecGroup \
--source-group $oldSecGroup --protocol -1
```

b. https://console.aws.amazon.com/cloudformation で AWS CloudFormation コンソール を開きます。

c. 古いノードスタックを選択します。

d. [削除] をクリックします。

e. [スタックの削除] 確認ダイアログボックスで、[スタックを削除] をクリックします。

13. aws-auth configmap を編集して、RBAC から古いノードインスタンスロールを削除します。

```
kubectl edit configmap -n kube-system aws-auth
```

古いノードグループの mapRoles エントリを削除します。クラスターが AWS GovCloud (米国東部) または AWS GovCloud (米国東部) の AWS リージョン にある場合は、arn:aws: を arn:aws-us-gov: に置き換えます。

```
apiVersion: v1
data:
```

```
  mapRoles: |
    - rolearn: arn:aws:iam::111122223333:role/nodes-1-16-NodeInstanceRole-
W70725MZQFF8
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
    - rolearn: arn:aws:iam::111122223333:role/nodes-1-15-NodeInstanceRole-
U11V27W93CX5
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes>
```

更新された configmap を適用するには、ファイルを保存して閉じます。

14. (オプション) Kubernetes Cluster Autoscaler を使用している場合は、デプロイメントのスケーリングを 1 レプリカに戻します。

> **Note**
>
> 必要に応じて、新しい Auto Scaling グループにタグ (k8s.io/cluster-autoscaler/enabled,k8s.io/cluster-autoscaler/*my-cluster* など) を付け、新しくタグ付けした Auto Scaling グループを参照するように、Cluster Autoscaler デプロイメントのコマンドを更新する必要があります。詳細については、「AWS の Cluster Autoscaler」を参照してください。

```
kubectl scale deployments/cluster-autoscaler --replicas=1 -n kube-system
```

15. (オプション) 最新バージョンの Amazon VPC CNI Plugin for Kubernetes を使用していることを確認します。サポートされている最新のインスタンスタイプを利用するには、CNI のバージョンを更新することが必要になる場合があります。詳細については、「Amazon VPC CNI plugin for Kubernetes Amazon EKS アドオンの使用」を参照してください。
16. クラスターの DNS 解決 (前のステップを参照) に kube-dns を使用している場合は、kube-dns デプロイメントを 1 レプリカにスケーリングしてください。

```
kubectl scale deployments/kube-dns --replicas=1 -n kube-system
```

既存のセルフマネージド型ノードグループの更新

このトピックは、新しい AMI を使用して既存の AWS CloudFormation セルフマネージド型ノードスタックを更新する方法について説明します。この手順を使用して、クラスターの更新後にノードを新しいバージョンの Kubernetes に更新することができます。それ以外の場合は、Kubernetes の既存バージョン用の Amazon EKS に最適化された最新の AMI に更新することができます。

> **Important**
>
> このトピックでは、セルフマネージド型ノードのノードの更新について説明します。マネージド型ノードグループ の使用の詳細については、「マネージド型ノードグループの更新」を参照してください。

最新のデフォルトの Amazon EKS ノード AWS CloudFormation テンプレートは、以前の AMI を一度に 1 つずつ削除する前に、新しい AMI でインスタンスをクラスターに起動するように設定されています。この設定により、ローリング更新中にクラスター内で Auto Scaling グループのアクティブなインスタンスが必要数確保されるようになります。

> **Note**
>
> この方法は、`eksctl` を使用して作成されたノードグループではサポートされていません。`eksctl` を使用してクラスターまたはノードグループを作成した場合は、「新しいノードグループへの移行」を参照してください。

既存のノードグループを更新するには

1. クラスターの DNS プロバイダーを決定します。

   ```
   kubectl get deployments -l k8s-app=kube-dns -n kube-system
   ```

   出力例は次のとおりです。このクラスターでは、DNS 解決に CoreDNS を使用していますが、クラスターからは `kube-dns` が返される場合があります。使用しているバージョンによって、出力が異なる場合があります。`kubectl`

   ```
   NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
   coredns   1         1         1            1           31m
   ```

2. 現在のデプロイメントで実行しているレプリカが 2 つ未満の場合は、そのデプロイメントを 2 つのレプリカにスケールアウトします。前のコマンドで、その出力が返された場合は、*coredns* を **kube-dns** に置き換えます。

```
kubectl scale deployments/coredns --replicas=2 -n kube-system
```

3. (オプション) Kubernetes Cluster Autoscaler を使用している場合は、スケーリングアクションの競合を回避するために、デプロイメントを 0 (ゼロ) レプリカにスケールダウンします。

```
kubectl scale deployments/cluster-autoscaler --replicas=0 -n kube-system
```

4. 現在のノードグループのインスタンスタイプと必要なインスタンス数を決定します。グループの AWS CloudFormation テンプレートを更新する場合は、後でこれらの値を入力します。

   a. Amazon EC2 コンソール https://console.aws.amazon.com/ec2/ を開きます。

   b. 左ナビゲーションペインの [Launch Configurations] (起動設定) を選択し、既存のノードの起動設定のインスタンスタイプを書き留めます。

   c. 左ナビゲーションペインの [Auto Scaling] (グループ) を選択し、既存の Auto Scaling グループのノードの、インスタンスの [Desired] (必要) 数を書き留めます。

5. https://console.aws.amazon.com/cloudformation で AWS CloudFormation コンソール を開きます。

6. ノードグループスタックを選択し、[更新] を選択します。

7. [Replace current template] (現在のテンプレートを置き換える) を選択し、Amazon S3 URL を選択します。

8. Amazon S3 URL で、ノードの AWS CloudFormation テンプレートの最新バージョンを使用していることを確認するために以下の URL をテキストエリアに貼り付けます。続いて、[次へ] を選択します。

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2022-12-23/amazon-eks-nodegroup.yaml
```

9. [Specify stack details] (スタックの詳細の指定) ページで、以下のパラメータを指定し、[Next] (次へ) を選択します。

   - [NodeAutoScalingGroupDesiredCapacity] – 前のステップで記録した必要なインスタンス数を入力します。または、スタック更新時にスケーリングする必要のある新しいノード数を入力します。

- NodeAutoScalingGroupMaxSize - ノードの Auto Scaling グループがスケールアウトする最大ノード数を入力します。この値は、必要な容量よりも少なくとも 1 ノード多い必要があります。これは、更新時にノード数を減らさずにノードのローリング更新を実行できるようにするためです。
- NodeInstanceType — 前のステップで記録したインスタンスタイプを選択します。または、ノードに別のインスタンスタイプを選択します。別のインスタンスタイプを選択する前に、Amazon EC2 インスタンスタイプを選択するを確認してください。各 Amazon EC2 インスタンスタイプは最大数の Elastic Network Interface (ネットワークインターフェイス) をサポートし、各ネットワークインターフェイスは最大数の IP アドレスをサポートします。ワーカーノードと Pod には、それぞれ IP アドレスが割り当てられています。そのため、各 Amazon EC2 ノードで実行させたい Pods の最大数をサポートするインスタンスタイプを選択することが重要です。インスタンスタイプごとにサポートされるネットワークインターフェイスと IP アドレスのリストについては、「各インスタンスタイプのネットワークインターフェイスごとの IP アドレス」を参照してください。例えば `m5.large` インスタンスタイプは、ワーカーノードと Pods に対して最大 30 の IP アドレスをサポートします。

> **Note**
>
> Amazon VPC CNI plugin for Kubernetes の最新バージョン用のサポートされているインスタンスタイプは、GitHub の vpc_ip_resource_limit.go に表示されています。サポートされている最新のインスタンスタイプを使用するため、Amazon VPC CNI plugin for Kubernetes のバージョンを更新することが必要になる場合があります。詳細については、「Amazon VPC CNI plugin for Kubernetes Amazon EKS アドオンの使用」を参照してください。

> **Important**
>
> AWS リージョン によっては利用できないインスタンスタイプがあります。

- NodeImageIdSSMParam - 更新する AMI ID の Amazon EC2 Systems Manager パラメータです。次の値では、Kubernetes バージョン `1.30` 用の Amazon EKS に最適化された最新の AMI が使用されています。

```
/aws/service/eks/optimized-ami/1.30/amazon-linux-2/recommended/image_id
```

1.30 は、同一のサポートされている Kubernetes バージョンに置き換えることができます。または、コントロールプレーンで実行されている Kubernetes バージョンより 1 つ前までのバージョンである必要があります。ノードは、コントロールプレーンと同じバージョンにしておくことをお勧めします。*amazon-linux-2* を別の AMI タイプに置き換えることもできます。詳細については、「Amazon EKS 最適化 Amazon Linux AMI ID の取得」を参照してください。

> **Note**
>
> Amazon EC2 Systems Manager パラメータを使用すると、AMI ID を検索して指定することなく、後でノードを更新できます。AWS CloudFormation スタックがこの値を使用している場合、スタックの更新によって常に、指定された Kubernetes バージョンで推奨される Amazon EKS に最適化された最新の AMI が起動されます。これは、テンプレートの値を変更しない場合も同様です。

- NodeImageId - 独自のカスタム AMI を使用するには、使用する AMI の ID を入力します。

> **Important**
>
> この値は、NodeImageIdSSMParam に指定されたすべての値を上書きします。NodeImageIdSSMParam 値を使用する場合は、 NodeImageId の値が空白であることを確認します。

- [DisableIMDSv1] – 各ノードは、デフォルトでインスタンスメタデータサービスバージョン 1 (IMDSv1) および IMDSv2 をサポートします。ただし、IMDSv1 を無効にできます。ノードグループでスケジュールされているノードまたは Pods で IMDSv1 を使用しない場合は、[true] (正) を選択します。IDMS の詳細については、「インスタンスメタデータサービスの設定」を参照してください。サービスアカウントの IAM ロールを実装した場合、AWS サービスへのアクセスを必要とするすべての Pods に必要なアクセス許可を直接割り当ててください。これにより、クラスター内の Pods は、現在の AWS リージョン の取得など、その他の理由で IMDS へのアクセスを必要としません。次に、ホストネットワークを使用しない Pods の IMDSv2 へのアクセスを無効にすることもできます。詳細については、ワーカーノードに割り当てられたインスタンスプロファイルへのアクセスを制限する を参照してください。

10. (オプション) オプション ページで、スタックリソースをタグ付けします。[次へ] を選択します。

11. [確認] ページで情報を確認し、スタックで IAM リソースを作成することを承認して、[スタックの更新] を選択します。

    > **Note**
    >
    > クラスター内の各ノードの更新には数分かかります。すべてのノードの更新が完了するのを待ってから、次の手順を実行します。

12. クラスターの DNS プロバイダーは kube-dns の場合は、kube-dns デプロイを 1 レプリカにスケールインします。

    ```
    kubectl scale deployments/kube-dns --replicas=1 -n kube-system
    ```

13. (オプション) Kubernetes Cluster Autoscaler を使用している場合は、デプロイメントのスケーリングを目的のレプリカ数に戻します。

    ```
    kubectl scale deployments/cluster-autoscaler --replicas=1 -n kube-system
    ```

14. (オプション) 最新バージョンの Amazon VPC CNI plugin for Kubernetes を使用していることを確認します。サポートされている最新のインスタンスタイプを使用するため、Amazon VPC CNI plugin for Kubernetes のバージョンを更新することが必要になる場合があります。詳細については、「Amazon VPC CNI plugin for Kubernetes Amazon EKS アドオンの使用」を参照してください。

AWS Fargate

> **Important**
>
> Amazon EKS を使用した AWS Fargate は、AWS GovCloud (米国東部) および AWS GovCloud (米国西部) ではご利用いただけません。

このトピックでは、Amazon EKS を使用して Kubernetes で Pods AWS Fargate を実行する方法について説明します。Fargate は、コンテナに適切なサイズのコンピューティング能力をオンデマンドで提供するテクノロジーです。 を使用すると、コンテナを実行するために仮想マシンのグループをプロビジョニング、設定、スケールする必要がありません。これにより、サーバータイプの選択、ノードグループをスケールするタイミングの決定、クラスターのパッキングの最適化を行う必要がなくなります。

Fargate で起動する Pods と、Fargate プロファイル での実行方法をコントロールできます。Fargate プロファイルは Amazon EKS クラスターの一部として定義されています。Amazon EKS は、Kubernetes が提供するアップストリームの拡張可能なモデルを使用して AWS により構築されたコントローラーを使用して、Kubernetes と Fargate を統合します。これらのコントローラーは、Amazon EKS が管理する Kubernetes コントロールプレーンの一部として実行され、ネイティブ Kubernetes Pods を Fargate にスケジュールするロールを果たします。Fargate コントローラーには、いくつかの変更と検証アドミッションコントローラに加えて、デフォルトの Kubernetes スケジューラとともに実行される新しいスケジューラが含まれています。Fargate で実行する条件を満たす Pod を起動すると、クラスターで実行されている Fargate コントローラーは Pod を認識し、更新し、Fargate にスケジューリングします。

このトピックでは、Fargate で実行されている Pods のさまざまなコンポーネントについて説明し、Amazon EKS で Fargate を使用する際の特別な考慮事項を示します。

AWS Fargate に関する考慮事項

Amazon EKS での Fargate の使用についての考慮事項を以下に示します。

- Fargate で実行される各 Pod には、独自の分離境界があります。基本となるカーネル、CPU リソース、メモリリソース、または Elastic Network Interface を別の Pod と共有しません。
- Network Load Balancer と Application Load Balancer (ALBs) は、IPターゲットでのみ Fargate で使用できます。詳細については、「ネットワークロードバランサーを作成する」および「Amazon EKS でのアプリケーション負荷分散」を参照してください。
- Fargate 公開サービスは、ターゲットタイプの IP モードでのみ実行され、ノード IP モードでは実行されません。マネージド型ノードで実行されているサービスと Fargate で実行されているサービスからの接続を確認するためのおすすめの方法は、サービス名を使用して接続することです。
- Fargate で実行するには、ポッドがスケジューリングされた時点で Fargate プロファイルと一致する必要があります。Fargate プロファイルに一致しない Pod は `Pending` としてスタックする可能性があります。一致する Fargate プロファイルが存在する場合、Fargate に再スケジューリングするために作成した保留中の Pods を削除できます。
- デーモンセットは Fargate ではサポートされていません。アプリケーションでデーモンが必要な場合は、Pods でサイドカーコンテナとして実行するようにデーモンを再設定します。
- 特権を持つコンテナは Fargate ではサポートされていません。
- Fargate で実行されているポッドは、`HostPort` または `HostNetwork` を Pod マニフェストに指定できません。

- Fargate Pods の場合、デフォルトの `nofile` および `nproc` のソフトリミットは 1024 で、ハードリミットは 65535 です。
- GPU は現在、Fargate では使用できません。
- Fargate で実行されているポッドは、プライベートサブネットでのみサポートされています (AWS のサービスへの NAT ゲートウェイアクセスができますが、インターネットゲートウェイへの直接ルーティングはできません)。そのため、クラスターの VPC ではプライベートサブネットを使用できるようにする必要があります。アウトバウンドインターネットアクセスのないクラスターについては、「」を参照してくださいプライベートクラスターの要件
- Vertical Pod Autoscaler を使用して Fargate Pods の CPU とメモリの適切な初期サイズを設定し、Horizontal Pod Autoscaler を使用してそれらの Pods をスケールできます。Vertical Pod Autoscaler で、より大きい CPU とメモリの組み合わせを持つ Fargate に Pods を自動的に再デプロイする場合は、正しい機能を確保するため、Vertical Pod Autoscaler のモードを `Auto` または `Recreate` に設定します。詳細については、「GitHub」で「Vertical Pod Autoscaler のドキュメント」を参照してください。
- VPC で DNS 解決と DNS ホスト名を有効にする必要があります。詳細については、「VPC の DNS サポートを更新する」を参照してください。
- Amazon EKS Fargate は、仮想マシン (VM) 内の各ポッドを分離することで、Kubernetes アプリケーションの多層防御を追加します。この VM 境界は、コンテナエスケープが発生した場合に他のポッドが使用するホストベースのリソースへのアクセスを防ぎます。これは、コンテナ化されたアプリケーションを攻撃し、コンテナ外のリソースにアクセスする一般的な方法です。

  Amazon EKS を使用しても、責任共有モデルに基づくお客様の責任は変わりません。クラスターのセキュリティとガバナンスのコントロールの設定について、慎重に検討する必要があります。アプリケーションを分離する最も安全な方法は、常に別のクラスターで実行することです。
- Fargate プロファイルでは、VPC セカンダリ CIDR ブロックからのサブネットの指定がサポートされています。セカンダリ CIDR ブロックを指定することもできます。サブネットの使用できる IP アドレスの数は限られています。その結果、クラスター内に作成できる Pods の数が制限されます。Pods に別のサブネットを使用することで、使用可能な IP アドレスの数を増やすことができます。詳細については、「IPv4 CIDR ブロックの VPC への追加」を参照してください。
- Amazon EC2 インスタンスメタデータサービス (IMDS) は、Fargate ノードにデプロイされた Pods では使用できません。IAM 認証情報を必要とする Fargate にデプロイされた Pods がある場合は、サービスアカウントの IAM ロール を使用して Pods に割り当てます。Pods が IMDS を通じて利用可能な他の情報にアクセスする必要がある場合は、この情報を Pod 仕様にハードコーディングする必要があります。これには、Pod がデプロイされる AWS リージョン またはアベイラビリティーゾーンが含まれます。

- Fargate Pods は AWS Outposts、AWS Wavelength、AWS ローカルゾーンのいずれにもデプロイできません。
- Amazon EKS は定期的に Fargate Pods にパッチを適用して安全に保つ必要があります。影響を軽減する方法で更新を試みますが、ポッドのエビクションが正常に行われなかった場合、Pods を削除しなければならない場合があります。中断を最小限に抑えるために行えるアクションがいくつかあります。詳細については、「Fargate OS のパッチ適用」を参照してください。
- Amazon VPC CNI plugin for Amazon EKS は、Fargate ノードにインストールされています。互換性のある代替 CNI プラグイン を Fargate ノードで使用することはできません。
- Fargate で実行されている Pod が、Amazon EFS ファイルシステムを自動的にマウントします。Fargate ノードでは動的永続ボリュームプロビジョニングを使用できませんが、静的プロビジョニングを使用することはできます。
- Amazon EBS ボリュームを Fargate Pods にマウントすることはできません。
- Amazon EBS CSI コントローラーは Fargate ノードで実行できますが、Amazon EBS CSI ノード DaemonSet は Amazon EC2 インスタンスでのみ実行できます。
- Kubernetes Job が `Completed` または `Failed` とマークされた後も、Job が作成した Pods は通常存在し続けます。この動作によりログと結果を表示できますが、Fargate を使用する場合、その後 Job をクリーンアップしないとコストが発生します。

  Job が完了または失敗した後に、関連する Pods を自動的に削除するには、有効期限 (TTL) コントローラを使用して期間を指定できます。次の例では、Job マニフェストで `.spec.ttlSecondsAfterFinished` を指定する方法を示しています。

```
apiVersion: batch/v1
kind: Job
metadata:
  name: busybox
spec:
  template:
    spec:
      containers:
      - name: busybox
        image: busybox
        command: ["/bin/sh", "-c", "sleep 10"]
      restartPolicy: Never
  ttlSecondsAfterFinished: 60 # <-- TTL controller
```

Amazon EKS を使用した AWS Fargate の使用開始

> **Important**
>
> Amazon EKS を使用した AWS Fargate は、AWS GovCloud (米国東部) および AWS GovCloud (米国西部) ではご利用いただけません。

このトピックでは、Amazon EKS クラスターを使用して AWS Fargate で Pods の実行を開始する方法について説明します。

CIDR ブロックを使用してクラスターのパブリックエンドポイントへのアクセスを制限する場合は、プライベートエンドポイントアクセスも有効にすることをお勧めします。こうすることで、Fargate Pods がクラスターと通信できるようになります。プライベートエンドポイントが有効になっていない場合、パブリックアクセスに指定する CIDR ブロックに、VPC からのアウトバウンドソースを含める必要があります。詳細については、「Amazon EKS クラスターエンドポイントアクセスコントロール」を参照してください。

前提条件

既存のクラスター。既存の Amazon EKS クラスターがなければ、Amazon EKS の使用開始 を参照してください。

既存のノードが Fargate Pods と通信できることを確認します

ノードのない新しいクラスター、またはマネージド型ノードグループのみを持つクラスターを使用している場合は、「Fargate Pod 実行ロールを作成する」に進みます。

既にそれに関連付けられているノードがある既存のクラスターで作業していると仮定します。これらのノードの Pods が Fargate で実行されている Pods と自由に通信できることを確認します。Fargate で実行されている Pods は、関連付けられているクラスターのクラスターセキュリティグループを使用するように自動的に設定されます。クラスター内の既存のノードが、クラスターセキュリティグループとの間でトラフィックを送受信できることを確認してください。マネージド型ノードグループ は、クラスターセキュリティグループも使用するように自動的に設定されるため、この互換性のためにノードを変更したり確認したりする必要はありません。

`eksctl` または Amazon EKS マネージド型 AWS CloudFormationテンプレートで作成された既存のノードグループの場合、クラスターセキュリティグループをノードに手動で追加できます。または、ノードグループの Auto Scaling グループ起動テンプレートを変更して、クラスターセキュリティグ

ループをインスタンスにアタッチすることもできます。詳細については、Amazon VPC ユーザーガイドの「インスタンスのセキュリティグループの変更」を参照してください。

AWS Management Console で、クラスターの [Networking] (ネットワーク) セクションでクラスターのクラスターセキュリティグループを確認できます。これを行うには、次の AWS CLI コマンドを実行します。このコマンドを使用する場合は、*my-cluster* を自分のクラスター名に置き換えます。

```
aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.clusterSecurityGroupId
```

Fargate Pod 実行ロールを作成する

クラスターが AWS Fargate で Pods を作成する場合、Fargate インフラストラクチャで実行されるコンポーネントは、ユーザーに代わって AWS API にコールを実行す必要があります。Amazon EKS の Pod 実行ロールにより、これらを行うための IAM アクセス許可が付与されます。AWS Fargate Pod 実行ロールを作成するには、「Amazon EKS Pod 実行 IAM ロール」を参照してください。

> **Note**
>
> `--fargate` オプションを使用して `eksctl` でクラスターを作成した場合は、クラスターに Pod 実行ロールが既にあり、これはパターン `eksctl-`*`my-cluster`*`-FargatePodExecutionRole-`*`ABCDEFGHIJKL`* で IAM コンソールに表示されます。同様に、`eksctl` を使用して Fargate プロファイルを作成する場合、`eksctl` は Pod 実行ロールを作成します (まだ存在しない場合)。

クラスターの Fargate プロファイルを作成する

クラスターの Fargate で実行されている Pods をスケジューリングする前に、起動時に Fargate を使用する Pods を指定する Fargate プロファイルを定義する必要があります。詳細については、「AWS Fargate プロファイル」を参照してください。

> **Note**
>
> `--fargate` オプションを使用して `eksctl` でクラスターを作成した場合、クラスターの Fargate プロファイルは、`kube-system` と `default` の名前空間のすべての Pods のセレクターを使用して既に作成されています。Fargate で使用するその他の名前空間の Fargate プロファイルを作成するには、以下の手順に従います。

Fargate プロファイルを作成するには、eksctl または AWS Management Consoleを使用します。

eksctl

この手順には、eksctl バージョン 0.183.0 以降が必要です。お使いのバージョンは、以下のコマンドを使用して確認できます。

```
eksctl version
```

eksctl のインストールまたはアップグレードの手順については、eksctl ドキュメントの「インストール」を参照してください。

eksctl で、Fargate プロファイルを作成するには

以下の eksctl コマンドで Fargate プロファイルを作成し、すべての *example value* を自分の値に置き換えます。名前空間を指定する必要があります。ただし、--labels オプションは必須ではありません。

```
eksctl create fargateprofile \
    --cluster my-cluster \
    --name my-fargate-profile \
    --namespace my-kubernetes-namespace \
    --labels key=value
```

my-kubernetes-namespace および *key=value* ラベルには、特定のワイルドカードを使用できます。詳細については、「Fargate プロファイルのワイルドカード」を参照してください。

AWS Management Console

AWS Management Console でクラスターの Fargate プロファイルを作成するには

1. https://console.aws.amazon.com/eks/home#/clusters で Amazon EKS コンソールを開きます。
2. Fargate プロファイルを作成するクラスターを選択します。
3. [コンピューティング] タブを開きます。
4. [Fargate プロファイル] で、[Fargate プロファイルを追加] を選択します。
5. [Fargate プロファイルを設定] ページで、次の操作を行います。
    a. [名前] に Fargate プロファイルの名前を入力します。名前は一意である必要があります。

b. Pod execution role (ポッド実行ロール) で、Fargate プロファイルで使用する Pod 実行ロールを選択します。`eks-fargate-pods.amazonaws.com` サービスプリンシパルを持つ IAM ロールのみが表示されます。ここにロールが表示されない場合は、ロールを作成する必要があります。詳細については、「Amazon EKS Pod 実行 IAM ロール」を参照してください。

c. 選択した [サブネット] を必要に応じて変更します。

> **Note**
>
> Fargate で実行される Pods では、プライベートサブネットのみがサポートされます。

d. [タグ] では、オプションで Fargate プロファイルにタグを付けることができます。これらのタグは、Pods など、プロファイルに関連付けられた他のリソースには伝達されません。

e. [Next] を選択します。

6. [Pod の選択を設定] ページで、次の操作を行います。

a. [名前空間] に、Pods と照合する名前空間を入力します。

- **`kube-system`** または **`default`** など、特定の名前空間を使用して照合できます。
- 特定のワイルドカード (例: **`prod-*`**) を使用して、複数の名前空間 (例: `prod-deployment` および `prod-test`) と照合することができます。詳細については、「Fargate プロファイルのワイルドカード」を参照してください。

b. (オプション) セレクターに Kubernetes ラベルを追加します。特に、指定された名前空間内の Pods が一致する必要があるものにそれらを追加します。

- ラベル **`infrastructure: fargate`** をセレクターに追加して、`infrastructure: fargate` Kubernetes ラベルも持つ指定された名前空間の Pods のみがセレクターと一致するようにすることができます。
- 特定のワイルドカード (例: **`key?: value?`**) を使用して、複数の名前空間 (例: `keya: valuea` および `keyb: valueb`) と照合することができます。詳細については、「Fargate プロファイルのワイルドカード」を参照してください。

c. [Next] を選択します。

7. [確認と作成] ページで、Fargate プロファイルの情報を確認し、[作成] を選択します。

CoreDNS の更新

デフォルトでは、CoreDNS は Amazon EKS クラスターの Amazon EC2 インフラストラクチャで実行するように設定されています。クラスター内の Fargate でのみ Pods を実行する場合は、次のステップを実行します。

> **Note**
>
> `--fargate` オプションを使用して `eksctl` でクラスターを作成した場合は、次のステップにスキップできます。

1. 次のコマンドを使用して、CoreDNS の Fargate プロファイルを作成します。*`my-cluster`* をクラスター名に、*`111122223333`* を アカウント ID に、*`AmazonEKSFargatePodExecutionRole`* を Pod 実行ロール名に、そして *`0000000000000001`*、*`0000000000000002`*、*`0000000000000003`* をプライベートサブネットの ID に置き換えます。Pod 実行ロールがない場合は、最初に 作成する 必要があります。

   > **Important**
   >
   > ロール ARN に / 以外のパスを含めることはできません。例えば、ロールの名前が `development/apps/my-role` の場合、ロールの ARN を指定するときに `my-role` に変更する必要があります。ロール ARN の形式は `arn:aws:iam::111122223333:role/role-name` であることが必要です。

   ```
   aws eks create-fargate-profile \
       --fargate-profile-name coredns \
       --cluster-name my-cluster \
       --pod-execution-role-arn
    arn:aws:iam::111122223333:role/AmazonEKSFargatePodExecutionRole \
       --selectors namespace=kube-system,labels={k8s-app=kube-dns} \
       --subnets subnet-0000000000000001 subnet-0000000000000002
    subnet-0000000000000003
   ```

2. 次のコマンドを実行して、CoreDNS Pods から `eks.amazonaws.com/compute-type : ec2` アノテーションを削除します。

   ```
   kubectl patch deployment coredns \
   ```

```
    -n kube-system \
    --type json \
    -p='[{"op": "remove", "path": "/spec/template/metadata/annotations/
eks.amazonaws.com~1compute-type"}]'
```

次のステップ

- 以下のワークフローを使用すると、Fargate で実行するために既存のアプリケーションの移行を開始できます。
  1. アプリケーションの Kubernetes 名前空間と Kubernetes ラベルに一致する Fargate プロファイルを作成します。
  2. 既存のいずれかの Pods を削除および再作成し、Fargate でスケジューリングされるようにします。例えば、以下のコマンドでは、coredns デプロイのロールアウトがトリガーされます。名前空間とデプロイタイプを変更して、特定の Pods を更新できます。

```
kubectl rollout restart -n kube-system deployment coredns
```

- Amazon EKS でのアプリケーション負荷分散 をデプロイして、Fargate で実行されている Pods が Ingress オブジェクトを使用できるようにします。
- Vertical Pod Autoscaler を使用して Fargate Pods の CPU とメモリの適切な初期サイズを設定し、Horizontal Pod Autoscaler を使用してそれらの Pods をスケールできます。Vertical Pod Autoscaler で、より上位の CPU とメモリの組み合わせを持つ Fargate に Pods を自動的に再デプロイする場合は、Vertical Pod Autoscaler のモードを `Auto` または `Recreate` に設定します。これは、正しい機能を保証するためです。詳細については、「GitHub」で「Vertical Pod Autoscaler のドキュメント」を参照してください。
- これらの手順 に従って、アプリケーションをモニタリングするための AWS Distro for OpenTelemetry (ADOT) コレクターをセットアップします。

AWS Fargate プロファイル

> **Important**
>
> Amazon EKS を使用した AWS Fargate は、AWS GovCloud (米国東部) および AWS GovCloud (米国西部) ではご利用いただけません。

クラスター内の Fargate で Pods をスケジュールする前に、起動時にどの Pods が Fargate を使用するかを指定する Fargate プロファイルを少なくとも 1 つ定義する必要があります。

管理者として、Fargate プロファイルを使用してどの Pods が Fargate で実行されるかを宣言できます。これはプロファイルのセレクターを介して行うことができます。1 つのプロファイルに最大 5 つのセレクターを追加できます。各セレクターには名前空間が含まれている必要があります。セレクターにはラベルを含めることもできます。ラベルフィールドは、複数のオプションのキーと値のペアで構成されます。セレクターに一致するポッドは Fargate でスケジュールされます。ポッドは、セレクターで指定された名前空間とラベルを使用して照合されます。名前空間セレクターがラベルなしで定義されている場合、Amazon EKS は、プロファイルを使用して、その名前空間で実行されるすべての Pods を Fargate にスケジュールしようとします。スケジュールされる Pod が Fargate プロファイルのセレクターのいずれかと一致する場合、その Pod は Fargate でスケジュールされます。

Pod が複数の Fargate プロファイルと一致する場合、次の Kubernetes ラベルを Pod 仕様に追加することで、Pod がどのプロファイルを使用するかを指定することができます: `eks.amazonaws.com/fargate-profile: my-fargate-profile`。Pod を Fargate にスケジューリングするには、そのプロファイル内のセレクターに一致させる必要があります。Kubernetes アフィニティ/アンチアフィニティルールは適用されず、Amazon EKS Fargate Pods では不要です。

Fargate プロファイルを作成する際は、Pod 実行ロールを指定する必要があります。この実行ロールは、プロファイルを使用して Fargate インフラストラクチャで実行される Amazon EKS コンポーネントのためのものです。認証のためにクラスターの Kubernetes ロールベースのアクセスコントロール (RBAC) に追加されます。これにより、Fargate インフラストラクチャで実行されている `kubelet` が Amazon EKS クラスターに登録され、クラスター内でノードとして表示されるようになります。Pod 実行ロールは、Amazon ECR イメージリポジトリへの読み取りアクセスを許可するために、Fargate インフラストラクチャへの IAM アクセス許可も提供します。詳細については、「Amazon EKS Pod 実行 IAM ロール」を参照してください。

Fargate プロファイルは変更できません。ただし、新しい更新されたプロファイルを作成して既存のプロファイルを置き換え、その後元のプロファイルを削除することはできます。

> **Note**
>
> Fargate プロファイルを使用して実行中の Pods は、プロファイルが削除されると停止され、保留状態になります。

クラスターのいずれかの Fargate プロファイルが DELETING ステータスである場合は、Fargate プロファイルの削除が完了するのを待ってから、そのクラスターに他のプロファイルを作成する必要があります。

Amazon EKS と Fargate は、Fargate プロファイルで定義されている各サブネットに Pods を分散させます。ただし、分散が偏ってしまう場合があります。均等な分散が必要な場合は、2 つの Fargate プロファイルを使用してください。2 つのレプリカをデプロイし、ダウンタイムを発生させたくないシナリオでは、均等な分散が重要になります。各プロファイルにはサブネットを 1 つだけ含めることをお勧めします。

Fargate プロファイルのコンポーネント

Fargate プロファイルには、次のコンポーネントが含まれています。

ポッド実行ロール

クラスターが AWS Fargate で Pods を作成するとき、Fargate インフラストラクチャ上で実行されている kubelet は、ユーザーに代わって AWS API にコールを実行する必要があります。例えば、Amazon ECR からコンテナイメージを取得するためのコールを行う必要があります。Amazon EKS の Pod 実行ロールにより、これらを行うための IAM アクセス許可が付与されます。

Fargate プロファイルを作成する際は、Pods で使用する Pod 実行ロールを指定する必要があります。このロールは、認証のためにクラスターの Kubernetes ロールベースのアクセスコントロール (RBAC) に追加されます。これにより、Fargate インフラストラクチャで実行されている kubelet が Amazon EKS クラスターに登録され、クラスター内でノードとして表示されるようになります。詳細については、「Amazon EKS Pod 実行 IAM ロール」を参照してください。

サブネット

このプロファイルを使用する Pods を起動するサブネットの ID。現時点では、Fargate で実行されている Pods にはパブリック IP アドレスが割り当てられていません。したがって、このパラメータには、プライベートサブネット (インターネットゲートウェイへの直接ルートなし) のみが受け入れられます。

セレクター

この Fargate プロファイルを使用するために Pods に一致するセレクター。1 つの Fargate プロファイルに最大 5 つのセレクターを指定できます。セレクターには次のコンポーネントがあります:

- 名前空間 – セレクターの名前空間を指定する必要があります。セレクターはこの名前空間で作成された Pods のみを照合します。ただし、複数のセレクターを作成して複数の名前空間をターゲットにすることはできます。
- ラベル – オプションで、セレクターに一致する Kubernetes ラベルを指定できます。セレクターは、セレクターで指定されているすべてのラベルを持つ Pods のみに一致します。

Fargate プロファイルのワイルドカード

名前空間、ラベルキー、およびラベル値のセレクター基準では、Kubernetes で許可されている文字に加えて、***** および **?** の使用が許可されています。

- ***** は、文字なし、1 文字、または複数の文字を表します。例: **prod*** は `prod` および `prod-metrics` を表します。
- **?** は 1 文字を表します (例: **value?** は `valuea` を表します)。しかし、`value` および `value-a` を表すことはできません。? は厳密に 1 文字だけを表すためです。

これらのワイルドカード文字は、任意の位置や組み合わせで使用できます (例: **prod***、***dev**、および **frontend*?**)。正規表現など、他のワイルドカードやパターンマッチング形式はサポートされていません。

Pod 仕様内の名前空間とラベルに一致するプロファイルが複数ある場合、Fargate はプロファイル名でソートされた英数字に基づいてプロファイルを選択します。例えば、(`beta-workload` という名前の) プロファイル A と (`prod-workload` という名前の) プロファイル B の両方に、起動する Pods に一致するセレクターがある場合、Fargate は Pods にプロファイル A (`beta-workload`) を選択します。Pods には、Pods のプロファイル A にラベルが付いています。(例: `eks.amazonaws.com/fargate-profile=beta-workload`)

ワイルドカードを使用する新しいプロファイルに既存の Fargate Pods を移行するには、次の 2 つの方法があります。

- 一致するセレクターを持つ新しいプロファイルを作成し、古いプロファイルを削除します。古いプロファイルでラベル付けされたポッドは、一致する新しいプロファイルに再スケジュールされます。
- ワークロードを移行したいものの、各 Fargate Pod にどの Fargate ラベルが付いているかわからない場合は、次の方法を使用します。新しいプロファイルを作成し、同じクラスター上のプロファイルの中でアルファベット順で最初に並べられる名前を付けます。次に、新しいプロファイルに移行する必要がある Fargate Pods をリサイクルします。

Fargate プロファイルの作成

このトピックでは、Fargate プロファイルを作成する方法について説明します。また、Fargate プロファイルに使用する Pod 実行ロールを作成しておく必要があります。詳細については、「Amazon EKS Pod 実行 IAM ロール」を参照してください。Fargate で実行されている Pods は、AWS のサービス への NAT ゲートウェイアクセスを持つプライベートサブネットでのみサポートされます。インターネットゲートウェイへの直接ルートはサポートされません。これは、クラスターの VPC がプライベートサブネットを使用できるようにするためです。プロファイルを作成するには eksctl または AWS Management Console を使用します。

この手順には、eksctl バージョン 0.183.0 以降が必要です。お使いのバージョンは、以下のコマンドを使用して確認できます。

```
eksctl version
```

eksctl のインストールまたはアップグレードの手順については、eksctl ドキュメントの「インストール」を参照してください。

eksctl

eksctl で、Fargate プロファイルを作成するには

以下の eksctl コマンドで Fargate プロファイルを作成し、すべての *example value* を自分の値に置き換えます。名前空間を指定する必要があります。ただし、--labels オプションは必須ではありません。

```
eksctl create fargateprofile \
    --cluster my-cluster \
    --name my-fargate-profile \
    --namespace my-kubernetes-namespace \
    --labels key=value
```

my-kubernetes-namespace および *key=value* ラベルには、特定のワイルドカードを使用できます。詳細については、「Fargate プロファイルのワイルドカード」を参照してください。

AWS Management Console

AWS Management Console でクラスターの Fargate プロファイルを作成するには

1. https://console.aws.amazon.com/eks/home#/clusters で Amazon EKS コンソールを開きます。

2. Fargate プロファイルを作成するクラスターを選択します。
3. [コンピューティング] タブを開きます。
4. [Fargate プロファイル] で、[Fargate プロファイルを追加] を選択します。
5. [Fargate プロファイルを設定] ページで、次の操作を行います。

   a. [名前] に、Fargate プロファイルの一意の名前 (***my-profile*** など) を入力します。

   b. Pod execution role (ポッド実行ロール) で、Fargate プロファイルで使用する Pod 実行ロールを選択します。`eks-fargate-pods.amazonaws.com` サービスプリンシパルを持つ IAM ロールのみが表示されます。ここにロールが表示されない場合は、ロールを作成する必要があります。詳細については、「Amazon EKS Pod 実行 IAM ロール」を参照してください。

   c. 選択した [サブネット] を必要に応じて変更します。

   > **Note**
   >
   > Fargate で実行される Pods では、プライベートサブネットのみがサポートされます。

   d. [タグ] では、オプションで Fargate プロファイルにタグを付けることができます。これらのタグは、Pods など、プロファイルに関連付けられた他のリソースには伝達されません。

   e. [Next] を選択します。
6. [Pod の選択を設定] ページで、次の操作を行います。

   a. [名前空間] に、Pods と照合する名前空間を入力します。

   - **`kube-system`** または **`default`** など、特定の名前空間を使用して照合できます。
   - 特定のワイルドカード (例: **`prod-*`**) を使用して、複数の名前空間 (例: `prod-deployment` および `prod-test`) と照合することができます。詳細については、「Fargate プロファイルのワイルドカード」を参照してください。

   b. (オプション) セレクターに Kubernetes ラベルを追加します。特に、指定された名前空間内の Pods が一致する必要があるものにそれらを追加します。

   - ラベル **`infrastructure: fargate`** をセレクターに追加して、`infrastructure: fargate` Kubernetes ラベルも持つ指定された名前空間の Pods のみがセレクターと一致するようにすることができます。

- 特定のワイルドカード (例: **key?: value?**) を使用して、複数の名前空間 (例: keya: valuea および keyb: valueb) と照合することができます。詳細については、「Fargate プロファイルのワイルドカード」を参照してください。

c. [Next] を選択します。

7. [確認と作成] ページで、Fargate プロファイルの情報を確認し、[作成] を選択します。

Fargate プロファイルの削除

このトピックでは、Fargate プロファイルを削除する方法について説明します。

Fargate プロファイルを削除すると、そのプロファイルで Fargate にスケジューリングされていた Pods はすべて削除されます。これらの Pods が別の Fargate プロファイルと一致する場合、それらのポッドはそのプロファイルで Fargate にスケジューリングされます。Fargate プロファイルがどのプロファイルとも一致しなくなった場合は、Fargate へのスケジューリングは行われず、保留中のままになる可能性があります。

一度にクラスター内の 1 つの Fargate プロファイルのみが、DELETING ステータスになることができます。そのクラスター内の他のプロファイルを削除する前に、Fargate プロファイルの削除が完了するまでお待ちください。

プロファイルを削除するには eksctl 、AWS Management Console、またはAWS CLI を使用します。ロールの削除に使用するツールの名前が付いているタブを選択します。

eksctl

eksctl で、Fargate プロファイルを削除するには

クラスターからプロファイルを削除するには、次のコマンドを使用します。*example value* をすべて自分の値に置き換えてください。

```
eksctl delete fargateprofile  --name my-profile --cluster my-cluster
```

AWS Management Console

AWS Management Console マネジメントコンソールを使用してクラスターから Fargate プロファイルを削除するには

1. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。

2. 左のナビゲーションペインで [クラスター] を選択します。クラスターの一覧で Fargate プロファイルを削除するクラスターを選択します。
3. [コンピューティング] タブを開きます。
4. 削除する Fargate プロファイルを選択し、[削除] を選択します。
5. [Fargate プロファイルの削除] ページで、プロファイルの名前を入力し、[削除] を選択します。

AWS CLI

AWS CLI で、Fargate プロファイルを削除するには

クラスターからプロファイルを削除するには、次のコマンドを使用します。*example value* をすべて自分の値に置き換えてください。

```
aws eks delete-fargate-profile --fargate-profile-name my-profile --cluster-name my-cluster
```

Fargate Pod の設定

> **Important**
>
> Amazon EKS を使用した AWS Fargate は、AWS GovCloud (米国東部) および AWS GovCloud (米国西部) ではご利用いただけません。

このセクションでは、AWS Fargate で Kubernetes Pods を実行する場合の固有な Pod 設定の詳細について説明します。

Pod CPU とメモリ

Kubernetes を使用すると、Pod の各コンテナに割り当てられるリクエスト、最小 vCPU 量、メモリリソースを定義できます。Pods が Kubernetes によってスケジュールされ、少なくとも各 Pod に対して要求されたリソースが、コンピューティングリソースで使用可能な状態を確保します。詳細については、Kubernetes ドキュメントの「コンテナのリソース管理」を参照してください。

> **Note**
>
> Amazon EKS Fargate はノードごとに 1 つの Pod しか実行しないため、リソースが少ない場合に Pods を削除するシナリオは発生しません。すべての Amazon EKS Fargate Pods は保証された優先度で実行されるため、リクエストされた CPU とメモリは、すべてのコンテナの制限に等しくなければなりません。詳細については、Kubernetes ドキュメントの「Pods にサービスの品質を設定する」を参照してください。

Pods が Fargate にスケジュールされている場合、Pod 仕様内の vCPU とメモリの予約によって、Pod にプロビジョニングする CPU とメモリの量が決まります。

- Init コンテナからの最大リクエストは、Init リクエスト vCPU およびメモリ要件を決定するために使用されます。
- 実行時間の長いコンテナのリクエストは、実行時間の長いリクエスト vCPU とメモリ要件を決定するために合計されます。
- 前記の 2 つの値のうち大きい方が、Pod 用に使用する vCPU とメモリリクエストに対して選択されます。
- Fargate は、必要な Kubernetes コンポーネント (`kubelet`、`kube-proxy`、および `containerd`) の各 Pod のメモリ予約に 256 MB を追加します。

Fargate では、Pods の実行に必要なリソースを常に確保するために、vCPU とメモリのリクエストの合計に最も近い、次に示すコンピューティング設定に切り上げられます。

vCPU とメモリの組み合わせを指定しない場合は、使用可能な最小の組み合わせ (.25 vCPU と 0.5 GB のメモリ) が使用されます。

次の表は、Fargate で実行されている Pods で使用可能な vCPU とメモリの組み合わせを示しています。

vCPU 値	メモリの値
.25 vCPU	0.5 GB、1 GB、2 GB
.5 vCPU	1 GB、2 GB、3 GB、4 GB

vCPU 値	メモリの値
1 vCPU	2 GB、3 GB、4 GB、5 GB、6 GB、7 GB、8 GB
2 vCPU	4 GB ~ 16 GB (1 GB のインクリメント)
4 vCPU	8 GB ~ 30 GB (1 GB のインクリメント)
8 vCPU	16 GB ~ 60 GB (4 GB のインクリメント)
16 vCPU	32 GB ~ 120 GB (8 GB のインクリメント)

Kubernetes コンポーネント用に予約されている追加メモリにより、要求よりも多くの vCPU で Fargate タスクがプロビジョニングされる可能性があります。例えば、1 つの vCPU と 8 GB のメモリを要求すると、メモリ要求に 256 MB が追加され、2 つの vCPU と 9 GB のメモリの Fargate タスクがプロビジョニングされます。これは、1 つの vCPU と 9 GB のメモリのタスクがないためです。

Fargate で実行されている Pod のサイズと、Kubernetes が `kubectl get nodes` でレポートするノードサイズに相関はありません。レポートされるノードサイズは、多くの場合 Pod の容量よりも大きくなります。次のコマンドを使用して、Pod 容量を確認できます。*`default`* を Pod の名前空間に、*`pod-name`* を Pod の名前に置き換えます。

```
kubectl describe pod --namespace default pod-name
```

出力例は次のとおりです。

```
[...]
annotations:
    CapacityProvisioned: 0.25vCPU 0.5GB
[...]
```

`CapacityProvisioned` アノテーションは、強制された Pod 容量を表し、Fargate で実行されている Pod のコストを決定します。コンピューティング設定の料金情報については、「AWS Fargate の料金」を参照してください。

Fargate ストレージ

Fargate で実行されている Pod が、Amazon EFS ファイルシステムを自動的にマウントします。Fargate ノードでは動的永続ボリュームプロビジョニングを使用できませんが、静的プロビジョニングを使用することはできます。詳細については、GitHub の「Amazon EFS CSI ドライバー」を参照してください。

プロビジョニングすると、Fargate で実行されている各 Pod に対して、デフォルトで 20 GiB のエフェメラルストレージが割り当てられます。このタイプのストレージは、Pod の停止後に削除されます。Fargate に起動される新しい Pods では、エフェメラルストレージボリュームの暗号化がデフォルトで有効になっています。エフェメラル Pod ストレージは、AWS Fargate で管理されるキーを使用して AES-256 暗号化アルゴリズムで暗号化されます。

> **Note**
>
> Fargate で実行される Amazon EKS Pods のデフォルトで使用可能なストレージは、20 GiB 未満です。これは、一部のスペースが `kubelet` と Pod 内に読み込まれるその他の Kubernetes モジュールによって使用されるためです。

エフェメラルストレージの総量は、最大 175 GiB まで増やすことができます。Kubernetes でサイズを設定するには、Pod 内の各コンテナに対して、`ephemeral-storage` リソースのリクエストを指定します。Kubernetes が Pods をスケジュールすると、各 Pod に対するリソース要求の合計が Fargate タスクの容量を下回ることが保証されます。詳細については、Kubernetes ドキュメントの「Pods とコンテナのリソース管理」を参照してください。

Amazon EKS Fargate は、システム使用目的でリクエストされた数よりも大きなエフェメラルストレージをプロビジョニングします。たとえば、100 GiB のリクエストでは、115 GiB のエフェメラルストレージの Fargate タスクがプロビジョニングされます。

Fargate OS のパッチ適用

> **Important**
>
> Amazon EKS を使用した AWS Fargate は、AWS GovCloud (米国東部) および AWS GovCloud (米国西部) ではご利用いただけません。

Amazon EKS は定期的に AWS Fargate ノードの OS にパッチを適用し、ノードを安全に保ちます。パッチ適用プロセスの一環として、ノードをリサイクルして OS パッチをインストールします。更新は、サービスへの影響を最小限に抑える方法で試行されます。ただし、Pods のエビクションが正常に行われない場合は、ポッドを削除する必要がある場合があります。以下に示しているのは、中断の可能性を最小限に抑えるために実行できるアクションです。

- 適切な Pod の中断予算 (PDB) を設定して、同時にダウンする Pods の数をコントロールします。
- Pods が削除される前に、失敗したエビクションに対応する Amazon EventBridge ルールを作成します。
- AWS User Notifications で通知設定を作成します。

Amazon EKS は、Kubernetes コミュニティと緊密に協力して、できるだけ早く、バグ修正とセキュリティパッチが入手可能になるようにしています。すべての Fargate Pods は最新の Kubernetes パッチバージョンで起動されます。このバージョンは、Amazon EKS からクラスターの Kubernetes バージョンとして入手できます。以前のパッチバージョンの Pod をお持ちの場合、Amazon EKS はそれをリサイクルして最新バージョンに更新することがあります。これにより、Pods に最新のセキュリティアップデートが確実に装備されます。そのため、重大な「Common Vulnerabilities and Exposures」(共通脆弱性識別子) (CVE) 問題があっても、最新の状態に保たれてセキュリティリスクが軽減されています。

Pods がリサイクルされるときに一度にダウンする Pods の数を制限するために、Pod の中断予算 (PDB) を設定できます。PDB を使用して、更新の実行を許可しながら、各アプリケーションの要件に基づいて最小限使用可能なポッドを定義できます。詳細については、「Kubernetes ドキュメント」の「Specifying a Disruption Budget for your Application」(アプリケーションの中断予算を指定する) を参照してください。

Amazon EKS は エビクションAPI を使用して、アプリケーションに設定した PDB を尊重しながら Pod を安全に排出します。ポッドのエビクションは、影響を最小限に抑えるために、アベイラビリティーゾーンごとに行われます。エビクションが成功した場合、新しい Pod は最新のパッチを取得し、それ以上のアクションは必要ありません。

Pod のエビクションが失敗すると、Amazon EKS はエビクションに失敗した Pods の詳細を含むイベントをアカウントに送信します。スケジュールされた終了時刻より前にメッセージに対応できます。具体的な時刻は、パッチの緊急性によって異なります。その時刻になると、Amazon EKS は Pods のエビクションを再び試行します。ただし、今回は、エビクションが失敗しても新しいイベントは送信されません。エビクションが再び失敗すると、新しい Pods に最新のパッチが適用できるように、既存の Pods が定期的に削除されます。

Pod のエビクションが失敗したときに受信されるイベントのサンプルを次に示します。これには、クラスター、Pod 名、Pod 名前空間、Fargate プロファイル、およびスケジュールされた終了時刻に関する詳細が含まれます。

```
{
    "version": "0",
    "id": "12345678-90ab-cdef-0123-4567890abcde",
    "detail-type": "EKS Fargate Pod Scheduled Termination",
    "source": "aws.eks",
    "account": "111122223333",
    "time": "2021-06-27T12:52:44Z",
    "region": "region-code",
    "resources": [
        "default/my-database-deployment"
    ],
    "detail": {
        "clusterName": "my-cluster",
        "fargateProfileName": "my-fargate-profile",
        "podName": "my-pod-name",
        "podNamespace": "default",
        "evictErrorMessage": "Cannot evict pod as it would violate the pod's disruption
 budget",
        "scheduledTerminationTime": "2021-06-30T12:52:44.832Z[UTC]"
    }
}
```

さらに、複数の PDB が Pod に関連付けられていると、エビクション失敗イベントが発生する可能性があります。このイベントは次のエラーメッセージを返します。

```
"evictErrorMessage": "This pod has multiple PodDisruptionBudget, which the eviction
 subresource does not support",
```

このイベントに基づいて、目的のアクションを作成できます。例えば、Pod の中断予算 (PDB) を調整して、Pods のエビクションの方法をコントロールできます。具体的には、利用可能な Pods の目標パーセンテージを指定する PDB から始めたとします。アップグレード中に Pods が強制終了される前に、PDB を異なる割合の Pods に調整できます。このイベントを受信するには、クラスターが属する AWS アカウント および AWS リージョン で Amazon EventBridge ルールを作成する必要があります。ルールでは、次のカスタムパターンを使用する必要があります。EventBridge ルールの作成の詳細については、Amazon EventBridge ユーザーガイドの「イベントに反応する Amazon EventBridge ルールの作成」を参照してください 。

```
{
  "source": ["aws.eks"],
  "detail-type": ["EKS Fargate Pod Scheduled Termination"]
}
```

キャプチャするためのイベントに適切なターゲットを設定できます。詳細については、「Amazon EventBridge ユーザーガイド」の「Amazon EventBridge ターゲット」を参照してください。AWS User Notifications で通知設定を作成することもできます。AWS Management Console を使用して通知を作成する場合は、[イベントルール] で、[AWS のサービス 名] に [Elastic Kubernetes Service (EKS)] を選択し、[イベントタイプ] に [EKS Fargate Pod のスケジュールされた終了時刻] を選択します。詳細については、「AWS User Notifications ユーザーガイド」の「AWS User Notifications の開始方法」を参照してください。

Fargate メトリクス

> **Important**
>
> Amazon EKS を使用した AWS Fargate は、AWS GovCloud (米国東部) および AWS GovCloud (米国西部) ではご利用いただけません。

AWS Fargate のシステムメトリクスおよび CloudWatch 使用状況メトリクスを収集できます。

アプリケーションメトリクス

Amazon EKS や AWS Fargate で実行されているアプリケーションをの場合、AWS Distro for OpenTelemetry (ADOT) を使用できます。ADOT では、システムメトリクスを収集し、CloudWatch コンテナインサイトダッシュボードに送信できます。Fargate で実行されているアプリケーションの ADOT の使用を開始するには、ADOT ドキュメントの「AWS Distro for OpenTelemetry ディストリビューションで CloudWatch Container Insights を使用する」を参照してください。

使用状況メトリクス

CloudWatch 使用状況メトリクスを使用して、アカウントのリソースの使用状況を把握できます。これらのメトリクスを使用して、CloudWatch グラフやダッシュボードで現在のサービスの使用状況を可視化できます。

AWS Fargate 使用状況メトリクスは、AWS のサービスクォータに対応しています。使用量がサービスクォータに近づいたときに警告するアラームを設定することもできます。Fargate のサービスのクォータの詳細については、「Amazon EKS Service Quotas」を参照してください。

AWS Fargate は、AWS/Usage 名前空間に以下のメトリクスを公開します。

メトリクス	説明
ResourceCount	アカウントで実行されている指定されたリソースの合計数。リソースは、メトリクスに関連付けられたディメンションによって定義されます。

以下のディメンションは、AWS Fargate によって発行される使用状況メトリクスを絞り込むために使用されます。

ディメンション	説明
Service	リソースを含む AWS のサービスの名前。AWS Fargate 使用状況メトリクスの場合、このディメンションの値は Fargate です。
Type	報告されるエンティティのタイプ。現在、AWS Fargate 使用状況メトリクスの有効な値は Resource のみです。
Resource	実行中のリソースのタイプ。 現在、AWS Fargate は Fargate のオンデマンド 使用状況に関する情報を返します。Fargate のオンデマンド使用状況のリソース値は OnDemand です。 **Note** Fargate のオンデマンド使用状況は、Fargate を使用した Amazon EKS Pods、Fargate 起動タイプを使用した Amazon ECS タスク、および FARGATE キャパシティプロバイダーを使用した Amazon ECS タスクの組み合わせです。

ディメンション	説明
Class	追跡されているリソースのクラス。現在、AWS Fargate はクラスディメンションを使用していません。

Fargate のリソース使用状況メトリクスをモニタリングするための CloudWatch アラームの作成

AWS Fargate は、Fargate オンデマンドリソース使用状況の AWS のサービスクォータに対応する CloudWatch 使用状況メトリクスを提供します。Service Quotas コンソールでは、使用状況をグラフで可視化できます。また、使用量がサービスクォータに近づいたときに警告するアラームも設定できます。詳細については、「Fargate メトリクス」を参照してください。

以下の手順を使用して、Fargate リソース使用状況メトリクスに基づく CloudWatch アラームを作成します。

Fargate 使用量クォータに基づいてアラームを作成するには (AWS Management Console)

1. https://console.aws.amazon.com/servicequotas/ で Service Quotas コンソールを開きます。
2. 左側のナビゲーションペインで [AWS services] (のサービス) を選択します。
3. [AWS サービス] リストから、[AWS Fargate]]] を探して選択します。
4. [Service quotas] (サービスクォータ) リストで、アラームを作成する Fargate 使用量クォータを選択します。
5. Amazon CloudWatch アラームのセクションで [Create] (作成) を選択します。
6. [アラームのしきい値] で、適用されたクォータ値からアラーム値として設定する値の割合を選択します。
7. [アラーム名] にアラームの名前を入力し、[Create (作成)] を選択します。

Fargate ログ記録

> **Important**
>
> Amazon EKS を使用した AWS Fargate は、AWS GovCloud (米国東部) および AWS GovCloud (米国西部) ではご利用いただけません。

Fargate の Amazon EKS では、Fluent Bit をベースにした組み込みのログルーターが利用できます。Fluent Bit コンテナをサイドカーとして明示的に実行する必要はなく、この実行は Amazon によって行われます。必要となるのは、ログルーターの設定だけです。設定は専用の `ConfigMap` を介して行い、その際は以下の基準を満たす必要があります。

- 名前のついた `aws-logging`
- `aws-observability` と呼ばれる専用の名前空間での作成
- 5,300 文字以内にしてください。

`ConfigMap` を作成すると、Fargate の Amazon EKS は自動的にそれを検出しログルーターの設定を行います。Fargate は、Fluent Bit の AWS のバージョンを使用しています。これは、AWS によって管理される Fluent Bit の上流対応のディストリビューションです。詳細については、GitHub の「AWS 用の Fluent Bit」を参照してください。

ログルーターを使用すると、AWS のさまざまなサービスをログの分析と保管に使用できます。Fargate からは、Amazon CloudWatch、Amazon OpenSearch Service に対し直接ログをストリーミングできます。また、Amazon Data Firehose 経由で Amazon S3、Amazon Kinesis Data Streams、およびパートナーツールなどの送信先にログをストリーミングすることも可能です。

前提条件

- Fargate Pods をデプロイする既存の Kubernetes 名前空間を指定する既存の Fargate プロファイルです。詳細については、「クラスターの Fargate プロファイルを作成する」を参照してください。
- 既存の Fargate Pod の実行ロールです。詳細については、「Fargate Pod 実行ロールを作成する」を参照してください。

ログルーターの設定

ログルーターを設定するには

以下のステップでは、すべての *`example value`* を独自の値に置き換えます。

1. `aws-observability` という名前の専用の Kubernetes 名前空間を作成します。
    a. 次の内容をコンピュータ上の *`aws-observability-namespace`*`.yaml` という名前のファイルに保存します。name の値は `aws-observability` である必要があり、`aws-observability: enabled` ラベルが必須です。

```
kind: Namespace
apiVersion: v1
metadata:
  name: aws-observability
  labels:
    aws-observability: enabled
```

b. 名前空間を作成します。

```
kubectl apply -f aws-observability-namespace.yaml
```

2. データ値 Fluent Conf を使用して ConfigMap を作成し、コンテナログを送信先に送ります。Fluent Conf とは Fluent Bit であり、コンテナログを任意のログ送信先にルーティングするために使用される、高速で軽量なログプロセッサ設定言語です。詳細については、Fluent Bit ドキュメントの「Configuration File」(設定ファイル) を参照してください。

> **Important**
>
> 典型的な Fluent Conf に含まれている主なセクションは、ServiceInput、Filter および Output です。ただし、Fargate のログルータでは、以下だけを受け入れます。
>
> - Filter および Output セクション。
> - Parser セクション。
>
> 他のセクションを提供した場合、拒否されます。

Fargate ログルーターは Service および Input セクションを管理します。次の Input セクションがありますが、これらは変更できず、ConfigMap には必要ありません。ただし、メモリバッファー制限やログに適用されるタグなどの洞察は得られます。

```
[INPUT]
    Name tail
    Buffer_Max_Size 66KB
    DB /var/log/flb_kube.db
    Mem_Buf_Limit 45MB
    Path /var/log/containers/*.log
    Read_From_Head On
    Refresh_Interval 10
```

```
    Rotate_Wait 30
    Skip_Long_Lines On
    Tag kube.*
```

ConfigMap の作成時は、以下の (Fargate がフィールドの検証に使用する) ルールを考慮に入れます。

- [FILTER]、[OUTPUT]および [PARSER] は、それぞれが対応するキーにより指定する必要があります。例: [FILTER] が filters.conf の下にある必要があります。filters.conf には、複数の [FILTER] を含められます。また、[OUTPUT] および [PARSER] セクションは、それぞれと対応するキーの下に置く必要があります。複数の [OUTPUT] セクションを指定することで、ログを異なる送信先に同時にルーティングできます。
- Fargate は各セクションに必要なキーを検証します。Name および match がそれぞれの [FILTER] および [OUTPUT] に必要です。Name および format がそれぞれの [PARSER] に必要です。キーの大文字と小文字は区別されません。
- ${ENV_VAR} などの環境変数は ConfigMap では許可されていません。
- インデントは、それぞれの filters.conf、output.conf、および parsers.conf の中で、ディレクティブまたはキーと値のペアで同じである必要があります。キーと値のペアは、ディレクティブよりも深いインデントにする必要があります。
- Fargate は、サポートされている次のフィルターに対して検証します。grep、parser、record_modifier、rewrite_tag、throttle、nest、modify、および kubernetes。
- Fargate は、サポートされている次の出力に対して検証します。es、firehose、kinesis_firehose、cloudwatch、cloudwatch_logs、および kinesis。
- ログ記録を有効にするには、サポートされている Output プラグインが少なくとも 1 つ ConfigMap にあることが必要です。Filter および Parser は、ログ記録を有効にするために必要ありません。

また、希望の設定を使用して Amazon EC2 で Fluent Bit を実行し、検証によって発生する問題をトラブルシューティングすることもできます。以下のいずれかの例に従って、ConfigMap を作成します。

> **Important**
>
> Amazon EKS Fargate のログ記録では、ConfigMaps での動的設定をサポートしていません。ConfigMaps に対する変更はすべて、新しい Pods に対してのみ適用されます。既存の Pods に変更は適用されません。

例を使用して、必要なログ送信先用に ConfigMap を作成します。

> **Note**
>
> また、Amazon Kinesis Data Streams をログの宛先として使用できます。Kinesis Data Streams を使用する場合は、ポッド実行ロールに kinesis:PutRecords 権限が付与されていることを確認してください。詳細については、Fluent Bit: 公式マニュアルの Amazon Kinesis Data Streams の「アクセス許可」を参照してください。

CloudWatch

CloudWatch の **ConfigMap** を作成するには

CloudWatch を使用する場合、次の 2 つの出力オプションがあります。

- C で記述された出力プラグイン
- Golang で記述された出力プラグイン

次の例は、cloudwatch_logs プラグインを使用して CloudWatch にログを送信する方法を示しています。

1. 次の内容を *aws-logging-cloudwatch-configmap*.yaml という名前のファイルに保存します。*region-code* をクラスターのある AWS リージョン に置き換えます。[OUTPUT] のパラメータは必須です。

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
```

```
data:
  flb_log_cw: "false"  # Set to true to ship Fluent Bit process logs to
 CloudWatch.
  filters.conf: |
    [FILTER]
        Name parser
        Match *
        Key_name log
        Parser crio
    [FILTER]
        Name kubernetes
        Match kube.*
        Merge_Log On
        Keep_Log Off
        Buffer_Size 0
        Kube_Meta_Cache_TTL 300s
  output.conf: |
    [OUTPUT]
        Name cloudwatch_logs
        Match   kube.*
        region region-code
        log_group_name my-logs
        log_stream_prefix from-fluent-bit-
        log_retention_days 60
        auto_create_group true
  parsers.conf: |
    [PARSER]
        Name crio
        Format Regex
        Regex ^(?<time>[^ ]+) (?<stream>stdout|stderr) (?<logtag>P|F) (?
<log>.*)$
        Time_Key    time
        Time_Format %Y-%m-%dT%H:%M:%S.%L%z
```

2. マニフェストをクラスターに適用します。

```
kubectl apply -f aws-logging-cloudwatch-configmap.yaml
```

3. CloudWatch IAM ポリシーをコンピュータにダウンロードします。GitHub でポリシーの表示をすることもできます。

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-eks-fluent-
logging-examples/mainline/examples/fargate/cloudwatchlogs/permissions.json
```

Amazon OpenSearch Service

Amazon OpenSearch Service の **ConfigMap** を作成するには

Amazon OpenSearch Service にログを送信するには、es 出力を使用します。これは C で書かれたプラグインです。次の例は、プラグインを使用して OpenSearch にログを送信する方法を示しています。

1. 次の内容を *aws-logging-opensearch-configmap*.yaml という名前のファイルに保存します。*example value* をすべて自分の値に置き換えてください。

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
data:
  output.conf: |
    [OUTPUT]
      Name  es
      Match *
      Host  search-example-gjxdcilagiprbglqn42jsty66y.region-
code.es.amazonaws.com
      Port  443
      Index example
      Type  example_type
      AWS_Auth On
      AWS_Region region-code
      tls   On
```

2. マニフェストをクラスターに適用します。

```
kubectl apply -f aws-logging-opensearch-configmap.yaml
```

3. OpenSearch IAM ポリシーをコンピュータにダウンロードします。GitHub でポリシーの表示をすることもできます。

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-eks-
fluent-logging-examples/mainline/examples/fargate/amazon-elasticsearch/
permissions.json
```

OpenSearch Dashboards のアクセスコントロールが適切に設定されていることを確認します。OpenSearch Dashboards の `all_access role` には、Fargate Pod の実行ロールと IAM ロールがマッピングされている必要があります。同様のマッピングが、`security_manager` ロールに対しても必要です。以前のマッピングを追加するには、`Menu`、`Security`、`Roles` の順にクリックした後、それぞれに対応するロールを選択します。詳細については、「CloudWatch Logs が Amazon ES ドメインにストリーミングされるようにトラブルシューティングする方法を教えてください。」を参照してください。

Firehose

Firehose 用の **ConfigMap** を作成するには

Firehose にログを送信する場合、次の二つの出力オプションがあります。

- `kinesis_firehose` — C で記述された出力プラグイン。
- `firehose` — Golang で記述された出力プラグイン。

次の例は、Firehose にログを送信するために `kinesis_firehose` プラグインを使用する方法を示しています。

1. 次の内容を *aws-logging-firehose-configmap*.yaml という名前のファイルに保存します。*region-code* をクラスターのある AWS リージョン に置き換えます。

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
data:
  output.conf: |
    [OUTPUT]
     Name  kinesis_firehose
     Match *
     region region-code
     delivery_stream my-stream-firehose
```

2. マニフェストをクラスターに適用します。

```
kubectl apply -f aws-logging-firehose-configmap.yaml
```

3. Firehose IAM ポリシーをコンピュータにダウンロードします。GitHub でポリシーの表示をすることもできます。

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-eks-fluent-logging-examples/mainline/examples/fargate/kinesis-firehose/permissions.json
```

3. 前のステップでダウンロードしたポリシー ファイルを使用して、IAM ポリシーを作成します。

```
aws iam create-policy --policy-name eks-fargate-logging-policy --policy-document file://permissions.json
```

4. 次のコマンドを使用して、Fargate プロファイルに指定されたポッド実行ロールに IAM ポリシーを添付します。*111122223333* をアカウントID に置き換えます。*AmazonEKSFargatePodExecutionRole* を Pod の実行ロールに置き換えます (詳細については、「Fargate Pod 実行ロールを作成する」を参照してください)。

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::111122223333:policy/eks-fargate-logging-policy \
  --role-name AmazonEKSFargatePodExecutionRole
```

Kubernetes フィルターのサポート

この機能には、最低限、次の Kubernetes バージョンとプラットフォームレベル以上が必要です。

Kubernetes バージョン	プラットフォームレベル
1.23 以降	eks.1

Fluent Bit Kubernetes フィルターを使用すると、ログファイルに Kubernetes メタデータを追加できます。フィルターの詳細については、「Fluent Bit ドキュメント」の「Kubernetes」を参照してください。API サーバーエンドポイントを使用して、フィルターを適用できます。

```
filters.conf: |
    [FILTER]
        Name             kubernetes
        Match            kube.*
```

```
        Merge_Log           On
        Buffer_Size         0
        Kube_Meta_Cache_TTL 300s
```

Important

- Kube_URL、Kube_CA_File、Kube_Token_Command、および Kube_Token_File はサービス所有の設定パラメータであるため、指定しないでください。Amazon EKS Fargate がこれらの値を設定します。
- Kube_Meta_Cache_TTL は、Fluent Bit が最新のメタデータを取得するために API サーバーと通信するまで待機する時間です。Kube_Meta_Cache_TTL が指定されていない場合、Amazon EKS Fargate は API サーバーの負荷を軽減するためにデフォルト値である 30 分を追加します。

Fluent Bit プロセスログをアカウントに送付するには

オプションで、Fluent Bit プロセスログを、次の ConfigMap を使用して Amazon CloudWatch に送付できます。Fluent Bit プロセスログを CloudWatch に送信するには、追加のログの取り込みおよびストレージのコストがかかります。*region-code* をクラスターのある AWS リージョン に置き換えます。

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
  labels:
data:
  # Configuration files: server, input, filters and output
  # ======================================================
  flb_log_cw: "true"  # Ships Fluent Bit process logs to CloudWatch.

  output.conf: |
    [OUTPUT]
        Name cloudwatch
        Match kube.*
        region region-code
        log_group_name fluent-bit-cloudwatch
        log_stream_prefix from-fluent-bit-
```

```
        auto_create_group true
```

ログは CloudWatch でクラスターが存在する AWS リージョン にあります。ロググループ名は *my-cluster*-fluent-bit-logs で、Fluent Bit のログストリーム名は fluent-bit-*podname*-*pod-namespace* です。

> **Note**
>
> - プロセスログは、Fluent Bit プロセスが正常に開始された場合にのみ送付されます。Fluent Bit の起動中に障害が発生すると、プロセスログを取得できません。プロセスログは CloudWatch にのみ送付できます。
> - プロセスログのアカウントへの送付をデバッグするには、前に使用した ConfigMap を適用して、プロセスログを取得します。Fluent Bit が起動に失敗するのは通常、開始時に Fluent Bit が ConfigMap を解析しないか、受け付けないためです。

Fluent Bit プロセスログの送信を停止するには

Fluent Bit プロセスログを CloudWatch に送信するには、追加のログの取り込みおよびストレージのコストがかかります。既存の ConfigMap 設定でプロセスログを除外するには、次のステップを実行します。

1. Fargate ログ記録を有効にした後、Amazon EKS クラスターの Fluent Bit プロセスログ用に自動的に作成された CloudWatch ロググループを見つけます。これはフォーマット {cluster_name}-fluent-bit-logs に従います。
2. CloudWatch ロググループ内の各 Pod's のプロセスログ用に作成された既存の CloudWatch ログストリームを削除します。
3. ConfigMap を編集して flb_log_cw: "false" を設定します。
4. クラスター内の既存の Pods を再起動します。

アプリケーションをテストする

1. サンプル Pod をデプロイします。
   a. 次の内容をコンピュータ上の *sample-app*.yaml という名前のファイルに保存します。

```
apiVersion: apps/v1
```

```
kind: Deployment
metadata:
  name: sample-app
  namespace: same-namespace-as-your-fargate-profile
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:latest
          ports:
            - name: http
              containerPort: 80
```

b. マニフェストをクラスターに適用します。

```
kubectl apply -f sample-app.yaml
```

2. ConfigMap で設定した送信先を使用して、NGINX ログを表示します。

サイズに関する考慮事項

ログルーター用のメモリは、最大 50 MBに収まるようにすることをお勧めします。アプリケーションで非常に高いスループットでログが生成されることが予想される場合は、最大 100 MB を想定して計画する必要があります。

トラブルシューティング

ログ記録の機能が有効か無効かを確認するには、Pod イベントを **kubectl describe pod *pod_name*** でチェックしてください。ConfigMap が無効になっているなど何らかの理由で無効の場合は、その理由を確認できます。出力には、次の出力例のように、ログ記録が有効かどうかを明確にする Pod イベントが含まれる場合があります。

```
[...]
Annotations:          CapacityProvisioned: 0.25vCPU 0.5GB
```

```
                    Logging: LoggingDisabled: LOGGING_CONFIGMAP_NOT_FOUND
                    kubernetes.io/psp: eks.privileged
[...]
Events:
  Type     Reason           Age        From
              Message
  ----     ------           ----       ----
              -------
  Warning  LoggingDisabled  <unknown>  fargate-scheduler
               Disabled logging because aws-logging configmap was not found. configmap
 "aws-logging" not found
```

Pod イベントは一時的なもので、その期間は設定によります。**kubectl describe pod *pod-name*** を使用して、Pod's アノテーションを表示することもできます。Pod アノテーションには、ログ記録機能が有効か無効か、またその理由に関する情報があります。

Amazon EC2 インスタンスタイプを選択する

Amazon EC2 では、ワーカーノード用のインスタンスタイプが幅広く用意されています。インスタンスタイプごとに、コンピューティング、メモリ、ストレージの異なる機能が提供されます。また、各インスタンスファミリーは、これらの機能に基づきグループ化されています。リストについては、「Amazon EC2 ユーザーガイド」の「使用可能なインスタンスタイプ」および「Amazon EC2 ユーザーガイド」の「使用可能なインスタンスタイプ」を参照してください。Amazon EKS は、(特定の) サポートを有効にするために、Amazon EC2 AMI のいくつかのバリエーションをリリースしています。選択したインスタンスタイプに Amazon EKS との互換性があることを確認する際は、次の基準を考慮してください。

- すべての Amazon EKS AMI は、現在、g5g および mac ファミリをサポートしていません。
- Arm およびアクセラレータ付きではない Amazon EKS AMI では、g3、g4、inf、および p ファミリはサポートされません。
- アクセラレータ付き Amazon EKS AMI は、a、c、hpc、m、および t ファミリをサポートしていません。
- ARM ベースのインスタンスの場合、Amazon Linux 2023 (AL2023) は Graviton2 以降のプロセッサを使用するインスタンスタイプのみをサポートします。AL2023 は A1 インスタンスをサポートしていません。

Amazon EKS でサポートされているインスタンスタイプを選択する場合は、各タイプで次の機能を考慮してください。

ノードグループ内のインスタンス数。

特に多くの Daemonsets が存在する場合などは、一般的に、数少ない大型のインスタンスの使用が適しています。各インスタンスには API サーバーへの API コールが必要です。したがって、インスタンス数が多いほど、API サーバーの負荷が高くなります。

オペレーティングシステム

Linux、Windows、および Bottlerocket に対応しているインスタンスタイプを確認します。Windows インスタンスを作成する前に、Amazon EKS クラスター の Windows サポートの有効化 を確認してください。

ハードウェアアーキテクチャ

x86 または Arm のどちらが必要か検討します。Linux は Arm にのみデプロイすることができます。Arm インスタンスをデプロイする前に、Amazon EKS 最適化 Arm Amazon Linux AMI を確認します。Nitro System (Linux または Windows) 上に構築されたインスタンス、またはアクセラレータ付き機能インスタンが必要かを検討します。高速化された機能が必要な場合は、Amazon EKS でのみ Linux を使用できます。

Pods の最大数

各 Pod には独自の IP アドレスが割り当てられているため、インスタンスタイプでサポートされている IP アドレスの数が、インスタンスで実行できる Pods の数を決定するための要因になります。インスタンスタイプがサポートする Pods の数を手動で確認するには、「各 Amazon EC2 インスタンスタイプの Amazon EKS 推奨最大 Pods 数」を参照してください。

> **Note**
>
> Amazon EKS に最適化された Amazon Linux 2 AMI の `v20220406` 以降を使用している場合、AMI を最新版にアップグレードしなくても、新しいインスタンスタイプを使用できます。これらの AMI は、`max-pods` の値が `eni-max-pods.txt` ファイルにリストされていな場合、必要な値を自動計算します。現在プレビュー中のインスタンスタイプは、Amazon EKS のデフォルトではサポートされていない場合があります。これらのタイプでの `max-pods` 値は、依然として AMI 内の `eni-max-pods.txt` に追加する必要があります

AWS Nitro システムのインスタンスタイプには、サポートできる IP アドレスの数を、非 Nitro のシステムのインスタンスタイプよりも大幅に多くできるオプションがあります。ただし、インスタンスに割り当てられたすべての IP アドレスが Pods で使用できるわけではありません。インス

タンスに割り当てる IP アドレスの数を大幅に増やすには、クラスターにバージョン 1.9.0 以降の Amazon VPC CNI アドオンをインストールし、適切に設定する必要があります。詳細については、「Amazon EC2 ノードで使用可能な IP アドレスの量を増やす」を参照してください。インスタンスに最大数の IP アドレスを割り当てるには、バージョン 1.10.1 以降の Amazon VPC CNI アドオンをクラスターにインストールし、IPv6 ファミリーを使用してそのクラスターをデプロイする必要があります。

IP ファミリー

クラスターで IPv4 ファミリーを使用している場合、サポートされているすべてのインスタンスタイプが使用可能になります。これによりクラスターは、プライベート IPv4 アドレスを Pods とサービスに割り当てることができます。ただし、クラスターに IPv6 ファミリを使用する場合は、AWS Nitro システムインスタンスタイプ、またはベアメタルインスタンスタイプを使用する必要があります。IPv4 のみが Windows インスタンスでサポートされています。クラスターでは、バージョン 1.10.1 以降の Amazon VPC CNI アドオンを実行している必要があります。IPv6 の使用の詳細については、「クラスター、Pods、services 用の IPv6 アドレス」を参照してください。

実行している Amazon VPC CNI アドオンのバージョン

最新バージョンの Amazon VPC CNI Plugin for Kubernetes は、こちらのインスタンスタイプをサポートしています。サポートされている最新のインスタンスタイプを利用するには、Amazon VPC CNI アドオンのバージョンを更新する必要があります。詳細については、「Amazon VPC CNI plugin for Kubernetes Amazon EKS アドオンの使用」を参照してください。最新バージョンでは、Amazon EKS で使用できる最新の機能をサポートしています。以前のバージョンでは、すべての機能がサポートされているわけではありません。さまざまなバージョンでサポートされている機能は、GitHub の [Changelog] (変更履歴) で確認できます。

ノードを作成している AWS リージョン

AWS リージョンによっては使用できないインスタンスタイプがあります。

Pods にセキュリティグループを使用しているかどうか

Pods にセキュリティグループを使用している場合は、特定のインスタンスタイプのみがサポートされます。詳細については、「Pods のセキュリティグループ」を参照してください。

各 Amazon EC2 インスタンスタイプの Amazon EKS 推奨最大 Pods 数

各 Pod には独自の IP アドレスが割り当てられているため、インスタンスタイプでサポートされている IP アドレスの数が、インスタンスで実行できる Pods の数を決定するための要因になりま

す。Amazon EKS には、ダウンロードして実行できるスクリプトが用意されており、各インスタンスタイプで実行する Amazon EKS で推奨される最大の Pods 数を決定できます。このスクリプトでは、各インスタンスのハードウェア属性と設定オプションを使用して、Pods の最大数を決定します。これらの手順で返された番号を使用して、インスタンスのとは異なるサブネットから IP アドレスを Pods に割り当てたり、インスタンスの IP アドレスの数を大幅に増やす などの機能を有効にできます。複数のインスタンスタイプを持つマネージド型ノードグループを使用している場合は、それらすべてのインスタンスタイプで機能する値を使用します。

1. 各インスタンスタイプにおける Pods の最大数を計算するために使用できるスクリプトをダウンロードします。

```
curl -O https://raw.githubusercontent.com/awslabs/amazon-eks-ami/master/templates/
al2/runtime/max-pods-calculator.sh
```

2. コンピュータ上で、そのスクリプトを実行可能としてマークします。

```
chmod +x max-pods-calculator.sh
```

3. *m5.large* をデプロイ予定のインスタンスタイプに置き換え、*1.9.0-eksbuild.1* を Amazon VPC CNI アドオンバージョンに置き換えて、そのスクリプトを実行します。アドオンのバージョンを確認するには、Amazon VPC CNI plugin for Kubernetes Amazon EKS アドオンの使用の更新手順を参照してください。

```
./max-pods-calculator.sh --instance-type m5.large --cni-version 1.9.0-eksbuild.1
```

出力例は次のとおりです。

```
29
```

次のオプションをスクリプトに追加して、オプション機能を使用する際にサポートされる最大の Pods 数を確認できます。

- `--cni-custom-networking-enabled` — インスタンスとは異なるサブネットから IP アドレスを割り当てる場合は、このオプションを使用します。詳細については、「ポッド用のカスタムネットワーク」を参照してください。同じサンプル値を使用して前のスクリプトにこのオプションを追加すると、`20` が得られます。
- `--cni-prefix-delegation-enabled` — 各 elastic network interface にかなり多くの IP アドレスを割り当てる場合は、このオプションを使用します。この機能を使用するには、Nitro System で実行する Amazon Linux インスタンスと、Amazon VPC CNI アドオンのバージョン

1.9.0 以降が必要です。詳細については、「Amazon EC2 ノードで使用可能な IP アドレスの量を増やす」を参照してください。同じサンプル値を使用して前のスクリプトにこのオプションを追加すると、110 が得られます。

--help オプションを指定してスクリプトを実行し、使用可能なすべてのオプションを表示することもできます。

> **Note**
>
> 最大 Pods 計算スクリプトは、Kubernetes スケーラビリティのしきい値と推奨設定に基づいて戻り値を 110 に制限します。インスタンスタイプに 30 を超える vCPU がある場合、この制限は、内部の Amazon EKS スケーラビリティチームのテストに基づく数値である 250 に跳ね上がります。詳細については、「Amazon VPC CNI プラグインがノードあたりのポッド数の制限を引き上げ」のブログ記事を参照してください。

Amazon EKS 最適化 AMI

事前に構築された Amazon EKS 最適化 Amazon マシンイメージ (AMI)、または独自のカスタム AMI のいずれかを使用して、ノードをデプロイできます。Amazon EKS 最適化 AMI の各タイプの詳細については、以下のいずれかのトピックを参照してください。独自のカスタム AMI を作成する手順については、「Amazon EKS 最適化 Amazon Linux AMI のビルドスクリプト」を参照してください。

トピック

- Amazon EKS は Dockershim のサポートを終了しました
- Amazon EKS 最適化 Amazon Linux AMI
- Amazon EKS 最適化 Bottlerocket AMI
- Amazon EKS 最適化 Ubuntu Linux AMI
- Amazon EKS 最適化 Windows AMI

Amazon EKS は **Dockershim** のサポートを終了しました

Kubernetes では Dockershim サポートは終了しています。Kubernetes チームは Kubernetes バージョン 1.24 でランタイムを削除しました。詳細については、Kubernetes ブログの「Kubernetes は Dockershim から移行しています: コミットメントと次のステップ」を参照してください。

Amazon EKS は、Kubernetes バージョン 1.24 のリリース以降、Dockershim のサポートも終了しました。バージョン 1.24 以降、公式に公開される Amazon EKS AMI のランタイムは、containerd のみです。このトピックでは詳細をいくつかピックアップして説明していますが、より詳細な情報については「Amazon EKS の containerd に移行するにあたり知っておくべきこと」を参照してください。

Docker ソケットボリュームをマウントする Kubernetes ワークロードを表示するために使用できる kubectl プラグインがあります。詳細については、GitHub の「Docker ソケット (DDS) の検出器」を参照してください。1.24 より前のバージョンの Kubernetes を実行する Amazon EKS AMI は、Docker をデフォルトのランタイムとして使用します。ただし、これらの Amazon EKS AMI には、containerd を使用してサポートされているクラスターでワークロードをテストに使用できるブートストラップフラグオプションがあります。詳細については、「Docker から containerd への移行テスト」を参照してください。

既存の Kubernetes バージョン向け AMI の公開は、そのサポート終了日まで継続します。詳細については、「Amazon EKS Kubernetes リリースカレンダー」を参照してください。containerd でのワークロードのテストにさらに時間が必要な場合は、1.24 より前のサポートされているバージョンをご利用いただけます。しかし、公式の Amazon EKS AMI をそのバージョン 1.24 以降にアップグレードする場合は、ワークロードが containerd で実行可能なことを検証するようにしてください。

containerd ランタイムは、より信頼性の高いパフォーマンスとセキュリティを提供します。containerd は Amazon EKS 全体で標準化されているランタイムです。Fargate と Bottlerocket はすでに containerd のみを使用しています。containerd は、Dockershim 一般的な脆弱性と露出 (CVE) に対応するために必要な Amazon EKS AMI のリリースの数を最小限に抑えるのに役立ちます。Dockershim は内部では containerd を既に使用しているため、変更を加える必要がない場合があります。ただし、次のように変更が必要な状況があります。これには、変更が必要な可能性がある状況も含まれます。

- Docker ソケットをマウントしているアプリケーションを変更する必要があります。例えば、コンテナで構築されたコンテナイメージが影響を受けます。多くのモニタリングツールも、Docker ソケットをマウントしています。ランタイムのモニタリング用のワークロードについては、その更新を待つか、再デプロイする必要がある場合があります。
- 特定の Docker 設定に依存するアプリケーションでは、場合により変更が必要です。例えば、HTTPS_PROXY プロトコルのサポートが終了しているなどです。このプロトコルを使用するアプリケーションを更新する必要があります。詳細については、「Docker Docs」の「dockerd」を参照してください。

- Amazon ECR 認証情報ヘルパーを使用してイメージをプルする場合は、kubelet イメージ認証情報プロバイダーへの切り替えが必要です。詳細については、「Kubernetes ドキュメント」の「kubelet イメージ認証情報プロバイダーを設定する」を参照してください。
- Amazon EKS 1.24 は Docker をサポートしなくなったため、「Amazon EKS bootstrap script」(Amazon EKS ブートストラップスクリプト) が以前サポートしていた一部のフラグはサポートされなくなりました。Amazon EKS 1.24 以降に移行する前に、現在サポートされていないフラグへの参照をすべて削除する必要があります。
  - --container-runtime dockerd (containerd はサポートされる唯一の値です)
  - --enable-docker-bridge
  - --docker-config-json
- Fluentd がすでに Container Insights 用に構成されている場合、containerd に変更する前に Fluentd を Fluent Bit に移行する必要があります。Fluentd パーサーは JSON 形式のログメッセージのみを解析するように構成されています。dockerd とは異なり、containerd コンテナランタイムには JSON 形式ではないログメッセージがあります。Fluent Bit に移行しないと、構成された Fluentd's パーサーの一部が Fluentd コンテナー内で大量のエラーを生成します。移行の詳細については、「CloudWatch Logs へログを送信する DaemonSet として Fluent Bit を設定する」を参照してください。
- カスタム AMI を使用して Amazon EKS 1.24 にアップグレードする場合は、ワーカーノードで IP 転送が有効になっていることを確認する必要があります。この設定は Docker では必要ありませんでしたが、containerd では必須です。Pod-Pod、Pod-外部、または Pod-apiserver 間のネットワーク接続のトラブルシューティングに必要です。

  ワーカーノードでこの設定を確認するには、次のコマンドのいずれかを実行します。

  - sysctl net.ipv4.ip_forward
  - cat /proc/sys/net/ipv4/ip_forward

  出力が 0 の場合、次のコマンドのいずれかを実行して net.ipv4.ip_forward カーネル変数をアクティブにします。

  - sysctl -w net.ipv4.ip_forward=1
  - echo 1 > /proc/sys/net/ipv4/ip_forward

  containerd ランタイムの Amazon EKS AMI での設定のアクティブ化については、「GitHub」の「install-worker.sh」を参照してください。

Amazon EKS 最適化 Amazon Linux AMI

Amazon EKS 最適化 Amazon Linux AMI は、Amazon Linux 2 (AL2) および Amazon Linux 2023 (AL2023) 上に構築されています。Amazon EKS ノードのベースイメージとして機能するように構成されています。AMI は、Amazon EKS と連携するように構成されており、次のコンポーネントが含まれています。

- `kubelet`
- AWS IAM Authenticator
- Docker (Amazon EKS バージョン `1.23` 以前)
- `containerd`

> **Note**
>
> - AL2 のセキュリティやプライバシーに関するイベントは、Amazon Linux Security Centerで追跡したり、関連する RSS フィードをサブスクライブしたりできます。セキュリティおよびプライバシーイベントには、問題の概要、影響を受けるパッケージ、および問題を修正するためにインスタンスを更新する方法などがあります。
> - 高速 AMI または Arm AMI をデプロイする前に、「Amazon EKS 最適化高速 Amazon Linux AMI」および「Amazon EKS 最適化 Arm Amazon Linux AMI」の情報を確認してください。
> - Kubernetes バージョン `1.23` では、オプションのブートストラップフラグを使用して、Docker から `containerd` への移行をテストできます。詳細については、「Docker から `containerd` への移行テスト」を参照してください。
> - Kubernetes バージョン `1.25` 以降、Amazon EC2 P2 インスタンスを Amazon EKS に最適化された高速 Amazon Linux AMI で使用するには、追加設定が必要です。これらの Kubernetes バージョン `1.25` 以降向けの AMI は `NVIDIA 525` シリーズまたはそれ以降のドライバーに対応しており、P2 インスタンスとの互換性がありません。`NVIDIA 525` シリーズおよびそれ以降のドライバーは P3、P4、P5 インスタンスとの互換性があるため、これらのインスタンスを Kubernetes バージョン `1.25` またはそれ以降の AMI で使用することができます。Amazon EKS クラスターをバージョン `1.25` にアップグレードする前に、P2 インスタンスを P3、P4、P5 インスタンスに移行します。また、`NVIDIA 525` シリーズまたはそれ以降と連携するように、アプリケーションを自発的にアップグレードする必要があります。新しい `NVIDIA 525` シリーズ以降のドライバーは、2024 年 1 月後半に Kubernetes バージョン `1.23` および `1.24` にバックポートする予定です。

- バージョン 1.30 以降のクラスターで新しく作成されたマネージド型ノードグループは、ノードオペレーティングシステムとして自動的に AL2023 を使用するようデフォルトで設定されます。以前は、新しいノードグループはデフォルトで AL2 を使用するよう設定されていました。新しいノードグループを作成するときに AL2 を AMI タイプとして選択すれば、AL2 を引き続き使用できます。
- AL2 のサポートは 2025 年 6 月 30 日に終了します。詳細については、「Amazon Linux 2 に関するよくある質問」を参照してください。

AL2 から AL2023 へのアップグレード

Amazon EKS 最適化 AMI は、AL2 および AL2023 をベースとする 2 つのファミリーで使用できます。AL2023 は、クラウドアプリケーションに安全かつ安定した高パフォーマンス環境を提供するように設計された、新しい Linux ベースのオペレーティングシステムです。これは Amazon Web Services の次世代 Amazon Linux であり、サポートされているすべての Amazon EKS バージョンで利用できます (延長サポート中のバージョン 1.23 や 1.24 を含む)。AL2023 をベースとする Amazon EKS 高速 AMI は、後日利用可能になります。高速ワークロードがある場合は、AL2 高速 AMI または Bottlerocket を引き続き使用する必要があります。

AL2023 には、AL2 に比べていくつかの改善点があります。詳細については、「Amazon Linux 2023 ユーザーガイド」の「Comparing AL2 and AL2023」を参照してください。AL2 からいくつかのパッケージが追加、アップグレード、削除されています。アップグレードする前に、AL2023 でアプリケーションをテストすることを強くお勧めします。AL2023 のパッケージに関するすべての変更一覧については、「Amazon Linux 2023 リリースノート」の「Package changes in Amazon Linux 2023」を参照してください。

これらの変更に加えて、以下の点に注意してください。

- AL2023 では、YAML 設定スキーマを使用する新しいノード初期化プロセス nodeadm が導入されています。セルフマネージド型ノードグループまたは起動テンプレートを持つ AMI を使用している場合は、新しいノードグループの作成時に追加のクラスターメタデータを明示的に指定する必要があります。最低限必要なパラメータの例を以下に示します。ここで、apiServerEndpoint、certificateAuthority、サービスの cidr が必要になります。

```
---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
```

```
  cluster:
    name: my-cluster
    apiServerEndpoint: https://example.com
    certificateAuthority: Y2VydGlmaWNhdGVBdXRob3JpdHk=
    cidr: 10.100.0.0/16
```

AL2 では、これらのパラメータからのメタデータは Amazon EKS `DescribeCluster` API コールから検出されていました。AL2023 では、ノードの大規模なスケールアップ中に API コールによってスロットリングが発生するリスクがあるため、この動作が変更されました。この変更は、起動テンプレートのないマネージド型ノードグループを使用している場合や、Karpenter を使用している場合には影響しません。詳細については、「Amazon EKS API リファレンス」の `certificateAuthority`、サービスの `cidr`、`DescribeCluster` を参照してください。

- AL2023 では、サポートされているすべての Amazon EKS バージョンで Docker が サポートされているとは限りません。AL2 では、Amazon EKS バージョン `1.24` 以降で Docker のサポートは終了し、削除されました。廃止の詳細については、「Amazon EKS は Dockershim のサポートを終了しました」を参照してください。
- AL2023 には Amazon VPC CNI バージョン `1.16.2` 以降が必要です。
- AL2023 にはデフォルトで `IMDSv2` が必要です。`IMDSv2` には、セキュリティ体制の改善に役立ついくつかの利点があります。セッション指向の認証方式を使用しており、セッションを開始するためにシンプルな HTTP PUT リクエストでシークレットトークンを作成する必要があります。セッショントークンの有効期間は 1 秒 ~ 6 時間です。`IMDSv1` から `IMDSv2` への移行方法の詳細については、「インスタンスメタデータサービスバージョン 2 の使用への移行」および「Get the full benefits of IMDSv2 and disable IMDSv1 across your AWS infrastructure」を参照してください。`IMDSv1` を使用する場合は、インスタンスのメタデータオプションの起動プロパティで設定を手動で上書きすることで使用可能になります。

  > **Note**
  >
  > `IMDSv2` の場合、マネージド型ノードグループのデフォルトのホップカウントは 1 に設定されています。つまり、コンテナが IMDS を使用してノードの認証情報にアクセスすることはできません。ノードの認証情報へのコンテナアクセスが必要な場合は、カスタム Amazon EC2 起動テンプレートの `HttpPutResponseHopLimit` を手動で上書きして 2 に増やすことでアクセスが可能になります。または、`IMDSv2` の代わりに Amazon EKS Pod Identity を使用して、認証情報を提供することもできます。

- AL2023 は、次世代の統合コントロールグループ階層 (`cgroupv2`) を備えています。`cgroupv2` は、コンテナランタイムを実装するために `systemd` によって使用されます。AL2023 に

は、cgroupv1 を使用してシステムを実行できるコードが引き続き含まれていますが、この設定は推奨されません。またサポート対象でもありません。この設定は、今後の Amazon Linux のメジャーリリースで完全に削除される予定です。

- AL2023 をサポートするには、eksctl に eksctl のバージョン 0.176.0 以降が必要です。

既存のマネージド型ノードグループの場合は、起動テンプレートの使用方法に応じて、インプレースアップグレードまたは Blue/Green アップグレードを実行できます。

- マネージド型ノードグループでカスタム AMI を使用している場合は、起動テンプレートの AMI ID を入れ替えることで、インプレースアップグレードを実行できます。このアップグレード戦略を実行する前に、まずアプリケーションとユーザーデータが AL2023 に転送されていることを必ず確認してください。
- 標準の起動テンプレートまたは AMI ID を指定しないカスタム起動テンプレートでマネージド型ノードグループを使用している場合は、Blue/Green 戦略を使用してアップグレードする必要があります。通常、Blue/Green アップグレードはより複雑で、AMI タイプとして AL2023 を指定する完全に新しいノードグループを作成する必要があります。新しいノードグループの設定は、AL2 ノードグループのすべてのカスタムデータが新しい OS と互換性があることを確認して、慎重に行う必要があります。新しいノードグループがアプリケーションでテストおよび検証されると、Pods を古いノードグループから新しいノードグループに移行できます。移行が完了したら、古いノードグループを削除できます。

Karpenter を利用していて、AL2023 を使用する場合は、EC2NodeClass amiFamily フィールドを AL2023 に変更する必要があります。デフォルトでは、Karpenter ではドリフトが有効になっています。つまり、amiFamily フィールドが変更されると、Karpenter はワーカーノードを使用可能な最新の AMI に自動的に更新します。

Amazon EKS 最適化高速 Amazon Linux AMI

> **Note**
>
> AL2023 をベースとする Amazon EKS 高速 AMI は、後日利用可能になります。高速ワークロードがある場合は、AL2 高速 AMI または Bottlerocket を引き続き使用する必要があります。

Amazon EKS 最適化高速 Amazon Linux AMI は、標準的な Amazon EKS 最適化 Amazon Linux AMI 上に構築されています。これは、Amazon EKS ノードのオプションのイメージとして機能し、GPU、Inferentia、Trainium ベースのワークロードをサポートするよう設定されています。

標準の Amazon EKS 最適化 AMI 設定に加えて、高速 AMI には、以下が備わっています。

- NVIDIA ドライバー
- `nvidia-container-runtime`
- AWS Neuron ドライバー

高速 AMI に含まれる最新のコンポーネント一覧については、GitHub の`amazon-eks-ami` リリースを参照してください。

> **Note**
>
> - Amazon EKS 最適化高速 AMI では、GPU と Inferentia をベースとしたインスタンスタイプのみをサポートします。ノードの AWS CloudFormation テンプレートには、これらのインスタンスタイプを指定するようにしてください。Amazon EKS 最適化高速 AMI を使用することで、NVIDIA のユーザーライセンス契約 (EULA) に同意したものとみなされます。
> - Amazon EKS 最適化高速 AMI は、以前は、GPU をサポートする Amazon EKS 最適化 AMI と呼ばれていたものです。
> - これまでのバージョンの Amazon EKS 最適化高速 AMI では、`nvidia-docker` リポジトリがインストールされていました。このリポジトリは、Amazon EKS AMI バージョン `v20200529` 以降では包含されなくなります。

AWS Neuron (ML アクセラレーター) ベースのワークロードを有効にする

Amazon EKS Neuron で を使用するトレーニングおよび推論ワークロードの詳細については、以下のリファレンスを参照してください。

- コンテナ - Kubernetes - 開始方法 (AWS Neuron ドキュメント)
- GitHub での AWS Neuron EKS サンプルでのトレーニング
- AWS Inferentia を使用した機械学習推論

GPU ベースのワークロードを有効化するには

次の手順に、Amazon EKS 最適化高速 AMI を使用しながら GPU ベースのインスタンス上でワークロードを実行する方法を示します。

1. GPU ノードをクラスターに加えた後、Kubernetes 用 NVIDIA デバイスプラグイン をクラスターの DaemonSet として適用する必要があります。次のコマンドを実行する前に、*vX.X.X* を必要となる NVIDIA/k8s-device-plugin バージョンに置き換えます。

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/
nvidia-device-plugin.yml
```

2. ノードに割り当て可能な GPU があることは、次のコマンドで確認できます。

```
kubectl get nodes "-o=custom-
columns=NAME:.metadata.name,GPU:.status.allocatable.nvidia\.com/gpu"
```

Pod をデプロイして、GPU ノードが適切に設定されていることをテストするには

1. 次の内容で、nvidia-smi.yaml という名前のファイルを作成します。*tag* を必要な nvidia/cuda のタグに置き換えます。このマニフェストは、ノード上で nvidia-smi を実行する NVIDIA CUDA コンテナを起動します。

```
apiVersion: v1
kind: Pod
metadata:
  name: nvidia-smi
spec:
  restartPolicy: OnFailure
  containers:
  - name: nvidia-smi
    image: nvidia/cuda:tag
    args:
    - "nvidia-smi"
    resources:
      limits:
        nvidia.com/gpu: 1
```

2. 次のコマンドを使用してマニフェストを適用します。

```
kubectl apply -f nvidia-smi.yaml
```

3. Pod の実行の終了後、次のコマンドを使用してログを表示します。

```
kubectl logs nvidia-smi
```

出力例は次のとおりです。

```
Mon Aug  6 20:23:31 20XX
+-----------------------------------------------------------------------------+
| NVIDIA-SMI XXX.XX                 Driver Version: XXX.XX                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  On   | 00000000:00:1C.0 Off |                    0 |
| N/A   46C    P0    47W / 300W |      0MiB / 16160MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Amazon EKS 最適化 Arm Amazon Linux AMI

Arm インスタンスは、ウェブサーバー、コンテナ化されたマイクロサービス、キャッシュフリート、および分散データストアといったスケールアウト型で Arm ベースのアプリケーションにおいて、コストを大幅に削減します。クラスターに Arm ノードを追加する際には、次の考慮事項を確認してください。

考慮事項

- クラスターが 2020 年 8 月 17 日より前にデプロイされている場合、クラスターの重要なアドオンマニフェストを 1 回だけアップグレードする必要があります。これは、クラスター内で使用中の各ハードウェアアーキテクチャのイメージを、Kubernetes が正しく取得できるようにするためです。クラスターのアドオン更新の詳細については、「Amazon EKS クラスターに必要な

Kubernetes バージョンを更新する」を参照してください。2020 年 8 月 17 日以降にクラスターをデプロイしている場合、ご使用の CoreDNS、kube-proxy、および Amazon VPC CNI plugin for Kubernetes のアドオンは、すでにマルチアーキテクチャに対応済みです。

- Arm ノードにデプロイされたアプリケーションは、Arm 用にコンパイルする必要があります。
- 既存のクラスターでデプロイ済みの DaemonSets がある場合、または Arm ノードをデプロイする新しいクラスターにこれをデプロイする場合は、クラスター内のすべてのハードウェアアーキテクチャで DaemonSet が実行可能であることを確認します。
- 同じクラスター内で、Arm ノードグループと x86 ノードグループを実行することができます。その場合、Pod をデプロイできるハードウェアアーキテクチャを Kubernetes が認識できるようにするために、マルチアーキテクチャのコンテナイメージを Amazon Elastic Container Registry などのコンテナリポジトリにデプロイした上で、ノードセレクターをマニフェストに追加することも考慮に入れてください。詳細については、Amazon ECR ユーザーガイドのマルチアーキテクチャイメージのプッシュと、ブログ記事 Introducing multi-architecture container images for Amazon ECR を参照してください。

Docker から **containerd** への移行テスト

Amazon EKS は、Kubernetes バージョン 1.24 のリリース以降、Docker のサポートを終了しました。詳細については、「Amazon EKS は Dockershim のサポートを終了しました」を参照してください。

Kubernetes バージョン 1.23 では、オプションのブートストラップフラグを使用して、Amazon EKS 最適化 AL2 AMI の containerd ランタイムを有効にすることができます。この機能により、バージョン 1.24 以降に更新するときに containerd に移行するための明確なパスが提供されます。Amazon EKS は、Kubernetes バージョン 1.24 のリリース以降、Docker のサポートを終了しました。containerd ランタイムは Kubernetes コミュニティで広く導入されていて、CNCF で段階的に進めているプロジェクトです。これは、新しいクラスターまたは既存のクラスターにノードグループを追加することでテストできます。

ブートストラップフラグを有効にするには、以下のいずれかのタイプでノードグループを作成します。

セルフマネージド型

セルフマネージド型の Amazon Linux ノードの起動 の手順に従ってノードグループを作成します。BootstrapArguments パラメータでは、Amazon EKS 最適化 AMI と、以下のテキストを指定します。

```
--container-runtime containerd
```

マネージド

eksctl を使用する場合には、次の内容の *my-nodegroup*.yaml という名前のファイルを作成します。*example value* をすべて自分の値に置き換えてください。ノードグループ名は 63 文字以下である必要があります。先頭は文字または数字でなければなりませんが、残りの文字にはハイフンおよびアンダースコアを含めることもできます。ami-*1234567890abcdef0* の最適化された AMI ID を取得するには、「Amazon EKS 最適化 Amazon Linux AMI ID の取得」を参照してください。

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: my-cluster
  region: region-code
  version: 1.23
managedNodeGroups:
  - name: my-nodegroup
    ami: ami-1234567890abcdef0
    overrideBootstrapCommand: |
      #!/bin/bash
      /etc/eks/bootstrap.sh my-cluster --container-runtime containerd
```

> **Note**
>
> 多数のノードを同時に起動する場合は、エラーを回避するために、ブートストラップ引数 --apiserver-endpoint、--b64-cluster-ca、および --dns-cluster-ip にも値を指定することをお勧めします。詳細については、「AMI を指定する」を参照してください。

以下のコマンドを実行して、ノードグループを作成します。

```
eksctl create nodegroup -f my-nodegroup.yaml
```

異なるツールを使用してマネージド型のノードグループを作成する場合、そのノードグループのデプロイには起動テンプレートを使用する必要があります。起動テンプレート内で Amazon EKS 最適化 AMI ID を指定した上で、その起動テンプレートによりノードグループをデプロイし、次

のユーザーデータを設定します。このユーザーデータは、引数を bootstrap.sh ファイルに渡します。ブートストラップファイルの詳細については、GitHub の bootstrap.sh を参照してください。

```
/etc/eks/bootstrap.sh my-cluster --container-runtime containerd
```

詳細情報

Amazon EKS 最適化 Amazon Linux AMI の詳細については、以下のセクションを参照してください。

- Amazon Linux をマネージド型ノードグループと使用するには、「マネージド型ノードグループ」を参照してください。
- セルフマネージド型の Amazon Linux ノードの起動には、「Amazon EKS 最適化 Amazon Linux AMI ID の取得」を参照してください。
- バージョンについては、「Amazon EKS 最適化 Amazon Linux AMI のバージョン」を参照してください。
- Amazon EKS 最適化 Amazon Linux AMI の最新の ID を取得するには、「Amazon EKS 最適化 Amazon Linux AMI ID の取得」を参照してください。
- Amazon EKS 最適化 AMI の構築に使用されているオープンソーススクリプトについては、「Amazon EKS 最適化 Amazon Linux AMI のビルドスクリプト」を参照してください。

Amazon EKS 最適化 Amazon Linux AMI のバージョン

Amazon EKS 最適化 Amazon Linux AMI は、Kubernetes バージョンと AMI のリリース日によって次の形式でバージョニングされています。

```
k8s_major_version.k8s_minor_version.k8s_patch_version-release_date
```

各 AMI リリースには、kubelet、Docker、Linux カーネル、containerd のさまざまなバージョンが含まれます。また高速 AMI には、さまざまなバージョンの NVIDIA ドライバーも含まれます。このバージョンに関する情報は、GitHub の「Changelog」で確認できます。

Amazon EKS 最適化 Amazon Linux AMI ID の取得

AWS Systems Manager パラメータストア API をクエリすることで、 Amazon EKS 最適化 AMI の Amazon マシンイメージ (AMI、Amazon Machine Image) ID をプログラムで取得できます。この API

から提供されるパラメータにより、Amazon EKS 最適化 AMI ID を手動で検索する必要がなくなります。Systems Manager Parameter Store API の詳細については、GetParameter を参照してください。

AWS CLI を使用して Amazon EKS 最適化 AMI の AMI ID を取得するには

1. us-west-2 など、ノードインスタンスをデプロイするリージョンを決定します。
2. 必要な AMI のタイプを決定します。Amazon EC2 インスタンスタイプの詳細については、「インスタンスタイプ」を参照してください。
   - amazon-linux-2 は、Amazon Linux 2 (AL2) x86 ベースのインスタンス用です。
   - amazon-linux-2-arm64 は、AWS Graviton ベースのインスタンスなどの AL2 ARM インスタンス用です。
   - amazon-linux-2-gpu は、AL2 GPU 高速インスタンス用です。
   - amazon-linux-2023/x86_64/standard は、Amazon Linux 2023 (AL2023) x86 ベースのインスタンス用です。
   - amazon-linux-2023/arm64/standard は、AL2023 ARM インスタンス用です。
3. 1.30 など、ノードがアタッチされるクラスターの Kubernetes バージョンを確認します。
4. 次の AWS CLI コマンドを実行して、適切な AMI ID を取得します。必要に応じて、AWS リージョン、Kubernetes バージョン、およびプラットフォームを置き換えます。Amazon EKS 最適化 AMI メタデータを取得するための ssm:GetParameter IAM アクセス許可を持つ IAM プリンシパルを使用して AWS CLI にログインする必要があります。

```
aws ssm get-parameter --name /aws/service/eks/optimized-ami/1.30/amazon-linux-2/
recommended/image_id \
                --region region-code --query "Parameter.Value" --output text
```

出力例は次のとおりです。

```
ami-1234567890abcdef0
```

Amazon EKS 最適化 Amazon Linux AMI のビルドスクリプト

Amazon Elastic Kubernetes Service (Amazon EKS) には、Amazon EKS 最適化 AMI の構築に使用される、オープンソースのスクリプトが備わっています。これらのビルドスクリプトは、GitHub で 利用可能です。

Amazon EKS 最適化 Amazon Linux AMI は、Amazon Linux 2 (AL2) および Amazon Linux 2023 (AL2023) 上に構築されており、特に Amazon EKS クラスター内ではノードとして使用されます。このリポジトリを使用すると、Amazon EKS チームが `kubelet` や Docker、AWS IAM Authenticator for Kubernetes などを設定する際の詳細な設定方法を確認したり、独自の Amazon Linux ベースの AMI を最初から構築したりできます。

ビルドスクリプトレポジトリには、HashiCorp Packer テンプレートと、AMI を生成するビルドスクリプトが含まれています。ここに置かれているものは、Amazon EKS 最適化 AMI のビルド用として最も信頼できるスクリプトであるため、GitHub リポジトリの状態をフォローすることで、AMI に対し実施された変更をモニタリングできます。例えば、独自の AMI で、Amazon EKS チームが公式の AMI に使用するのと同じバージョンの Docker を使用できます。

さらに、GitHub リポジトリには、インスタンスの証明書データ、コントロールプレーンエンドポイント、クラスター名などを設定するために起動時に実行される、特別なブートストラップスクリプトおよび nodeadm スクリプトも含まれます。

加えて、GitHub リポジトリには、Amazon EKS ノード用の AWS CloudFormation テンプレートも含まれています。これらのテンプレートにより、Amazon EKS 最適化 AMI を実行するインスタンスの準備と、クラスターへの登録を簡単に行うことができます。

詳細については、GitHub のリポジトリ (https://github.com/awslabs/amazon-eks-ami) を参照してください。

Amazon EKS 最適化 AL2 には、`containerd` ランタイムを有効化するためのオプションのブートストラップフラグが含まれています。

カスタム Amazon Linux AMI 用の VT1 を設定する

Amazon EKS のカスタム Amazon Linux AMI は、Amazon Linux 2 (AL2)、Ubuntu 18、Ubuntu 20 用の VT1 ビデオトランスコーディングインスタンスファミリーをサポートできます。VT1 は、高速 H.264/AVC および H.265/HEVC コーデックを備えた Xilinx U30 メディアトランスコーディングカードをサポートしています。これらのアクセラレートインスタンスのメリットを享受するには、次のステップを実行する必要があります。

1. AL2、Ubuntu 18、または Ubuntu 20 からベース AMI を作成して起動します。
2. ベースの AMI が起動したら、ノードに XRT ドライバー とランタイムをインストールします。
3. Amazon EKS クラスターの作成.
4. Kubernetes FPGA plugin をクラスターにインストールします。

```
kubectl apply -f fpga-device-plugin.yml
```

このプラグインが、Amazon EKS クラスターのノードごとに Xilinx U30 デバイスをアドバタイズするようになりました。FFMPEG Docker イメージを使用して、Amazon EKS クラスターでサンプルビデオトランスコーディングワークロードを実行できます。

カスタム Amazon Linux 2 AMI 用の DL1 を設定する

Amazon EKS のカスタム Amazon Linux 2 (AL2) AMI は、追加の設定と Kubernetes アドオンを通じて、大規模な深層学習ワークロードをサポートできます。このドキュメントでは、オンプレミスセットアップ用または大規模なクラウド設定のベースラインとして汎用 Kubernetes ソリューションをセットアップするために必要なコンポーネントについて説明します。この機能をサポートするには、カスタム環境で次の手順を実行する必要があります。

- システムにロードされた SynapaseAI® Software ドライバ - これらは、Github で入手できる AMI に含まれています。
- Habana デバイスプラグイン - Kubernetes クラスターへの Habana デバイスの登録を自動的に有効にし、デバイスの状態を追跡できるようにする DaemonSet。
- Helm 3.x
- MPI Operator をインストールするための Helm チャート。
- MPI Operator

1. AL2、Ubuntu 18、または Ubuntu 20 からベース AMI を作成して起動します。
2. これらの手順に従って、DL1 の環境をセットアップします。

Amazon EKS 最適化 Bottlerocket AMI

Bottlerocket は、AWS によってスポンサーおよびサポートされているオープンソースの Linux ディストリビューションです。Bottlerocket はコンテナワークロードのホスティング専用です。Bottlerocket を使用すると、コンテナインフラストラクチャの更新を自動化することにより、コンテナ化されたデプロイの可用性を向上させ、運用コストを削減できます。Bottlerocket は、コンテナの実行に不可欠なソフトウェアのみが含まれており、リソース使用率の向上、セキュリティ上の脅威の軽減、管理オーバーヘッドの軽減が実現されます。Bottlerocket AMI には、`containerd`、`kubelet`、および AWS IAM オーセンティケーターが含まれます。Bottlerocket

は、マネージド型ノードグループとセルフマネージド型ノードに加え、Karpenter でもサポートされています。

利点

Bottlerocket を Amazon EKS クラスターと使用すると、次のような利点があります。

- 運用コストが低く、管理の複雑さが軽減されて増加した稼働 - Bottlerocket は、他の Linux ディストリビューションよりもリソースフットプリントが小さく、起動時間が短く、セキュリティの脅威に対する脆弱性が低くなります。Bottlerocket's のより小さなフットプリントは、ストレージ、コンピューティング、ネットワーキングのリソースを少なく使用することでコストを削減できます。
- OS の自動更新によるセキュリティの向上 — Bottlerocket への更新は単一のユニットとして適用されるので、必要に応じてロールバックできます。これにより、システムが使用不能な状態にさらす更新プログラムの破損または失敗のリスクをなくします。Bottlerocket を使用すると、セキュリティ更新プログラムが利用可能になり次第、中断を最小限に抑えながら自動的に適用され、障害が発生した場合はロールバックできます。
- プレミアムサポート — Amazon EC2 で AWS が提供する Bottlerocket のビルドは、Amazon EC2、Amazon EKS、Amazon ECR などの AWS サービスも対象とする同じ AWS Support プランの対象となります。

考慮事項

Bottlerocket を AMI タイプに使用する際、次の事項を考慮してください。

- Bottlerocket は、`x86_64` と `arm64` プロセッサを搭載した Amazon EC2 インスタンスをサポートします。Bottlerocket AMI を Inferentia チップを搭載した Amazon EC2 インスタンスで使用することはお勧めしません。
- 現在、Bottlerocket ノードをデプロイするために使用できる AWS CloudFormation テンプレートはありません。
- Bottlerocket イメージには、SSH サーバーまたはシェルは含まれません。帯域外のアクセス方法を使用して SSH を許可できます。これらの手法は、管理者コンテナを有効にし、ユーザーデータでブートストラップの設定ステップの渡しを可能にします。詳細については、GitHub の「Bottlerocket OS」内にある次のセクションを参照してください。
  - [探査]
  - 管理者コンテナ
  - KubernetesX 設定

- Bottlerocket はさまざまなコンテナタイプを使用します。
  - デフォルトで、「コントロールコンテナ」は有効になっています。このコンテナは、Amazon EC2 Bottlerocket インスタンス上でのコマンドの実行やシェルセッションの開始に使用できる AWS Systems Manager エージェントを実行します。詳細については、AWS Systems Manager ユーザーガイドの Session Manager のセットアップを参照してください。
  - ノードグループの作成時に SSH キーが与えられる場合、管理者コンテナが有効になります。admin container は、開発とテストのシナリオにのみ使用することをお勧めします。本番環境で使用することはお勧めしません。詳細については、GitHub の「管理コンテナ」を参照してください。

詳細情報

Amazon EKS 最適化 Bottlerocket AMI の詳細については、以下のセクションを参照してください。

- Bottlerocket の詳細については、GitHub のドキュメントとリリースを参照してください。
- Bottlerocket をマネージド型ノードグループと使用するには、「マネージド型ノードグループ」を参照してください。
- セルフマネージド型 Bottlerocket ノードを起動するには、「セルフマネージド型 Bottlerocket ノードの起動」を参照してください。
- Amazon EKS 最適化 Bottlerocket AMI の最新の ID を取得するには、「Amazon EKS 最適化 Bottlerocket AMI ID の取得」を参照してください。
- コンプライアンスサポートの詳細については、「Bottlerocket コンプライアンスサポート」を参照してください。

Amazon EKS 最適化 Bottlerocket AMI ID の取得

AWS Systems Manager パラメータストア API をクエリすることで、Amazon EKS 最適化 AMI の Amazon マシンイメージ (AMI) ID を取得できます。このパラメータを使用することで、Amazon EKS 最適化 AMI ID を手動で検索する必要がなくなります。Systems Manager Parameter Store API の詳細については、GetParameter を参照してください。使用する IAM プリンシパルには、Amazon EKS 最適化 AMI メタデータを取得するための `ssm:GetParameter` IAM アクセス許可が必要です。

サブパラメータ `image_id` を指定しながら次の AWS CLI コマンドを使用することで、推奨される最新の Amazon EKS 最適化 Bottlerocket AMI のイメージ ID を取得できます。*`1.30`* をサポートされているバージョンで置き換え、*`region-code`* を、AMI ID を必要とする Amazon EKS がサポートされているリージョンで置き換えます。

```
aws ssm get-parameter --name /aws/service/bottlerocket/aws-k8s-1.30/x86_64/latest/
image_id --region region-code --query "Parameter.Value" --output text
```

出力例は次のとおりです。

```
ami-1234567890abcdef0
```

Bottlerocket コンプライアンスサポート

Bottlerocket はさまざまな組織によって定義された推奨事項に準拠しています。

- 「CIS ベンチマーク」は Bottlerocket に定義されています。デフォルト設定では、Bottlerocket イメージは CIS レベル 1 構成プロファイルに必要なほとんどのコントロールが含まれています。CIS レベル 2 構成プロファイルに必要なコントロールを実装できます。詳細については、「AWS ブログ」の「Amazon EKS に最適化された Bottlerocket AMI を CIS ベンチマークと照合して検証」を参照してください。
- 最適化された機能セットおよび最小化したアタックサーフェスは、攻撃対象領域が小さくなるため、Bottlerocket インスタンスが PCI DSS 要件を満たすために必要な構成が少なくて済みます。Bottlerocket の CIS Benchmark は、ガイダンスを強化するために優れたリソースであり、PCI DSS 要件 2.2 に基づく安全な構成標準の要件をサポートします。Fluent Bit を活用して、PCI DSS 要件 10.2 に基づくオペレーティングシステムレベルの監査ログ記録の要件をサポートすることもできます。AWS は、PCI DSS 要件 6.2 (v3.2.1 用) および要件 6.3.3 (v4.0 用) を満たすために役立つ新しい (パッチが適用された) Bottlerocket インスタンスを定期的に公開します。
- Bottlerocket は、Amazon EC2 および Amazon EKS の両方に規制対象のワークロードで使用が承認された HIPAA 対象の機能です。詳細については、「Amazon EKS で HIPAA セキュリティとコンプライアンス向けの設計」ホワイトペーパーを参照してください。

Amazon EKS 最適化 Ubuntu Linux AMI

Canonical 社は Amazon EKS と提携し、クラスターで使用できる ノード AMI を作成しました。

Canonical は、特定用途向けの Kubernetes ノード OS イメージを提供しています。この最小化された Ubuntu イメージは、Amazon EKS 用に最適化されており、AWS と共同開発されたカスタム AWS カーネルが含まれています。詳細については、「AUbuntu on Amazon Elastic Kubernetes Service (EKS)」と「セルフマネージド型 Ubuntu ノードの起動」を参照してください。サポートの詳細については、「AWS プレミアムサポートのよくある質問」の「サードパーティソフトウェア」セクションを参照してください。

Amazon EKS 最適化 Windows AMI

Windows Amazon EKS 最適化 AMI は Windows Server 2019 と Windows Server 2022 上に構築されています。Amazon EKS ノードのベースイメージとして機能するように構成されています。デフォルトでは、AMI には次のコンポーネントが含まれています。

- kubelet
- kube-proxy
- AWS IAM Authenticator for Kubernetes
- csi-proxy
- containerd

> **Note**
>
> Microsoft セキュリティアップデートガイド を使用して、Windows サーバーのセキュリティまたはプライバシーイベントを追跡できます。

Amazon EKS は、次のバリアントで Windows コンテナに最適化された AMI を提供しています。

- Amazon EKS 最適化 Windows Server 2019 Core AMI
- Amazon EKS 最適化 Windows Server 2019 Full AMI
- Amazon EKS 最適化 Windows Server 2022 Core AMI
- Amazon EKS 最適化 Windows Server 2022 Full AMI

> **Important**
>
> - Amazon EKS 最適化 Windows Server 20H2 Core AMI は非推奨です。この AMI の新しいバージョンはリリースされません。
> - 最新のセキュリティ更新プログラムがデフォルトで適用されるように、Amazon EKS は直近 4 か月間の最適化 Windows AMI を維持します。新しい AMI はそれぞれ、初回リリースから 4 か月間使用できます。この期間が過ぎると、古い AMI は非公開になり、アクセス不可になります。セキュリティの脆弱性を回避し、サポート期間が終了した古い AMI にアクセスできなくなることを防ぐために、最新の AMI を使用することをお勧めします。非公開

になった AMI へのアクセスを可能にする事は保証できませんが、AWS Supportにチケットを提出することでアクセスをリクエストできます。

リリースカレンダー

以下の表は、Amazon EKS における Windows バージョンのサポートリリース日、および終了日を示しています。終了日が空白の場合、そのバージョンのサポートは継続されています。

Windows バージョン	Amazon EKS リリース	Amazon EKS のサポート終了
Windows Server 2022 Core	10/17/2022	
Windows Server 2022 Full	10/17/2022	
Windows Server 20H2 Core	8/12/2021	8/9/2022
Windows Server 2004 Core	8/19/2020	12/14/2021
Windows Server 2019 Core	10/7/2019	
Windows Server 2019 Full	10/7/2019	
Windows Server 1909 Core	10/7/2019	12/8/2020

ブートストラップスクリプトの設定パラメータ

Windows ノードを作成すると、ノード上にさまざまなパラメータが設定できるスクリプトができます。設定によりますが、このスクリプトはノード上の次のような場所で確認できます。`C:\Program Files\Amazon\EKS\Start-EKSBootstrap.ps1`。カスタムパラメータ値は、ブートストラップスクリプトの引数として指定できます。例えば、起動テンプレートのユーザーデータを更新できます。詳細については、「Amazon EC2 ユーザーデータ」を参照してください。

このスクリプトには次のコマンドラインパラメータが含まれます。

- `-EKSClusterName` - このワーカーノードを参加させるための Amazon EKS クラスター名を指定します。
- `-KubeletExtraArgs` - kubelet の追加引数を指定します (オプション)。
- `-KubeProxyExtraArgs` — kube-proxy の追加引数を指定します (オプション)。

- -APIServerEndpoint — Amazon EKS クラスター API サーバーエンドポイントを指定します (オプション)。-Base64ClusterCA と共に使用した場合のみ有効となります。Get-EKSCluster の呼び出しをバイパスします。
- -Base64ClusterCA — Base64 でエンコードされたクラスターの CA の内容を指定します (オプション)。-APIServerEndpoint と共に使用した場合のみ有効となります。Get-EKSCluster の呼び出しをバイパスします。
- -DNSClusterIP — クラスター内の DNS クエリに使用する IP アドレスを上書きします (オプション)。プライマリインターフェイスの IP アドレスに基づく 10.100.0.10 または 172.20.0.10 のデフォルトです。
- -ServiceCIDR— クラスターサービスの宛先となる Kubernetes サービスの IP アドレス範囲をオーバーライドします。プライマリインターフェイスの IP アドレスに基づく 172.20.0.0/16 または 10.100.0.0/16 のデフォルトです。
- -ExcludedSnatCIDRs — 送信元ネットワークアドレス変換 (SNAT) から除外する IPv4 CIDR のリストです。つまり、VPC アドレスで呼び出し可能なポッドプライベート IP は、アウトバウンドトラフィック用のインスタンス ENI のプライマリ IPv4 アドレスの IP アドレスに変換されません。デフォルトでは、Amazon EKS Windows ノードの VPC の IPv4 CIDR が追加されます。このパラメータに CIDR を指定すると、指定した CIDR も除外されます。詳細については、「Pods の SNAT」を参照してください。

コマンドラインパラメータに加えて、いくつかの環境変数パラメータを指定することもできます。コマンドラインパラメータを指定する場合、そのパラメータはそれぞれの環境変数よりも優先されます。ブートストラップスクリプトはマシンスコープの変数のみを読み取るため、環境変数はマシン (またはシステム) スコープとして定義する必要があります。

このスクリプトでは、次の環境変数が考慮されます。

- SERVICE_IPV4_CIDR — 定義については、ServiceCIDR コマンドラインパラメータを参照してください。
- EXCLUDED_SNAT_CIDRS — コンマ区切りの文字列にする必要があります。定義については ExcludedSnatCIDRs コマンドラインパラメータを参照してください。

セルフマネージド型 Windows Server 2022 ノードを **eksctl** で起動する

以下の **test-windows-2022**.yaml は、Windows Server 2022 をセルフマネージド型ノードとして実行するためのリファレンスとして使用できます。*example value* をすべて自分の値に置き換えてください。

> **Note**
>
> セルフマネージド型 Windows Server 2022 ノードを実行するには、eksctl バージョン 0.116.0 以降を使用する必要があります。

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: windows-2022-cluster
  region: region-code
  version: '1.30'

nodeGroups:
  - name: windows-ng
    instanceType: m5.2xlarge
    amiFamily: WindowsServer2022FullContainer
    volumeSize: 100
    minSize: 2
    maxSize: 3
  - name: linux-ng
    amiFamily: AmazonLinux2
    minSize: 2
    maxSize: 3
```

ノードグループは、次のコマンドを使用して作成できます。

```
eksctl create cluster -f test-windows-2022.yaml
```

gMSA 認証サポート

Amazon EKS Windows Pods は、さまざまなタイプのグループ管理サービスアカウント (gMSA) 認証が可能です。

- Amazon EKS は認証用の Active Directory ドメインアイデンティティをサポートしています。ドメイン参加型 gMSA の詳細については、AWS ブログの「Amazon EKS Windowspods の Windows 認証」を参照してください。
- Amazon EKS では、非ドメイン参加型 Windows ノードがポータブルなユーザーアイデンティティを使用して gMSA 認証情報を取得できるようにするプラグインが提供されています。ドメイ

ンレス gMSA の詳細については、AWS ブログの「Amazon EKS Windowspods のドメインレス Windows 認証」を参照してください。

キャッシュされたコンテナイメージ

Amazon EKS Windows 最適化 AMI には、containerd のランタイムのためにキャッシュされたコンテナイメージが含まれます。コンテナイメージは、Amazon 管理ビルドコンポーネントを使用してカスタム AMI を構築するときにキャッシュされます。詳細については、「Amazon 管理ビルドコンポーネントを使用する」を参照してください。

containerd ランタイムのキャッシュされたコンテナイメージは以下のとおりです。

- amazonaws.com/eks/pause-windows
- mcr.microsoft.com/windows/nanoserver
- mcr.microsoft.com/windows/servercore

詳細情報

Amazon EKS 最適化 Windows AMI の詳細については、以下のセクションを参照してください。

- Windows をマネージド型ノードグループと使用するには、「マネージド型ノードグループ」を参照してください。
- セルフマネージド型 Windows ノードを起動するには、「セルフマネージド型 Windows ノードの起動」を参照してください。
- バージョンについては、「Amazon ECS に最適化された Windows AMI バージョン」を参照してください。
- Amazon EKS 最適化 Windows AMI の最新の ID を取得するには、「Amazon EKS 最適化 Windows AMI ID の取得」を参照してください。
- Amazon EC2 Image Builder を使用して、カスタムの Amazon EKS 最適化 Windows AMI を作成するには、「Amazon EKS に最適化されたカスタム Windows AMI の作成」を参照してください。
- ベストプラクティスについては、「EKS ベストプラクティスガイド」の「Amazon EKS optimized Windows AMI management」を参照してください。

Amazon ECS に最適化された Windows AMI バージョン

> **Important**
>
> AWS によって公開されている Amazon EKS 最適化 Windows AMI の延長サポートは、Kubernetes バージョン `1.23` では利用できませんが、Kubernetes バージョン `1.24` 以降では利用できます。

このトピックでは、Amazon EKS 最適化の Windows AMI のバージョンと、それに対応する `kubelet`、`containerd`、および `csi-proxy` のバージョンを示します。

AMI ID を含む、Amazon EKS 最適化 AMI のメタデータでは、各バリアントをプログラム的に取得することができます。詳細については、「Amazon EKS 最適化 Windows AMI ID の取得」を参照してください。

各 AMI は、Kubernetes のバージョンと AMI のリリース日によって次の形式でバージョン付けされています。

```
k8s_major_version.k8s_minor_version-release_date
```

> **Note**
>
> Amazon EKS マネージド型ノードグループは、Windows AMI の 2022 年 11 月以降のリリースをサポートしています。

Amazon EKS 最適化 Windows Server 2022 Core AMI

次の表は、Amazon EKS 最適化 Windows Server 2022 Core AMI の現在および以前のバージョンを示しています。

Kubernetes version 1.30

Kubernetes バージョン **1.30**

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Kubernetes version 1.29

Kubernetes バージョン **1.29**

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.29-2024.05.15	1.29.3	1.7.11	1.1.2	containerd を 1.7.11 にアップグレードしました。kubelet を 1.29.3 にアップグレードしました。
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	containerd を 1.6.28 にアップグレードしました。CNI と csi-proxy を golang 1.22.1 を使って再構築しました。
1.29-2024.03.12	1.29.0	1.6.25	1.1.2	
1.29-2024.02.13	1.29.0	1.6.25	1.1.2	

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.29-2024.02.06	1.29.0	1.6.25	1.1.2	一時停止画像が kubelet ガベージコレクション処理によって誤って削除されてしまうバグを修正しました。
1.29-2024.01.11	1.29.0	1.6.18	1.1.2	Windows Server 2022 Core AMI でスタンドアロン Windows Update KB5034439 を除外しました。KB は、Amazon EKS に最適化された Windows AMI に含まれていない、個別の WinRE パーティションのある Windows インストールにのみ適用されます。

Kubernetes version 1.28

Kubernetes バージョン **1.28**

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.28-2024.05.14	1.28.8	1.6.28	1.1.2	containerd を 1.6.28 にアップグレードしました。kubelet を 1.28.8 にアップグレードしました。
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	containerd を 1.6.25 にアップグレードしました。CNI と csi-proxy を golang

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
				1.22.1 を使って再構築しました。
1.28-2024.03.12	1.28.5	1.6.18	1.1.2	
1.28-2024.02.13	1.28.5	1.6.18	1.1.2	
1.28-2024.01.11	1.28.5	1.6.18	1.1.2	Windows Server 2022 Core AMI でスタンドアロン Windows Update KB5034439 を除外しました。KB は、Amazon EKS に最適化された Windows AMI に含まれていない、個別の WinRE パーティションのある Windows インストールにのみ適用されます。
1.28-2023.12.12	1.28.3	1.6.18	1.1.2	
1.28-2023.11.14	1.28.3	1.6.18	1.1.2	CVE-2023-5528 のパッチが含まれます。
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	containerd を 1.6.18 にアップグレードしました。新しいブートストラップスクリプト環境変数 (SERVICE_IPV4_CIDR と EXCLUDED_SNAT_CIDRS) を追加しました。
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	kubelet のセキュリティ勧告を修正しました。

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

Kubernetes version 1.27

Kubernetes バージョン **1.27**

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.27-2024.05.14	1.27.12	1.6.28	1.1.2	containerd を 1.6.28 にアップグレードしました。kubelet を 1.27.12 にアップグレードしました。
1.27-2024.04.09	1.27.9	1.6.25	1.1.2	containerd を 1.6.25 にアップグレードしました。CNI と csi-proxy を golang 1.22.1 を使って再構築しました。
1.27-2024.03.12	1.27.9	1.6.18	1.1.2	
1.27-2024.02.13	1.27.9	1.6.18	1.1.2	
1.27-2024.01.11	1.27.9	1.6.18	1.1.2	Windows Server 2022 Core AMI でスタンドアロン Windows Update KB5034439 を除外しました。KB は、Amazon EKS に最

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
				適化された Windows AMI に含まれていない、個別の WinRE パーティションのある Windows インストールにのみ適用されます。
1.27-2023.12.12	1.27.7	1.6.18	1.1.2	
1.27-2023.11.14	1.27.7	1.6.18	1.1.2	CVE-2023-5528 のパッチが含まれます。
1.27-2023.10.19	1.27.6	1.6.18	1.1.2	containerd を 1.6.18 にアップグレードしました。新しいブートストラップスクリプト環境変数 (SERVICE_IPV4_CIDR と EXCLUDED_SNAT_CIDRS) を追加しました。
1.27-2023-09.27	1.27.6	1.6.6	1.1.2	kubelet のセキュリティ勧告を修正しました。
1.27-2023.09.12	1.27.4	1.6.6	1.1.2	Kubernetes API サーバーから Pod IP アドレスを取得する Kubernetes コネクタバイナリを使用するように、Amazon VPC CNI プラグインをアップグレードしました。プルリクエスト #100 をマージしました。
1.27-2023.08.17	1.27.4	1.6.6	1.1.2	CVE-2023-3676 、CVE-2023-3893 、および CVE-2023-3955 のパッチが含まれます。

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.27-2023.08.08	1.27.3	1.6.6	1.1.1	
1.27-2023.07.11	1.27.3	1.6.6	1.1.1	
1.27-2023.06.20	1.27.1	1.6.6	1.1.1	DNS サフィックス検索リストに正しく入力されない問題を解決しました。
1.27-2023.06.14	1.27.1	1.6.6	1.1.1	CNI のホストポートマッピングのサポートが追加されました。プルリクエスト #93 をマージしました。
1.27-2023.06.06	1.27.1	1.6.6	1.1.1	ノードがプライベート Amazon ECR イメージの取り込みに失敗する原因となっていた containers-roadmap 問題 #2042 を修正しました。
1.27-2023.05.17	1.27.1	1.6.6	1.1.1	

Kubernetes version 1.26

Kubernetes バージョン **1.26**

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.26-2024.05.14	1.26.15	1.6.28	1.1.2	containerd を 1.6.28 にアップグレードしました。kubelet を 1.26.15 にアップグレードしました。
1.26-2024.04.09	1.26.12	1.6.25	1.1.2	containerd を 1.6.25 にアップグレードしました。CNI と csi-proxy を golang 1.22.1 を使って再構築しました。
1.26-2024.03.12	1.26.12	1.6.18	1.1.2	
1.26-2024.02.13	1.26.12	1.6.18	1.1.2	
1.26-2024.01.11	1.26.12	1.6.18	1.1.2	Windows Server 2022 Core AMI でスタンドアロン Windows Update KB5034439 を除外しました。KB は、Amazon EKS に最適化された Windows AMI に含まれていない、個別の WinRE パーティションのある Windows インストールにのみ適用されます。
1.26-2023.12.12	1.26.10	1.6.18	1.1.2	

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.26-2023.11.14	1.26.10	1.6.18	1.1.2	CVE-2023-5528 のパッチが含まれます。
1.26-2023.10.19	1.26.9	1.6.18	1.1.2	containerd を 1.6.18 にアップグレードしました。kubelet を 1.26.9 にアップグレードしました。新しいブートストラップスクリプト環境変数 (SERVICE_IPV4_CIDR と EXCLUDED_SNAT_CIDRS) を追加しました。
1.26-2023.09.12	1.26.7	1.6.6	1.1.2	Kubernetes API サーバーから Pod IP アドレスを取得する Kubernetes コネクタバイナリを使用するように、Amazon VPC CNI プラグインをアップグレードしました。プルリクエスト #100 をマージしました。
1.26-2023.08.17	1.26.7	1.6.6	1.1.2	CVE-2023-3676 、CVE-2023-3893 、および CVE-2023-3955 のパッチが含まれます。
1.26-2023.08.08	1.26.6	1.6.6	1.1.1	
1.26-2023.07.11	1.26.6	1.6.6	1.1.1	
1.26-2023.06.20	1.26.4	1.6.6	1.1.1	DNS サフィックス検索リストに正しく入力されない問題を解決しました。

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.26-2023.06.14	1.26.4	1.6.6	1.1.1	Kubernetes を 1.26.4 にアップグレードしました。CNI のホストポートマッピングのサポートが追加されました。プルリクエスト #93 をマージしました。
1.26-2023.05.09	1.26.2	1.6.6	1.1.1	ノードの再起動後にポッドでネットワーク接続の問題 #1126 を引き起こすバグを修正しました。新しいブートストラップスクリプトの設定パラメータ (ExcludedSnatCIDRs) を導入しました。
1.26-2023.04.26	1.26.2	1.6.6	1.1.1	
1.26-2023.04.11	1.26.2	1.6.6	1.1.1	サービスクラッシュ時の kubelet および kube-proxy の回復メカニズムを追加しました。
1.26-2023.03.24	1.26.2	1.6.6	1.1.1	

Kubernetes version 1.25

Kubernetes バージョン **1.25**

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.25-2024.05.14	1.25.16	1.6.28	1.1.2	containerd を 1.6.28 にアップグレードしました。
1.25-2024.04.09	1.25.16	1.6.25	1.1.2	containerd を 1.6.25 にアップグレードしました。CNI と csi-proxy を golang 1.22.1 を使って再構築しました。
1.25-2024.03.12	1.25.16	1.6.18	1.1.2	
1.25-2024.02.13	1.25.16	1.6.18	1.1.2	
1.25-2024.01.11	1.25.16	1.6.18	1.1.2	Windows Server 2022 Core AMI でスタンドアロン Windows Update KB5034439 を除外しました。KB は、Amazon EKS に最適化された Windows AMI に含まれていない、個別の WinRE パーティションのある Windows インストールにのみ適用されます。
1.25-2023.12.12	1.25.15	1.6.18	1.1.2	
1.25-2023.11.14	1.25.15	1.6.18	1.1.2	CVE-2023-5528 のパッチが含まれます。

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.25-2023.10.19	1.25.14	1.6.18	1.1.2	containerd を 1.6.18 にアップグレードしました。kubelet を 1.25.14 にアップグレードしました。新しいブートストラップスクリプト環境変数 (SERVICE_IPV4_CIDR と EXCLUDED_SNAT_CIDRS) を追加しました。
1.25-2023.09.12	1.25.12	1.6.6	1.1.2	Kubernetes API サーバーから Pod IP アドレスを取得する Kubernetes コネクタバイナリを使用するように、Amazon VPC CNI プラグインをアップグレードしました。プルリクエスト #100 をマージしました。
1.25-2023.08.17	1.25.12	1.6.6	1.1.2	CVE-2023-3676 、CVE-2023-3893 、および CVE-2023-3955 のパッチが含まれます。
1.25-2023.08.08	1.25.9	1.6.6	1.1.1	
1.25-2023.07.11	1.25.9	1.6.6	1.1.1	
1.25-2023.06.20	1.25.9	1.6.6	1.1.1	DNS サフィックス検索リストに正しく入力されない問題を解決しました。

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.25-2023.06.14	1.25.9	1.6.6	1.1.1	Kubernetes を 1.25.9 にアップグレードしました。CNI のホストポートマッピングのサポートが追加されました。プルリクエスト #93 をマージしました。
1.25-2023.05.09	1.25.7	1.6.6	1.1.1	ノードの再起動後にポッドでネットワーク接続の問題 #1126 を引き起こすバグを修正しました。新しいブートストラップスクリプトの設定パラメータ (ExcludedSnatCIDRs) を導入しました。
1.25-2023.04.11	1.25.7	1.6.6	1.1.1	サービスクラッシュ時の kubelet および kube-proxy の回復メカニズムを追加しました。
1.25-2023.03.27	1.25.6	1.6.6	1.1.1	Amazon EKS 上で Windows コンテナの gMSA 認証を容易にするため、「ドメインレス gMSA プラグイン」をインストールしました。
1.25-2023.03.20	1.25.6	1.6.6	1.1.1	
1.25-2023.02.14	1.25.6	1.6.6	1.1.1	

Kubernetes version 1.24

Kubernetes バージョン **1.24**

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.24-2024.05.14	1.24.17	1.6.28	1.1.2	containerd を 1.6.28 にアップグレードしました。
1.24-2024.04.09	1.24.17	1.6.25	1.1.2	containerd を 1.6.25 にアップグレードしました。CNI と csi-proxy を golang 1.22.1 を使って再構築しました。
1.24-2024.03.12	1.24.17	1.6.18	1.1.2	
1.24-2024.02.13	1.24.17	1.6.18	1.1.2	
1.24-2024.01.11	1.24.17	1.6.18	1.1.2	Windows Server 2022 Core AMI でスタンドアロン Windows Update KB5034439 を除外しました。KB は、Amazon EKS に最適化された Windows AMI に含まれていない、個別の WinRE パーティションのある Windows インストールにのみ適用されます。
1.24-2023.12.12	1.24.17	1.6.18	1.1.2	
1.24-2023.11.14	1.24.17	1.6.18	1.1.2	CVE-2023-5528 のパッチが含まれます。

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.24-2023.10.19	1.24.17	1.6.18	1.1.2	containerd を 1.6.18 にアップグレードしました。kubelet を 1.24.17 にアップグレードしました。新しいブートストラップスクリプト環境変数 (SERVICE_IPV4_CIDR と EXCLUDED_SNAT_CIDRS) を追加しました。
1.24-2023.09.12	1.24.16	1.6.6	1.1.2	Kubernetes API サーバーから Pod IP アドレスを取得する Kubernetes コネクタバイナリを使用するように、Amazon VPC CNI プラグインをアップグレードしました。プルリクエスト #100 をマージしました。
1.24-2023.08.17	1.24.16	1.6.6	1.1.2	CVE-2023-3676 、CVE-2023-3893 、および CVE-2023-3955 のパッチが含まれます。
1.24-2023.08.08	1.24.13	1.6.6	1.1.1	
1.24-2023.07.11	1.24.13	1.6.6	1.1.1	
1.24-2023.06.20	1.24.13	1.6.6	1.1.1	DNS サフィックス検索リストに正しく入力されない問題を解決しました。

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.24-2023.06.14	1.24.13	1.6.6	1.1.1	Kubernetes を 1.24.13 にアップグレードしました。CNI のホストポートマッピングのサポートが追加されました。プルリクエスト #93 をマージしました。
1.24-2023.05.09	1.24.7	1.6.6	1.1.1	ノードの再起動後にポッドでネットワーク接続の問題 #1126 を引き起こすバグを修正しました。新しいブートストラップスクリプトの設定パラメータ (ExcludedSnatCIDRs) を導入しました。
1.24-2023.04.11	1.24.7	1.6.6	1.1.1	サービスクラッシュ時の kubelet および kube-proxy の回復メカニズムを追加しました。
1.24-2023.03.27	1.24.7	1.6.6	1.1.1	Amazon EKS 上で Windows コンテナの gMSA 認証を容易にするため、「ドメインレス gMSA プラグイン」をインストールしました。
1.24-2023.03.20	1.24.7	1.6.6	1.1.1	1.24.10 については kube-proxy で問題が報告されているため、Kubernetes バージョンは 1.24.7 にダウングレードされました。
1.24-2023.02.14	1.24.10	1.6.6	1.1.1	

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.24-2023.01.23	1.24.7	1.6.6	1.1.1	
1.24-2023.01.11	1.24.7	1.6.6	1.1.1	
1.24-2022.12.13	1.24.7	1.6.6	1.1.1	
1.24-2022.10.11	1.24.7	1.6.6	1.1.1	

Amazon EKS 最適化 Windows Server 2022 Full AMI

次の表は、Amazon EKS 最適化 Windows Server 2022 Full AMI の現在および以前のバージョンを示しています。

Kubernetes version 1.30

Kubernetes バージョン **1.30**

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Kubernetes version 1.29

Kubernetes バージョン **1.29**

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.29-2024.05.15	1.29.3	1.7.11	1.1.2	containerd を 1.7.11 にアップグレードしました。kubelet を 1.29.3 にアップグレードしました。
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	containerd を 1.6.28 にアップグレードしました。CNI と csi-proxy を golang 1.22.1 を使って再構築しました。
1.29-2024.03.12	1.29.0	1.6.25	1.1.2	
1.29-2024.02.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.06	1.29.0	1.6.25	1.1.2	一時停止画像が kubelet ガベージコレクション処理によって誤って削除されてしまうバグを修正しました。
1.29-2024.01.09	1.29.0	1.6.18	1.1.2	

Kubernetes version 1.28

Kubernetes バージョン **1.28**

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.28-2024.05.14	1.28.8	1.6.28	1.1.2	containerd を 1.6.28 にアップグレードしました。kubelet を 1.28.8 にアップグレードしました。
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	containerd を 1.6.25 にアップグレードしました。CNI と csi-proxy を golang 1.22.1 を使って再構築しました。
1.28-2024.03.12	1.28.5	1.6.18	1.1.2	
1.28-2024.02.13	1.28.5	1.6.18	1.1.2	
1.28-2024.01.09	1.28.5	1.6.18	1.1.2	
1.28-2023.12.12	1.28.3	1.6.18	1.1.2	
1.28-2023.11.14	1.28.3	1.6.18	1.1.2	CVE-2023-5528 のパッチが含まれます。
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	containerd を 1.6.18 にアップグレードしました。新しいブートストラップスクリプト環境変数 (SERVICE_IPV4_CIDR

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
				と EXCLUDED_SNAT_CIDRS) を追加しました。
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	kubelet のセキュリティ勧告を修正しました。
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

Kubernetes version 1.27

Kubernetes バージョン **1.27**

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.27-2024.05.14	1.27.12	1.6.28	1.1.2	containerd を 1.6.28 にアップグレードしました。kubelet を 1.27.12 にアップグレードしました。
1.27-2024.04.09	1.27.9	1.6.25	1.1.2	containerd を 1.6.25 にアップグレードしました。CNI と csi-proxy を golang 1.22.1 を使って再構築しました。
1.27-2024.03.12	1.27.9	1.6.18	1.1.2	

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.27-2024.02.13	1.27.9	1.6.18	1.1.2	
1.27-2024.01.09	1.27.9	1.6.18	1.1.2	
1.27-2023.12.12	1.27.7	1.6.18	1.1.2	
1.27-2023.11.14	1.27.7	1.6.18	1.1.2	CVE-2023-5528 のパッチが含まれます。
1.27-2023.10.19	1.27.6	1.6.18	1.1.2	containerd を 1.6.18 にアップグレードしました。新しいブートストラップスクリプト環境変数 (SERVICE_IPV4_CIDR と EXCLUDED_SNAT_CIDRS) を追加しました。
1.27-2023-09.27	1.27.6	1.6.6	1.1.2	kubelet のセキュリティ勧告を修正しました。
1.27-2023.09.12	1.27.4	1.6.6	1.1.2	Kubernetes API サーバーから Pod IP アドレスを取得する Kubernetes コネクタバイナリを使用するように、Amazon VPC CNI プラグインをアップグレードしました。プルリクエスト #100 をマージしました。
1.27-2023.08.17	1.27.4	1.6.6	1.1.2	CVE-2023-3676 、CVE-2023-3893 、および CVE-2023-3955 のパッチが含まれます。

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.27-2023.08.08	1.27.3	1.6.6	1.1.1	
1.27-2023.07.11	1.27.3	1.6.6	1.1.1	
1.27-2023.06.20	1.27.1	1.6.6	1.1.1	DNS サフィックス検索リストに正しく入力されない問題を解決しました。
1.27-2023.06.14	1.27.1	1.6.6	1.1.1	CNI のホストポートマッピングのサポートが追加されました。プルリクエスト #93 をマージしました。
1.27-2023.06.06	1.27.1	1.6.6	1.1.1	ノードがプライベート Amazon ECR イメージの取り込みに失敗する原因となっていた containers-roadmap 問題 #2042 を修正しました。
1.27-2023.05.18	1.27.1	1.6.6	1.1.1	

Kubernetes version 1.26

Kubernetes バージョン **1.26**

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.26-2024.05.14	1.26.15	1.6.28	1.1.2	containerd を 1.6.28 にアップグレードしました。kubelet を 1.26.15 にアップグレードしました。
1.26-2024.04.09	1.26.12	1.6.25	1.1.2	containerd を 1.6.25 にアップグレードしました。CNI と csi-proxy を golang 1.22.1 を使って再構築しました。
1.26-2024.03.12	1.26.12	1.6.18	1.1.2	
1.26-2024.02.13	1.26.12	1.6.18	1.1.2	
1.26-2024.01.09	1.26.12	1.6.18	1.1.2	
1.26-2023.12.12	1.26.10	1.6.18	1.1.2	
1.26-2023.11.14	1.26.10	1.6.18	1.1.2	CVE-2023-5528 のパッチが含まれます。
1.26-2023.10.19	1.26.9	1.6.18	1.1.2	containerd を 1.6.18 にアップグレードしました。kubelet を 1.26.9 にアップグレードしました。新しいブートストラップスクリプト環境変

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
				数 (SERVICE_IPV4_CIDR と EXCLUDED_SNAT_CIDRS) を追加しました。
1.26-2023.09.12	1.26.7	1.6.6	1.1.2	Kubernetes API サーバーから Pod IP アドレスを取得する Kubernetes コネクタバイナリを使用するように、Amazon VPC CNI プラグインをアップグレードしました。プルリクエスト #100 をマージしました。
1.26-2023.08.17	1.26.7	1.6.6	1.1.2	CVE-2023-3676 、CVE-2023-3893 、および CVE-2023-3955 のパッチが含まれます。
1.26-2023.08.08	1.26.6	1.6.6	1.1.1	
1.26-2023.07.11	1.26.6	1.6.6	1.1.1	
1.26-2023.06.20	1.26.4	1.6.6	1.1.1	DNS サフィックス検索リストに正しく入力されない問題を解決しました。
1.26-2023.06.14	1.26.4	1.6.6	1.1.1	Kubernetes を 1.26.4 にアップグレードしました。CNI のホストポートマッピングのサポートが追加されました。プルリクエスト #93 をマージしました。

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.26-2023.05.09	1.26.2	1.6.6	1.1.1	ノードの再起動後にポッドでネットワーク接続の問題 #1126 を引き起こすバグを修正しました。新しいブートストラップスクリプトの設定パラメータ (ExcludedSnatCIDRs) を導入しました。
1.26-2023.04.26	1.26.2	1.6.6	1.1.1	
1.26-2023.04.11	1.26.2	1.6.6	1.1.1	サービスクラッシュ時の kubelet および kube-proxy の回復メカニズムを追加しました。
1.26-2023.03.24	1.26.2	1.6.6	1.1.1	

Kubernetes version 1.25

Kubernetes バージョン **1.25**

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.25-2024.05.14	1.25.16	1.6.28	1.1.2	containerd を 1.6.28 にアップグレードしました。
1.25-2024.04.09	1.25.16	1.6.25	1.1.2	containerd を 1.6.25 にアップグレードしました。CNI と csi-proxy を golang

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
				1.22.1 を使って再構築しました。
1.25-2024.03.12	1.25.16	1.6.18	1.1.2	
1.25-2024.02.13	1.25.16	1.6.18	1.1.2	
1.25-2024.01.09	1.25.16	1.6.18	1.1.2	
1.25-2023.12.12	1.25.15	1.6.18	1.1.2	
1.25-2023.11.14	1.25.15	1.6.18	1.1.2	CVE-2023-5528 のパッチが含まれます。
1.25-2023.10.19	1.25.14	1.6.18	1.1.2	containerd を 1.6.18 にアップグレードしました。kubelet を 1.25.14 にアップグレードしました。新しいブートストラップスクリプト環境変数 (SERVICE_IPV4_CIDR と EXCLUDED_SNAT_CIDRS) を追加しました。

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.25-2023.09.12	1.25.12	1.6.6	1.1.2	Kubernetes API サーバーから Pod IP アドレスを取得する Kubernetes コネクタバイナリを使用するように、Amazon VPC CNI プラグインをアップグレードしました。プルリクエスト #100 をマージしました。
1.25-2023.08.17	1.25.12	1.6.6	1.1.2	CVE-2023-3676 、CVE-2023-3893 、および CVE-2023-3955 のパッチが含まれます。
1.25-2023.08.08	1.25.9	1.6.6	1.1.1	
1.25-2023.07.11	1.25.9	1.6.6	1.1.1	
1.25-2023.06.20	1.25.9	1.6.6	1.1.1	DNS サフィックス検索リストに正しく入力されない問題を解決しました。
1.25-2023.06.14	1.25.9	1.6.6	1.1.1	Kubernetes を 1.25.9 にアップグレードしました。CNI のホストポートマッピングのサポートが追加されました。プルリクエスト #93 をマージしました。

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.25-2023.05.09	1.25.7	1.6.6	1.1.1	ノードの再起動後にポッドでネットワーク接続の問題 #1126 を引き起こすバグを修正しました。新しいブートストラップスクリプトの設定パラメータ (ExcludedSnatCIDRs) を導入しました。
1.25-2023.04.11	1.25.7	1.6.6	1.1.1	サービスクラッシュ時の kubelet および kube-proxy の回復メカニズムを追加しました。
1.25-2023.03.27	1.25.6	1.6.6	1.1.1	Amazon EKS 上で Windows コンテナの gMSA 認証を容易にするため、「ドメインレス gMSA プラグイン」をインストールしました。
1.25-2023.03.20	1.25.6	1.6.6	1.1.1	
1.25-2023.02.14	1.25.6	1.6.6	1.1.1	

Kubernetes version 1.24

Kubernetes バージョン **1.24**

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.24-2024.05.14	1.24.17	1.6.28	1.1.2	containerd を 1.6.28 にアップグレードしました。
1.24-2024.04.09	1.24.17	1.6.25	1.1.2	containerd を 1.6.25 にアップグレードしました。CNI と csi-proxy を golang 1.22.1 を使って再構築しました。
1.24-2024.03.12	1.24.17	1.6.18	1.1.2	
1.24-2024.02.13	1.24.17	1.6.18	1.1.2	
1.24-2024.01.09	1.24.17	1.6.18	1.1.2	
1.24-2023.12.12	1.24.17	1.6.18	1.1.2	
1.24-2023.11.14	1.24.17	1.6.18	1.1.2	CVE-2023-5528 のパッチが含まれます。
1.24-2023.10.19	1.24.17	1.6.18	1.1.2	containerd を 1.6.18 にアップグレードしました。kubelet を 1.24.17 にアップグレードしました。新しいブートストラップスクリプト環境変数 (SERVICE_IPV4_CIDR

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
				と EXCLUDED_SNAT_CIDRS) を追加しました。
1.24-2023.09.12	1.24.16	1.6.6	1.1.2	Kubernetes API サーバーから Pod IP アドレスを取得する Kubernetes コネクタバイナリを使用するように、Amazon VPC CNI プラグインをアップグレードしました。プルリクエスト #100 をマージしました。
1.24-2023.08.17	1.24.16	1.6.6	1.1.2	CVE-2023-3676 、CVE-2023-3893 、および CVE-2023-3955 のパッチが含まれます。
1.24-2023.08.08	1.24.13	1.6.6	1.1.1	
1.24-2023.07.11	1.24.13	1.6.6	1.1.1	
1.24-2023.06.20	1.24.13	1.6.6	1.1.1	DNS サフィックス検索リストに正しく入力されない問題を解決しました。
1.24-2023.06.14	1.24.13	1.6.6	1.1.1	Kubernetes を 1.24.13 にアップグレードしました。CNI のホストポートマッピングのサポートが追加されました。プルリクエスト #93 をマージしました。

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.24-2023.05.09	1.24.7	1.6.6	1.1.1	ノードの再起動後にポッドでネットワーク接続の問題 #1126 を引き起こすバグを修正しました。新しいブートストラップスクリプトの設定パラメータ (ExcludedSnatCIDRs) を導入しました。
1.24-2023.04.11	1.24.7	1.6.6	1.1.1	サービスクラッシュ時の kubelet および kube-proxy の回復メカニズムを追加しました。
1.24-2023.03.27	1.24.7	1.6.6	1.1.1	Amazon EKS 上で Windows コンテナの gMSA 認証を容易にするため、「ドメインレス gMSA プラグイン」をインストールしました。
1.24-2023.03.20	1.24.7	1.6.6	1.1.1	1.24.10 については kube-proxy で問題が報告されているため、Kubernetes バージョンは 1.24.7 にダウングレードされました。
1.24-2023.02.14	1.24.10	1.6.6	1.1.1	
1.24-2023.01.23	1.24.7	1.6.6	1.1.1	
1.24-2023.01.11	1.24.7	1.6.6	1.1.1	

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.24-2022.12.14	1.24.7	1.6.6	1.1.1	
1.24-2022.10.11	1.24.7	1.6.6	1.1.1	

Amazon EKS に最適化された Windows サーバー 2019 コア AMI

次の表は、Amazon EKS 最適化 Windows Server 2019 Core AMI の現在および以前のバージョンを示しています。

Kubernetes version 1.30

Kubernetes バージョン **1.30**

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Kubernetes version 1.29

Kubernetes バージョン **1.29**

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.29-2024.05.15	1.29.3	1.7.11	1.1.2	containerd を 1.7.11 にアップグレードしました。kubelet を 1.29.3 にアップグレードしました。
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	containerd を 1.6.28 にアップグレードしました。CNI と csi-proxy を golang 1.22.1 を使って再構築しました。
1.29-2024.03.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.06	1.29.0	1.6.25	1.1.2	一時停止画像が kubelet ガベージコレクション処理によって誤って削除されてしまうバグを修正しました。
1.29-2024.01.09	1.29.0	1.6.18	1.1.2	

Kubernetes version 1.28

Kubernetes バージョン **1.28**

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.28-2024.05.14	1.28.8	1.6.28	1.1.2	containerd を 1.6.28 にアップグレードしました。kubelet を 1.28.8 にアップグレードしました。
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	containerd を 1.6.25 にアップグレードしました。CNI と csi-proxy を golang 1.22.1 を使って再構築しました。
1.28-2024.03.13	1.28.5	1.6.18	1.1.2	
1.28-2024.02.13	1.28.5	1.6.18	1.1.2	
1.28-2024.01.09	1.28.5	1.6.18	1.1.2	
1.28-2023.12.12	1.28.3	1.6.18	1.1.2	
1.28-2023.11.14	1.28.3	1.6.18	1.1.2	CVE-2023-5528 のパッチが含まれます。
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	containerd を 1.6.18 にアップグレードしました。新しいブートストラップスクリプト環境変数 (SERVICE_IPV4_CIDR

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
				と EXCLUDED_SNAT_CIDRS) を追加しました。
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	kubelet のセキュリティ勧告を修正しました。
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

Kubernetes version 1.27

Kubernetes バージョン **1.27**

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.27-2024.05.14	1.27.12	1.6.28	1.1.2	containerd を 1.6.28 にアップグレードしました。kubelet を 1.27.12 にアップグレードしました。
1.27-2024.04.09	1.27.9	1.6.25	1.1.2	containerd を 1.6.25 にアップグレードしました。CNI と csi-proxy を golang 1.22.1 を使って再構築しました。
1.27-2024.03.13	1.27.9	1.6.18	1.1.2	

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.27-2024.02.13	1.27.9	1.6.18	1.1.2	
1.27-2024.01.09	1.27.9	1.6.18	1.1.2	
1.27-2023.12.12	1.27.7	1.6.18	1.1.2	
1.27-2023.11.14	1.27.7	1.6.18	1.1.2	CVE-2023-5528 のパッチが含まれます。
1.27-2023.10.19	1.27.6	1.6.18	1.1.2	containerd を 1.6.18 にアップグレードしました。新しいブートストラップスクリプト環境変数 (SERVICE_IPV4_CIDR と EXCLUDED_SNAT_CIDRS) を追加しました。
1.27-2023-09.27	1.27.6	1.6.6	1.1.2	kubelet のセキュリティ勧告を修正しました。
1.27-2023.09.12	1.27.4	1.6.6	1.1.2	Kubernetes API サーバーから Pod IP アドレスを取得する Kubernetes コネクタバイナリを使用するように、Amazon VPC CNI プラグインをアップグレードしました。プルリクエスト #100 をマージしました。
1.27-2023.08.17	1.27.4	1.6.6	1.1.2	CVE-2023-3676 、CVE-2023-3893 、および CVE-2023-3955 のパッチが含まれます。

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.27-2023.08.08	1.27.3	1.6.6	1.1.1	
1.27-2023.07.11	1.27.3	1.6.6	1.1.1	
1.27-2023.06.20	1.27.1	1.6.6	1.1.1	DNS サフィックス検索リストに正しく入力されない問題を解決しました。
1.27-2023.06.14	1.27.1	1.6.6	1.1.1	CNI のホストポートマッピングのサポートが追加されました。プルリクエスト #93 をマージしました。
1.27-2023.06.06	1.27.1	1.6.6	1.1.1	ノードがプライベート Amazon ECR イメージの取り込みに失敗する原因となっていた containers-roadmap 問題 #2042 を修正しました。
11.27-2023.05.18	1.27.1	1.6.6	1.1.1	

Kubernetes version 1.26

Kubernetes バージョン **1.26**

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.26-2024.05.14	1.26.15	1.6.28	1.1.2	containerd を 1.6.28 にアップグレードしました。kubelet を 1.26.15 にアップグレードしました。
1.26-2024.04.09	1.26.12	1.6.25	1.1.2	containerd を 1.6.25 にアップグレードしました。CNI と csi-proxy を golang 1.22.1 を使って再構築しました。
1.26-2024.03.13	1.26.12	1.6.18	1.1.2	
1.26-2024.02.13	1.26.12	1.6.18	1.1.2	
1.26-2024.01.09	1.26.12	1.6.18	1.1.2	
1.26-2023.12.12	1.26.10	1.6.18	1.1.2	
1.26-2023.11.14	1.26.10	1.6.18	1.1.2	CVE-2023-5528 のパッチが含まれます。
1.26-2023.10.19	1.26.9	1.6.18	1.1.2	containerd を 1.6.18 にアップグレードしました。kubelet を 1.26.9 にアップグレードしました。新しいブートストラップスクリプト環境変

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
				数 (SERVICE_IPV4_CIDR と EXCLUDED_SNAT_CIDRS) を追加しました。
1.26-2023.09.12	1.26.7	1.6.6	1.1.2	Kubernetes API サーバーから Pod IP アドレスを取得する Kubernetes コネクタバイナリを使用するように、Amazon VPC CNI プラグインをアップグレードしました。プルリクエスト #100 をマージしました。
1.26-2023.08.17	1.26.7	1.6.6	1.1.2	CVE-2023-3676 、CVE-2023-3893 、および CVE-2023-3955 のパッチが含まれます。
1.26-2023.08.08	1.26.6	1.6.6	1.1.1	
1.26-2023.07.11	1.26.6	1.6.6	1.1.1	
1.26-2023.06.20	1.26.4	1.6.6	1.1.1	DNS サフィックス検索リストに正しく入力されない問題を解決しました。
1.26-2023.06.14	1.26.4	1.6.6	1.1.1	Kubernetes を 1.26.4 にアップグレードしました。CNI のホストポートマッピングのサポートが追加されました。プルリクエスト #93 をマージしました。

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.26-2023.05.09	1.26.2	1.6.6	1.1.1	ノードの再起動後にポッドでネットワーク接続の問題 #1126 を引き起こすバグを修正しました。新しいブートストラップスクリプトの設定パラメータ (ExcludedSnatCIDRs) を導入しました。
1.26-2023.04.26	1.26.2	1.6.6	1.1.1	
1.26-2023.04.11	1.26.2	1.6.6	1.1.1	サービスクラッシュ時の kubelet および kube-proxy の回復メカニズムを追加しました。
1.26-2023.03.24	1.26.2	1.6.6	1.1.1	

Kubernetes version 1.25

Kubernetes バージョン **1.25**

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.25-2024.05.14	1.25.16	1.6.28	1.1.2	containerd を 1.6.28 にアップグレードしました。
1.25-2024.04.09	1.25.16	1.6.25	1.1.2	containerd を 1.6.25 にアップグレードしました。CNI と csi-proxy を golang

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
				1.22.1 を使って再構築しました。
1.25-2024.03.13	1.25.16	1.6.18	1.1.2	
1.25-2024.02.13	1.25.16	1.6.18	1.1.2	
1.25-2024.01.09	1.25.16	1.6.18	1.1.2	
1.25-2023.12.12	1.25.15	1.6.18	1.1.2	
1.25-2023.11.14	1.25.15	1.6.18	1.1.2	CVE-2023-5528 のパッチが含まれます。
1.25-2023.10.19	1.25.14	1.6.18	1.1.2	containerd を 1.6.18 にアップグレードしました。kubelet を 1.25.14 にアップグレードしました。新しいブートストラップスクリプト環境変数 (SERVICE_IPV4_CIDR と EXCLUDED_SNAT_CIDRS) を追加しました。

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.25-2023.09.12	1.25.12	1.6.6	1.1.2	Kubernetes API サーバーから Pod IP アドレスを取得する Kubernetes コネクタバイナリを使用するように、Amazon VPC CNI プラグインをアップグレードしました。プルリクエスト #100 をマージしました。
1.25-2023.08.17	1.25.12	1.6.6	1.1.2	CVE-2023-3676 、CVE-2023-3893 、および CVE-2023-3955 のパッチが含まれます。
1.25-2023.08.08	1.25.9	1.6.6	1.1.1	
1.25-2023.07.11	1.25.9	1.6.6	1.1.1	
1.25-2023.06.20	1.25.9	1.6.6	1.1.1	DNS サフィックス検索リストに正しく入力されない問題を解決しました。
1.25-2023.06.14	1.25.9	1.6.6	1.1.1	Kubernetes を 1.25.9 にアップグレードしました。CNI のホストポートマッピングのサポートが追加されました。プルリクエスト #93 をマージしました。

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.25-2023.05.09	1.25.7	1.6.6	1.1.1	ノードの再起動後にポッドでネットワーク接続の問題 #1126 を引き起こすバグを修正しました。新しいブートストラップスクリプトの設定パラメータ (ExcludedSnatCIDRs) を導入しました。
1.25-2023.04.11	1.25.7	1.6.6	1.1.1	サービスクラッシュ時の kubelet および kube-proxy の回復メカニズムを追加しました。
1.25-2023.03.27	1.25.6	1.6.6	1.1.1	Amazon EKS 上で Windows コンテナの gMSA 認証を容易にするため、「ドメインレス gMSA プラグイン」をインストールしました。
1.25-2023.03.20	1.25.6	1.6.6	1.1.1	
1.25-2023.02.14	1.25.6	1.6.6	1.1.1	

Kubernetes version 1.24

Kubernetes バージョン **1.24**

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.24-2024.05.14	1.24.17	1.6.28	1.1.2	containerd を 1.6.28 にアップグレードしました。
1.24-2024.04.09	1.24.17	1.6.25	1.1.2	containerd を 1.6.25 にアップグレードしました。CNI と csi-proxy を golang 1.22.1 を使って再構築しました。
1.24-2024.03.13	1.24.17	1.6.18	1.1.2	
1.24-2024.02.13	1.24.17	1.6.18	1.1.2	
1.24-2024.01.09	1.24.17	1.6.18	1.1.2	
1.24-2023.12.12	1.24.17	1.6.18	1.1.2	
1.24-2023.11.14	1.24.17	1.6.18	1.1.2	CVE-2023-5528 のパッチが含まれます。
1.24-2023.10.19	1.24.17	1.6.18	1.1.2	containerd を 1.6.18 にアップグレードしました。kubelet を 1.24.17 にアップグレードしました。新しいブートストラップスクリプト環境変数 (SERVICE_IPV4_CIDR

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
				と EXCLUDED_SNAT_CIDRS) を追加しました。
1.24-2023.09.12	1.24.16	1.6.6	1.1.2	Kubernetes API サーバーから Pod IP アドレスを取得する Kubernetes コネクタバイナリを使用するように、Amazon VPC CNI プラグインをアップグレードしました。プルリクエスト #100 をマージしました。
1.24-2023.08.17	1.24.16	1.6.6	1.1.2	CVE-2023-3676 、CVE-2023-3893 、および CVE-2023-3955 のパッチが含まれます。
1.24-2023.08.08	1.24.13	1.6.6	1.1.1	
1.24-2023.07.11	1.24.13	1.6.6	1.1.1	
1.24-2023.06.20	1.24.13	1.6.6	1.1.1	DNS サフィックス検索リストに正しく入力されない問題を解決しました。
1.24-2023.06.14	1.24.13	1.6.6	1.1.1	Kubernetes を 1.24.13 にアップグレードしました。CNI のホストポートマッピングのサポートが追加されました。プルリクエスト #93 をマージしました。

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.24-2023.05.09	1.24.7	1.6.6	1.1.1	ノードの再起動後にポッドでネットワーク接続の問題 #1126 を引き起こすバグを修正しました。新しいブートストラップスクリプトの設定パラメータ (ExcludedSnatCIDRs) を導入しました。
1.24-2023.04.11	1.24.7	1.6.6	1.1.1	サービスクラッシュ時の kubelet および kube-proxy の回復メカニズムを追加しました。
1.24-2023.03.27	1.24.7	1.6.6	1.1.1	Amazon EKS 上で Windows コンテナの gMSA 認証を容易にするため、「ドメインレス gMSA プラグイン」をインストールしました。
1.24-2023.03.20	1.24.7	1.6.6	1.1.1	1.24.10 については kube-proxy で問題が報告されているため、Kubernetes バージョンは 1.24.7 にダウングレードされました。
1.24-2023.02.14	1.24.10	1.6.6	1.1.1	
1.24-2023.01.23	1.24.7	1.6.6	1.1.1	
1.24-2023.01.11	1.24.7	1.6.6	1.1.1	

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.24-2022.12.13	1.24.7	1.6.6	1.1.1	
1.24-2022.11.08	1.24.7	1.6.6	1.1.1	

Amazon EKS 最適化 Windows Server 2019 Full AMI

次の表は、Amazon EKS 最適化 Windows Server 2019 Full AMI の現在および以前のバージョンを示しています。

Kubernetes version 1.30

Kubernetes バージョン **1.30**

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Kubernetes version 1.29

Kubernetes バージョン **1.29**

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.29-2024.05.15	1.29.3	1.7.11	1.1.2	containerd を 1.7.11 にアップグレードしました。kubelet を 1.29.3 にアップグレードしました。
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	containerd を 1.6.28 にアップグレードしました。CNI と csi-proxy を golang 1.22.1 を使って再構築しました。
1.29-2024.03.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.06	1.29.0	1.6.25	1.1.2	一時停止画像が kubelet ガベージコレクション処理によって誤って削除されてしまうバグを修正しました。
1.29-2024.01.09	1.29.0	1.6.18	1.1.2	

Kubernetes version 1.28

Kubernetes バージョン **1.28**

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.28-2024.05.14	1.28.8	1.6.28	1.1.2	containerd を 1.6.28 にアップグレードしました。kubelet を 1.28.8 にアップグレードしました。
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	containerd を 1.6.25 にアップグレードしました。CNI と csi-proxy を golang 1.22.1 を使って再構築しました。
1.28-2024.03.13	1.28.5	1.6.18	1.1.2	
1.28-2024.02.13	1.28.5	1.6.18	1.1.2	
1.28-2024.01.09	1.28.5	1.6.18	1.1.2	
1.28-2023.12.12	1.28.3	1.6.18	1.1.2	
1.28-2023.11.14	1.28.3	1.6.18	1.1.2	CVE-2023-5528 のパッチが含まれます。
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	containerd を 1.6.18 にアップグレードしました。新しいブートストラップスクリプト環境変数 (SERVICE_IPV4_CIDR

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
				と EXCLUDED_SNAT_CIDRS) を追加しました。
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	kubelet のセキュリティ勧告を修正しました。
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

Kubernetes version 1.27

Kubernetes バージョン **1.27**

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.27-2024.05.14	1.27.12	1.6.28	1.1.2	containerd を 1.6.28 にアップグレードしました。kubelet を 1.27.12 にアップグレードしました。
1.27-2024.04.09	1.27.9	1.6.25	1.1.2	containerd を 1.6.25 にアップグレードしました。CNI と csi-proxy を golang 1.22.1 を使って再構築しました。
1.27-2024.03.13	1.27.9	1.6.18	1.1.2	

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.27-2024.02.13	1.27.9	1.6.18	1.1.2	
1.27-2024.01.09	1.27.9	1.6.18	1.1.2	
1.27-2023.12.12	1.27.7	1.6.18	1.1.2	
1.27-2023.11.14	1.27.7	1.6.18	1.1.2	CVE-2023-5528 のパッチが含まれます。
1.27-2023.10.19	1.27.6	1.6.18	1.1.2	containerd を 1.6.18 にアップグレードしました。新しいブートストラップスクリプト環境変数 (SERVICE_IPV4_CIDR と EXCLUDED_SNAT_CIDRS) を追加しました。
1.27-2023-09.27	1.27.6	1.6.6	1.1.2	kubelet のセキュリティ勧告を修正しました。
1.27-2023.09.12	1.27.4	1.6.6	1.1.2	Kubernetes API サーバーから Pod IP アドレスを取得する Kubernetes コネクタバイナリを使用するように、Amazon VPC CNI プラグインをアップグレードしました。プルリクエスト #100 をマージしました。
1.27-2023.08.17	1.27.4	1.6.6	1.1.2	CVE-2023-3676 、CVE-2023-3893 、および CVE-2023-3955 のパッチが含まれます。

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.27-2023.08.08	1.27.3	1.6.6	1.1.1	
1.27-2023.07.11	1.27.3	1.6.6	1.1.1	
1.27-2023.06.20	1.27.1	1.6.6	1.1.1	DNS サフィックス検索リストに正しく入力されない問題を解決しました。
1.27-2023.06.14	1.27.1	1.6.6	1.1.1	CNI のホストポートマッピングのサポートが追加されました。プルリクエスト #93 をマージしました。
1.27-2023.06.06	1.27.1	1.6.6	1.1.1	ノードがプライベート Amazon ECR イメージの取り込みに失敗する原因となっていた containers-roadmap 問題 #2042 を修正しました。
1.27-2023.05.17	1.27.1	1.6.6	1.1.1	

Kubernetes version 1.26

Kubernetes バージョン **1.26**

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.26-2024.05.14	1.26.15	1.6.28	1.1.2	containerd を 1.6.28 にアップグレードしました。kubelet を 1.26.15 にアップグレードしました。
1.26-2024.04.09	1.26.12	1.6.25	1.1.2	containerd を 1.6.25 にアップグレードしました。CNI と csi-proxy を golang 1.22.1 を使って再構築しました。
1.26-2024.03.13	1.26.12	1.6.18	1.1.2	
1.26-2024.02.13	1.26.12	1.6.18	1.1.2	
1.26-2024.01.09	1.26.12	1.6.18	1.1.2	
1.26-2023.12.12	1.26.10	1.6.18	1.1.2	
1.26-2023.11.14	1.26.10	1.6.18	1.1.2	CVE-2023-5528 のパッチが含まれます。
1.26-2023.10.19	1.26.9	1.6.18	1.1.2	containerd を 1.6.18 にアップグレードしました。kubelet を 1.26.9 にアップグレードしました。新しいブートストラップスクリプト環境変

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
				数 (SERVICE_IPV4_CIDR と EXCLUDED_SNAT_CIDRS) を追加しました。
1.26-2023.09.12	1.26.7	1.6.6	1.1.2	Kubernetes API サーバーから Pod IP アドレスを取得する Kubernetes コネクタバイナリを使用するように、Amazon VPC CNI プラグインをアップグレードしました。プルリクエスト #100 をマージしました。
1.26-2023.08.17	1.26.7	1.6.6	1.1.2	CVE-2023-3676 、CVE-2023-3893 、および CVE-2023-3955 のパッチが含まれます。
1.26-2023.08.08	1.26.6	1.6.6	1.1.1	
1.26-2023.07.11	1.26.6	1.6.6	1.1.1	
1.26-2023.06.20	1.26.4	1.6.6	1.1.1	DNS サフィックス検索リストに正しく入力されない問題を解決しました。
1.26-2023.06.14	1.26.4	1.6.6	1.1.1	Kubernetes を 1.26.4 にアップグレードしました。CNI のホストポートマッピングのサポートが追加されました。プルリクエスト #93 をマージしました。

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.26-2023.05.09	1.26.2	1.6.6	1.1.1	ノードの再起動後にポッドでネットワーク接続の問題 #1126 を引き起こすバグを修正しました。新しいブートストラップスクリプトの設定パラメータ (ExcludedSnatCIDRs) を導入しました。
1.26-2023.04.26	1.26.2	1.6.6	1.1.1	
1.26-2023.04.11	1.26.2	1.6.6	1.1.1	サービスクラッシュ時の kubelet および kube-proxy の回復メカニズムを追加しました。
1.26-2023.03.24	1.26.2	1.6.6	1.1.1	

Kubernetes version 1.25

Kubernetes バージョン **1.25**

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.25-2024.05.14	1.25.16	1.6.28	1.1.2	containerd を 1.6.28 にアップグレードしました。
1.25-2024.04.09	1.25.16	1.6.25	1.1.2	containerd を 1.6.25 にアップグレードしました。CNI と csi-proxy を golang

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
				1.22.1 を使って再構築しました。
1.25-2024.03.13	1.25.16	1.6.18	1.1.2	
1.25-2024.02.13	1.25.16	1.6.18	1.1.2	
1.25-2024.01.09	1.25.16	1.6.18	1.1.2	
1.25-2023.12.12	1.25.15	1.6.18	1.1.2	
1.25-2023.11.14	1.25.15	1.6.18	1.1.2	CVE-2023-5528 のパッチが含まれます。
1.25-2023.10.19	1.25.14	1.6.18	1.1.2	containerd を 1.6.18 にアップグレードしました。kubelet を 1.25.14 にアップグレードしました。新しいブートストラップスクリプト環境変数 (SERVICE_IPV4_CIDR と EXCLUDED_SNAT_CIDRS) を追加しました。

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.25-2023.09.12	1.25.12	1.6.6	1.1.2	Kubernetes API サーバーから Pod IP アドレスを取得する Kubernetes コネクタバイナリを使用するように、Amazon VPC CNI プラグインをアップグレードしました。プルリクエスト #100 をマージしました。
1.25-2023.08.17	1.25.12	1.6.6	1.1.2	CVE-2023-3676 、CVE-2023-3893 、および CVE-2023-3955 のパッチが含まれます。
1.25-2023.08.08	1.25.9	1.6.6	1.1.1	
1.25-2023.07.11	1.25.9	1.6.6	1.1.1	
1.25-2023.06.20	1.25.9	1.6.6	1.1.1	DNS サフィックス検索リストに正しく入力されない問題を解決しました。
1.25-2023.06.14	1.25.9	1.6.6	1.1.1	Kubernetes を 1.25.9 にアップグレードしました。CNI のホストポートマッピングのサポートが追加されました。プルリクエスト #93 をマージしました。

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.25-2023.05.09	1.25.7	1.6.6	1.1.1	ノードの再起動後にポッドでネットワーク接続の問題 #1126 を引き起こすバグを修正しました。新しいブートストラップスクリプトの設定パラメータ (ExcludedSnatCIDRs) を導入しました。
1.25-2023.04.11	1.25.7	1.6.6	1.1.1	サービスクラッシュ時の kubelet および kube-proxy の回復メカニズムを追加しました。
1.25-2023.03.27	1.25.6	1.6.6	1.1.1	Amazon EKS 上で Windows コンテナの gMSA 認証を容易にするため、「ドメインレス gMSA プラグイン」をインストールしました。
1.25-2023.03.20	1.25.6	1.6.6	1.1.1	
1.25-2023.02.14	1.25.6	1.6.6	1.1.1	

Kubernetes version 1.24

Kubernetes バージョン **1.24**

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.24-2024.05.14	1.24.17	1.6.28	1.1.2	containerd を 1.6.28 にアップグレードしました。
1.24-2024.04.09	1.24.17	1.6.25	1.1.2	containerd を 1.6.25 にアップグレードしました。CNI と csi-proxy を golang 1.22.1 を使って再構築しました。
1.24-2024.03.13	1.24.17	1.6.18	1.1.2	
1.24-2024.02.13	1.24.17	1.6.18	1.1.2	
1.24-2024.01.09	1.24.17	1.6.18	1.1.2	
1.24-2023.12.12	1.24.17	1.6.18	1.1.2	
1.24-2023.11.14	1.24.17	1.6.18	1.1.2	CVE-2023-5528 のパッチが含まれます。
1.24-2023.10.19	1.24.17	1.6.18	1.1.2	containerd を 1.6.18 にアップグレードしました。kubelet を 1.24.17 にアップグレードしました。新しいブートストラップスクリプト環境変数 (SERVICE_IPV4_CIDR

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
				と EXCLUDED_SNAT_CIDRS) を追加しました。
1.24-2023.09.12	1.24.16	1.6.6	1.1.2	Kubernetes API サーバーから Pod IP アドレスを取得する Kubernetes コネクタバイナリを使用するように、Amazon VPC CNI プラグインをアップグレードしました。プルリクエスト #100 をマージしました。
1.24-2023.08.17	1.24.16	1.6.6	1.1.2	CVE-2023-3676 、CVE-2023-3893 、および CVE-2023-3955 のパッチが含まれます。
1.24-2023.08.08	1.24.13	1.6.6	1.1.1	
1.24-2023.07.11	1.24.13	1.6.6	1.1.1	
1.24-2023.06.21	1.24.13	1.6.6	1.1.1	DNS サフィックス検索リストに正しく入力されない問題を解決しました。
1.24-2023.06.14	1.24.13	1.6.6	1.1.1	Kubernetes を 1.24.13 にアップグレードしました。CNI のホストポートマッピングのサポートが追加されました。プルリクエスト #93 をマージしました。

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.24-2023.05.09	1.24.7	1.6.6	1.1.1	ノードの再起動後にポッドでネットワーク接続の問題 #1126 を引き起こすバグを修正しました。新しいブートストラップスクリプトの設定パラメータ (ExcludedSnatCIDRs) を導入しました。
1.24-2023.04.11	1.24.7	1.6.6	1.1.1	サービスクラッシュ時の kubelet および kube-proxy の回復メカニズムを追加しました。
1.24-2023.03.27	1.24.7	1.6.6	1.1.1	Amazon EKS 上で Windows コンテナの gMSA 認証を容易にするため、「ドメインレス gMSA プラグイン」をインストールしました。
1.24-2023.03.20	1.24.7	1.6.6	1.1.1	1.24.10 については kube-proxy で問題が報告されているため、Kubernetes バージョンは 1.24.7 にダウングレードされました。
1.24-2023.02.14	1.24.10	1.6.6	1.1.1	
1.24-2023.01.23	1.24.7	1.6.6	1.1.1	
1.24-2023.01.11	1.24.7	1.6.6	1.1.1	

AMI のバージョン	**kubelet** バージョン	**containerd** バージョン	**csi-proxy** バージョン	リリースノート
1.24-2022.12.14	1.24.7	1.6.6	1.1.1	
1.24-2022.10.12	1.24.7	1.6.6	1.1.1	

Amazon EKS 最適化 Windows AMI ID の取得

AWS Systems Manager パラメータストア API をクエリすることで、 Amazon EKS 最適化 AMI の Amazon マシンイメージ (AMI、Amazon Machine Image) ID をプログラムで取得できます。この API から提供されるパラメータにより、Amazon EKS 最適化 AMI ID を手動で検索する必要がなくなります。Systems Manager Parameter Store API の詳細については、GetParameter を参照してください。使用する IAM プリンシパルには、Amazon EKS 最適化 AMI メタデータを取得するための `ssm:GetParameter` IAM アクセス許可が必要です。

サブパラメータ `image_id` を指定しながら次のコマンドを使用することで、推奨される最新の Amazon EKS 最適化 Windows AMI のイメージ ID を取得できます。*`1.30`* を、サポートされている任意の Amazon EKS バージョンで置き換えることができます。また、*`region-code`* を、AMI ID を必要とする Amazon EKS がサポートされているリージョンで置き換えることができます。Windows Server の完全な AMI ID を確認するには、*`Core`* を `Full` に置き換えます。Kubernetes バージョン `1.24` 以降の場合は、*`2019`* Server 2022 AMI ID を表示するには、`2022` を Windows に置き換えることができます。

```
aws ssm get-parameter --name /aws/service/ami-windows-latest/Windows_Server-2019-
English-Core-EKS_Optimized-1.30/image_id --region region-code --query "Parameter.Value"
 --output text
```

出力例は次のとおりです。

```
ami-1234567890abcdef0
```

Amazon EKS に最適化されたカスタム Windows AMI の作成

EC2 Image Builder を使用し、次のオプションのいずれかを使用して、カスタムの Amazon EKS 最適化 Windows AMI を作成できます。

- Amazon EKS 最適化 Windows AMI をベースとして使用する
- Amazon 管理ビルドコンポーネントを使用する

どちらの方法でも、独自の Image Builder レシピを作成する必要があります。詳細については、「Image Builder ユーザーガイド」の「イメージレシピの新しいバージョンの作成」を参照してください。

> **Important**
>
> 以下の eks の [Amazon 管理] コンポーネントには、CVE-2023-5528 のパッチが含まれます。
>
> - 1.24.3 以上
> - 1.25.2 以上
> - 1.26.2 以上
> - 1.27.0 以上
> - 1.28.0 以上

Amazon EKS 最適化 Windows AMI をベースとして使用する

このオプションは、カスタム Windows AMI を構築するのに推奨される方法です。当社が提供する Amazon EKS 最適化 Windows AMI は、Amazon 管理のビルドコンポーネントよりも頻繁に更新されます。

1. 新しい Image Builder レシピを始める
   a. https://console.aws.amazon.com/imagebuilder で、EC2 Image Builder コンソールを開きます。
   b. 左側のナビゲーションペインで、[イメージレシピ] を選択します。
   c. [イメージレシピの作成] を選択します。
2. [レシピの詳細] セクションで、[名前] と [バージョン] を入力します。

3. [ベースイメージ] セクションで Amazon EKS 最適化 Windows AMI の ID を指定します。

   a. [カスタム AMI ID を入力] を選択します。

   b. 必要な Windows OS バージョンの AMI ID を取得します。詳細については、「Amazon EKS 最適化 Windows AMI ID の取得」を参照してください。

   c. カスタム [AMI ID] を入力します。AMI ID が見つからない場合は、AMI ID の AWS リージョン がコンソールの右上に表示されている AWS リージョン と一致しているか確認してください。

4. (オプション) 最新のセキュリティアップデートを入手するには、`update-windows` コンポーネントを [ビルドコンポーネント -] セクションに追加します。

   a. [名前でコンポーネントを検索する] 検索ボックスの右側にあるドロップダウンリストから、[Amazon 管理] を選択します。

   b. [名前でコンポーネントを検索する] 検索ボックスに **`update-windows`** と入力します。

   c. **`update-windows`** の検索結果のチェックボックスを選択します。このコンポーネントは、オペレーティングシステムの最新 Windows パッチを含みます。

5. 必要な設定で残りのイメージレシピの入力を完了します。詳細については、「Image Builder ユーザーガイド」の「イメージレシピの新しいバージョンの作成 (コンソール)」を参照してください。

6. [レシピを作成する] を選択します。

7. 新しいイメージレシピを新規または既存のイメージパイプラインで使用します。イメージパイプラインが正常に実行されると、カスタム AMI が出力イメージとして一覧表示され、使用できるようになります。詳細については、「EC2 Image Builder コンソールウィザードを使用してイメージパイプラインを作成する」を参照してください。

Amazon 管理ビルドコンポーネントを使用する

Amazon EKS 最適化 Windows AMI をベースとして使用できない場合は、代わりに Amazon 管理ビルドコンポーネントを使用できます。このオプションは、サポートされている最新の Kubernetes バージョンよりも遅れる可能性があります。

1. 新しい Image Builder レシピを始める

   a. https://console.aws.amazon.com/imagebuilder で、EC2 Image Builder コンソールを開きます。

   b. 左側のナビゲーションペインで、[イメージレシピ] を選択します。

c. [イメージレシピの作成] を選択します。

2. [レシピの詳細] セクションで、[名前] と [バージョン] を入力します。
3. [ベースイメージ] のセクションで、カスタム AMI の作成に使用するオプションを決定します。

   - [管理イメージの選択] — [イメージオペレーティングシステム (OS)] で [Windows] を選択します。次に、[イメージのオリジン] について、以下のいずれかのオプションを選択します。
     - [クイックスタート (Amazon 管理)] – [イメージ名] のドロップダウンから、Amazon EKS がサポートされている Windows Server のバージョンを選択します。詳細については、「Amazon EKS 最適化 Windows AMI」を参照してください。
     - [ユーザー所有のイメージ] – [イメージ名] で、独自ライセンスを持つ独自のイメージの ARN を選択します。Amazon EKS コンポーネントをインストール済みのイメージを指定することはできません。
   - [カスタム AMI ID を入力] – [AMI ID] に、独自ライセンスを持つ AMI の ID を入力します。Amazon EKS コンポーネントをインストール済みのイメージを指定することはできません。
4. [ビルドコンポーネント - Windows] セクションで、次の操作を行います。

   a. [名前でコンポーネントを検索する] 検索ボックスの右側にあるドロップダウンリストから、[Amazon 管理] を選択します。
   b. [名前でコンポーネントを検索する] 検索ボックスに **eks** と入力します。
   c. たとえ目的のバージョンではない場合でも、**eks-optimized-ami-windows** 検索結果のチェックボックスを選択します。
   d. [名前でコンポーネントを検索する] 検索ボックスに **update-windows** と入力します。
   e. update-windows の検索結果のチェックボックスを選択します。このコンポーネントは、オペレーティングシステムの最新 Windows パッチを含みます。
5. [選択したコンポーネント] セクションで、次の手順を実行します。

   a. [**eks-optimized-ami-windows**] の [バージョニングオプション] を選択してください。
   b. [コンポーネントバージョンを指定する] を選択します。
   c. [コンポーネントバージョン] フィールドに ***version.x*** と入力し、*version* をサポートされている Kubernetes バージョンに置き換えます。バージョン番号の一部に *x* を入力すると、明示的に定義したバージョンの一部と一致する最新のコンポーネントバージョンが使用されます。コンソールの出力に注目してください。目的のバージョンが管理コンポーネントとして利用可能かどうかが示されます。ビルドコンポーネントでは、最新の Kubernetes バージョンを使用できない場合があります。使用できるバージョンの詳細については、

「eks-optimized-ami-windows コンポーネントのバージョンに関する情報を取得する」を参照してください。

> **Note**
>
> 次の eks-optimized-ami-windows ビルドコンポーネントのバージョンには、eksctl バージョン 0.129 以前のものが必要です。
>
> • 1.24.0

6. 必要な設定で残りのイメージレシピの入力を完了します。詳細については、「Image Builder ユーザーガイド」の「イメージレシピの新しいバージョンの作成 (コンソール)」を参照してください。
7. [レシピを作成する] を選択します。
8. 新しいイメージレシピを新規または既存のイメージパイプラインで使用します。イメージパイプラインが正常に実行されると、カスタム AMI が出力イメージとして一覧表示され、使用できるようになります。詳細については、「EC2 Image Builder コンソールウィザードを使用してイメージパイプラインを作成する」を参照してください。

eks-optimized-ami-windows コンポーネントのバージョンに関する情報を取得する

各コンポーネントにインストールされているものに関する特定の情報を取得できます。例えば、どの kubelet バージョンがインストールされているかを確認できます。コンポーネントには、Amazon EKS がサポートする Windows オペレーティングシステムのバージョンで機能テストが実施されます。詳細については、「リリースカレンダー」を参照してください。その他の Windows OS バージョンで、サポートされていない、またはサポート終了としてリストにあるものは、このコンポーネントと互換性がない可能性があります。

1. https://console.aws.amazon.com/imagebuilder で、EC2 Image Builder コンソールを開きます。
2. 左にあるナビゲーションペインで、[コンポーネント] をクリックします。
3. [名前でコンポーネントを検索する] 検索ボックスの右側にあるドロップダウンリストで、[自己所有] を [クイックスタート (Amazon 管理)] に変更します。
4. [名前でコンポーネントを検索] ボックスに **eks** と入力します。
5. (オプション) 最新バージョンを使用している場合は、[バージョン] 列を 2 回選択して降順にソートします。
6. 目的のバージョンの [**eks-optimized-ami-windows**] リンクを選択します。

結果のページの [説明] には、特定の情報が表示されます。

ストレージ

この章では、Amazon EKS クラスターのストレージオプションについて説明します。

トピック

Amazon EBS CSI ドライバー

Amazon Elastic Block Store (Amazon EBS) コンテナストレージインターフェイス (CSI) ドライバーは、Amazon EBS ボリュームのライフサイクルを、作成した Kubernetes ボリュームのストレージとして管理します。Amazon EBS CSI ドライバーは、ジェネリックエフェメラルボリュームとパーシステントボリュームという種類の Kubernetes ボリューム用の Amazon EBS ボリュームを作成します。

Amazon EBS CSI ドライバーを使用する際の考慮事項を以下に示します。

- Amazon EBS CSI プラグインでは、ユーザーに代わって AWS API の呼び出しを行うための IAM アクセス許可が必要です。詳細については、「Amazon EBS CSI ドライバー IAM ロールの作成」を参照してください。
- Amazon EBS ボリュームを Fargate Pods にマウントすることはできません。
- Amazon EBS CSI コントローラーは Fargate ノードで実行できますが、Amazon EBS CSI ノード DaemonSet は Amazon EC2 インスタンスでのみ実行できます。

Amazon EBS CSI ドライバーは、最初にクラスターを作成したときにインストールされません。ドライバーを使用するには、Amazon EKS アドオンまたはセルフマネージド型のアドオンとして追加する必要があります。

- Amazon EKS アドオンとして追加する方法については、Amazon EBS CSI ドライバーを Amazon EKS アドオンとして管理するを参照してください。
- セルフマネージド型のインストールとして追加する方法については、GitHub の「Amazon EBS のコンテナストレージインターフェイス (CSI) ドライバー」プロジェクトを参照してください。

どちらかの方法で CSI ドライバーをインストールした後、サンプルアプリケーションで機能をテストできます。詳細については、「サンプルアプリケーションをデプロイし、CSI ドライバーが動作していることを確認する」を参照してください。

Amazon EBS CSI ドライバー IAM ロールの作成

Amazon EBS CSI プラグインでは、ユーザーに代わって AWS API の呼び出しを行うための IAM アクセス許可が必要です。詳細については、GitHub の「ドライバー権限の設定」を参照してください。

> **Note**
>
> Pods は、IMDS へのアクセスをブロックする場合を除き、IAM ロールに割り当てられたアクセス許可にアクセスできます。詳細については、「Amazon EKS のセキュリティベストプラクティス」を参照してください。

前提条件

- 既存のクラスター。
- クラスター用の既存 AWS Identity and Access Management IAM OpenID Connect (OIDC) プロバイダー。既に存在しているかどうかを確認する、または作成するには「クラスターの IAM OIDC プロバイダーを作成する」を参照してください。

以下の手順は、IAM ロールを作成し、それに AWS マネージドポリシーをアタッチする方法を示しています。`eksctl`、AWS Management Console、または AWS CLI を使用できます。

> **Note**
>
> この手順には、ドライバーを Amazon EKS アドオンとして使用するための特定のステップが書かれています。ドライバーをセルフマネージドのアドオンとして使用するには、さまざま

なステップが必要です。詳細については、「GitHub」 の「ドライバー権限の設定」を参照してください。

eksctl

`eksctl` で Amazon EBS CSI プラグイン IAM ロールを作成するには

1. IAM ロールを作成して、ポリシーをアタッチします。AWS が AWS マネージドポリシーを維持しますが、ユーザーが独自のカスタムポリシーを作成することもできます。次のコマンドを使用して、IAM ロールを作成し、それに AWS マネージドポリシーをアタッチできます。*`my-cluster`* を自分のクラスター名に置き換えます。このコマンドは IAM ロールを作成して IAM ポリシーをアタッチする AWS CloudFormation スタックをデプロイします。クラスターが AWS GovCloud (米国東部) または AWS GovCloud (米国東部) の AWS リージョン にある場合は、`arn:aws:` を `arn:aws-us-gov:` に置き換えます。

   ```
   eksctl create iamserviceaccount \
       --name ebs-csi-controller-sa \
       --namespace kube-system \
       --cluster my-cluster \
       --role-name AmazonEKS_EBS_CSI_DriverRole \
       --role-only \
       --attach-policy-arn arn:aws:iam::aws:policy/service-role/
   AmazonEBSCSIDriverPolicy \
       --approve
   ```

2. Amazon EBS ボリュームでの暗号化にカスタム「KMS キー」を使用する場合、必要に応じて IAM ロールをカスタマイズします。例えば、以下を実行してください。

   a. 次のコードをコピーし、新しい *`kms-key-for-encryption-on-ebs`*`.json` ファイルに貼り付けます。*`custom-key-arn`* をカスタム「KMS キー ARN」に置き換えます。

      ```
      {
        "Version": "2012-10-17",
        "Statement": [
          {
            "Effect": "Allow",
            "Action": [
              "kms:CreateGrant",
              "kms:ListGrants",
              "kms:RevokeGrant"
      ```

```
      ],
      "Resource": ["custom-key-arn"],
      "Condition": {
        "Bool": {
          "kms:GrantIsForAWSResource": "true"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": ["custom-key-arn"]
    }
  ]
}
```

b. ポリシーを作成します。*KMS_Key_For_Encryption_On_EBS_Policy* は別の名前に変更できます。ただし、変更する場合は、必ず後の手順で変更してください。

```
aws iam create-policy \
  --policy-name KMS_Key_For_Encryption_On_EBS_Policy \
  --policy-document file://kms-key-for-encryption-on-ebs.json
```

c. 次のコマンドを使用して、IAM ポリシーをロールに添付します。*111122223333* をアカウントID に置き換えます。クラスターが AWS GovCloud (米国東部) または AWS GovCloud (米国東部) の AWS リージョン にある場合は、`arn:aws:` を `arn:aws-us-gov:` に置き換えます。

```
aws iam attach-role-policy \
  --policy-arn
 arn:aws:iam::111122223333:policy/KMS_Key_For_Encryption_On_EBS_Policy \
  --role-name AmazonEKS_EBS_CSI_DriverRole
```

AWS Management Console

AWS Management Console で Amazon EBS CSI プラグイン IAM ロールを作成するには

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. 左のナビゲーションペインで、[ロール] を選択します。
3. [ロール] ページで、[ロールの作成] を選択します。
4. [信頼されたエンティティを選択] ページで、以下の操作を実行します。
   a. [信頼されたエンティティの種類] セクションで、[ウェブ アイデンティティ] を選択します。
   b. [ID プロバイダー] で、(Amazon EKS の [概要] に示されているように) クラスターに [OpenID Connect プロバイダーの URL] を選択します。
   c. [対象者] で [sts.amazonaws.com] を選択します。
   d. [Next] を選択します。
5. [アクセス許可を追加] ページで、以下を実行します。
   a. [フィルタポリシー] ボックスに AmazonEBSCSIDriverPolicy と入力します。
   b. 検索で返された AmazonEBSCSIDriverPolicy の左にあるチェックボックスを選択します。
   c. [Next] を選択します。
6. [名前を付けて、レビューし、作成する] ページで、以下の操作を実行します。
   a. [ロール名] に、***AmazonEKS_EBS_CSI_DriverRole*** などのロールの一意の名前を入力します。
   b. [タグの追加 (オプション)] で、タグをキーバリューのペアとして添付して、メタデータをロールに追加します。IAM でのタグの使用に関する詳細については、『IAM ユーザーガイド』の「IAM リソースにタグを付ける」を参照してください。
   c. [ロールの作成] を選択します。
7. ロールが作成されたら、コンソールでロールを選択して編集用に開きます。
8. [信頼関係] タブを選択し、続いて [信頼ポリシーの編集] を選択します。
9. 次の行と似ている行を探してます。

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud":
 "sts.amazonaws.com"
```

前の行の末尾にカンマを追加し、前の行の後に次の行を追加します。*region-code* をクラスターのある AWS リージョン に置き換えます。*EXAMPLED539D4633E53DE1B71EXAMPLE* をクラスターの OIDC プロバイダー ID に置き換えます。

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub":
 "system:serviceaccount:kube-system:ebs-csi-controller-sa"
```

10. [ポリシーの更新] を選択して終了します。
11. Amazon EBS ボリュームでの暗号化にカスタム「KMS キー」を使用する場合、必要に応じて IAM ロールをカスタマイズします。例えば、以下を実行してください。

    a. 左のナビゲーションペインの [ポリシー] を選択します。

    b. [ポリシー] ページで、[ポリシーの作成] を選択します。

    c. [ポリシーの作成] ページで、[JSON] タブを選択します。

    d. 次のコードをコピーしてエディタに貼り付け、*custom-key-arn* をカスタム「KMS キー ARN」に置き換えます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:CreateGrant",
        "kms:ListGrants",
        "kms:RevokeGrant"
      ],
      "Resource": ["custom-key-arn"],
      "Condition": {
        "Bool": {
          "kms:GrantIsForAWSResource": "true"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
```

```
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": ["custom-key-arn"]
    }
  ]
}
```

e. [次へ: タグ] を選択します。

f. [タグの追加 (オプション)] ページで、[次へ: 確認] を選択します。

g. [名前] に、ポリシーの一意の名前を入力します (例えば、***KMS_Key_For_Encryption_On_EBS_Policy***)。

h. [ポリシーを作成] を選択します。

i. 左のナビゲーションペインで、[ロール] を選択します。

j. コンソールで [***AmazonEKS_EBS_CSI_DriverRole***] を選択し、編集用に開きます。

k. [アクセス許可を追加] ドロップダウンリストから [ポリシーをアタッチ] を選択します。

l. [フィルタポリシー] ボックスに *KMS_Key_For_Encryption_On_EBS_Policy* と入力します。

m. 検索で返された *KMS_Key_For_Encryption_On_EBS_Policy* の左にあるチェックボックスを選択します。

n. [ポリシーのアタッチ] を選択します。

AWS CLI

AWS CLI で Amazon EBS CSI プラグイン IAM ロールを作成するには

1. クラスターの OIDC プロバイダーの URL を表示します。*my-cluster* をクラスター名に置き換えます。コマンドの出力が None の場合は、「前提条件」を確認してください。

```
aws eks describe-cluster --name my-cluster --query
 "cluster.identity.oidc.issuer" --output text
```

出力例は次のとおりです。

```
https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

2. IAM ロールを作成して AssumeRoleWithWebIdentity アクションを付与します。

a. 次の内容を *aws-ebs-csi-driver-trust-policy*.json という名前のファイルにコピーします。*111122223333* はご使用のアカウントの ID に置き換えます。*EXAMPLED539D4633E53DE1B71EXAMPLE* と *region-code* を、前のステップで返された値に置き換えます。クラスターが AWS GovCloud (米国東部) または AWS GovCloud (米国東部) の AWS リージョン にある場合は、`arn:aws:` を `arn:aws-us-gov:` に置き換えます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:oidc-provider/
oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
          "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-
system:ebs-csi-controller-sa"
        }
      }
    }
  ]
}
```

b. ロールを作成します。*AmazonEKS_EBS_CSI_DriverRole* は別の名前に変更できます。変更する場合は、必ず後の手順で変更してください。

```
aws iam create-role \
  --role-name AmazonEKS_EBS_CSI_DriverRole \
  --assume-role-policy-document file://"aws-ebs-csi-driver-trust-
policy.json"
```

3. ポリシーをアタッチします。AWS が AWS マネージドポリシーを維持しますが、ユーザーが独自のカスタムポリシーを作成することもできます。次のコマンドを使用して、AWS マネージドポリシーをロールにアタッチします。クラスターが AWS GovCloud (米国東部) また

は AWS GovCloud (米国東部) の AWS リージョン にある場合は、`arn:aws:` を `arn:aws-us-gov:` に置き換えます。

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy \
  --role-name AmazonEKS_EBS_CSI_DriverRole
```

4. Amazon EBS ボリュームでの暗号化にカスタム「KMS キー」を使用する場合、必要に応じて IAM ロールをカスタマイズします。例えば、以下を実行してください。

   a. 次のコードをコピーし、新しい *kms-key-for-encryption-on-ebs*.json ファイルに貼り付けます。*custom-key-arn* をカスタム「KMS キー ARN」に置き換えます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:CreateGrant",
        "kms:ListGrants",
        "kms:RevokeGrant"
      ],
      "Resource": ["custom-key-arn"],
      "Condition": {
        "Bool": {
          "kms:GrantIsForAWSResource": "true"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": ["custom-key-arn"]
    }
  ]
```

```
}
```

b. ポリシーを作成します。*KMS_Key_For_Encryption_On_EBS_Policy* は別の名前に変更できます。ただし、変更する場合は、必ず後の手順で変更してください。

```
aws iam create-policy \
  --policy-name KMS_Key_For_Encryption_On_EBS_Policy \
  --policy-document file://kms-key-for-encryption-on-ebs.json
```

c. 次のコマンドを使用して、IAM ポリシーをロールに添付します。*111122223333* をアカウントID に置き換えます。クラスターが AWS GovCloud (米国東部) または AWS GovCloud (米国東部) の AWS リージョン にある場合は、`arn:aws:` を `arn:aws-us-gov:` に置き換えます。

```
aws iam attach-role-policy \
  --policy-arn
 arn:aws:iam::111122223333:policy/KMS_Key_For_Encryption_On_EBS_Policy \
  --role-name AmazonEKS_EBS_CSI_DriverRole
```

Amazon EBS CSI ドライバーの IAM ロールを作成したので、続けて「Amazon EBS CSI ドライバーアドオンの追加」に進めます。その手順でプラグインがデプロイされると、`ebs-csi-controller-sa` という名前のサービスアカウントを作成して使用するように設定されます。サービスアカウントは、必要な Kubernetes アクセス許可が割り当てられた Kubernetes `clusterrole` にバインドされます。

Amazon EBS CSI ドライバーを Amazon EKS アドオンとして管理する

セキュリティを向上させ作業量を減らすため、Amazon EBS CSI ドライバーを Amazon EKS アドオンとして管理できます。Amazon EKS アドオンについては、Amazon EKS アドオンを参照してください。Amazon EBS CSI アドオンを追加するには、Amazon EBS CSI ドライバーアドオンの追加の手順に従います。

Amazon EBS CSI アドオンを追加した場合は、Amazon EBS CSI ドライバーを Amazon EKS アドオンとして更新する および Amazon EBS CSI アドオンの削除 セクションの手順に従って管理できます。

前提条件

- 既存のクラスター。必要なプラットフォームのバージョンを確認するには、次のコマンドを実行します。

```
aws eks describe-addon-versions --addon-name aws-ebs-csi-driver
```

- クラスター用の既存 AWS Identity and Access Management IAM OpenID Connect (OIDC) プロバイダー。既に存在しているかどうかを確認する、または作成するには「クラスターの IAM OIDC プロバイダーを作成する」を参照してください。
- Amazon EBS CSI ドライバーの IAM ロール。この前提条件を満たしていない場合、アドオンをインストールして `kubectl describe pvc` を実行しようとすると、`could not create volume in EC2: UnauthorizedOperation` エラーとともに `failed to provision volume with StorageClass` が表示されます。詳細については、「Amazon EBS CSI ドライバー IAM ロールの作成」を参照してください。
- クラスター全体の制限された `PodSecurityPolicy` を使用している場合、デプロイのための十分な権限がアドオンに付与されていることを確認してください。各アドオン Pod に必要なアクセス許可については、GitHub の「関連するアドオンマニフェストの定義」を参照してください。

> **Important**
>
> Amazon EBS CSI ドライバーのスナップショット機能を使用するには、アドオンをインストールする前に外部の Snapshotter をインストールする必要があります。外部の Snapshotter のコンポーネントは、次の順序でインストールする必要があります。
>
> - `volumesnapshotclasses`、`volumesnapshots`、および `volumesnapshotcontents` 用の `CustomResourceDefinition` (CRD)
> - RBAC (`ClusterRole`、`ClusterRoleBinding` など)
> - コントローラーデプロイメント
>
> 詳細については、「GitHub」の「CSI Snapshotter」を参照してください。

Amazon EBS CSI ドライバーアドオンの追加

> **Important**
>
> Amazon EBS ドライバーを Amazon EKS アドオンとして追加する前に、クラスターにセルフマネージドバージョンのドライバーがインストールされていないことを確認してください。インストールされている場合は、GitHub の「セルフマネージド型 Amazon EBS CSI ドライバーのアンインストール」を参照してください。

`eksctl`、AWS Management Console、または AWS CLI を使用して、Amazon EBS CSI アドオンをクラスターに追加できます。

eksctl

`eksctl` を使用して Amazon EBS CSI アドオンを追加するには

以下のコマンドを実行します。*`my-cluster`* をクラスターの名前に、*`111122223333`* をアカウント ID に、*`AmazonEKS_EBS_CSI_DriverRole`* を前に作成した IAM ロールの名前に置き換えます。クラスターが AWS GovCloud (米国東部) または AWS GovCloud (米国東部) の AWS リージョン にある場合は、`arn:aws:` を `arn:aws-us-gov:` に置き換えます。

```
eksctl create addon --name aws-ebs-csi-driver --cluster my-cluster --service-
account-role-arn arn:aws:iam::111122223333:role/AmazonEKS_EBS_CSI_DriverRole --force
```

`--force` オプションを削除した状態で Amazon EKS アドオンを更新する際に、そのアドオン設定のいずれかが現状の設定と競合している場合、その更新は失敗します。この場合、競合を解決するためのエラーメッセージが表示されます。このオプションを指定する前に、自分が管理する必要がある設定を Amazon EKS アドオンが管理していないことを確認してください。これらの設定は、このオプションの指定により上書きされます。この設定のその他のオプションの詳細については、`eksctl` ドキュメントの「アドオン」を参照してください。Amazon EKS Kubernetes フィールド管理の詳細については、「 Kubernetes フィールド管理」を参照してください。

AWS Management Console

AWS Management Console を使用して Amazon EBS CSIアドオンを追加するには

1. Ahttps://console.aws.amazon.com/eks/home#/clusters で Amazon EKS コンソールを開きます。

2. 左のナビゲーションペインで [クラスター] を選択します。
3. Amazon EBS CSI アドオンを設定するクラスターの名前を選択します。
4. [アドオン] タブを選択します。
5. [その他のアドオンを入手] を選択します。
6. [アドオンを選択] ページで、次の操作を行います。
   a. [Amazon EKS アドオン] セクションで、[Amazon EBS CSI ドライバー] チェックボックスを選択します。
   b. [次へ] をクリックします。
7. [選択したアドオンセッティングの設定] ページで、次の操作を行います。
   a. 使用する [バージョン] を選択します。
   b. [IAM ロールを選択] で、Amazon EBS CSI ドライバーの IAM ポリシーをアタッチした IAM ロールの名前を選択します。
   c. (オプション) [オプションの構成設定] を展開できます。[コンフリクト解決方法] で [上書きする] を選択すると、既存のアドオンでの 1 つ以上の設定が、Amazon EKS アドオンの設定で上書きされます。このオプションが有効でない状態で既存の設定との競合が発生する場合は、オペレーションが失敗します。表示されたエラーメッセージを使用して、競合をトラブルシューティングできます。このオプションを選択する前に、自分で管理する必要のある設定が、Amazon EKS アドオンにより管理されていないことを確認してください。
   d. [Next] (次へ) をクリックします。
8. [確認と追加] ページで、[作成] を選択します。アドオンのインストールが完了した後、インストールしたアドオンが表示されます。

AWS CLI

AWS CLI を使用して Amazon EBS CSIアドオンを追加するには

以下のコマンドを実行します。*my-cluster* をクラスターの名前に置き換え、*111122223333* をアカウント ID に置き換え、*AmazonEKS_EBS_CSI_DriverRole* を前に作成したロールの名前に置き換えます。クラスターが AWS GovCloud (米国東部) または AWS GovCloud (米国東部) の AWS リージョン にある場合は、`arn:aws:` を `arn:aws-us-gov:` に置き換えます。

```
aws eks create-addon --cluster-name my-cluster --addon-name aws-ebs-csi-driver \
```

```
  --service-account-role-arn
arn:aws:iam::111122223333:role/AmazonEKS_EBS_CSI_DriverRole
```

Amazon EBS CSI ドライバーを Amazon EKS アドオンとして追加したので、続けて「サンプルアプリケーションをデプロイし、CSI ドライバーが動作していることを確認する」に進めます。この手順はストレージクラスの設定を含みます。

Amazon EBS CSI ドライバーを Amazon EKS アドオンとして更新する

Amazon EKS では、新しいバージョンがリリースされたり、新しい Kubernetes マイナーバージョンにクラスターを更新しても、クラスターで Amazon EBS CSI が自動的に更新されることはありません。既存のクラスターで Amazon EBS CSI を更新するには、まずはお客様が更新を開始してください。その後、Amazon EKS でアドオンが更新されます。

eksctl

`eksctl` を使用して Amazon EBS CSI アドオンを更新するには

1. Amazon EBS CSI アドオンの現在のバージョンを確認します。*my-cluster* をクラスター名に置き換えます。

   ```
   eksctl get addon --name aws-ebs-csi-driver --cluster my-cluster
   ```

   出力例は次のとおりです。

   ```
   NAME                    VERSION                 STATUS  ISSUES  IAMROLE
    UPDATE AVAILABLE
   aws-ebs-csi-driver      v1.11.2-eksbuild.1      ACTIVE  0
     v1.11.4-eksbuild.1
   ```

2. アドオンを、前のステップの出力で UPDATE AVAILABLE の下に表示されたバージョンに更新します。

   ```
   eksctl update addon --name aws-ebs-csi-driver --version v1.11.4-eksbuild.1 --
   cluster my-cluster \
     --service-account-role-arn
    arn:aws:iam::111122223333:role/AmazonEKS_EBS_CSI_DriverRole --force
   ```

--force オプションを削除した状態で Amazon EKS アドオンを更新する際に、そのアドオン設定のいずれかが現状の設定と競合している場合、その更新は失敗します。この場合、競合を解決するためのエラーメッセージが表示されます。このオプションを指定する前に、自分が管理する必要がある設定を Amazon EKS アドオンが管理していないことを確認してください。これらの設定は、このオプションの指定により上書きされます。この設定のその他のオプションの詳細については、eksctl ドキュメントの「アドオン」を参照してください。Amazon EKS Kubernetes フィールド管理の詳細については、「Kubernetes フィールド管理」を参照してください。

AWS Management Console

AWS Management Console を使用して Amazon EBS CSI アドオンを更新するには

1. Ahttps://console.aws.amazon.com/eks/home#/clusters で Amazon EKS コンソールを開きます。
2. 左のナビゲーションペインで [クラスター] を選択します。
3. Amazon EBS CSI アドオンを更新するクラスターの名前を選択します。
4. [アドオン] タブを選択します。
5. [Amazon EBS CSI ドライバー] を選択します。
6. [編集] を選択します。
7. [Amazon EBS CSI ドライバーを設定] ページで、次の操作を行います。
   a. 使用する [バージョン] を選択します。
   b. [IAM ロールを選択] で、Amazon EBS CSI ドライバーの IAM ポリシーをアタッチした IAM ロールの名前を選択します。
   c. (オプション) [オプションの構成設定] を展開し、必要に応じて変更することができます。
   d. [変更の保存] をクリックします。

AWS CLI

AWS CLI を使用して Amazon EBS CSI アドオンを更新するには

1. Amazon EBS CSI アドオンの現在のバージョンを確認します。*my-cluster* をクラスター名に置き換えます。

```
aws eks describe-addon --cluster-name my-cluster --addon-name aws-ebs-csi-driver
 --query "addon.addonVersion" --output text
```

出力例は次のとおりです。

```
v1.11.2-eksbuild.1
```

2. クラスターのバージョンで使用できる Amazon EBS CSI アドオンのバージョンを確認します。

```
aws eks describe-addon-versions --addon-name aws-ebs-csi-driver --kubernetes-
version 1.23 \
  --query "addons[].addonVersions[].[addonVersion,
 compatibilities[].defaultVersion]" --output text
```

出力例は次のとおりです。

```
v1.11.4-eksbuild.1
True
v1.11.2-eksbuild.1
False
```

下に True と表示されているバージョンは、アドオンの作成時にデプロイされたデフォルトのバージョンです。アドオン作成時にデプロイされたバージョンは、利用可能な最新バージョンではない場合があります。先ほどの出力では、アドオンの作成時に最新バージョンがデプロイされます。

3. アドオンを、前のステップの出力で True として返されたバージョンに更新します。出力に返されたバージョンであれば、さらに新しいバージョンに更新することもできます。

```
aws eks update-addon --cluster-name my-cluster --addon-name aws-ebs-csi-driver
 --addon-version v1.11.4-eksbuild.1 \
  --service-account-role-arn
 arn:aws:iam::111122223333:role/AmazonEKS_EBS_CSI_DriverRole --resolve-
conflicts PRESERVE
```

[##] オプションは、アドオンに設定したカスタム設定を保持します。この設定のその他のオプションの詳細については、「Amazon EKS コマンドラインリファレンス」の「更新

アドオン」を参照してください。Amazon EKS アドオン設定管理の詳細については、「Kubernetes フィールド管理」を参照してください。

Amazon EBS CSI アドオンの削除

Amazon EKS アドオンを削除するには、次の 2 つのオプションがあります。

- クラスター上のアドオンソフトウェアを保持する — このオプションでは、すべての設定の Amazon EKS 管理が削除されます。また、Amazon EKS による更新の通知機能や、更新の開始後の Amazon EKS アドオンの自動更新機能も削除されます。ただし、クラスター上のアドオンソフトウェアは保持されます。このオプションを選択すると、アドオンは Amazon EKS アドオンではなく、セルフマネージド型インストールになります。このオプションを使用すると、アドオンのダウンタイムは発生しません。この手順のコマンドには、このオプションを使用します。
- クラスターからアドオンソフトウェアを完全に削除する — クラスターから Amazon EKS アドオンを削除するのは、アドオンに依存するリソースがクラスター上にない場合のみとすることをお勧めします。このオプションを実行するには、この手順で使用するコマンドから `--preserve` を削除します。

アドオンに IAM アカウントが関連付けられている場合、その IAM アカウントは削除されません。

`eksctl`、AWS Management Console、または AWS CLI を使用して、Amazon EBS CSI アドオンを削除できます。

eksctl

`eksctl` を使用して Amazon EBS CSIアドオンを削除するには

`my-cluster` をクラスター名に置き換えた後、次のコマンドを実行します。

```
eksctl delete addon --cluster my-cluster --name aws-ebs-csi-driver --preserve
```

AWS Management Console

AWS Management Console を使用して Amazon EBS CSI アドオンを削除するには

1. Ahttps://console.aws.amazon.com/eks/home#/clusters で Amazon EKS コンソールを開きます。
2. 左のナビゲーションペインで [クラスター] を選択します。
3. Amazon EBS CSI アドオンを削除するクラスターの名前を選択します。

4. [アドオン] タブを選択します。
5. [Amazon EBS CSI ドライバー] を選択します。
6. [削除] を選択します。
7. [削除: aws-ebs-csi-driver] 確認ダイアログボックスで、以下の操作を行います。
   a. Amazon EKS でのアドオンの設定の管理を停止する場合は、[クラスターで保持する] を選択します。クラスターでアドオンソフトウェアを保持する場合は、この操作を実行します。これは、アドオンのすべての設定を自分で管理できるようにするためです。
   b. **aws-ebs-csi-driver** と入力します。
   c. [削除] を選択します。

AWS CLI

AWS CLI を使用して Amazon EBS CSI アドオンを削除するには

my-cluster をクラスター名に置き換えた後、次のコマンドを実行します。

```
aws eks delete-addon --cluster-name my-cluster --addon-name aws-ebs-csi-driver --
preserve
```

サンプルアプリケーションをデプロイし、CSI ドライバーが動作していることを確認する

CSI ドライバーの機能は、サンプルアプリケーションを使用してテストできます。このトピックでは、1 つの例を示しますが、以下の操作を行うこともできます。

- 外部の Snapshotter を使用してボリュームスナップショットを作成するサンプルアプリケーションをデプロイします。詳細については、「GitHub」の「Volume Snapshots」(ボリュームスナップショット) を参照してください。
- ボリュームのサイズ変更を使用するサンプルアプリケーションをデプロイします。詳細については、「GitHub」の「Volume Resizing」(ボリュームのサイズ変更) を参照してください。

この手順では、「Amazon EBS Container Storage Interface (CSI) ドライバー」GitHub リポジトリの「動的ボリュームプロビジョニング」の例を使用して、動的にプロビジョニングされた Amazon EBS ボリュームを消費します。

1. Amazon EBS Container Storage Interface (CSI) ドライバー GitHub リポジトリをローカルシステムにクローンします。

```
git clone https://github.com/kubernetes-sigs/aws-ebs-csi-driver.git
```

2. dynamic-provisioning サンプルディレクトリに移動します。

```
cd aws-ebs-csi-driver/examples/kubernetes/dynamic-provisioning/
```

3. (オプション) manifests/storageclass.yaml ファイルは、デフォルトで gp2 Amazon EBS ボリュームをプロビジョニングします。代わりに gp3 ボリュームを使用するには、type: gp3 を manifests/storageclass.yaml に追加します。

```
echo "parameters:
  type: gp3" >> manifests/storageclass.yaml
```

4. manifests ディレクトリから、ebs-sc ストレージクラス、ebs-claim 永続ボリュームクレーム、および app サンプルアプリケーションをデプロイします。

```
kubectl apply -f manifests/
```

5. ebs-sc ストレージクラスについて説明します。

```
kubectl describe storageclass ebs-sc
```

出力例は次のとおりです。

```
Name:            ebs-sc
IsDefaultClass:  No
Annotations:     kubectl.kubernetes.io/last-applied-
configuration={"apiVersion":"storage.k8s.io/v1","kind":"StorageClass","metadata":
{"annotations":{},"name":"ebs-
sc"},"provisioner":"ebs.csi.aws.com","volumeBindingMode":"WaitForFirstConsumer"}

Provisioner:           ebs.csi.aws.com
Parameters:            <none>
AllowVolumeExpansion:  <unset>
MountOptions:          <none>
ReclaimPolicy:         Delete
VolumeBindingMode:     WaitForFirstConsumer
```

```
Events:                <none>
```

> **Note**
>
> このストレージクラスは WaitForFirstConsumer ボリュームバインドモードを使用します。これは、Pod が永続的なボリュームリクエストを行うまで、ボリュームが動的にプロビジョニングされないことを意味します。詳細については、Kubernetes ドキュメントの「Volume Binding Mode」(ボリュームバインドモード) を参照してください。

6. デフォルトの名前空間の Pods をモニタリングします。数分後、app Pod のステータスが Running に変わります。

```
kubectl get pods --watch
```

Ctrl+C を入力して、シェルプロンプトに戻ります。

7. デフォルトの名前空間の永続的ボリュームを一覧表示します。default/ebs-claim クレームがある永続的なボリュームを探します。

```
kubectl get pv
```

出力例は次のとおりです。

```
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY
   STATUS   CLAIM               STORAGECLASS   REASON   AGE
pvc-37717cd6-d0dc-11e9-b17f-06fad4858a5a   4Gi        RWO            Delete
   Bound    default/ebs-claim   ebs-sc                  30s
```

8. 永続的ボリュームの詳細を表示します。pvc-*37717cd6-d0dc-11e9-b17f-06fad4858a5a* の部分は、前の手順の出力の値に置き換えます。

```
kubectl describe pv pvc-37717cd6-d0dc-11e9-b17f-06fad4858a5a
```

出力例は次のとおりです。

```
Name:              pvc-37717cd6-d0dc-11e9-b17f-06fad4858a5a
Labels:            <none>
Annotations:       pv.kubernetes.io/provisioned-by: ebs.csi.aws.com
Finalizers:        [kubernetes.io/pv-protection external-attacher/ebs-csi-aws-com]
```

```
StorageClass:      ebs-sc
Status:            Bound
Claim:             default/ebs-claim
Reclaim Policy:    Delete
Access Modes:      RWO
VolumeMode:        Filesystem
Capacity:          4Gi
Node Affinity:
  Required Terms:
    Term 0:        topology.ebs.csi.aws.com/zone in [region-code]
Message:
Source:
    Type:              CSI (a Container Storage Interface (CSI) volume source)
    Driver:            ebs.csi.aws.com
    VolumeHandle:      vol-0d651e157c6d93445
    ReadOnly:          false
    VolumeAttributes:      storage.kubernetes.io/
csiProvisionerIdentity=1567792483192-8081-ebs.csi.aws.com
Events:                <none>
```

Amazon EBS ボリューム ID は、前の出力の VolumeHandle の値です。

9. Pod がボリュームにデータを書き込んでいることを確認します。

```
kubectl exec -it app -- cat /data/out.txt
```

出力例は次のとおりです。

```
Wed May 5 16:17:03 UTC 2021
Wed May 5 16:17:08 UTC 2021
Wed May 5 16:17:13 UTC 2021
Wed May 5 16:17:18 UTC 2021
[...]
```

10. 完了したら、このサンプルアプリケーションのリソースを削除します。

```
kubectl delete -f manifests/
```

Amazon EBS CSI 移行に関するよくある質問

> **Important**
>
> バージョン 1.22 またはそれ以前のクラスターで Pods を実行している場合は、サービスの中断を避けるために、クラスターをバージョン 1.23 に更新する前に Amazon EBS CSI ドライバーをインストールする必要があります。

Amazon EBS コンテナストレージインターフェイス (CSI) 移行機能は、ストレージオペレーションを処理する責任を Amazon EBS ツリー内 EBS ストレージプロビジョナーから Amazon EBS CSI ドライバーに移します。

CSI ドライバーとは何ですか?

CSI ドライバー:

- Kubernetes プロジェクトのソースコードに存在する Kubernetes の「ツリー内」のストレージドライバーを置き換えます。
- Amazon EBS などのストレージプロバイダーと連携します。
- 単純化されたプラグインモデルを提供することで、AWS などのストレージプロバイダーが機能をリリースし、Kubernetes リリースサイクルに依存せずにサポートを維持することが容易になります。

詳細については、Kubernetes CSI ドキュメントの「Introduction」(はじめに) を参照してください。

CSI 移行とは何ですか?

Kubernetes CSI 移行機能は、ストレージオペレーションを処理する責任を、`kubernetes.io/aws-ebs` などの既存のツリー内ストレージプラグインから、対応する CSI ドライバーに移します。既存の `StorageClass`、`PersistentVolume`、および `PersistentVolumeClaim` (PVC) オブジェクトは、対応する CSI ドライバーがインストールされている限り、引き続き機能します。この機能を有効にすると、次のようになります。

- PVC を使用する既存のワークロードは、従来どおりに機能します。
- Kubernetes は、すべてのストレージ管理オペレーションのコントロールを CSI ドライバーに渡します。

詳細については、Kubernetes ブログの「Kubernetes1.23: Kubernetes ツリー内から CSI ボリュームへの移行ステータスの更新」を参照してください。

ツリー内プラグインから CSI ドライバーへの移行をサポートするために、Amazon EKS バージョン 1.23 以降のクラスターでは CSIMigration および CSIMigrationAWS フラグがデフォルトで有効になっています。これらのフラグにより、クラスターはツリー内 API を同等の CSI API に変換できます。これらのフラグは、Amazon EKS によって管理される Kubernetes コントロールプレーンと、Amazon EKS 最適化 AMI で構成された kubelet 設定で設定されます。クラスターで Amazon EBS ボリュームを使用している Pods がある場合は、クラスターをバージョン **1.23** に更新する前に、Amazon EBS CSI ドライバーをインストールする必要があります。そうしないと、プロビジョニングやマウントなどのボリュームオペレーションが想定どおりに機能しない可能性があります。詳細については、「Amazon EBS CSI ドライバー」を参照してください。

> **Note**
>
> ツリー内 StorageClass プロビジョナーの名前は kubernetes.io/aws-ebs です。Amazon EBS CSI StorageClass プロビジョナーの名前は ebs.csi.aws.com です。

バージョン **1.23** 以降のクラスターで **kubernetes.io/aws-ebs StorageClass** ボリュームをマウントできますか?

はい。Amazon EBS CSI ドライバーがインストールされている限りマウントできます。新しく作成されたバージョン 1.23 以降のクラスターの場合、クラスター作成プロセスの一環として Amazon EBS CSI ドライバーをインストールすることをお勧めします。また、ebs.csi.aws.com プロビジョナーに基づいて StorageClasses のみを使用することをお勧めします。

クラスターコントロールプレーンをバージョン 1.23 に更新し、ノードを 1.23 にまだ更新していない場合、CSIMigration および CSIMigrationAWS kubelet フラグは有効になっていません。この場合、ツリー内ドライバーを使用して kubernetes.io/aws-ebs ベースボリュームをマウントします。ただし、Pods による kubernetes.io/aws-ebs ベースボリュームの確実な使用をスケジュールできるようにするには、Amazon EBS CSI ドライバーをインストールする必要があります。ドライバーは、他のボリュームオペレーションを成功させるためにも必要です。

Amazon EKS **1.23** 以降のクラスターで **kubernetes.io/aws-ebs StorageClass** ボリュームをプロビジョニングできますか?

はい。Amazon EBS CSI ドライバーがインストールされている限りプロビジョニングできます。

Amazon EKS から **kubernetes.io/aws-ebs StorageClass** プロビジョナーが削除されることはありますか?

kubernetes.io/aws-ebs StorageClass プロビジョナーと awsElasticBlockStore ボリュームタイプはサポートされなくなりましたが、削除する予定はありません。これらのリソースは、Kubernetes API の一部として扱われます。

Amazon EBS CSI ドライバーをインストールするにはどうすればよいですか?

Amazon EBS CSI ドライバーは Amazon EKS アドオンを使用してインストールすることをお勧めします。Amazon EKS アドオンの更新が必要な場合、ユーザーが更新を開始すると、Amazon EKS がアドオンを更新します。ドライバーを自ら管理したい場合は、オープンソースの Helm チャートを使用してインストールできます。

> **Important**
>
> Kubernetes ツリー内 Amazon EBS ドライバーは、Kubernetes コントロールプレーンで実行されます。Amazon EKS クラスター の IAM ロール に割り当てられた IAM 許可を使用して、Amazon EBS ボリュームをプロビジョニングします。Amazon EBS CSI ドライバーはノードで実行されます。ドライバーには、ボリュームをプロビジョニングするために IAM 許可が必要です。詳細については、「Amazon EBS CSI ドライバー IAM ロールの作成」を参照してください。

Amazon EBS CSI ドライバーがクラスターにインストールされているかどうかを確認するにはどうすればよいですか?

ドライバーがクラスターにインストールされているかどうかを確認するには、以下のコマンドを実行します。

```
kubectl get csidriver ebs.csi.aws.com
```

インストールが Amazon EKS によって管理されているかどうか確認するには、次のコマンドを実行します。

```
aws eks list-addons --cluster-name my-cluster
```

Amazon EBS CSI ドライバーをまだインストールしていない場合、Amazon EKS によってクラスターのバージョン **1.23** への更新が妨げられますか?

いいえ。

クラスターをバージョン 1.23 に更新する前に Amazon EBS CSI ドライバーをインストールし忘れた場合はどうなりますか? クラスターを更新した後にドライバーをインストールできますか?

はい。ただし、Amazon EBS CSI ドライバーを必要とするボリュームオペレーションは、クラスターの更新後、ドライバーがインストールされるまで失敗します。

新しく作成された Amazon EKS バージョン **1.23** 以降のクラスターで適用されるデフォルトの **StorageClass** は何ですか?

デフォルトの StorageClass 動作は変更されないままとなります。新しいクラスターごとに、Amazon EKS は gp2 という名前の kubernetes.io/aws-ebs ベースの StorageClass を適用します。新しく作成されたクラスターからこの StorageClass を削除する予定はありません。クラスターのデフォルトの StorageClass とは別に、ボリュームタイプを指定せずに ebs.csi.aws.com ベースの StorageClass を作成すると、Amazon EBS CSI ドライバーはデフォルトで gp3 を使用します。

クラスターをバージョン **1.23** に更新すると、Amazon EKS は既存のクラスターに既に存在する **StorageClasses** に対して変更を加えますか?

いいえ。

スナップショットを使用して永続ボリュームを **kubernetes.io/aws-ebsStorageClass** から **ebs.csi.aws.com** に移行するにはどうすればよいですか?

永続ボリュームを移行するには、AWS ブログの「Migrating Amazon EKS clusters from gp2 to gp3 EBS volumes」(Amazon EKS クラスターを gp2 から gp3 EBS ボリュームに移行する) を参照してください。

アノテーションを使用して Amazon EBS ボリュームを変更する方法を教えてください。

aws-ebs-csi-driver v1.19.0-eksbuild.2 以降では、PersistentVolumeClaim (PVC) 内のアノテーションを使用して Amazon EBS ボリュームを変更できます。新しいボリューム変更機

能は、「volumemodifier」と呼ばれる追加のサイドカーとして実装されています。詳細については、AWS ブログの「EBS CSI ドライバーを使用した Kubernetes での Amazon EBS ボリュームの移行と変更の簡素化」を参照してください。

移行は Windows ワークロード向けにサポートされていますか?

はい。オープンソースの Helm チャートを使用して Amazon EBS CSI ドライバーをインストールする場合は、node.enableWindows を true に設定します。Amazon EBS CSI ドライバーを Amazon EKS アドオンとしてインストールする場合、これはデフォルトで設定されます。StorageClasses を作成するときは、fsType を ntfs などの Windows ファイルシステムに設定します。その後、Windows ワークロードのボリュームオペレーションは、Linux ワークロードの場合と同じように、Amazon EBS CSI ドライバーに移行されます。

Amazon EFS CSI ドライバー

Amazon Elastic File System (Amazon EFS) は、サーバーレスで伸縮自在なファイルストレージを提供するため、ストレージ容量およびパフォーマンスのプロビジョニングや管理を行うことなくファイルデータを共有できます。Amazon EFS Container Storage Interface (CSI) ドライバーは、AWS で動作する Kubernetes クラスターが Amazon EFS ファイルシステムのライフサイクルを管理できるようにする CSI インターフェイスを提供します。このトピックでは、Amazon EFS CSI ドライバーを Amazon EKS クラスターにデプロイする方法を示します。

考慮事項

- Amazon EFS CSI ドライバーは、Windows ベースのコンテナイメージと互換性がありません。
- Fargate ノードでは永続ボリュームのために動的プロビジョニングを使用できませんが、静的プロビジョニングは使用できます。
- 動的プロビジョニングには、1.2 以降のドライバーが必要です。サポートされている Amazon EKS クラスターのバージョンのドライバーのバージョン 1.1 を使用して、永続ボリュームのために静的プロビジョニングを使用できます。
- このドライバーのバージョン 1.3.2 以降では、Amazon EC2 Graviton ベースのインスタンスを含む Arm64 アーキテクチャがサポートされています。
- バージョン 1.4.2 以降のドライバーでは、ファイルシステムのマウントに FIPS を使用できます。
- Amazon EFS のリソースクォータに注意してください。例えば、Amazon EFS ファイルシステムごとに作成できるアクセスポイントのクォータは 1,000 です。詳細については、「変更できない Amazon EFS リソースクォータ」を参照してください。

前提条件

- クラスター用の既存 AWS Identity and Access Management IAM OpenID Connect (OIDC) プロバイダー。既に存在しているかどうかを確認する、または作成するには「クラスターの IAM OIDC プロバイダーを作成する」を参照してください。
- ご使用のデバイスまたは AWS CloudShell で、バージョン `2.12.3` 以降、または AWS Command Line Interface (AWS CLI) のバージョン `1.27.160` 以降がインストールおよび設定されていること。現在のバージョンを確認するには、「**`aws --version | cut -d / -f2 | cut -d ' ' -f1`**」を参照してください。macOS の `yum`、`apt-get`、または Homebrew などのパッケージマネージャは、AWS CLI の最新バージョンより数バージョン遅れることがあります。最新バージョンをインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI のインストール、更新、およびアンインストール」と「aws configure でのクイック設定」を参照してください。AWS CloudShell にインストールされている AWS CLI バージョンは、最新バージョンより数バージョン遅れている可能性もあります。更新するには、「AWS CloudShell ユーザーガイド」の「ホームディレクトリへの AWS CLI のインストール」を参照してください。
- デバイスまたは AWS CloudShell に、`kubectl` コマンドラインツールがインストールされていること。バージョンは、ご使用のクラスターの Kubernetes バージョンと同じか、1 つ前のマイナーバージョン以前、あるいはそれより新しいバージョンが使用できます。例えば、クラスターのバージョンが `1.29` である場合、`kubectl` のバージョン `1.28`、`1.29`、または `1.30` が使用できます。`kubectl` をインストールまたはアップグレードする方法については、「`kubectl` のインストールまたは更新」を参照してください。

> **Note**
>
> AWS Fargate で実行されている Pod は、Amazon EFS ファイルシステムを自動的にマウントします。

IAM ロールの作成

Amazon EFS CSI ドライバーには、ファイルシステムと対話するための IAM アクセス許可が必要です。IAM ロールを作成して、必要な AWS 管理ポリシーをアタッチします。`eksctl`、AWS Management Console、または AWS CLI を使用できます。

> **Note**
>
> この手順には、ドライバーを Amazon EKS アドオンとして使用するための特定のステップが書かれています。セルフマネージド型インストールについての詳細は、GitHub の「ドライバー権限の設定」を参照してください。

eksctl

`eksctl` で Amazon EFS CSI ドライバーの IAM ロールを作成するには

次のコマンドを実行して、IAM ロールを作成します。*`my-cluster`* をクラスターの名前に、*`AmazonEKS_EFS_CSI_DriverRole`* をロールの名前に置き換えます。

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
eksctl create iamserviceaccount \
    --name efs-csi-controller-sa \
    --namespace kube-system \
    --cluster $cluster_name \
    --role-name $role_name \
    --role-only \
    --attach-policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEFSCSIDriverPolicy \
    --approve
TRUST_POLICY=$(aws iam get-role --role-name $role_name --query
 'Role.AssumeRolePolicyDocument' | \
    sed -e 's/efs-csi-controller-sa/efs-csi-*/' -e 's/StringEquals/StringLike/')
aws iam update-assume-role-policy --role-name $role_name --policy-document
 "$TRUST_POLICY"
```

AWS Management Console

AWS Management Console で Amazon EFS CSI ドライバー IAM ロールを作成するには

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. 左のナビゲーションペインで、[ロール] を選択します。
3. [ロール] ページで、[ロールの作成] を選択します。
4. [信頼されたエンティティを選択] ページで、以下の操作を実行します。

a. [信頼されたエンティティの種類] セクションで、[ウェブ アイデンティティ] を選択します。

b. [ID プロバイダー] で、(Amazon EKS の [概要] に示されているように) クラスターに [OpenID Connect プロバイダーの URL] を選択します。

c. [対象者] で [sts.amazonaws.com] を選択します。

d. [Next] を選択します。

5. [アクセス許可を追加] ページで、以下を実行します。

a. [フィルタポリシー] ボックスに *AmazonEFSCSIDriverPolicy* と入力します。

b. 検索で返された *AmazonEFSCSIDriverPolicy* の左にあるチェックボックスを選択します。

c. [Next] を選択します。

6. [名前を付けて、レビューし、作成する] ページで、以下の操作を実行します。

a. [ロール名] に、***AmazonEKS_EFS_CSI_DriverRole*** などのロールの一意の名前を入力します。

b. [タグの追加 (オプション)] で、タグをキーバリューのペアとして添付して、メタデータをロールに追加します。IAM でのタグの使用に関する詳細については、『IAM ユーザーガイド』の「IAM リソースにタグを付ける」を参照してください。

c. [ロールの作成] を選択します。

7. ロールが作成されたら、コンソールでロールを選択して編集用に開きます。

8. [信頼関係] タブを選択し、続いて [信頼ポリシーの編集] を選択します。

9. 次の行と似ている行を探してます。

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud":
 "sts.amazonaws.com"
```

前の行の上に、次の行を追加します。*region-code* をクラスターのある AWS リージョンに置き換えます。*EXAMPLED539D4633E53DE1B71EXAMPLE* をクラスターの OIDC プロバイダー ID に置き換えます。

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub":
 "system:serviceaccount:kube-system:efs-csi-*",
```

10. Condition オペレーターを "StringEquals" から "StringLike" に修正します。

11. [ポリシーの更新] を選択して終了します。

AWS CLI

AWS CLI で Amazon EFS CSI ドライバー IAM ロールを作成するには

1. クラスターの OIDC プロバイダーの URL を表示します。*my-cluster* をクラスター名に置き換えます。コマンドの出力が None の場合は、「前提条件」を確認してください。

```
aws eks describe-cluster --name my-cluster --query
 "cluster.identity.oidc.issuer" --output text
```

出力例は次のとおりです。

```
https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

2. IAM ロールを作成して AssumeRoleWithWebIdentity アクションを付与します。

a. 次の内容を *aws-efs-csi-driver-trust-policy*.json という名前のファイルにコピーします。*111122223333* はご使用のアカウントの ID に置き換えます。*EXAMPLED539D4633E53DE1B71EXAMPLE* と *region-code* を、前のステップで返された値に置き換えます。クラスターが AWS GovCloud (米国東部) または AWS GovCloud (米国東部) の AWS リージョン にある場合は、arn:aws: を arn:aws-us-gov: に置き換えます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:oidc-provider/
oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringLike": {
          "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-
system:efs-csi-*",
```

```
          "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com"
        }
      }
    }
  ]
}
```

b. ロールを作成する。*AmazonEKS_EFS_CSI_DriverRole* を別の名前に変更できますが、変更する場合は、後の手順でも変更してください。

```
aws iam create-role \
  --role-name AmazonEKS_EFS_CSI_DriverRole \
  --assume-role-policy-document file://"aws-efs-csi-driver-trust-
policy.json"
```

3. 次のマンドを実行して、必要な AWS マネージドポリシーをロールに添付します。クラスターが AWS GovCloud (米国東部) または AWS GovCloud (米国東部) の AWS リージョン にある場合は、`arn:aws:` を `arn:aws-us-gov:` に置き換えます。

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy \
  --role-name AmazonEKS_EFS_CSI_DriverRole
```

Amazon EFS CSI ドライバーのインストール

Amazon EFS CSI ドライバーは Amazon EKS アドオンを使用してインストールすることをお勧めします。Amazon EKS アドオンをクラスターに追加するには、「アドオンの作成」を参照してください。アドオンの詳細については、「Amazon EKS アドオン」を参照してください。Amazon EKS アドオンを使用できない場合は、その理由について、問題をコンテナロードマップの GitHub リポジトリに送信することをお勧めします。

または、Amazon EFS CSI ドライバーのセルフマネージド型インストールが必要な場合は、GitHub の「インストール」を参照してください。

Amazon EFS ファイルシステムの作成

Amazon EFS ファイルシステムを作成するには、GitHub の「Amazon EKS 用の Amazon EFS ファイルシステムを作成する」を参照してください。

サンプルアプリケーションをデプロイする

さまざまなサンプルアプリケーションをデプロイし、必要に応じて変更できます。詳細については、GitHub の「例」を参照してください。

Amazon FSx for Lustre CSI ドライバー

FSx for Lustre コンテナストレージインターフェイス (CSI) ドライバーは、Amazon EKS クラスターが FSx for Lustre ファイルシステムのライフサイクルを管理できるようにする CSI インターフェイスを提供します。詳細については、「FSx for Lustre ユーザーガイド」を参照してください。

このトピックでは、FSx for Lustre CSI ドライバーを Amazon EKS クラスターにデプロイし、動作することを確認する方法を示します。最新バージョンのドライバーを使用することをお勧めします。利用可能なバージョンについては、GitHub の「CSI Specification Compatibility Matrix」(CSI 仕様互換性マトリックス) を参照してください。

> **Note**
>
> Fargate では、ドライバーはサポートされていません。

使用可能なパラメータの詳細と、ドライバーの機能を示す完全な例については、「GitHub」の「FSx for Lustre コンテナストレージインターフェイス (CSI) ドライバー」プロジェクトを参照してください。

前提条件

必要なもの:

- ご使用のデバイスまたは AWS CloudShell で、バージョン `2.12.3` 以降、または AWS Command Line Interface (AWS CLI) のバージョン `1.27.160` 以降がインストールおよび設定されていること。現在のバージョンを確認するには、「**`aws --version | cut -d / -f2 | cut -d ' ' -f1`**」を参照してください。macOS の `yum`、`apt-get`、または Homebrew などのパッケージマネージャは、AWS CLI の最新バージョンより数バージョン遅れることがあります。最新バージョンをインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI のインストール、更新、およびアンインストール」と「aws configure でのクイック設定」を参照してください。AWS CloudShell にインストールされている AWS CLI バージョンは、最新バージョンより数バージョン遅れている可能性もあります。更新するには、「AWS CloudShell ユーザーガイド」の「ホームディレクトリへの AWS CLI のインストール」を参照してください。

- デバイスまたは AWS CloudShell にインストールされている eksctl コマンドラインツールのバージョン 0.183.0 以降。eksctl をインストールまたはアップグレードするには、eksctl ドキュメントの「インストール」を参照してください。
- デバイスまたは AWS CloudShell に、kubectl コマンドラインツールがインストールされていること。バージョンは、ご使用のクラスターの Kubernetes バージョンと同じか、1 つ前のマイナーバージョン以前、あるいはそれより新しいバージョンが使用できます。例えば、クラスターのバージョンが 1.29 である場合、kubectl のバージョン 1.28、1.29、または 1.30 が使用できます。kubectl をインストールまたはアップグレードする方法については、「kubectl のインストールまたは更新」を参照してください。

次の手順は、FSx for Lustre CSI ドライバーを使用して簡単なテストクラスターを作成して、動作を確認できるようにするのに役立ちます。本番ワークロードにテストクラスターを使用することはお勧めしません。このチュートリアルでは、*example values* を使用することをお勧めしますが、置き換えるように書かれている箇所はその限りではありません。本番クラスター向けに手順を完了するときには、どの *example value* も置き換えることができます。変数は、この手順全体で設定および使用され、別のターミナルには存在しないため、同じターミナルですべての手順を完了することをお勧めします。

FSx for Lustre CSI ドライバーを Amazon EKS クラスターにデプロイするには

1. 残りの手順で使用する変数をいくつか設定します。*my-csi-fsx-cluster* を作成するテストクラスターの名前に置き換え、*region-code* をテストクラスターを作成する AWS リージョンに置き換えます。

```
export cluster_name=my-csi-fsx-cluster
export region_code=region-code
```

2. テストクラスターを作成します。

```
eksctl create cluster \
  --name $cluster_name \
  --region $region_code \
  --with-oidc \
  --ssh-access \
  --ssh-public-key my-key
```

クラスターのプロビジョニングには数分かかります。クラスターの作成中に、数行の出力が表示されます。出力の最後の行は、次のサンプル行のようになります。

```
[#]  EKS cluster "my-csi-fsx-cluster" in "region-code" region is ready
```

3. ドライバーの Kubernetes サービスアカウントを作成し、次のコマンドを使用して AmazonFSxFullAccessAWS マネージドポリシーをサービスアカウントにアタッチします。クラスターが AWS GovCloud (米国東部) または AWS GovCloud (米国東部) の AWS リージョンにある場合は、arn:aws: を arn:aws-us-gov: に置き換えます。

```
eksctl create iamserviceaccount \
  --name fsx-csi-controller-sa \
  --namespace kube-system \
  --cluster $cluster_name \
  --attach-policy-arn arn:aws:iam::aws:policy/AmazonFSxFullAccess \
  --approve \
  --role-name AmazonEKSFSxLustreCSIDriverFullAccess \
  --region $region_code
```

サービスアカウントが作成されると、数行の出力が表示されます。出力の最後の数行は、次のようになります。

```
[#]  1 task: {
    2 sequential sub-tasks: {
        create IAM role for serviceaccount "kube-system/fsx-csi-controller-sa",
        create serviceaccount "kube-system/fsx-csi-controller-sa",
    } }
[#]  building iamserviceaccount stack "eksctl-my-csi-fsx-cluster-addon-
iamserviceaccount-kube-system-fsx-csi-controller-sa"
[#]  deploying stack "eksctl-my-csi-fsx-cluster-addon-iamserviceaccount-kube-
system-fsx-csi-controller-sa"
[#]  waiting for CloudFormation stack "eksctl-my-csi-fsx-cluster-addon-
iamserviceaccount-kube-system-fsx-csi-controller-sa"
[#]  created serviceaccount "kube-system/fsx-csi-controller-sa"
```

デプロイされた AWS CloudFormation スタックの名前を書き留めておきます。前述の出力例では、スタックの名前は eksctl-*my-csi-fsx-cluster*-addon-iamserviceaccount-kube-system-*fsx-csi-controller-sa* です。

4. 次のコマンドでドライバーをデプロイします。*release-X.XX* を目的のブランチに置き換えます。マスターブランチは、現在リリースされている安定版のドライバーと互換性のない今後リリースされる機能が含まれている可能性があるため、サポート対象外です。最新のリリース済

みバージョンをダウンロードすることをお勧めします。アクティブなブランチのリストについては、「GitHub」の「aws-fsx-csi-driver」を参照してください。

> **Note**
>
> GitHub の aws-fsx-csi-driver で、適用されているコンテンツを表示できます。

```
kubectl apply -k "github.com/kubernetes-sigs/aws-fsx-csi-driver/deploy/kubernetes/
overlays/stable/?ref=release-X.XX"
```

出力例は次のとおりです。

```
serviceaccount/fsx-csi-controller-sa created
serviceaccount/fsx-csi-node-sa created
clusterrole.rbac.authorization.k8s.io/fsx-csi-external-provisioner-role created
clusterrole.rbac.authorization.k8s.io/fsx-external-resizer-role created
clusterrolebinding.rbac.authorization.k8s.io/fsx-csi-external-provisioner-binding
 created
clusterrolebinding.rbac.authorization.k8s.io/fsx-csi-resizer-binding created
deployment.apps/fsx-csi-controller created
daemonset.apps/fsx-csi-node created
csidriver.storage.k8s.io/fsx.csi.aws.com created
```

5. 作成されたロールの ARN を書き留めます。書き留めていなくて、AWS CLI 出力として表示されなくなっている場合は、次の操作で AWS Management Console に表示できます。

   a. https://console.aws.amazon.com/cloudformation で AWS CloudFormation コンソール を開きます。

   b. コンソールが IAM ロールを作成した AWS リージョン に設定されていることを確認してから、[スタック] を選択します。

   c. eksctl-*my-csi-fsx-cluster*-addon-iamserviceaccount-kube-system-*fsx-csi-controller-sa* という名前のスタックを選択します。

   d. [出力] タブを選択します。[Role1] ARN が [出力 (1)] ページに表示されます。

6. 次のコマンドを使用して、ドライバーのデプロイにパッチを適用し、前に作成したサービスアカウントを追加します。ARN を、メモした ARN に置き換えます。*111122223333* をアカウント ID に置き換えます。クラスターが AWS GovCloud (米国東部) または AWS GovCloud (米国東部) の AWS リージョン にある場合は、arn:aws: を arn:aws-us-gov: に置き換えます。

```
kubectl annotate serviceaccount -n kube-system fsx-csi-controller-sa \
  eks.amazonaws.com/role-
arn=arn:aws:iam::111122223333:role/AmazonEKSFSxLustreCSIDriverFullAccess --
overwrite=true
```

出力例は次のとおりです。

```
serviceaccount/fsx-csi-controller-sa annotated
```

Kubernetes ストレージクラス、永続的なボリューム要求、およびサンプルアプリケーションをデプロイして、CSI ドライバーが動作していることを確認するには

この手順では、「FSx for Lustre Container Storage Interface (CSI) driver」(FSx for Lustre コンテナストレージインターフェイス (CSI) ドライバー) GitHub リポジトリを使用して、動的にプロビジョニングされた FSx for Lustre ボリュームを使用します。

1. クラスターのセキュリティグループを書き留めます。AWS Management Console の [ネットワーク] セクションまたは次の AWS CLI コマンドを使用して確認できます。

```
aws eks describe-cluster --name $cluster_name --query
 cluster.resourcesVpcConfig.clusterSecurityGroupId
```

2. 「Amazon FSx for Lustre ユーザーガイド」の「Amazon VPC セキュリティグループ」に示す基準に従って、Amazon FSx ファイルシステムのセキュリティグループを作成します。[VPC] で、[ネットワーク] セクションに示されているようにクラスターの VPC を選択します。「Lustre クライアントに関連付けられているセキュリティグループ」には、クラスターセキュリティグループを使用します。アウトバウンドルールをそのままにして、[すべてのトラフィック] を許可することができます。
3. 次のコマンドを使用して、ストレージクラスマニフェストをダウンロードします。

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-fsx-csi-driver/
master/examples/kubernetes/dynamic_provisioning/specs/storageclass.yaml
```

4. `storageclass.yaml` ファイルの parameters セクションを編集します。*example value* をすべて自分の値に置き換えてください。

```
parameters:
  subnetId: subnet-0eabfaa81fb22bcaf
```

```
  securityGroupIds: sg-068000ccf82dfba88
  deploymentType: PERSISTENT_1
  automaticBackupRetentionDays: "1"
  dailyAutomaticBackupStartTime: "00:00"
  copyTagsToBackups: "true"
  perUnitStorageThroughput: "200"
  dataCompressionType: "NONE"
  weeklyMaintenanceStartTime: "7:09:00"
  fileSystemTypeVersion: "2.12"
```

- **subnetId** – Amazon FSx for Lustre ファイルシステムが作成されるサブネット ID。Amazon FSx for Lustre は、すべてのアベイラビリティーゾーンでサポートされているわけではありません。https://console.aws.amazon.com/fsx/ で Amazon FSx for Lustre コンソールを開き、使用するサブネットが、サポートされているアベイラビリティーゾーンにあることを確認します。次のように、サブネットにノードを含めることも、別のサブネットまたは VPC にすることもできます。
  - AWS Management Console で [コンピューティング] セクションのノードグループを選択すると、ノードサブネットを確認できます。
  - 指定するサブネットが、ノードがあるサブネットと同じでない場合は、VPC が接続されている必要があり、セキュリティグループで必要なポートが開いていることを確認する必要があります。
- **securityGroupIds** – ファイルシステム用に作成したセキュリティグループの ID。
- **deploymentType** (オプション) – ファイルシステムのデプロイのタイプ。有効な値は、SCRATCH_1、SCRATCH_2、PERSISTENT_1、および PERSISTENT_2 です。デプロイのタイプの詳細については、「Amazon FSx for Lustre ファイルシステムを作成する」を参照してください。
- 他のパラメータ (オプション) - 他のパラメータについては、「GitHub」の「StorageClass の編集」を参照してください。

5. ストレージクラスマニフェストを作成します。

```
kubectl apply -f storageclass.yaml
```

出力例は次のとおりです。

```
storageclass.storage.k8s.io/fsx-sc created
```

6. 永続的なボリューム要求マニフェストをダウンロードします。

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-fsx-csi-driver/
master/examples/kubernetes/dynamic_provisioning/specs/claim.yaml
```

7. (オプション) claim.yaml ファイルを編集します。ストレージ要件と前のステップで選択した deploymentType に基づいて、*1200Gi* を次のいずれかの増分値に変更します。

```
storage: 1200Gi
```

   - SCRATCH_2 および PERSISTENT – **1.2 TiB**、**2.4 TiB**、または 2.4 TiB を超えると 2.4 TiB の増分。
   - SCRATCH_1 – **1.2 TiB**、**2.4 TiB**、**3.6 TiB**、または 3.6 TiB を超えると 3.6 TiB の増分。

8. 永続的なボリューム要求を作成します。

```
kubectl apply -f claim.yaml
```

   出力例は次のとおりです。

```
persistentvolumeclaim/fsx-claim created
```

9. ファイルシステムがプロビジョニングされていることを確認します。

```
kubectl describe pvc
```

   出力例は次のとおりです。

```
Name:          fsx-claim
Namespace:     default
StorageClass:  fsx-sc
Status:        Bound
[...]
```

> **Note**
>
> Status は、Pending になる前に 5 ~ 10 分間 Bound と表示されることがあります。Status が Bound になるまで次のステップに進まないでください。Status が 10

分を超えて Pending になっている場合は、Events 内の警告メッセージを問題に対処するための参考として使用します。

10. サンプルアプリケーションをデプロイします。

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-fsx-csi-driver/master/examples/kubernetes/dynamic_provisioning/specs/pod.yaml
```

11. サンプルアプリケーションが実行中であることを確認します。

```
kubectl get pods
```

出力例は次のとおりです。

```
NAME      READY   STATUS              RESTARTS   AGE
fsx-app   1/1     Running             0          8s
```

12. ファイルシステムがアプリケーションによって正しくマウントされていることを確認します。

```
kubectl exec -ti fsx-app -- df -h
```

出力例は次のとおりです。

```
Filesystem                  Size  Used Avail Use% Mounted on
overlay                      80G  4.0G   77G   5% /
tmpfs                        64M     0   64M   0% /dev
tmpfs                       3.8G     0  3.8G   0% /sys/fs/cgroup
192.0.2.0@tcp:/abcdef01     1.1T  7.8M  1.1T   1% /data
/dev/nvme0n1p1               80G  4.0G   77G   5% /etc/hosts
shm                          64M     0   64M   0% /dev/shm
tmpfs                       6.9G   12K  6.9G   1% /run/secrets/kubernetes.io/
serviceaccount
tmpfs                       3.8G     0  3.8G   0% /proc/acpi
tmpfs                       3.8G     0  3.8G   0% /sys/firmware
```

13. データがサンプルアプリケーションによって FSx for Lustre ファイルシステムに書き込まれたことを確認します。

```
kubectl exec -it fsx-app -- ls /data
```

出力例は次のとおりです。

```
out.txt
```

この出力例は、サンプルアプリケーションが `out.txt` ファイルをファイルシステムに正常に書き込んだことを示しています。

> **Note**
>
> クラスターを削除する前に FSx for Lustre ファイルシステムを必ず削除してください。詳細については、「FSx for Lustre ユーザーガイド」の「リソースをクリーンアップする」を参照してください。

Amazon FSx for NetApp ONTAP CSI ドライバ

NetApp's Astra Trident は、コンテナストレージインターフェイス (CSI) 準拠のドライバーを使用して、動的なストレージオーケストレーションを提供します。これにより、Amazon EKS クラスターが Amazon FSx for NetApp ONTAP ファイルシステムによってバックアップされた永続的ボリューム (PV) のライフサイクルを管理できるようになります。開始するには、「Astra Trident ドキュメント」の「 Use Astra Trident with Amazon FSx for NetApp ONTAP」を参照してください。

Amazon FSx for NetApp ONTAP は、クラウド上でフルマネージドの ONTAP ファイルシステムを起動して実行できるストレージサービスです。ONTAP は、幅広く採用されているデータアクセスとデータ管理の機能を利用できる NetApp's ファイルシステム技術です。FSx for ONTAP によって、オンプレミスの NetApp ファイルシステムの機能、パフォーマンス、および API に、フルマネージド型 AWS のサービスの俊敏性、スケーラビリティが加わり、簡素化されます。詳細については、「FSx for ONTAP ユーザーガイド」を参照してください。

Amazon FSx for OpenZFS CSI ドライバー

Amazon FSX for OpenZFS は、オンプレミスの ZFS またはその他の Linux ベースのファイルサーバーから AWS にデータを簡単に移動できるようにするフルマネージド型のファイルストレージサービスです。データの移動は、アプリケーションコードやデータの管理方法を変更しなくても実行できます。Amazon FSX for OpenZFS は、オープンソースの OpenZFS ファイルシステム上に構築された、信頼性が高く、スケーラブルで効率的で機能豊富なファイルストレージを提供しています。これ

らの機能は、フルマネージド型の AWS サービスの俊敏性、スケーラビリティ、およびシンプルさを兼ね備えています。詳細については、「Amazon FSx for OpenZFS ユーザーガイド」を参照してください。

Amazon FSx for OpenZFS コンテナストレージインターフェイス (CSI) ドライバーは、Amazon EKS クラスターが Amazon FSx for OpenZFS ボリュームのライフサイクルを管理できるようにする CSI インターフェイスを提供します。Amazon FSx for OpenZFS CSI ドライバーを Amazon EKS クラスターにデプロイするには、GitHub の「aws-fsx-openzfs-csi-driver」を参照してください。

Amazon File Cache CSI ドライバー

Amazon File Cache は、データの保存場所にかかわらず、ファイルデータの処理に使用される AWS のフルマネージド型の高速キャッシュです。Amazon File Cache は、初回アクセス時にデータをキャッシュに自動的にロードし、使用されていないときにデータを解放します。詳細については、「Amazon File Cache ユーザーガイド」を参照してください。

Amazon File Cache Container Storage Interface (CSI) ドライバーは、Amazon EKS クラスターが Amazon ファイルキャッシュのライフサイクルを管理できるようにする CSI インターフェイスを提供します。Amazon File Cache CSI ドライバーを Amazon EKS クラスターにデプロイするには、GitHub の「aws-file-cache-csi-driver」を参照してください。

Mountpoint for Amazon S3 CSI driver

Mountpoint for Amazon S3 Container Storage Interface (CSI) ドライバーを使用すると、Kubernetes アプリケーションはファイルシステムインターフェイスを介して Amazon S3 オブジェクトにアクセスできるため、アプリケーションコードを変更せずに高い総スループットを実現できます。Amazon S3 Mountpoint 向けに構築された CSI ドライバーは、Amazon S3 バケットを Amazon EKS Kubernetes 内のコンテナや自己管理型クラスターからアクセスできるボリュームとして提供します。このトピックでは、Mountpoint for Amazon S3 CSI ドライバーを、Amazon EKS クラスターにデプロイする方法を示します。

考慮事項

- Mountpoint for Amazon S3 CSI ドライバーは、Windows ベースのコンテナイメージと互換性がありません。
- Mountpoint for Amazon S3 CSI ドライバーは、AWS Fargate をサポートしていません。ただし、Amazon EC2 で実行されているコンテナ (Amazon EKS またはカスタム Kubernetes インストール) はサポートされています。

- Mountpoint for Amazon S3 CSI ドライバーは、静的プロビジョニングのみをサポートします。動的プロビジョニング、つまり新しいバケットの作成はサポートされていません。

> **Note**
>
> 静的プロビジョニングとは、`PersistentVolume` オブジェクトの `volumeAttributes` で `bucketName` として指定されている既存の Amazon S3 バケットを使用することです。ユーザーのプロビジョニングの詳細については、「GitHub」を参照してください。

- Mountpoint for Amazon S3 CSI ドライバーでマウントされたボリュームは、すべての POSIX ファイルシステム機能をサポートしているわけではありません。ファイルシステムの動作の詳細については、GitHub の「Amazon S3 ファイルシステム用 Mountpoint の動作について」を参照してください。

前提条件

- クラスター用の既存 AWS Identity and Access Management IAM OpenID Connect (OIDC) プロバイダー。既に存在しているかどうかを確認する、または作成するには「クラスターの IAM OIDC プロバイダーを作成する」を参照してください。
- ご使用のデバイスまたは AWS CloudShell で、AWS CLI のバージョン 2.12.3 以降がインストールおよび設定されていること。
- デバイスまたは AWS CloudShell に、`kubectl` コマンドラインツールがインストールされていること。バージョンは、ご使用のクラスターの Kubernetes バージョンと同じか、1 つ前のマイナーバージョン以前、あるいはそれより新しいバージョンが使用できます。例えば、クラスターのバージョンが `1.29` である場合、`kubectl` のバージョン `1.28`、`1.29`、または `1.30` が使用できます。`kubectl` をインストールまたはアップグレードする方法については、「`kubectl` のインストールまたは更新」を参照してください。

IAM ポリシーの作成

Mountpoint for Amazon S3 CSI ドライバーには、ファイルシステムと対話するための Amazon S3 アクセス許可が必要です。このセクションでは、必要な権限を付与する IAM ポリシーを作成する方法を説明します。

以下のポリシー例は、Mountpoint の IAM アクセス許可に関する推奨事項に従っています。別の方法として、AWS 管理ポリシー AmazonS3FullAccess を使用することもできますが、この管理ポリシーでは Mountpoint に必要以上のアクセス権限が付与されます。

Mountpoint の推奨アクセス許可の詳細については、「GitHub」 の「Mountpoint IAM アクセス許可」を参照のこと。

IAM コンソールを使用して IAM ポリシーを作成するには

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. 左のナビゲーションペインの [ポリシー] を選択します。
3. [ポリシー] ページで、[ポリシーの作成] を選択します。
4. [ポリシーエディター] には [JSON] を選択します。
5. 以下の [ポリシー] をコピーして、エディタに貼り付けます。

> **Important**
>
> DOC-EXAMPLE-BUCKET1 を Amazon S3 バケットの名前に置き換えます。

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
            "Sid": "MountpointFullBucketAccess",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET1"
            ]
        },
        {
            "Sid": "MountpointFullObjectAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:AbortMultipartUpload",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET1/*"
```

```
            ]
        }
    ]
}
```

Amazon S3 Express One Zone ストレージクラスで導入されたディレクトリバケットは、汎用バケットとは異なる認証メカニズムを使用します。`s3:*` アクションではなく、`s3express:CreateSession` アクションを使用する必要があります。ディレクトリバケットの詳細については、Amazon S3 ユーザーガイドの「ディレクトリバケット」を参照してください。

以下は、ディレクトリバケットに使用する最小特権ポリシーの例です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3express:CreateSession",
            "Resource": "arn:aws:s3express:aws-region:111122223333:bucket/DOC-EXAMPLE-BUCKET1--az_id--x-s3"
        }
    ]
}
```

6. [Next] を選択します。
7. [確認と作成] ページで、ポリシーに名前を付けます。このウォークスルーでは、名前に `AmazonS3CSIDriverPolicy` を使用します。
8. [ポリシーを作成] を選択します。

IAM ロールの作成

Mountpoint for Amazon S3 CSI ドライバーには、ファイルシステムと対話するための IAM アクセス許可が必要です。このセクションでは、IAM ロールを作成して、これらの権限を委任する方法を示します。このロールは、`eksctl`、IAM コンソール、または AWS CLI を使用して作成できます。

> **Note**
>
> IAM ポリシー `AmazonS3CSIDriverPolicy` は、前のセクションで作成しました。

eksctl

`eksctl` を使用して Mountpoint for Amazon S3 CSI ドライバーの IAM ロールを作成するには

以下のコマンドを実行して、IAM ロールと Kubernetes サービスアカウントを作成します。これらのコマンドはまた、`AmazonS3CSIDriverPolicy` IAM ポリシーをロールにアタッチし、Kubernetes サービスアカウント (`s3-csi-controller-sa`) に IAM ロールの Amazon リソース名 (ARN) をアノテーションし、Kubernetes サービスアカウント名を IAMロールの信頼ポリシーに追加します。

```
CLUSTER_NAME=my-cluster
REGION=region-code
ROLE_NAME=AmazonEKS_S3_CSI_DriverRole
POLICY_ARN=AmazonEKS_S3_CSI_DriverRole_ARN
eksctl create iamserviceaccount \
    --name s3-csi-driver-sa \
    --namespace kube-system \
    --cluster $CLUSTER_NAME \
    --attach-policy-arn $POLICY_ARN \
    --approve \
    --role-name $ROLE_NAME \
    --region $REGION \
    --role-only
```

IAM console

AWS Management Console を使用して Mountpoint for Amazon S3 CSIドライバー の IAM ロールを作成するには

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. 左のナビゲーションペインで、[ロール] を選択します。
3. [ロール] ページで、[ロールの作成] を選択します。
4. [信頼されたエンティティを選択] ページで、以下の操作を実行します。
    a. [信頼されたエンティティの種類] セクションで、[ウェブ アイデンティティ] を選択します。
    b. [アイデンティティプロバイダー] で、(Amazon EKS の [概要] に示されているように) クラスターに [OpenID Connect プロバイダーの URL] を選択します。

        URL が表示されない場合は、「前提条件」セクションを確認してください。

c. [対象者] で [sts.amazonaws.com] を選択します。

d. [Next] を選択します。

5. [アクセス許可を追加] ページで、以下を実行します。

a. [フィルタポリシー] ボックスに **AmazonS3CSIDriverPolicy** と入力します。

> **Note**
>
> このポリシーは、前のセクションで作成されました。

b. 検索で返された AmazonS3CSIDriverPolicy 結果の左にあるチェックボックスを選択します。

c. [Next] を選択します。

6. [名前を付けて、レビューし、作成する] ページで、以下の操作を実行します。

a. [ロール名] に、**AmazonEKS_S3_CSI_DriverRole** などのロールの一意の名前を入力します。

b. [タグの追加 (オプション)] で、タグをキーバリューのペアとして添付して、メタデータをロールに追加します。IAM でのタグの使用に関する詳細については、『IAM ユーザーガイド』の「IAM リソースにタグを付ける」を参照してください。

c. [ロールの作成] を選択します。

7. ロールが作成されたら、コンソールでロールを選択して編集用に開きます。

8. [信頼関係] タブを選択し、続いて [信頼ポリシーの編集] を選択します。

9. 次のように表示されます。

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud":
 "sts.amazonaws.com"
```

前の行の末尾にカンマを追加し、前の行の後に次の行を追加します。*region-code* をクラスターのある AWS リージョン に置き換えます。*EXAMPLED539D4633E53DE1B71EXAMPLE* をクラスターの OIDC プロバイダー ID に置き換えます。

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub":
 "system:serviceaccount:kube-system:s3-csi-*"
```

10. Condition オペレーターを "StringEquals" から "StringLike" に修正します。

11. [ポリシーの更新] を選択して終了します。

AWS CLI

AWS CLI を使用して Mountpoint for Amazon S3 CSI ドライバー の IAM ロールを作成するには

1. クラスターの OIDC プロバイダーの URL を表示します。*my-cluster* を自分のクラスター名に置き換えます。コマンドの出力が None の場合は、「前提条件」を確認してください。

```
aws eks describe-cluster --name my-cluster --query
 "cluster.identity.oidc.issuer" --output text
```

出力例は次のとおりです。

```
https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

2. IAM ロールを作成し、Kubernetes サービスアカウントに AssumeRoleWithWebIdentity アクションを許可します。

a. 次の内容を *aws-s3-csi-driver-trust-policy*.json という名前のファイルにコピーします。*111122223333* はご使用のアカウントの ID に置き換えます。*EXAMPLED539D4633E53DE1B71EXAMPLE* と *region-code* を、前のステップで返された値に置き換えます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:oidc-provider/
oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringLike": {
          "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-
system:s3-csi-*",
          "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com"
```

```
        }
      }
    }
  ]
}
```

b. ロールを作成する。*AmazonEKS_S3_CSI_DriverRole* を別の名前に変更できますが、変更する場合は、後の手順でも変更してください。

```
aws iam create-role \
  --role-name AmazonEKS_S3_CSI_DriverRole \
  --assume-role-policy-document file://"aws-s3-csi-driver-trust-policy.json"
```

3. 次のコマンドを使用して IAM ロールを作成し、そのロールに IAM ポリシーをアタッチします。

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonS3CSIDriverPolicy \
  --role-name AmazonEKS_S3_CSI_DriverRole
```

> **Note**
>
> IAM ポリシー AmazonS3CSIDriverPolicy は、前のセクションで作成しました。

4. ドライバーを Amazon EKS アドオンとしてインストールする場合は、この手順をスキップしてください。ドライバーのセルフマネージド型インストールでは、作成した IAM ロールの ARN で注釈が付けられた Kubernetes サービスアカウントを作成します。

a. 次の内容を *mountpoint-s3-service-account*.yaml という名前のファイルに保存します。*111122223333* をアカウント ID に置き換えます。

```
---
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/name: aws-mountpoint-s3-csi-driver
  name: mountpoint-s3-csi-controller-sa
  namespace: kube-system
  annotations:
```

```
    eks.amazonaws.com/role-arn:
arn:aws:iam::111122223333:role/AmazonEKS_S3_CSI_DriverRole
```

b. クラスター上で Kubernetes サービスアカウントを作成します。mountpoint-s3-csi-controller-sa という名前の Kubernetes サービスアカウントは、*AmazonEKS_S3_CSI_DriverRole* という名前で作成した IAM ロールで注釈が付けられています。

```
kubectl apply -f mountpoint-s3-service-account.yaml
```

> **Note**
>
> その手順でプラグインがデプロイされると、s3-csi-driver-sa という名前のサービスアカウントを作成して使用するように設定されます。

Mountpoint for Amazon S3 CSI ドライバーのインストール

Mountpoint for Amazon S3 CSI ドライバーは、Amazon EKS アドオンを使用してインストールできます。eksctl、AWS Management Console、または AWS CLI を使用して、アドオンをクラスターに追加できます。

必要に応じて、セルフマネージドインストールとして Amazon S3 CSI ドライバー用 Mountpoint をインストールできます。セルフマネージド型インストールとして追加する方法については、「GitHub」の「インストール」を参照してください。

eksctl

eksctl を使用して Amazon S3 CSI アドオンを追加するには

以下のコマンドを実行します。*my-cluster* をクラスターの名前に、*111122223333* をアカウント ID に、*AmazonEKS_S3_CSI_DriverRole* を前に作成した IAM ロールの名前に置き換えます。

```
eksctl create addon --name aws-mountpoint-s3-csi-driver --cluster my-cluster --
service-account-role-arn arn:aws:iam::111122223333:role/AmazonEKS_S3_CSI_DriverRole
 --force
```

`--force` オプションを削除した状態で Amazon EKS アドオンを更新する際に、そのアドオン設定のいずれかが現状の設定と競合している場合、その更新は失敗します。この場合、競合を解決するためのエラーメッセージが表示されます。このオプションを指定する前に、自分が管理する必要がある設定を Amazon EKS アドオンが管理していないことを確認してください。これらの設定は、このオプションの指定により上書きされます。この設定のその他のオプションの詳細については、`eksctl` ドキュメントの「アドオン」を参照してください。Amazon EKS Kubernetes フィールド管理の詳細については、「Kubernetes フィールド管理」を参照してください。

AWS Management Console

AWS Management Consoleを使用して Amazon S3 CSI アドオンの Mountpoint を追加するには

1. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。
2. 左のナビゲーションペインで [クラスター] を選択します。
3. Amazon S3 CSI アドオンの Mountpoint を設定するクラスターの名前を選択します。
4. [アドオン] タブを選択します。
5. [その他のアドオンを入手] を選択します。
6. [アドオンを選択] ページで、次の操作を行います。
   a. [Amazon EKS アドオン] セクションで、[Mountpoint for Amazon S3 CSI ドライバー] チェックボックスを選択します。
   b. [Next] を選択します。
7. [選択したアドオンセッティングの設定] ページで、次の操作を行います。
   a. 使用する [バージョン] を選択します。
   b. [IAM ロールを選択] で、Mountpoint for Amazon S3 CSI ドライバーの IAM ポリシー をアタッチした IAM ロールの名前を選択します。
   c. (オプション) [オプションの構成設定] を展開できます。[コンフリクト解決方法] で [上書きする] を選択すると、既存のアドオンでの 1 つ以上の設定が、Amazon EKS アドオンの設定で上書きされます。このオプションが有効でない状態で既存の設定との競合が発生する場合は、オペレーションが失敗します。表示されたエラーメッセージを使用して、競合をトラブルシューティングできます。このオプションを選択する前に、自分で管理する必要のある設定が、Amazon EKS アドオンにより管理されていないことを確認してください。
   d. [Next] を選択します。

8. [確認と追加] ページで、[作成] を選択します。アドオンのインストールが完了した後、インストールしたアドオンが表示されます。

AWS CLI

AWS CLI を使用して Amazon S3 CSI アドオンの Mountpoint を追加するには

以下のコマンドを実行します。*my-cluster* をクラスターの名前に置き換え、*111122223333* をアカウント ID に置き換え、*AmazonEKS_S3_CSI_DriverRole* を前に作成したロールの名前に置き換えます。

```
aws eks create-addon --cluster-name my-cluster --addon-name aws-mountpoint-s3-csi-driver \
  --service-account-role-arn
 arn:aws:iam::111122223333:role/AmazonEKS_S3_CSI_DriverRole
```

Mountpoint for Amazon S3 の設定

ほとんどの場合、Mountpoint for Amazon S3 はバケット名のみを設定できます。Mountpoint for Amazon S3 に設定する手順については、「GitHub」で「Mountpoint for Amazon S3 の設定」を参照してください。

サンプルアプリケーションをデプロイする

既存の Amazon S3 バケットのドライバーにデプロイできます。詳細については、「GitHub」の「静的プロビジョニング」を参照してください。

Mountpoint for Amazon S3 CSI ドライバーの削除

Amazon EKS アドオンを削除するには、次の 2 つのオプションがあります。

- クラスター上のアドオンソフトウェアを保持する — このオプションでは、すべての設定の Amazon EKS 管理が削除されます。また、Amazon EKS による更新の通知機能や、更新の開始後の Amazon EKS アドオンの自動更新機能も削除されます。ただし、クラスター上のアドオンソフトウェアは保持されます。このオプションを選択すると、アドオンは Amazon EKS アドオンではなく、セルフマネージド型インストールになります。このオプションを使用すると、アドオンのダウンタイムは発生しません。この手順のコマンドには、このオプションを使用します。

- クラスターからアドオンソフトウェアを完全に削除する — クラスターから Amazon EKS アドオンを削除するのは、アドオンに依存するリソースがクラスター上にない場合のみとすることをお勧めします。このオプションを実行するには、この手順で使用するコマンドから `--preserve` を削除します。

アドオンに IAM アカウントが関連付けられている場合、その IAM アカウントは削除されません。

`eksctl`、AWS Management Console、または AWS CLI を使用して、Amazon S3 CSI アドオンを削除できます。

eksctl

`eksctl` を使用して Amazon S3 CSIアドオンを削除するには

`my-cluster` をクラスター名に置き換えた後、次のコマンドを実行します。

```
eksctl delete addon --cluster my-cluster --name aws-mountpoint-s3-csi-driver --
preserve
```

AWS Management Console

AWS Management Consoleを使用して Amazon S3 CSI アドオンを削除するには

1. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。
2. 左のナビゲーションペインで [クラスター] を選択します。
3. Amazon EBS CSI アドオンを削除するクラスターの名前を選択します。
4. [アドオン] タブを選択します。
5. [Mountpoint for Amazon S3 CSI ドライバー]を選択します。
6. [削除] を選択します。
7. [削除: aws-mountpoint-s3-csi-driver] 確認ダイアログボックスで、以下の操作を行います。
   a. Amazon EKS でのアドオンの設定の管理を停止する場合は、[クラスターで保持する] を選択します。クラスターでアドオンソフトウェアを保持する場合は、この操作を実行します。これは、アドオンのすべての設定を自分で管理できるようにするためです。
   b. **`aws-mountpoint-s3-csi-driver`** と入力します。
   c. [削除] を選択します。

AWS CLI

AWS CLI を使用して Amazon S3 CSI アドオンを削除するには

my-cluster をクラスター名に置き換えた後、次のコマンドを実行します。

```
aws eks delete-addon --cluster-name my-cluster --addon-name aws-mountpoint-s3-csi-driver --preserve
```

CSI スナップショットコントローラー

コンテナストレージインターフェイス (CSI) スナップショットコントローラーを使用すると、Amazon EBS CSI ドライバーなどの互換性のある CSI ドライバーのスナップショット機能を使用できます。

CSI スナップショットコントローラーを使用する場合の考慮事項を以下に示します。

- スナップショットコントローラーは、スナップショット機能を備えた CSI ドライバーと一緒にインストールする必要があります。Amazon EBS CSI ドライバーは、Amazon EBS CSI マネージドボリュームの Amazon EBS スナップショットの作成をサポートします。インストール手順については、「Amazon EBS CSI ドライバー」を参照してください。
- Kuberneteswith プロビジョナーを使用する Amazon EBS ボリュームなど、CSI 移行経由で提供されるボリュームのスナップショットはサポートされていません。`StorageClass` `kubernetes.io/aws-ebs`ボリュームは CSI `StorageClass` ドライバープロビジョナーを参照するを使用して作成する必要があります。`ebs.csi.aws.com`移行に関する詳細については、「Amazon EBS CSI 移行に関するよくある質問」を参照してください。

Amazon EFS CSI ドライバーは Amazon EKS アドオンを使用してインストールすることをお勧めします。Amazon EKS アドオンをクラスターに追加するには、「アドオンの作成」を参照してください。アドオンの詳細については、「Amazon EKS アドオン」を参照してください。

または、Amazon EBS CSI スナップショットコントローラーの自己管理型インストールを希望する場合は、「アップストリームでの使用」を参照してください。Kubernetes `external-snapshotter` GitHub

Amazon EKS ネットワーク

Amazon EKS クラスターは VPC 内に作成されます。ポッドネットワークは、Amazon VPC コンテナネットワークインターフェイス (CNI) プラグインによって提供されます。この章では、クラスターのネットワークの詳細に関する以下のトピックについて説明します。

トピック

Amazon EKS VPC およびサブネットの要件と考慮事項

クラスターを作成する際には、VPC と、異なるアベイラビリティーゾーンに存在する 2 つ以上のサブネットを指定します。このトピックでは、クラスターで使用する VPC およびサブネットに関する Amazon EKS 固有の要件と考慮事項の概要について説明します。Amazon EKS で使用する VPC がない場合は、Amazon EKS で提供された AWS CloudFormation テンプレートを使用して作成できます。AWS Outposts でローカルクラスターまたは拡張クラスターを作成する場合は、このトピックの代わりに「Amazon EKS ローカルクラスター VPC およびサブネットの要件と考慮事項」を参照してください。

VPC の要件と考慮事項

クラスターを作成する際には、指定する VPC が次の要件と考慮事項を満たす必要があります。

- VPC には、作成するクラスター、ノード、およびその他の Kubernetes リソースで利用できる十分な数の IP アドレスが必要です。使用する VPC に十分な数の IP アドレスがない場合は、使用可能な IP アドレスの数を増やしてみてください。

  これを行うには、クラスター設定を更新して、クラスターが使用するサブネットとセキュリティグループを変更します。AWS Management Console、AWS CLI の最新バージョン、AWS CloudFormation、および `eksctl` のバージョン `v0.164.0-rc.0` 以降から更新できます。クラス

ターバージョンを正常にアップグレードするには、これを実行して、サブネットに利用可能な IP アドレスを増やす必要がある場合があります。

> **Important**
>
> 追加するサブネットはすべて、クラスターの作成時に最初に提供したのと同じ一連の AZ 内にある必要があります。新しいサブネットは、その他のすべての要件 (例えば、十分な IP アドレスを持っている必要がある) を満たす必要があります。
> 例えば、1 つのクラスターを作成し、4 つのサブネットを指定したとします。指定した順序では、1 番目のサブネットは us-west-2a アベイラビリティーゾーンにあり、2 番目と 3 番目のサブネットは us-west-2b アベイラビリティーゾーンにあり、4 番目のサブネットは us-west-2c アベイラビリティーゾーンにあります。サブネットを変更する場合は、3 つのアベイラビリティーゾーンのそれぞれに少なくとも 1 つのサブネットを指定する必要があります。また、サブネットは元のサブネットと同じ VPC 内にある必要があります。

VPC 内の CIDR ブロックよりも多くの IP アドレスが必要な場合は、VPC に追加の Classless Inter-Domain Routing (CIDR) ブロックを関連付けることで CIDR ブロックを追加できます。クラスターの作成前または作成後に、プライベート (RFC 1918) CIDR ブロックとパブリック (非 RFC 1918) CIDR ブロックを VPC に関連付けることができます。クラスターで VPC に関連付けた CIDR ブロックが認識されるまでに、最大 5 時間かかることがあります。

共有サービス VPC でトランジットゲートウェイを使用することで、IP アドレスの利用率を節約できます。詳細については、「共有サービスによる分離された VPC」および「Amazon EKS VPC routable IP address conservation patterns in a hybrid network (ハイブリッドネットワークにおける Amazon EKS VPC ルーティング可能な IP アドレス保全パターン)」を参照してください。

- Kubernetes で IPv6 アドレスを Pods およびサービスに割り当てる場合は、IPv6 CIDR ブロックを VPC に関連付けます。詳細については、「Amazon VPC ユーザーガイド」の「IPv6 CIDR ブロックと VPC の関連付け」を参照してください。
- VPC は、DNS ホスト名と DNS 解決がサポートされている必要があります。そうではない場合、ノードはクラスターに登録されません。詳細については、「Amazon VPC ユーザーガイド」の「DNS attributes for your VPC」(VPC の DNS 属性) を参照してください。
- VPC では、AWS PrivateLink を使用する VPC エンドポイントが必要になる場合があります。詳細については、「サブネットの要件と考慮事項」を参照してください。

Kubernetes 1.14 以前でクラスターを作成した場合、Amazon EKS は次のタグを VPC に追加しています。

キー	Value
kubernetes.io/cluster/ *my-cluster*	owned

このタグは Amazon EKS でのみ使用されています。サービスに影響を与えずに、このタグを削除できます。このタグは、バージョン 1.15 以降のクラスターでは使用されません。

サブネットの要件と考慮事項

クラスターを作成すると、Amazon EKS は、指定したサブネットに 2～4 つの Elastic Network Interface を作成します。これらのネットワークインターフェイスは、クラスターと VPC 間の通信を可能にします。これらのネットワークインターフェイスでは、kubectl exec や kubectl logs などの Kubernetes の機能も有効化されます。Amazon EKS が作成した各ネットワークインターフェイスの説明には、テキスト Amazon EKS *cluster-name* が書き込まれます。

Amazon EKS は、クラスターの作成時に指定した任意のサブネットにネットワークインターフェイスを作成することができます。クラスターの作成後に、Amazon EKS がネットワークインターフェイスを作成するサブネットを変更できます。クラスターの Kubernetes バージョンを更新すると、Amazon EKS は作成した元のネットワークインターフェイスを削除し、新しいネットワークインターフェイスを作成します。これらのネットワークインターフェイスは、元のネットワークインターフェイスと同じサブネット内に作成することも、元のネットワークインターフェイスとは異なるサブネット内に作成することもできます。ネットワークインターフェイスを作成するサブネットを制御する場合は、クラスターを作成するとき、またはクラスターの作成後にサブネットを更新するときに、指定するサブネットの数を 2 つだけに制限します。

クラスターのサブネットの要件

クラスターの作成または更新時に指定するサブネットは、次の要件を満たす必要があります。

- サブネットには、Amazon EKS での使用のために、それぞれ 6 個以上の IP アドレスが必要です。ただし、IP アドレスは 16 個以上を推奨します。
- サブネットは AWS Outposts、AWS Wavelength、または AWS ローカルゾーンに存在することはできません。ただし、VPC 内にサブネットが存在する場合は、セルフマネージド型ノード および Kubernetes のリソースをこれらのタイプのサブネットにデプロイできます。

- サブネットは、パブリックでもプライベートでもかまいません。ただし、可能であれば、プライベートサブネットを指定することをお勧めします。パブリックサブネットは、インターネットゲートウェイへのルートが含まれているルートテーブルを含むサブネットです。一方、プライベートサブネットは、インターネットゲートウェイへのルートが含まれていないルートテーブルを含むサブネットです。
- サブネットは以下のアベイラビリティーゾーンには配置できません。

AWS リージョン	リージョン名	拒否されたアベイラビリティーゾーン ID
us-east-1	米国東部 (バージニア北部)	use1-az3
us-west-1	米国西部 (北カリフォルニア)	usw1-az2
ca-central-1	カナダ (中部)	cac1-az3

各コンポーネントの IP アドレスファミリー使用状況

以下の表に、Amazon EKS の各コンポーネントで使用される IP アドレスファミリーを示します。ネットワークアドレス変換 (NAT) またはその他の互換システムを使用すると、表の値が "No" のファミリーに属する送信元 IP アドレスから、これらのコンポーネントに接続できます。

機能は、クラスターの IP family (ipFamily) 設定によって異なる場合があります。この設定は、Services が Kubernetes に割り当てる CIDR ブロックに使用される IP アドレスのタイプを変更します。設定値が IPv4 のクラスターは IPv4 cluster、設定値が IPv6 のクラスターは IPv6 cluster と呼ばれます。

コンポーネント	**IPv4** アドレスのみ	**IPv6** アドレスのみ	デュアルスタックアドレス
EKS API パブリックエンドポイント	はい	いいえ	いいえ
EKS API VPC エンドポイント	はい	いいえ	いいえ

コンポーネント	IPv4 アドレスのみ	IPv6 アドレスのみ	デュアルスタックアドレス
EKS Auth API パブリックエンドポイント	はい [1]	はい [1]	はい [1]
EKS Auth API VPC エンドポイント	はい [1]	はい [1]	はい [1]
EKS クラスターパブリックエンドポイント	はい	いいえ	いいえ
EKS クラスタープライベートエンドポイント	はい [2]	はい [2]	いいえ
EKS クラスターサブネット	はい [2]	いいえ	はい [2]
ノードのプライマリ IP アドレス	はい [2]	いいえ	はい [2]
Service IP アドレス用のクラスター CIDR 範囲	はい [2]	はい [2]	いいえ
VPC CNI による Pod IP アドレス	はい [2]	はい [2]	いいえ

Note

[1] エンドポイントは、IPv4 アドレスと IPv6 アドレスの両方を持つデュアルスタックです。AWS 外部のアプリケーション、クラスターのノード、およびクラスター内のポッドは、IPv4 または IPv6 のいずれかによってこのエンドポイントに到達できます。

[2] クラスターの作成時に、クラスターの IP family (ipFamily) 設定で IPv4 クラスターとIPv6 クラスターのどちらかを選択します。これは後から変更できません。代わりに、他

のクラスターを作成してワークロードを移行する際は、別の設定を選択する必要があります。

ノードのサブネットの要件

クラスターの作成時に指定するのと同じサブネットに、ノードと Kubernetes リソースをデプロイできます。ただし、これは必須ではありません。これは、クラスターの作成時に指定しなかったサブネットにも、ノードと Kubernetes リソースをデプロイできるためです。ノードを異なるサブネットにデプロイする場合、Amazon EKS は、それらのサブネットにクラスターネットワークインターフェイスを作成しません。ノードと Kubernetes リソースをデプロイするサブネットはすべて、次の要件を満たす必要があります。

- サブネットには、すべてのノードと Kubernetes リソースをデプロイするのに十分な数の使用可能な IP アドレスが必要です。
- Kubernetes が IPv6 アドレスを Pods とサービスに割り当てるようにする場合は、サブネットに関連付けられた 1 つの IPv6 CIDR ブロックと 1 つの IPv4 CIDR ブロックが必要です。詳細については、「Amazon VPC ユーザーガイド」の「IPv6 CIDR ブロックをサブネットに関連付ける」を参照してください。サブネットに関連付けられているルートテーブルには、IPv4 および IPv6 のアドレスへのルートを含める必要があります。詳細については、「Amazon VPC ユーザーガイド」の「Routes」(ルート) を参照してください。ポッドに割り当てられるのは IPv6 アドレスのみです。ただし、Amazon EKS がクラスターとノード用に作成するネットワークインターフェイスには、IPv4 および IPv6 のアドレスが割り当てられます。
- インターネットから Pods へのインバウンドアクセスが必要な場合は、ロードバランサーと ingress をデプロイするのに十分な数の利用可能な IP アドレスがあるパブリックサブネットが少なくとも 1 つあることを必ず確認してください。パブリックサブネットにロードバランサーをデプロイできます。ロードバランサーは、プライベートサブネットまたはパブリックサブネットの Pods に負荷分散を行います。可能であれば、プライベートサブネットにノードをデプロイすることをお勧めします。
- ノードをパブリック サブネットにデプロイする場合は、サブネットが IPv4 パブリックアドレスまたは IPv6 アドレスを自動割り当てする必要があります。ノードを IPv6 CIDR ブロックが関連付けられているプライベートサブネットにデプロイする場合、プライベートサブネットも IPv6 アドレスを自動割り当てする必要があります。2020 年 3 月 26 日以降に Amazon EKS AWS CloudFormation テンプレート を使用して VPC をデプロイした場合、この設定は有効になっています。テンプレートを使用してこの日付より前に VPC をデプロイした場合、または独自の VPC を使用した場合は、この設定を手動で有効にする必要があります。詳細については、「Amazon

VPC ユーザーガイド」の「サブネットのパブリック IPv4 アドレス指定属性を変更する」および「サブネットの IPv6 アドレス指定属性を変更する」を参照してください。

- ノードをデプロイするサブネットがプライベートサブネットであり、そのルートテーブルにネットワークアドレス変換 (NAT) デバイス (IPv4) または Egress-Only ゲートウェイ (IPv6) へのルートが含まれていない場合は、AWS PrivateLink を使用して VPC エンドポイントを VPC に追加します。VPC エンドポイントは、ノードと Pods が通信を行う必要があるすべての AWS のサービス で必要になります。例としては、Amazon ECR、Elastic Load Balancing、Amazon CloudWatch、AWS Security Token Service、Amazon Simple Storage Service (Amazon S3) などがあります。エンドポイントには、ノードが存在するサブネットを含める必要があります。すべての AWS のサービス で VPC エンドポイントがサポートされているわけではありません。詳細については、「AWS PrivateLink とは」および「AWS PrivateLink と統合する AWS サービス」を参照してください。Amazon EKS のその他の要件のリストについては、「プライベートクラスターの要件」を参照してください。
- ロードバランサーをサブネットにデプロイする場合は、サブネットに次のタグが必要です。
  - プライベートサブネット

キー	Value
kubernetes.io/role/internal-elb	1

  - パブリックサブネット

キー	Value
kubernetes.io/role/elb	1

バージョン 1.18 以前の Kubernetes クラスターが作成された場合、Amazon EKS は、指定したすべてのサブネットに次のタグを追加します。

キー	Value
kubernetes.io/cluster/ *my-cluster*	shared

ここで新しい Kubernetes クラスターを作成した場合、Amazon EKS はサブネットにタグを追加しません。1.19 より前のバージョンのクラスターが使用していたサブネットにタグがあった場合、クラスターが新しいバージョンに更新されても、タグはサブネットから自動的に削除されませんでした。AWS Load Balancer Controller のバージョン 2.1.1 以前では、このタグが必要です。新しいバージョンの Load Balancer Controller を使用している場合は、サービスを中断することなくタグを削除できます。

eksctl または Amazon EKS AWS CloudFormation VPC テンプレートのいずれかを使用して VPC をデプロイした場合、次が適用されます。

- 2020 年 3 月 26 日以降 - パブリック IPv4 アドレスは、パブリックサブネットにより、パブリックサブネットにデプロイした新しいノードに自動的に割り当てられます。
- 2020 年 3 月 26 日より前 - パブリック IPv4 アドレスは、パブリックサブネットにより、パブリックサブネットにデプロイした新しいノードに自動的に割り当てられません。

この変更は、パブリックサブネットにデプロイされた新しいノードグループに、次のような影響を与えます。

- マネージド型ノードグループ - 2020 年 4 月 22 日以降にノードグループをパブリックサブネットにデプロイした場合は、パブリックサブネットでパブリック IP アドレスの自動割り当てを有効にする必要があります。詳細については、「サブネットのパブリック IPv4 アドレッシング属性の変更」を参照してください。
- Linux、Windows、または Arm のセルフマネージド型ノードグループ - 2020 年 3 月 26 日以降にノードグループをパブリックサブネットにデプロイした場合は、パブリックサブネットでパブリック IP アドレスの自動割り当てを有効にする必要があります。それ以外の場合は、代わりにパブリック IP アドレスを使用してノードを起動する必要があります。詳細については、「サブネットのパブリック IPv4 アドレッシング属性の変更」または「インスタンスの起動時のパブリック IPv4 アドレスの割り当て」を参照してください。

共有サブネットの要件と考慮事項

VPC 共有を使用して、同じ AWS Organizations 内でサブネットを他の AWS アカウントと共有できます。共有サブネットに Amazon EKS クラスターを作成できますが、以下の考慮事項に注意してください。

- VPC サブネットの所有者は、参加者アカウントで Amazon EKS クラスターを作成する前に、そのアカウントとサブネットを共有する必要があります。

- VPC のデフォルトセキュリティグループは所有者に属しているため、デフォルトのセキュリティグループを使用してリソースを起動することはできません。また、参加者は、他の参加者または所有者が所有するセキュリティグループを使用してリソースを起動することはできません。
- 共有サブネットでは、参加者と所有者がそれぞれのアカウント内のセキュリティグループを個別に管理します。サブネットの所有者は、参加者が作成したセキュリティグループを表示できますが、これらのグループに対してアクションを実行することはできません。サブネットの所有者がこれらのセキュリティグループの削除または変更を希望する場合は、セキュリティグループを作成した参加者がそのアクションを実行する必要があります。
- 参加者がクラスターを作成する場合、以下の考慮事項が適用されます。
  - クラスター IAM ロールとノード IAM ロールは、そのアカウントで作成する必要があります。詳細については、Amazon EKS クラスター の IAM ロールおよびAmazon EKS ノードの IAM ロールを参照してください。
  - 管理対象ノードグループを含め、すべてのノードは同じ参加者が作成する必要があります。
- 共有 VPC の所有者は、参加者が共有サブネット内に作成したクラスターを表示、更新、削除することはできません。これは、アカウントごとに異なるアクセス権を持つ VPC リソースに加えて適用されます。詳細については、「Amazon VPC ユーザーガイド」の「所有者および参加者の責任と権限」を参照してください。
- Amazon VPC CNI plugin for Kubernetes のカスタムネットワーク機能を使用する場合、所有者アカウントに記載されているアベイラビリティーゾーン ID マッピングを使用して、それぞれの `ENIConfig` を作成する必要があります。詳細については、「ポッド用のカスタムネットワーク」を参照してください。

VPC サブネット共有の詳細については、「Amazon VPC ユーザーガイド」の「VPC を他のアカウントと共有する」を参照してください。

Amazon EKS クラスター VPC の作成

Amazon Virtual Private Cloud (Amazon VPC) を使用すると、定義した仮想ネットワーク内で AWS リソースを起動できます。この仮想ネットワークは、お客様自身のデータセンターで運用されている従来のネットワークによく似ています。ただし、Amazon Web Services のスケーラブルなインフラストラクチャを使用できるというメリットがあります。本番用に使う Amazon EKS クラスターをデプロイする前に、Amazon VPC サービスを十分に理解しておくことをお勧めします。詳細については、Amazon VPC ユーザーガイドを参照してください。

Amazon EKS クラスター、ノード、および Kubernetes リソースが VPC にデプロイされます。Amazon EKS で既存の VPC を使用する場合は、その VPC が「Amazon EKS VPC およびサブ

ネットの要件と考慮事項」に記載されている要件を満たしている必要があります。このトピックでは、Amazon EKS に用意されている AWS CloudFormation テンプレートを使用して Amazon EKS 要件を満たす VPC を作成する方法について説明します。テンプレートをデプロイすると、テンプレートによって作成されたリソースを表示して、作成されたリソースとそれらのリソースの設定を正確に把握できます。

前提条件

Amazon EKS 用の VPC を作成するには、Amazon VPC リソースを作成するのに必要な IAM アクセス許可が必要です。これらのリソースは、VPC、サブネット、セキュリティグループ、ルートテーブルとルート、およびインターネットと NAT ゲートウェイです。詳細については、「Amazon VPC ユーザーガイド」の「パブリックサブネットを持つ VPC を作成するポリシー例」を参照してください。完全なリストについては、「サービス認証リファレンス」の「Amazon EC2 のアクション、リソース、および条件キー」を参照してください。

パブリックサブネットとプライベートサブネット、パブリックサブネットのみ、またはプライベートサブネットのみで、VPC を作成できます。

Public and private subnets

この VPC には、2 つのパブリックサブネットと 2 つのプライベート サブネットがあります。パブリックサブネットに関連付けられているルートテーブルには、インターネットゲートウェイへのルートが含まれています。一方、プライベートサブネットのルートテーブルには、インターネットゲートウェイへのルートがありません。1 つのパブリックサブネットと 1 つのプライベートサブネットが同じアベイラビリティーゾーンにデプロイされます。他の公開サブネットとプライベートサブネットは、同じ AWS リージョン 内の二番目のアベイラビリティーゾーンにデプロイされます。ほとんどのデプロイにこのオプションをお勧めします。

このオプションを使用すると、プライベートサブネットにノードをデプロイできます。このオプションを使用すると、Kubernetes は、プライベートサブネット内のノードで実行される Pods へのトラフィックを負荷分散できるパブリックサブネットにロードバランサーをデプロイできます。パブリック IPv4 アドレスは、パブリックサブネットにデプロイされたノードに自動的に割り当てられます。一方、プライベートサブネットにデプロイされたノードに対しては、パブリック IPv4 アドレスは割り当てられません。

また、パブリックサブネットおよびプライベートサブネットに置かれたノードに対し、IPv6 アドレスを割り当てることもできます。プライベートサブネット内のノードは、クラスターや他の AWS のサービス との通信が可能です。Pods は、各アベイラビリティーゾーンにデプロイされた NAT ゲートウェイ (IPv4 アドレスを使用) または送信専用インターネットゲートウェイ (IPv6 アドレスを使用) を介して、インターネットへの通信が可能です。クラスターまたはノード以外

のソースからのすべてのインバウンドトラフィックを拒否するが、すべてのアウトバウンドトラフィックを許可するルールを持つセキュリティグループがデプロイされます。Kubernetes がサブネットにロードバランサーをデプロイできるように、サブネットにはタグが付けられています。

VPC を作成するには

1. https://console.aws.amazon.com/cloudformation で AWS CloudFormation コンソール を開きます。
2. ナビゲーションバーで、Amazon EKS をサポートする AWS リージョン を選択します。
3. [Create stack] (スタックの作成) を選択し、[With new resources (standard)] (新しいリソースの使用 (標準)) を選択します。
4. [Prerequisite - Prepare template] (前提条件 – テンプレートの準備) の下で、[Template is ready] (テンプレートの準備完了) がオンになっていることを確認した上で、[Specify template] (テンプレートの指定) の下で、[Amazon S3 URL] を選択します。
5. IPv4 のみをサポートする VPC、または IPv4 と IPv6 をサポートする VPC を作成できます。次の URL の 1 つを [Amazon S3 URL] の下にあるテキスト領域に貼り付けて、[Next] (次へ) を選択します。

   - IPv4

     ```
     https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml
     ```

   - IPv4 および IPv6

     ```
     https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-ipv6-vpc-public-private-subnets.yaml
     ```

6. [Specify stack details] (スタック詳細の指定) ページで、パラメータを入力し、[Next] (次へ) を選択します。

   - [スタック名]: AWS CloudFormation スタックのスタック名を選択します。例えば、前のステップで使用されたテンプレート名を使用できます。この名前には、英数字 (大文字と小文字が区別されます) とハイフンのみを使用できます。先頭の文字は英数字である必要があります。また、100 文字より長くすることはできません。名前は、クラスターを作成する AWS リージョン および AWS アカウント 内で一意である必要があります。
   - [VpcBlock]: VPC の IPv4 CIDR 範囲を選択します。デプロイする各ノード、Pod、およびロードバランサーには、このブロックの IPv4 アドレスが割り当てられます。デフォルト

の IPv4 値で、ほとんどの実装において十分な IP アドレスを取得できますが、不足する場合はこの値を変更できます。詳細については、「Amazon VPC ユーザーガイド」の「VPC とサブネットのサイズ設定」を参照してください。VPC を作成したら、追加の CIDR ブロックを VPC に追加することもできます。IPv6 VPC を作成している場合、IPv6 CIDR 範囲は、Amazon のグローバルユニキャストアドレススペースから自動的に割り当てられます。

- [PublicSubnet01Block]: パブリックサブネット 1 の IPv4 CIDR ブロックを指定します。デフォルト値は、ほとんどの実装で十分な IP アドレスを提供しますが、そうでない場合は、変更することができます。IPv6 VPC を作成する場合、テンプレートで指定済みブロックを使用できます。
- [PublicSubnet02Block]: パブリックサブネット 2 の IPv4 CIDR ブロックを指定します。デフォルト値は、ほとんどの実装で十分な IP アドレスを提供しますが、そうでない場合は、変更することができます。IPv6 VPC を作成する場合、テンプレートで指定済みブロックを使用できます。
- [PrivateSubnet01Block]: プライベートサブネット 1 の IPv4 CIDR ブロックを指定します。デフォルト値は、ほとんどの実装で十分な IP アドレスを提供しますが、そうでない場合は、変更することができます。IPv6 VPC を作成する場合、テンプレートで指定済みブロックを使用できます。
- [PrivateSubnet02Block]: プライベートサブネット 2 の IPv4 CIDR ブロックを指定します。デフォルト値は、ほとんどの実装で十分な IP アドレスを提供しますが、そうでない場合は、変更することができます。IPv6 VPC を作成する場合、テンプレートで指定済みブロックを使用できます。

7. (オプション) [Configure stack options] (スタックオプションの設定) ページで、スタックリソースにタグ付けを行った上で、[Next] (次へ) をクリックします。
8. Review ページで、スタックの作成を選択します。
9. スタックが作成されたら、コンソールで選択し、[Outputs] (出力) を選択します。
10. 作成された VPC の [VpcId] を記録します。これは、クラスターとノードを作成するときに必要です。
11. 作成されたサブネットの SubnetIds と、それらをパブリックサブネットとプライベートサブネットのどちらとして作成したかを記録します。クラスターとノードを作成するときは、これらのうち少なくとも 2 つが必要です。
12. IPv4 VPC を作成している場合、この手順をスキップしてください。IPv6 VPC を作成している場合は、テンプレートによって作成されたパブリックサブネットにおいて、IPv6 アドレスの自動割り当てオプションが有効化されている必要があります。この設定は、プライ

ベートサブネットに対してすでに有効になっています。次の手順を完了して、設定を有効にします。

a. Amazon VPC コンソール (https://console.aws.amazon.com/vpc/) を開きます。
b. 左のナビゲーションペインで [Subnets] (サブネット) を選択します。
c. パブリックサブネット (***stack-name***/SubnetPublic01 または ***stack-name***/SubnetPublic02 には、public という語が含まれています) の 1 つを選択し、[Actions] (アクション)、[Edit subnet settings] (サブネット設定の編集) の順にクリックします。
d. [Enable auto-assign **IPv6** address] (アドレスの自動割り当てを有効にする) チェックボックスを選択し、[Save] (保存) を選択します。
e. 他のパブリックサブネットに対しても、ここまでの手順を再度実行します。

Only public subnets

この VPC には、AWS リージョン 内の異なるアベイラビリティーゾーンにデプロイされる 3 つのパブリックサブネットがあります。すべてのノードには自動的にパブリック IPv4 アドレスが割り当てられ、インターネットゲートウェイを介してインターネットトラフィックを送受信できます。すべてのインバウンドトラフィックを拒否し、すべてのアウトバウンドトラフィックを許可するセキュリティグループがデプロイされます。Kubernetes がサブネットにロードバランサーをデプロイできるように、サブネットにはタグが付けられています。

VPC を作成するには

1. https://console.aws.amazon.com/cloudformation で AWS CloudFormation コンソール を開きます。
2. ナビゲーションバーで、Amazon EKS をサポートする AWS リージョン を選択します。
3. [Create stack] (スタックの作成) を選択し、[With new resources (standard)] (新しいリソースの使用 (標準)) を選択します。
4. [テンプレートの準備] の下で、[テンプレートの準備完了] が選択されていることを確認し、[テンプレートソース] の下で、[Amazon S3 URL] を選択します。
5. 次の URL を [Amazon S3 URL] の下のテキスト領域に貼り付けて、[Next] (次へ) を選択します。

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-vpc-sample.yaml
```

6. [Specify Details] (詳細の指定) ページで、パラメータを入力し、[Next] (次へ) を選択します。

- [スタック名]: AWS CloudFormation スタックのスタック名を選択します。例えば、***amazon-eks-vpc-sample*** という名前にすることができます。この名前には、英数字 (大文字と小文字が区別されます) とハイフンのみを使用できます。先頭の文字は英数字である必要があります。また、100 文字より長くすることはできません。名前は、クラスターを作成する AWS リージョン および AWS アカウント 内で一意である必要があります。
- [VpcBlock]: VPC の CIDR ブロックを選択します。デプロイする各ノード、Pod、およびロードバランサーには、このブロックの IPv4 アドレスが割り当てられます。デフォルトの IPv4 値で、ほとんどの実装において十分な IP アドレスを取得できますが、不足する場合はこの値を変更できます。詳細については、「Amazon VPC ユーザーガイド」の「VPC とサブネットのサイズ設定」を参照してください。VPC を作成したら、追加の CIDR ブロックを VPC に追加することもできます。
- Subnet01Block: サブネット 1 の CIDR ブロックを指定します。デフォルト値は、ほとんどの実装で十分な IP アドレスを提供しますが、そうでない場合は、変更することができます。
- Subnet02Block: サブネット 2 の CIDR ブロックを指定します。デフォルト値は、ほとんどの実装で十分な IP アドレスを提供しますが、そうでない場合は、変更することができます。
- Subnet03Block: サブネット 3 の CIDR ブロックを指定します。デフォルト値は、ほとんどの実装で十分な IP アドレスを提供しますが、そうでない場合は、変更することができます。

7. (オプション) [Options] (オプション) ページで、スタックリソースをタグ付けします。[Next] を選択します。
8. [Review] ページで、[Create] を選択します。
9. スタックが作成されたら、コンソールで選択し、[Outputs] (出力) を選択します。
10. 作成された VPC の [VpcId] を記録します。これは、クラスターとノードを作成するときに必要です。
11. 作成されたサブネットの [SubnetIds] を記録します。クラスターとノードを作成するときは、これらのうち少なくとも 2 つが必要です。
12. (オプション) この VPC にデプロイするクラスターは、Pods と services にプライベート IPv4 アドレスを割り当てることができます。この VPC にクラスターをデプロイし、Pods と services にプライベート IPv6 アドレスを割り当てる場合は、VPC、サブネット、ルートテーブル、およびセキュリティグループを更新します。詳細については、「Amazon VPC ユーザーガイド」の「既存の VPC を IPv4 から IPv6 に移行する」を参照してくださ

い。Amazon EKS では、サブネットにおいて、IPv6 アドレスの Auto-assign オプションが有効化されている必要があります。デフォルトでは、このオプションは無効となっています。

Only private subnets

この VPC には、AWS リージョン 内の異なるアベイラビリティーゾーンにデプロイされる 3 つのプライベートサブネットがあります。サブネットにデプロイされたリソースはインターネットにアクセスできず、インターネットからサブネット内のリソースにアクセスすることもできません。ノードが通常アクセスする必要があるいくつかの AWS のサービス の AWS PrivateLink を使用して、VPC エンドポイントがテンプレートによって作成されます。ノードにアウトバウンドインターネットアクセスが必要な場合は、VPC が作成された後で、各サブネットのアベイラビリティーゾーン内に、パブリック NAT ゲートウェイを追加できます。セキュリティグループは、サブネットにデプロイされたリソースからのトラフィックを除き、すべてのインバウンドトラフィックを拒否するように作成されます。また、セキュリティグループは、すべてのアウトバウンドトラフィックを許可します。Kubernetes がサブネットに内部ロードバランサーをデプロイできるように、サブネットにはタグが付けられています。この設定で VPC を作成する場合は、「プライベートクラスターの要件」を参照してその他の要件と考慮事項を確認してください。

VPC を作成するには

1. https://console.aws.amazon.com/cloudformation で AWS CloudFormation コンソール を開きます。
2. ナビゲーションバーで、Amazon EKS をサポートする AWS リージョン を選択します。
3. [Create stack] (スタックの作成) を選択し、[With new resources (standard)] (新しいリソースの使用 (標準)) を選択します。
4. [テンプレートの準備] の下で、[テンプレートの準備完了] が選択されていることを確認し、[テンプレートソース] の下で、[Amazon S3 URL] を選択します。
5. 次の URL を [Amazon S3 URL] の下のテキスト領域に貼り付けて、[Next] (次へ) を選択します。

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-fully-private-vpc.yaml
```

6. [Specify Details] (詳細の指定) ページで、パラメータを入力し、[Next] (次へ) を選択します。

- [スタック名]: AWS CloudFormation スタックのスタック名を選択します。例えば、***amazon-eks-fully-private-vpc*** という名前にすることができます。この名前には、英数字 (大文字と小文字が区別されます) とハイフンのみを使用できます。先頭の文字は英数字である必要があります。また、100 文字より長くすることはできません。名前は、クラスターを作成する AWS リージョン および AWS アカウント 内で一意である必要があります。
- [VpcBlock]: VPC の CIDR ブロックを選択します。デプロイする各ノード、Pod、およびロードバランサーには、このブロックの IPv4 アドレスが割り当てられます。デフォルトの IPv4 値で、ほとんどの実装において十分な IP アドレスを取得できますが、不足する場合はこの値を変更できます。詳細については、「Amazon VPC ユーザーガイド」の「VPC とサブネットのサイズ設定」を参照してください。VPC を作成したら、追加の CIDR ブロックを VPC に追加することもできます。
- PrivateSubnet01Block: サブネット 1 の CIDR ブロックを指定します。デフォルト値は、ほとんどの実装で十分な IP アドレスを提供しますが、そうでない場合は、変更することができます。
- PrivateSubnet02Block: サブネット 2 の CIDR ブロックを指定します。デフォルト値は、ほとんどの実装で十分な IP アドレスを提供しますが、そうでない場合は、変更することができます。
- PrivateSubnet03Block: サブネット 3 の CIDR ブロックを指定します。デフォルト値は、ほとんどの実装で十分な IP アドレスを提供しますが、そうでない場合は、変更することができます。

7. (オプション) [Options] (オプション) ページで、スタックリソースをタグ付けします。[Next] を選択します。
8. [Review] ページで、[Create] を選択します。
9. スタックが作成されたら、コンソールで選択し、[Outputs] (出力) を選択します。
10. 作成された VPC の [VpcId] を記録します。これは、クラスターとノードを作成するときに必要です。
11. 作成されたサブネットの [SubnetIds] を記録します。クラスターとノードを作成するときは、これらのうち少なくとも 2 つが必要です。
12. (オプション) この VPC にデプロイするクラスターは、Pods と services にプライベート IPv4 アドレスを割り当てることができます。この VPC にクラスターをデプロイし、Pods と services にプライベート IPv6 アドレスを割り当てる場合は、VPC、サブネット、ルートテーブル、およびセキュリティグループを更新します。詳細については、「Amazon VPC ユーザーガイド」の「既存の VPC を IPv4 から IPv6 に移行する」を参照してくださ

い。Amazon EKS では、サブネットにおいて、IPv6 アドレスの Auto-assign オプションが有効化されている必要があります (デフォルトでは無効)。

Amazon EKS セキュリティグループの要件および考慮事項

このトピックでは、Amazon EKS クラスターのセキュリティグループの要件について説明します。

クラスターを作成すると、Amazon EKS により eks-cluster-sg-*my-cluster*-*uniqueID* という名前のセキュリティグループが作成されます。セキュリティグループには、デフォルトで次のルールがあります。

[Rule type] (ルールタイプ)	プロトコル	ポート	送信元	デスティネーション
インバウンド	すべて	すべて	自分	
アウトバウンド	すべて	すべて		0.0.0.0/0 (IPv4) または ::/0 (IPv6)

> **Important**
>
> クラスターにアウトバウンドルールが必要ない場合は、削除できます。削除する場合でも、「クラスタートラフィックの制限」に記載されている最低限のルールが適用されます。インバウンドルールを削除すると、Amazon EKS はクラスターが更新されるたびにそのルールを再作成します。

Amazon EKS は次のタグをセキュリティグループに追加します。タグを削除した場合、Amazon EKS はクラスターが更新されるたびにこれらのタグをセキュリティグループに追加します。

キー	Value
kubernetes.io/cluster/ *my-cluster*	owned
aws:eks:cluster-name	*my-cluster*

キー	Value
Name	eks-cluster-sg- *my-cluster -uniqueid*

Amazon EKS は、同様に作成される次のリソースに、セキュリティグループを自動的に関連付けます。

- 2—4 エラスティックネットワークインターフェイス (これ以降、ネットワークインターフェイス) は、クラスターの作成時に作成されます。
- 作成したマネージドノードグループ内のノードのネットワークインターフェイス。

デフォルトのルールでは、すべてのトラフィックがクラスターとノード間で自由に行き来することができます。また、任意の送信先へのすべてのアウトバウンドトラフィックが許可されています。クラスターを作成すると、オプションで独自のセキュリティグループを指定できます。その場合、Amazon EKS は、指定したセキュリティグループをクラスター用に作成するネットワークインターフェイスにも関連付けます。ただし、作成したノードグループには関連付けられません。

クラスターのセキュリティグループの ID は、AWS Management Console のクラスターの [Networking] (ネットワーキング) セクションで確認できます。もしくは、次の AWS CLI コマンドを実行して確認できます。

```
aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.clusterSecurityGroupId
```

クラスタートラフィックの制限

クラスターとノード間で開いているポートの数を制限する必要がある場合は、デフォルトのアウトバウンドルールを削除し、クラスターに必要な次の最低限のルールを追加できます。デフォルトのインバウンドルールを削除すると、Amazon EKS はクラスターが更新されるたびにそのルールを再作成します。

[Rule type] (ルールタイプ)	プロトコル	ポート	デスティネーション
アウトバウンド	TCP	443	クラスターセキュリティグループ
アウトバウンド	TCP	10250	クラスターセキュリティグループ
アウトバウンド (DNS)	TCP と UDP	53	クラスターセキュリティグループ

次のトラフィックにもルールを追加する必要があります。

- ノード間通信にノードが使用するプロトコルおよびポート。
- アウトバウンドのインターネットアクセス。これによりノードが Amazon EKS API にアクセス可能になり、起動時のノードの登録やクラスターの詳細分析が行えます。ノードがインターネットにアクセスできない場合は、「プライベートクラスターの要件」でその他の考慮事項を参照してください。
- Amazon ECR や、イメージをプルする必要があるその他のコンテナレジストリ (DockerHub など) からコンテナイメージをプルするためのノードのアクセス。詳細については、AWS 全般のリファレンス の「AWS IP アドレスの範囲」を参照してください。
- Amazon S3 へのノードのアクセス。
- `IPv4` や `IPv6` アドレスが必要な個別のルール。

ルールの制限を検討している場合は、変更したルールを本番稼働用のクラスターに適用する前に、すべての Pods を徹底的にテストすることをお勧めします。

最初に Kubernetes `1.14` と `eks.3` 以前のプラットフォームバージョンを使用してクラスターをデプロイした場合は、次の点を考慮してください。

- コントロールプレーンおよびノードセキュリティグループがある場合もあります。これらのグループの作成時、前の表に示した制限付きルールが含まれていました。これらのセキュリティグループは不要で、削除できます。ただし、それらのグループに含まれるルールがクラスターセキュリティグループに含まれていることを確認する必要があります。

- API を使用してクラスターを直接デプロイした場合、または AWS CLI や AWS CloudFormation などのツールを使用してクラスターを作成しており、クラスター作成時にセキュリティグループを指定しなかった場合、Amazon EKS が作成したクラスターネットワークインターフェイスに VPC のデフォルトのセキュリティグループが適用されます。

Amazon EKS ネットワーキングアドオン

Amazon EKS クラスターで使用できるネットワーキングアドオンがいくつかあります。

組み込みアドオン

> **Note**
>
> コンソール以外の方法でクラスターを作成した場合、各クラスターにセルフマネージド版の組み込みアドオンが付属します。セルフマネージド型は、AWS Management Console、AWS Command Line Interface、SDK のいずれからも管理することはできません。セルフマネージド型アドオンの設定とアップグレードは、ユーザーが管理します。
> セルフマネージド型のアドオンを使用する代わりに、Amazon EKS タイプのアドオンをクラスターに追加することをお勧めします。コンソールを使用してクラスターを作成する場合、これらのアドオンの Amazon EKS タイプがインストールされます。

Amazon VPC CNI plugin for Kubernetes

この CNI アドオンは、Elastic Network Interface を作成し、Amazon EC2 ノードにアタッチします。またアドオンは、VPC のプライベート`IPv4` または `IPv6` アドレスを各 Pod およびサービスに割り当てます。このアドオンはデフォルトでクラスターにインストールされます。詳細については、「Amazon VPC CNI plugin for Kubernetes Amazon EKS アドオンの使用」を参照してください。

CoreDNS

CoreDNS は、Kubernetes クラスター DNS として機能する柔軟で拡張可能な DNS サーバーです。CoreDNS は、クラスター内の Pods すべてに名前解決を提供します。このアドオンはデフォルトでクラスターにインストールされます。詳細については、「CoreDNS Amazon EKS アドオンの使用」を参照してください。

kube-proxy

このアドオンは、Amazon EC2 ノード上のネットワークルールを維持し、Pods へのネットワーク通信を可能にします。このアドオンはデフォルトでクラスターにインストールされます。詳細については、「Kubernetes kube-proxy アドオンの使用」を参照してください。

オプションの AWS ネットワークアドオン

AWS Load Balancer Controller

`loadbalancer` タイプの Kubernetes サービスオブジェクトをデプロイするとき、コントローラーは AWS Network Load Balancer を作成します。Kubernetes 入力オブジェクトを作成するとき、コントローラーは AWS Application Load Balancer を作成します。Network Load Balancer のプロビジョニングには、Kubernetes に組み込まれている従来の「レガシーのクラウドプロバイダー」コントローラーではなく、このコントローラーを使用することをお勧めします。詳細については、AWS Load Balancer Controller ドキュメントを参照してください。

AWS ゲートウェイ API コントローラー

このコントローラーを使用すると、Kubernetes ゲートウェイ API を使用して複数の Kubernetes クラスターを通してサービスに接続できます。コントローラーは、「Amazon VPC Lattice」サービスを使用して Amazon EC2 インスタンス、コンテナ、サーバーレス機能で実行されている Kubernetes サービスを接続します。詳細については、「AWS ゲートウェイ API コントローラー」ドキュメントを参照してください。

アドオンの詳細については、「Amazon EKS アドオン」を参照してください。

Amazon VPC CNI plugin for Kubernetes Amazon EKS アドオンの使用

Amazon VPC CNI plugin for Kubernetes アドオンは Amazon EKS クラスター内の各 Amazon EC2 ノードにデプロイされます。アドオンは Elastic Network Interface を作成し、Amazon EC2 ノードにアタッチします。またアドオンは、VPC のプライベート `IPv4` または `IPv6` アドレスを各 Pod およびサービスに割り当てます。

アドオンのバージョンはクラスター内の各 Fargate ノードにデプロイされますが、Fargate ノードでは更新しません。Amazon EKS クラスターでは他の互換性のある CNI プラグインも使用できますが、これは Amazon EKS でサポートされている唯一の CNI プラグインです。

次の表は、各 Kubernetes バージョンの Amazon EKS アドオンタイプの利用可能な最新バージョンを示しています。

Kubernetes バージョン	1.30	1.29	1.28	1.27	1.26	1.25	1.24	1.23
Amazon EKS タイプの VPC CNI バージョン	v1.18.2-eksbuilc	v1.18.2-eksbuilc	v1.18.2-eksbuilc	v1.18.2-eksbuilc	v1.18.2-eksbuilc	v1.18.2-eksbuilc	v1.18.2-eksbuilc	v1.18.2-eksbuild.1

Important

このアドオンを自己管理している場合、表のバージョンは、利用可能なセルフマネージドバージョンと同じではない可能性があります。このアドオンのセルフマネージドタイプの更新の詳細については、「セルフマネージド型アドオンを更新する」を参照してください。

Important

VPC CNI v1.12.0 以降にアップグレードするには、まず VPC CNI v1.7.0 にアップグレードする必要があります。一度に 1 つのマイナーバージョンを更新することをお勧めします。

前提条件

- 既存の Amazon EKS クラスター。デプロイするには、「Amazon EKS の使用開始」を参照してください。
- クラスター用の既存 AWS Identity and Access Management IAM OpenID Connect (OIDC) プロバイダー。既に存在しているかどうかを確認する、または作成するには「クラスターの IAM OIDC プロバイダーを作成する」を参照してください。
- AmazonEKS_CNI_Policy IAM ポリシー (クラスターが IPv4 ファミリーを使用している場合)、または IPv6 ポリシー (クラスターが IPv6 ファミリーを使用している場合) がアタッチされた IAM ロール。詳細については、「サービスアカウントの IAM ロールを使用する Amazon VPC CNI plugin for Kubernetes の設定」を参照してください。
- Amazon VPC CNI plugin for Kubernetes のバージョン 1.7.0 以降の CNI プラグインを使用しており、カスタム Pod セキュリティポリシーを使用する場合は、「デフォルトの Amazon EKS Pod セキュリティポリシーを削除するポッドのセキュリティポリシー」を参照してください。

- > **Important**
  >
  > Amazon VPC CNI plugin for Kubernetes バージョン `v1.16.0` から `v1.16.1` では、Kubernetes バージョン `1.23` 以前との互換性を削除されています。VPC CNI バージョン `v1.16.2` では、 Kubernetes バージョン `1.23` 以前、および CNI 仕様 `v0.4.0` との互換性が復元されています。

  Amazon VPC CNI plugin for Kubernetes バージョン `v1.16.0` から `v1.16.1` では CNI 仕様バージョン `v1.0.0` が実装されています。CNI 仕様 `v1.0.0` は、Kubernetes バージョン `v1.24` 以降を実行する EKS クラスターでサポートされています。VPC CNI バージョン `v1.16.0` から `v1.16.1` および CNI 仕様 `v1.0.0` は、Kubernetes バージョン `v1.23` 以前ではサポートされていません。CNI 仕様の `v1.0.0` の詳細については、「Container Network Interface (CNI) 仕様」を参照してください

考慮事項

- バージョンは `major-version.minor-version.patch-version-eksbuild.build-number` として指定されます。
- 各機能のバージョン互換性を確認

  Amazon VPC CNI plugin for Kubernetes の各リリースの一部の機能には、特定の Kubernetes バージョンが必要です。異なる Amazon EKS の機能を使用する際、アドオンの特定のバージョンが必要な場合については、機能に関するドキュメントに記載されています。以前のバージョンを実行する特定の理由がある場合を除いて、最新のバージョンを実行することをお勧めします。

Amazon EKS アドオンの作成

Amazon EKS タイプのアドオンを作成します。

1. クラスターにインストールされているアドオンのバージョンを確認します。

   ```
   kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni:
    | cut -d : -f 3
   ```

   出力例は次のとおりです。

```
v1.16.4-eksbuild.2
```

2. クラスターにインストールされているアドオンのタイプを確認します。クラスターを作成するために使用したツールによっては、現在クラスターに Amazon EKS アドオンタイプがインストールされていない場合があります。*my-cluster* の部分は、自分のクラスター名に置き換えます。

```
$ aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 addon.addonVersion --output text
```

バージョン番号が返された場合、Amazon EKS タイプのアドオンがクラスターにインストールされているため、このステップの残りのステップを完了する必要はありません。エラーが返された場合、クラスターに Amazon EKS タイプのアドオンがインストールされていません。インストールするには、この手順の残りのステップを完了します。

3. 現在インストールされているアドオンの設定を保存します。

```
kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
```

4. AWS CLI を使用してアドオンを作成します。AWS Management Console または `eksctl` を使用してアドオンを作成する場合は、「アドオンの作成」を参照して、アドオン名の `vpc-cni` を指定します。デバイスに沿ったコマンドをコピーします。必要に応じてコマンドに次の変更を加え、変更したコマンドを実行します。

   - *my-cluster* を自分のクラスター名に置き換えます。
   - *v1.18.2-eksbuild.1* を、クラスターバージョンの最新バージョンの表に記載されている最新バージョンに置き換えます。
   - *111122223333* を、アカウント ID に置き換えます。また、*AmazonEKSVPCCNIRole* を、作成した既存の IAM ロールの名前に置き換えます。ロールを指定するには、クラスター用に IAM OpenID Connect (OIDC) プロバイダーが必要です。クラスター用に持っているかどうかを確認、あるいは作成するには、「クラスターの IAM OIDC プロバイダーを作成する」を参照してください。

```
aws eks create-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version v1.18.2-eksbuild.1 \
    --service-account-role-arn arn:aws:iam::111122223333:role/AmazonEKSVPCCNIRole
```

Amazon EKS アドオンのデフォルト設定と競合するカスタム設定を現在のアドオンに適用した場合、作成が失敗する可能性があります。作成が失敗した場合、エラーが表示されます。これは、問題の解決に役立てることができます。または、前のコマンドに **--resolve-conflicts OVERWRITE** を追加することもできます。これにより、アドオンは既存のカスタム設定を上書きできます。アドオンを作成したら、カスタム設定で更新できます。

5. クラスターの Kubernetes バージョン用のアドオンの最新バージョンがクラスターに追加されたことを確認します。*my-cluster* を自分のクラスター名に置き換えます。

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 addon.addonVersion --output text
```

アドオンの作成が完了するまでに数秒かかる場合があります。

出力例は次のとおりです。

```
v1.18.2-eksbuild.1
```

6. 元のアドオンに対してカスタム設定を行った場合は、Amazon EKS アドオンを作成する前に、前のステップで保存した設定を使用して、カスタム設定で Amazon EKS アドオンを更新します。
7. (オプション)cni-metrics-helper をクラスタにインストールします。 メトリクスヘルパーは、ネットワークインターフェイスと IP アドレス情報を収集し、クラスターレベルでメトリクスを集計し、メトリクスを Amazon CloudWatch に発行するために使用できるツールです。詳細については、GitHub の CNI を参照してください。

Amazon EKS アドオンの更新

Amazon EKS タイプのアドオンを更新します。Amazon EKS タイプのアドオンをクラスターに追加していない場合は、この手順を完了する代わりに、アドオンを追加するか、「セルフマネージド型アドオンを更新する」を参照してください。

1. クラスターにインストールされているアドオンのバージョンを確認します。*my-cluster* をクラスター名に置き換えます。

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 "addon.addonVersion" --output text
```

出力例は次のとおりです。

```
v1.16.4-eksbuild.2
```

返されたバージョンが、最新バージョンの表にあるクラスターの Kubernetes バージョンのバージョンと同じである場合は、既に最新バージョンがクラスターにインストールされているため、この手順の残りを完了する必要はありません。出力にバージョン番号ではなくエラーが表示される場合は、Amazon EKS タイプのアドオンがクラスターにインストールされていません。この手順でアドオンを更新する前に、アドオンを作成する必要があります。

2. 現在インストールされているアドオンの設定を保存します。

```
kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
```

3. AWS CLI を使用してアドオンを更新します。AWS Management Console または eksctl を使用してアドオンを更新する場合は、「アドオンの更新」を参照してください。デバイスに沿ったコマンドをコピーします。必要に応じてコマンドに次の変更を加え、変更したコマンドを実行します。

   - *my-cluster* を自分のクラスター名に置き換えます。
   - *v1.18.2-eksbuild.1* を、クラスターバージョンの最新バージョンの表に記載されている最新バージョンに置き換えます。
   - *111122223333* を、アカウント ID に置き換えます。また、*AmazonEKSVPCCNIRole* を、作成した既存の IAM ロールの名前に置き換えます。ロールを指定するには、クラスター用に IAM OpenID Connect (OIDC) プロバイダーが必要です。クラスター用に持っているかどうかを確認、あるいは作成するには、「クラスターの IAM OIDC プロバイダーを作成する」を参照してください。
   - **--resolve-conflicts** *PRESERVE* オプションはアドオンの既存の設定値を保存します。アドオン設定にカスタム値を設定していて、このオプションを使用しない場合、Amazon EKS は値をデフォルト値で上書きします。このオプションを使用する場合、実稼働クラスターのアドオンを更新する前に、非稼動クラスターのフィールドおよび値変更をテストすることをお勧めします。この値を OVERWRITE に変更する場合、すべての設定が Amazon EKS のデフォルト値に変更されます。いずれかの設定にカスタム値を設定した場合、Amazon EKS のデフォルト値で上書きされる可能性があります。この値を none に変更した場合、Amazon EKS は設定の値を一切変更しませんが、更新が失敗する可能性があります。更新に失敗した場合、競合の解決に役立つエラーメッセージが返されます。

- 構成設定を更新しない場合は、コマンドから **--configuration-values '{*"env":{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}*}'** を削除します。構成設定を更新する場合は、*"env":{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}* を、希望する設定に置き換えます。この例では、AWS_VPC_K8S_CNI_EXTERNALSNAT 環境変数は true に設定されています。指定する値は、設定スキーマに対して有効である必要があります。設定スキーマが不明である場合は、**aws eks describe-addon-configuration --addon-name vpc-cni --addon-version *v1.18.2-eksbuild.1*** を実行して、*v1.18.2-eksbuild.1* を、設定を表示するアドオンのバージョン番号に置き換えます。出力でスキーマが返されます。既存のカスタム設定があり、それをすべて削除してすべての設定の値を Amazon EKS のデフォルトに戻したい場合は、コマンドから *"env":{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}* を削除して、**{}** が空になるようにします。各設定の説明については、GitHub の「CNI 設定変数」を参照してください。

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version v1.18.2-eksbuild.1 \
    --service-account-role-arn arn:aws:iam::111122223333:role/AmazonEKSVPCCNIRole
 \
    --resolve-conflicts PRESERVE --configuration-values '{"env":
{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}}'
```

更新が完了するまでに数秒かかる場合があります。

4. アドオンのバージョンが更新されたことを確認します。*my-cluster* を自分のクラスター名に置き換えます。

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni
```

更新が完了するまでに数秒かかる場合があります。

出力例は次のとおりです。

```
{
    "addon": {
        "addonName": "vpc-cni",
        "clusterName": "my-cluster",
        "status": "ACTIVE",
        "addonVersion": "v1.18.2-eksbuild.1",
        "health": {
            "issues": []
        },
```

```
        "addonArn": "arn:aws:eks:region:111122223333:addon/my-cluster/vpc-
cni/74c33d2f-b4dc-8718-56e7-9fdfa65d14a9",
        "createdAt": "2023-04-12T18:25:19.319000+00:00",
        "modifiedAt": "2023-04-12T18:40:28.683000+00:00",
        "serviceAccountRoleArn":
 "arn:aws:iam::111122223333:role/AmazonEKSVPCCNIRole",
        "tags": {},
        "configurationValues": "{\"env\":{\"AWS_VPC_K8S_CNI_EXTERNALSNAT\":\"true
\"}}"
    }
}
```

セルフマネージド型アドオンを更新する

> **Important**
>
> セルフマネージド型のアドオンを使用する代わりに、Amazon EKS タイプのアドオンをクラスターに追加することをお勧めします。タイプの違いがよくわからない場合は、「the section called “Amazon EKS アドオン”」を参照してください。Amazon EKS アドオンをクラスターに追加する方法については、「the section called “アドオンの作成”」を参照してください。Amazon EKS アドオンを使用できない場合は、その理由に関する問題をコンテナロードマップ GitHub リポジトリに送信することをお勧めします。

1. Amazon EKS タイプのアドオンがクラスターにインストールされていないことを確認します。*my-cluster* の部分は、自分のクラスター名に置き換えます。

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
    addon.addonVersion --output text
   ```

   エラーメッセージが返された場合、クラスターに Amazon EKS タイプのアドオンがインストールされていません。アドオンを自己管理するには、この手順の残りのステップを完了してアドオンを更新します。バージョン番号が返された場合、クラスターに Amazon EKS タイプのアドオンがインストールされています。更新するには、この手順を使用するのではなく、「アドオンの更新」の手順を使用してください。アドオンタイプの違いがよくわからない場合は、「Amazon EKS アドオン」を参照してください。
2. クラスターに現在インストールされているコンテナイメージのバージョンを確認します。

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni:
 | cut -d : -f 3
```

出力例は次のとおりです。

```
v1.16.4-eksbuild.2
```

出力にビルド番号が含まれていない場合があります。

3. 現在の設定をバックアップして、バージョンを更新した後も同じ設定を構成できるようにします。

```
kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
```

4. 利用可能なバージョンを確認し、更新するバージョンの変更について理解するには、GitHub の releases を参照してください。新しいバージョンが GitHub で利用できる場合でも、利用可能な最新バージョンの表に記載されている同じ major.minor.patch バージョンに更新することをお勧めします。表に記載されているビルドバージョンは、GitHub に記載されているセルフマネージドバージョンでは指定されていません。次のオプションのいずれかでタスクを完了して、バージョンを更新します。

   - アドオンのカスタム設定がない場合は、更新先のリリースの GitHub の To apply this release: の見出しの下にあるコマンドを実行します。
   - カスタム設定がある場合は、次のコマンドを実行してマニフェストファイルをダウンロードします。*https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/v1.18.2/config/master/aws-k8s-cni.yaml* を、更新先の GitHub 上のリリースの URL に変更します。

     ```
     curl -O https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/v1.18.2/config/master/aws-k8s-cni.yaml
     ```

     必要に応じて、前のステップで作成したバックアップのカスタム設定でマニフェストを変更し、変更したマニフェストをクラスターに適用します。ノードがイメージの取得元のプライベート Amazon EKS Amazon ECR リポジトリにアクセスできない場合 (マニフェストの image: で始まる行を参照)、イメージをダウンロードして自分のリポジトリにコピーし、リポジトリからイメージを取得するようにマニフェストを変更する必要があります。詳細につい

ては、「あるリポジトリから別のリポジトリにコンテナイメージをコピーする」を参照してください。

```
kubectl apply -f aws-k8s-cni.yaml
```

5. 新しいバージョンがクラスターにインストールされたことを確認します。

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni:
 | cut -d : -f 3
```

出力例は次のとおりです。

```
v1.18.2
```

6. (オプション)cni-metrics-helper をクラスタにインストールします。 メトリクスヘルパーは、ネットワークインターフェイスと IP アドレス情報を収集し、クラスターレベルでメトリクスを集計し、メトリクスを Amazon CloudWatch に発行するために使用できるツールです。詳細については、GitHub の CNI を参照してください。

サービスアカウントの IAM ロールを使用する Amazon VPC CNI plugin for Kubernetes の設定

Amazon VPC CNI plugin for Kubernetes は、Amazon EKS クラスター内の Pod ネットワーキング用のネットワークプラグインです。プラグインは、Kubernetes ノードに VPC IP アドレスを割り当て、各ノードで Pods に必要なネットワークを設定するロールを果たします。プラグイン:

- AWS Identity and Access Management (IAM) のアクセス許可が必要。クラスターで IPv4 ファミリーを使用する場合、アクセス許可は AmazonEKS_CNI_Policy AWS 管理ポリシーで指定されます。クラスターが IPv6 ファミリーを使用する場合には、作成した IAM ポリシーにアクセス許可を追加する必要があります。このポリシーは、Amazon EKS ノード IAM ロール または 個別の IAM ロールにアタッチすることができます。このトピックで詳細に説明するように、別のロールに割り当てることをお勧めします。
- デプロイ時に作成すると、aws-node という名前の Kubernetes サービスアカウントを使用するように設定されます。このサービスアカウントは aws-node という名前の Kubernetes clusterrole にバインドされます。これには、必要な Kubernetes アクセス許可が割り当てられています。

> **Note**
>
> IMDS へのアクセスをブロックする場合を除き、Amazon VPC CNI plugin for Kubernetes 用の Pods には、Amazon EKS ノード IAM ロール に割り当てられたパーミッションへのアクセス権があります。詳細については、「ワーカーノードに割り当てられたインスタンスプロファイルへのアクセスを制限する」を参照してください。

前提条件

- 既存の Amazon EKS クラスター。デプロイするには、「Amazon EKS の使用開始」を参照してください。
- クラスター用の既存 AWS Identity and Access Management IAM OpenID Connect (OIDC) プロバイダー。既に存在しているかどうかを確認する、または作成するには「クラスターの IAM OIDC プロバイダーを作成する」を参照してください。

ステップ 1: Amazon VPC CNI plugin for Kubernetes IAM ロールを作成する

IAM ロールを作成するには

1. クラスターで使用する IP ファミリを決定します。

   ```
   aws eks describe-cluster --name my-cluster | grep ipFamily
   ```

   出力例は次のとおりです。

   ```
   "ipFamily": "ipv4"
   ```

   この出力では、代わりに `ipv6` が返されることがあります。

2. IAM ロールを作成します。IAM ロールを作成するには、`eksctl` または `kubectl` および AWS CLI を使用してます。

   eksctl

   クラスターの IP ファミリーに適合するコマンドを使用して IAM ロールを作成し、そのロールに IAM ポリシーをアタッチします。このコマンドでは、IAM ロールを作成する AWS CloudFormation スタックを作成およびデプロイし、そのために指定したポリシーをアタッ

チします。さらに、既存の aws-node Kubernetes サービスアカウントに対し、作成された IAM ロールの ARN をアノテーションします。

- IPv4

  my-cluster を独自の値に置き換えます。

  ```
  eksctl create iamserviceaccount \
      --name aws-node \
      --namespace kube-system \
      --cluster my-cluster \
      --role-name AmazonEKSVPCCNIRole \
      --attach-policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
      --override-existing-serviceaccounts \
      --approve
  ```

- IPv6

  my-cluster を独自の値に置き換えます。*111122223333* をアカウント ID に置き換え、*AmazonEKS_CNI_IPv6_Policy* を IPv6 ポリシーの名前に置き換えます。IPv6 ポリシーがない場合は、IPv6 ファミリーを使用するクラスター用に IAM ポリシーを作成します。 を参照して作成します。クラスターで IPv6 を使用するには、いくつかの要件を満たす必要があります。詳細については、「クラスター、Pods、services 用の IPv6 アドレス」を参照してください。

  ```
  eksctl create iamserviceaccount \
      --name aws-node \
      --namespace kube-system \
      --cluster my-cluster \
      --role-name AmazonEKSVPCCNIRole \
      --attach-policy-arn
   arn:aws:iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy \
      --override-existing-serviceaccounts \
      --approve
  ```

kubectl and the AWS CLI

1. クラスターの OIDC プロバイダーの URL を表示します。

```
aws eks describe-cluster --name my-cluster --query
 "cluster.identity.oidc.issuer" --output text
```

出力例は次のとおりです。

```
https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

出力が返されない場合は、クラスター用の IAM OIDC プロバイダーを作成する必要があります。

2. 次の内容を *vpc-cni-trust-policy.json* という名前のファイルにコピーします。*111122223333* をアカウント ID に置き換え、*EXAMPLED539D4633E53DE1B71EXAMPLE* を前のステップで返された値に置き換えます。*region-code* をクラスターのある AWS リージョン に置き換えます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::111122223333:oidc-provider/
oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
                    "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-
system:aws-node"
                }
            }
        }
    ]
}
```

3. ロールを作成します。*AmazonEKSVPCCNIRole* は、任意の名前で置き換えることができます。

```
aws iam create-role \
  --role-name AmazonEKSVPCCNIRole \
  --assume-role-policy-document file://"vpc-cni-trust-policy.json"
```

4. 必要な IAM ポリシーをロールにアタッチします。クラスターの IP ファミリに適合したコマンドを実行します。

   - IPv4

     ```
     aws iam attach-role-policy \
       --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
       --role-name AmazonEKSVPCCNIRole
     ```

   - IPv6

     111122223333 をアカウント名に置き換え、*AmazonEKS_CNI_IPv6_Policy* を IPv6 ポリシー名に置き換えます。IPv6 ポリシーがない場合は、IPv6 ファミリーを使用するクラスター用に IAM ポリシーを作成します。 を参照して作成します。クラスターで IPv6 を使用するには、いくつかの要件を満たす必要があります。詳細については、「クラスター、Pods、services 用の IPv6 アドレス」を参照してください。

     ```
     aws iam attach-role-policy \
       --policy-arn arn:aws:iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy \
       --role-name AmazonEKSVPCCNIRole
     ```

5. 次のコマンドを実行し、先に作成した IAM ロールの ARN で aws-node サービスアカウントをアノテーションします。*example values* を自分の値に置き換えてください。

   ```
   kubectl annotate serviceaccount \
       -n kube-system aws-node \
       eks.amazonaws.com/role-
   arn=arn:aws:iam::111122223333:role/AmazonEKSVPCCNIRole
   ```

3. (オプション) Kubernetes サービスアカウントで使用されている AWS Security Token Service エンドポイントタイプを設定します。詳細については、「サービスアカウントの AWS Security Token Service エンドポイントを設定する」を参照してください。

ステップ 2: Amazon VPC CNI plugin for KubernetesPods 再デプロイする

1. 認証情報環境変数を適用するために、サービスアカウントに関連付けられている既存の Pods を削除して再作成します。現在アノテーションなしで実行されている Pods には、アノテーションは適用されません。次のコマンドは、既存の aws-node DaemonSet Pods を削除し、サービスアカウントのアノテーションを使用してデプロイします。

```
kubectl delete Pods -n kube-system -l k8s-app=aws-node
```

2. Pods がすべて再起動したことを確認します。

```
kubectl get pods -n kube-system -l k8s-app=aws-node
```

3. Pods の 1 つの詳細を表示し、環境変数の AWS_WEB_IDENTITY_TOKEN_FILE および AWS_ROLE_ARN が存在することを確認します。*cpjw7* を、前のステップの出力で返された、いずれかの Pods の名前に置き換えます。

```
kubectl describe pod -n kube-system aws-node-cpjw7 | grep 'AWS_ROLE_ARN:\|
AWS_WEB_IDENTITY_TOKEN_FILE:'
```

出力例は次のとおりです。

```
AWS_ROLE_ARN:                 arn:aws:iam::111122223333:role/AmazonEKSVPCCNIRole
      AWS_WEB_IDENTITY_TOKEN_FILE:  /var/run/secrets/eks.amazonaws.com/
serviceaccount/token
      AWS_ROLE_ARN:
 arn:aws:iam::111122223333:role/AmazonEKSVPCCNIRole
      AWS_WEB_IDENTITY_TOKEN_FILE:           /var/run/secrets/eks.amazonaws.com/
serviceaccount/token
```

Pod には 2 つのコンテナが含まれているため、重複した結果の 2 つのセットが返されます。両方のコンテナの値は同じです。

Pod で AWS リージョン 内のエンドポイントを使用している場合、前の出力では下記の行も返されています。

```
AWS_STS_REGIONAL_ENDPOINTS=regional
```

ステップ 3: ノードの IAM ロールから CNI ポリシーを削除する

現在、Amazon EKS ノード IAM ロールに AmazonEKS_CNI_Policy IAM (IPv4) ポリシーまたはIPv6 ポリシー がアタッチされており、さらに、別の IAM ロールを作成し、このロールに対し前出のポリシーをアタッチし、そのロールを aws-node Kubernetes サービスアカウントに割り当てている場合、クラスターの IP ファミリーに適合する AWS CLI コマンドを使用して、ノードのロールからポリシーを削除することをお勧めします。*AmazonEKSNodeRole* をノードロールの名前に置き換えます。

- IPv4

```
aws iam detach-role-policy --role-name AmazonEKSNodeRole --policy-arn
 arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
```

- IPv6

  111122223333 をアカウント名に置き換え、*AmazonEKS_CNI_IPv6_Policy* を IPv6 ポリシー名に置き換えます。

```
aws iam detach-role-policy --role-name AmazonEKSNodeRole --policy-arn
 arn:aws:iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy
```

IPv6 ファミリーを使用するクラスター用に IAM ポリシーを作成します。

IPv6 ファミリーを使用するクラスターを作成し、そのクラスターでバージョン 1.10.1 以降の Amazon VPC CNI plugin for Kubernetes アドオンが設定されている場合は、IAM ロールに割り当てることができる IAM ポリシーを作成する必要があります。作成時に IPv6 ファミリーの使用を設定していない、既存のクラスターにおいて、IPv6 を使用する場合には、新しいクラスターを作成する必要があります。クラスターでの IPv6 使用の詳細については、「クラスター、Pods、services 用の IPv6 アドレス」を参照してください。

1. 次のテキストをコピーし、*vpc-cni-ipv6-policy*.json という名前のファイルに保存します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "ec2:AssignIpv6Addresses",
                "ec2:DescribeInstances",
                "ec2:DescribeTags",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeInstanceTypes"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*"
            ]
        }
    ]
}
```

2. IAM ポリシーを作成する

```
aws iam create-policy --policy-name AmazonEKS_CNI_IPv6_Policy --policy-document
 file://vpc-cni-ipv6-policy.json
```

Pod ネットワークのユースケースの選択

Amazon VPC CNI plugin for Kubernetes は Pods のネットワークを提供します. 次の表は、一緒に使用できるネットワーキングユースケースと、さまざまな Amazon EKS ノードタイプで使用できる機能および Amazon VPC CNI plugin for Kubernetes 設定を理解するのに役立ちます。この表のすべての情報は、Linux IPv4 ノードにのみ適用されます。

Amazon EKS ノードタイプ	Amazon EC2			Fargate
ユースケース	ネットワークインターフェイスに割り当てられた個々の IP アドレス	ネットワークインターフェイスに割り当てられた IP プレフィクス	Podsのセキュリティグループ	
ポッド用のカスタムネットワーク— ノードのサブネットとは異なるサブネットから IP アドレスを割り当てる	はい	はい	はい	はい (Fargate プロファイルでコントロールされるサブネット)
Pods の SNAT	はい (デフォルトは `false`)	はい (デフォルトは `false`)	はい (`true` のみ)	はい (`true` のみ)
機能				
セキュリティグループスコープ	ノード	ノード	ポッド (`POD_SECURITY_GROUP_ENFORCING_MODE =standard` および `AWS_VPC_K8S_CNI_EXTERNALSNAT =false` を設定した場合、VPC 外部のエンドポイント宛てのトラフィック	ポッド

Amazon EKS ノードタイプ	Amazon EC2			Fargate
ユースケース	ネットワークインターフェイスに割り当てられた個々の IP アドレス	ネットワークインターフェイスに割り当てられた IP プレフィクス	Podsのセキュリティグループ	
			は、Pod's セキュリティグループではなくノードのセキュリティグループを使用します)	
Amazon VPC サブネットタイプ	パブリックおよびプライベート IP	パブリックおよびプライベート IP	非公開のみ	非公開のみ
ネットワークポリシー (VPC CNI)	互換性あり	互換性あり	互換性あり Amazon VPC CNI プラグインのバージョン 1.14.0 以降のみ	サポートされていません
ノードあたりの Pod 密度	中程度	高い	低	1
Pod 起動時刻	= さらに良い	= ベスト	良好	中
Amazon VPC CNI プラグインの設定 (各設定の詳細については、「GitHub」で「amazon-vpc-cni-k8sGitHub」を参照してください)				
`WARM_ENI_TARGET`	はい	該当しない	該当しない	該当しない

Amazon EKS ノードタイプ	Amazon EC2			Fargate
ユースケース	ネットワークインターフェイスに割り当てられた個々の IP アドレス	ネットワークインターフェイスに割り当てられた IP プレフィクス	Podsのセキュリティグループ	
WARM_IP_TARGET	はい	はい	該当しない	該当しない
MINIMUM_IP_TARGET	はい	はい	該当しない	該当しない
WARM_PREFIX_TARGET	該当しない	はい	該当しない	該当しない

Note

- カスタムネットワークで IPv6 を使用することはできません。
- IPv6 アドレスの変換は行われず、SNAT は適用されません。
- セキュリティグループが関連付けられている Pods との間のトラフィックフローは、Calico ネットワークポリシーの適用の対象ではなく、Amazon VPC セキュリティ グループの適用のみに限定されます。
- Calico ネットワークポリシーの適用を使用する場合は、Kubernetes の既知の問題を回避するために、環境変数 ANNOTATE_POD_IP を true に設定することをお勧めします。この機能を使用するには、ポッドの patch のアクセス許可を aws-node ClusterRole に追加する必要があります。aws-node DaemonSet にパッチのアクセス許可を追加すると、プラグインのセキュリティ範囲が広がることに注意してください。詳細については、GitHub の VPC CNI リポジトリの ANNOTATE_POD_IP を参照してください。
- IP プレフィックスと IP アドレスは、標準の Amazon EC2 Elastic ネットワークインターフェイスに関連付けられています。特定のセキュリティグループを必要とする Pod には、ブランチネットワークインターフェイスのプライマリ IP アドレスが割り当てられます。IP アドレスを取得する Pods、または IP プレフィックスからの IP アドレスを、同じノード

上のブランチネットワークインターフェイスを取得する Pods と混在させることができます。

Windows 個のノード

各ノードは 1 つのネットワークインターフェースのみをサポートします。セカンダリ IPv4 アドレスと IPv4 プレフィックスを使用できます。デフォルトでは、ノードで使用可能な IPv4 アドレスの数は、各 Elastic Network Interface に割り当てることができるセカンダリ IPv4 アドレスの数から 1 を引いた数になります。ただし、IP プレフィックスを有効にすることで、ノードで使用可能な IPv4 アドレスと Pod 密度を増やすことができます。詳細については、「Amazon EC2 ノードで使用可能な IP アドレスの量を増やす」を参照してください。

Calico ネットワークポリシーは Windows でサポートされています。Windows では、Pods のセキュリティグループを使用したり、カスタムネットワークを使用したりすることはできません。

クラスター、Pods、services 用の **IPv6** アドレス

デフォルトでは、Kubernetes は IPv4 アドレスを Pods と services に割り当てます。Pods と services に IPv4 アドレスを割り当てる代わりに、IPv6 アドレスを割り当てるようにクラスターを設定できます。Amazon EKS は、Kubernetes がバージョン 1.23 以降でサポートしていても、デュアルスタックの Pods または services をサポートしません。つまり、IPv4 アドレスと IPv6 アドレスの両方を Pods と services に割り当てることはできません。

そのクラスターに使用する IP ファミリーは、クラスターの作成時に選択します。クラスターの作成後は、ファミリーを変更できません。

クラスターに **IPv6** ファミリーを使用する際の考慮事項

- 新しいクラスターを作成し、そのクラスターで IPv6 ファミリーの使用を指定する必要があります。これより前のバージョンからクラスターを更新して、IPv6 ファミリーを有効化することはできません。新しいクラスターを作成する手順については、「Amazon EKS クラスターの作成」を参照してください。
- クラスターにデプロイする Amazon VPC CNI アドオンには、バージョン 1.10.1 以降を使用する必要があります。このバージョン以降がデフォルトでデプロイされます。アドオンのデプロイ後は、クラスター内のすべてのノードグループ内にあるすべてのノードを削除しない限り、Amazon VPC CNI アドオンを 1.10.1 より前のバージョンにダウングレードすることはできません。
- Windows Pods および services はサポートされていません。

- Amazon EC2 ノードを使用する場合は、IP プレフィックスの委任と IPv6 を使用して Amazon VPC CNI アドオンを設定する必要があります。クラスターの作成時に IPv6 ファミリーを選択した場合は、バージョン 1.10.1 のアドオンが、この設定のデフォルトになります。これは、セルフマネージド型のアドオン、および Amazon EKS アドオンの両方に当てはまります。IP プレフィックス委任の詳細については、「Amazon EC2 ノードで使用可能な IP アドレスの量を増やす」を参照してください。
- クラスターを作成する際に指定する VPC とサブネットには、それらに割り当てられた IPv6 CIDR ブロックが必要です。同時に、IPv4 CIDR ブロックも割り当てられている必要があります。これは、IPv6 のみを使用する場合でも、VPC が機能するには IPv4 CIDR ブロックが必要になるためです。詳細については、「Amazon VPC ユーザーガイド」の「IPv6 CIDR ブロックと VPC の関連付け」を参照してください。
- クラスターとノードを作成する際は、IPv6 アドレスを自動割り当てするように設定されたサブネットを指定する必要があります。これを指定しない場合、作成したクラスターとノードのデプロイができなくなります。自動割り当ての設定はデフォルトで無効になっています。詳細については、「Amazon VPC ユーザーガイド」の「Modify the IPv6 addressing attribute for your subnet」(サブネットの アドレス属性を変更する) を参照してください。
- サブネットに割り当てられるルートテーブルには、IPv6 アドレス用のルートが必要です。詳細については、「Amazon VPC ユーザーガイド」の「既存の VPC を IPv4 から IPv6 に移行する」を参照してください。
- セキュリティグループでは IPv6 アドレスの使用を許可する必要があります。詳細については、「Amazon VPC ユーザーガイド」の「既存の VPC を IPv4 から IPv6 に移行する」を参照してください。
- IPv6 は、AWS Nitro ベースの Amazon EC2 または Fargate ノードでのみ使用が可能です。
- Amazon EC2 ノードを使用する Pods のセキュリティグループ で、IPv6 を使用することはできません。ただし、Fargate ノードでは使用が可能です。個々の Pods のそれぞれでセキュリティグループが必要な場合は、Amazon EC2 ノードでは引き続き IPv4 ファミリーを使用するか、代わりに Fargate ノードを使用します。
- IP アドレスの枯渇を緩和するために、これまで使用していたカスタムネットワーキングの代わりとして、IPv6 を使用することができます。IPv6 では、カスタムネットワーキングを使用することはできません。クラスターにおいて、ネットワークの分離用としてカスタムネットワーキングを使用する場合は、引き続き、IPv4 ファミリーによるカスタムネットワーキングを使用する必要があります。
- AWS Outposts に IPv6 を使用することはできません。

- Pods および services には、IPv6 アドレスのみが割り当てられます。これらには、IPv4 アドレスは割り当てられません。Pods は、インスタンス自体の NAT を介して IPv4 エンドポイントとの通信が可能なため、DNS64 および NAT64 は必要となりません。トラフィックがパブリック IP アドレスを必要とする場合、そのトラフィックは、送信元ネットワークアドレスとしてパブリック IP に変換されます。
- VPC の外部と通信する際の、送信元 Pod の IPv6 アドレスは、送信元のネットワークアドレスがノードの IPv6 アドレスに変換されたものではありません。このルーティングは、インターネットゲートウェイ、または送信専用インターネットゲートウェイを使用して行われます。
- すべてのノードには、IPv4 と IPv6 のアドレスが割り当てられています。
- Amazon FSx for Lustre CSI ドライバー はサポートされていません。
- IP モードでは、アプリケーションまたはネットワークからの IPv6 Pods へのトラフィックを負荷分散するために、バージョン 2.3.1 以降の AWS Load Balancer Controller が使用できますが、インスタンスモードでは使用できません。詳細については、「AWS Load Balancer Controller とは」を参照してください。
- ノード IAM または CNI IAM のロールには、IPv6 のIAM ポリシーをアタッチする必要があります。上記 2 つの中では、CNI IAM ロールへのアタッチが推奨されます。詳細については、IPv6 ファミリーを使用するクラスター用に IAM ポリシーを作成します。およびステップ 1: Amazon VPC CNI plugin for Kubernetes IAM ロールを作成するを参照してください。
- 各 Fargate Pod は、デプロイ先のサブネット用に指定された CIDR から、IPv6 アドレスを受け取ります。Fargate Pods を実行するための基盤ハードウェアユニットは、そのユニットがデプロイされているサブネットに割り当てられている CIDR から、一意の IPv4 および IPv6 アドレスを取得します。
- アプリケーション、Amazon EKS アドオン、および、IPv6 クラスターのデプロイ前に統合した AWS のサービスに関しては、包括的な評価を実施することをお勧めします。これにより、IPv6 を使用した場合も、すべてが想定どおりに動作することを保証できます。
- Amazon EC2 インスタンスメタデータサービスの IPv6 エンドポイントの使用は、Amazon EKS ではサポートされていません。
- IPv6 ファミリーを使用するクラスターでセルフマネージドノードグループを作成する場合、ユーザーデータには、ノードの起動時に実行される `bootstrap.sh` ファイルの次の `BootstrapArguments` が含まれている必要があります。*`your-cidr`* をクラスターの VPC の IPv6 CIDR 範囲に置き換えます。

```
--ip-family ipv6 --service-ipv6-cidr your-cidr
```

クラスターの IPv6 CIDR 範囲がわからない場合は、次のコマンドで確認できます (AWS CLI バージョン 2.4.9 以降が必要です)。

```
aws eks describe-cluster --name my-cluster --query
 cluster.kubernetesNetworkConfig.serviceIpv6Cidr --output text
```

IPv6 クラスターとマネージド Amazon Linux ノードをデプロイする

このチュートリアルでは、IPv6 ファミリーを使用する IPv6 Amazon VPC と Amazon EKS クラスターのデプロイ、および Amazon EC2 Amazon Linux ノードを使用するマネージド型ノードグループのデプロイ方法を説明します。IPv6 クラスター内では、Amazon EC2 Windows のノードをデプロイすることはできません。Fargate ノードをクラスターにデプロイすることもできまが、理解しやすくするため、これらの手順はこのトピックでは説明していません。

本番で使用するクラスターを作成する前に、すべての設定内容に習熟した上で、要件を満たす設定でクラスターをデプロイすることをお勧めします。詳細については、「Amazon EKS クラスターの作成」、「マネージド型ノードグループ」、ならびにこのトピックの「考慮事項」を参照してください。一部の設定の有効化は、クラスターの作成時にのみ行えます。

前提条件

このチュートリアルを開始する前に、Amazon EKS クラスターの作成と管理に必要な次のツールとリソースを、インストールおよび設定しておく必要があります。

- デバイスまたは AWS CloudShell に、kubectl コマンドラインツールがインストールされていること。バージョンは、ご使用のクラスターの Kubernetes バージョンと同じか、1 つ前のマイナーバージョン以前、あるいはそれより新しいバージョンが使用できます。例えば、クラスターのバージョンが 1.29 である場合、kubectl のバージョン 1.28、1.29、または 1.30 が使用できます。kubectl をインストールまたはアップグレードする方法については、「kubectl のインストールまたは更新」を参照してください。
- 使用している IAM セキュリティプリンシパルは、Amazon EKS の IAM ロール、サービスにリンクされたロール、AWS CloudFormation、VPC、関連リソースを使用するために許可が必要です。詳細については、「IAM ユーザーガイド」の「Amazon Elastic Kubernetes サービスのアクション、リソース、および条件キー」および「サービスにリンクされたロールの使用」を参照してください。

ここでの手順は、eksctl または AWS CLI のいずれかを使用して、リソースを作成するために提供されたものです。AWS Management Console を使用してリソースをデプロイすることも可能ですが、理解しやすさのために、このトピックではそのための手順を扱っていません。

eksctl

前提条件

eksctl のバージョン 0.183.0 以降がコンピュータにインストールされていること。インストールまたはアップグレードするには、eksctl ドキュメントの「インストール」を参照してください。

eksctl を使用して IPv6 クラスターをデプロイするには

1. ipv6-cluster.yaml ファイルを作成します。デバイスに沿ったコマンドをコピーします。必要に応じてコマンドに次の変更を加え、変更したコマンドを実行します。
   - *my-cluster* をクラスターの名前に置き換えます。この名前には、英数字 (大文字と小文字が区別されます) とハイフンのみを使用できます。先頭の文字は英数字である必要があります。また、100 文字より長くすることはできません。名前は、クラスターを作成する AWS リージョン および AWS アカウント 内で一意である必要があります。
   - *region-code* を Amazon EKS でサポートされている AWS リージョン に置き換えます。AWS リージョン の一覧については、AWS 全般的なリファレンスガイドの Amazon EKS エンドポイントとクォータ を参照してください。
   - version の値には、クラスターのバージョンを設定する必要があります。詳細については、「サポートされている Amazon EKS Kubernetes のバージョン」を参照してください。
   - *my-nodegroup* をノードグループの名前に置き換えます。ノードグループ名は 63 文字以下である必要があります。先頭は文字または数字でなければなりませんが、残りの文字にはハイフンおよびアンダースコアを含めることもできます。
   - *t3.medium* を、任意の AWS Nitro System インスタンスタイプに置き換えます。

```
cat >ipv6-cluster.yaml <<EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
```

```
  name: my-cluster
  region: region-code
  version: "X.XX"

kubernetesNetworkConfig:
  ipFamily: IPv6

addons:
  - name: vpc-cni
    version: latest
  - name: coredns
    version: latest
  - name: kube-proxy
    version: latest

iam:
  withOIDC: true

managedNodeGroups:
  - name: my-nodegroup
    instanceType: t3.medium
EOF
```

2. クラスターを作成する

```
eksctl create cluster -f ipv6-cluster.yaml
```

クラスターの作成には数分かかります。出力の最後の行が表示されるまで、次の手順には進まないでください。この行は、下記のような出力になります。

```
[...]
[#]  EKS cluster "my-cluster" in "region-code" region is ready
```

3. デフォルトの Pods に、IPv6 アドレスが割り当てられていることを確認します。

```
kubectl get pods -n kube-system -o wide
```

出力例は次のとおりです。

```
NAME                       READY   STATUS    RESTARTS   AGE     IP
                             NODE
 NOMINATED NODE   READINESS GATES
```

```
aws-node-rslts              1/1     Running   1          5m36s
  2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
aws-node-t74jh              1/1     Running   0          5m32s
  2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-cw7w2    1/1     Running   0          56m
  2600:1f13:b66:8203:34e5::                ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-tx6n8    1/1     Running   0          56m
  2600:1f13:b66:8203:34e5::1               ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
kube-proxy-btpbk            1/1     Running   0          5m36s
  2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
kube-proxy-jjk2g            1/1     Running   0          5m33s
  2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
```

4. デフォルトのサービスに、IPv6 アドレスが割り当てられていることを確認します。

```
kubectl get services -n kube-system -o wide
```

出力例は次のとおりです。

```
NAME       TYPE        CLUSTER-IP          EXTERNAL-IP   PORT(S)         AGE
 SELECTOR
kube-dns   ClusterIP   fd30:3087:b6c2::a   <none>        53/UDP,53/TCP   57m
 k8s-app=kube-dns
```

5. (オプション) サンプルアプリケーションをデプロイする か、ロードバランサーアプリケーション、または IPv6 Pods へのネットワークトラフィックのための、AWS Load Balancer Controller とサンプルアプリケーションをデプロイします。
6. このチュートリアルのために作成したクラスターとノードの使用が終了したら、それらのリソースは、次のコマンドにより削除しておきます。

```
eksctl delete cluster my-cluster
```

AWS CLI

前提条件

ご使用のデバイスまたは AWS CloudShell で、バージョン `2.12.3` 以降、または AWS Command Line Interface (AWS CLI) のバージョン `1.27.160` 以降がインストールおよび設定されていること。現在のバージョンを確認するには、「**`aws --version | cut -d / -f2 | cut -d ' ' -f1`**」を参照してください。macOS の `yum`、`apt-get`、または Homebrew などのパッケージマネージャは、AWS CLI の最新バージョンより数バージョン遅れることがあります。最新バージョンをインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI のインストール、更新、およびアンインストール」と「aws configure でのクイック設定」を参照してください。AWS CloudShell にインストールされている AWS CLI バージョンは、最新バージョンより数バージョン遅れている可能性もあります。更新するには、「AWS CloudShell ユーザーガイド」の「ホームディレクトリへの AWS CLI のインストール」を参照してください。AWS CloudShell を使用する場合は、バージョン `2.12.3` 以降または `1.27.160` 以降の AWS CLI をインストールする必要があります。AWS CloudShell には、デフォルトとして AWS CLI の古いバージョンがインストールされている可能性があります。

Important

- この手順のすべてのステップは、同一のユーザーとして実行する必要があります。現在のユーザーを確認するには、次のコマンドを実行します。

  ```
  aws sts get-caller-identity
  ```

- この手順のすべてのステップは、同一のシェル内で実行する必要があります。いくつかのステップには、これより前のステップで設定した変数を使用します。変数の値が異なるシェル内で設定されていると、その変数を使用するステップは正しく機能しません。AWS CloudShell を使用して次の手順を実行する際、約 20～30 分の間キーボードまたはポインタによる操作がないと、シェルセッションが終了することを銘記しておいてください。実行中のプロセスは、操作数としてカウントされません。
- 各命令は Bash シェル用に記述されているので、他のシェルでは修正が必要な場合があります。

AWS CLI を使用してクラスターを作成するには

この手順の各ステップにおいて、すべての *example values* は、ご自分が使用する値に置き換える必要があります。

1. 以下のコマンドを実行して、後のステップで使用するいくつかの変数を設定します。*region-code* は、リソースをデプロイする AWS リージョン に置き換えます。この値には、Amazon EKS でサポートされている任意の AWS リージョン が設定できます。AWS リージョン の一覧については、AWS 全般的なリファレンスガイドの Amazon EKS エンドポイントとクォータ を参照してください。*my-cluster* をクラスターの名前に置き換えます。この名前には、英数字 (大文字と小文字が区別されます) とハイフンのみを使用できます。先頭の文字は英数字である必要があります。また、100 文字より長くすることはできません。名前は、クラスターを作成する AWS リージョン および AWS アカウント内で一意である必要があります。*my-nodegroup* をノードグループの名前に置き換えます。ノードグループ名は 63 文字以下である必要があります。先頭は文字または数字でなければなりませんが、残りの文字にはハイフンおよびアンダースコアを含めることもできます。*111122223333* をアカウント ID に置き換えます。

```
export region_code=region-code
export cluster_name=my-cluster
export nodegroup_name=my-nodegroup
export account_id=111122223333
```

2. パブリックサブネットとプライベートサブネットを持ち、Amazon EKS と IPv6 の要件を満たす Amazon VPC を作成します。

   a. 次のコマンドを実行して、AWS CloudFormation スタック名のための変数を設定します。*my-eks-ipv6-vpc* は、好みの任意の名前で置き換えることができます。

   ```
   export vpc_stack_name=my-eks-ipv6-vpc
   ```

   b. AWS CloudFormation テンプレートを使用して IPv6 VPC を作成します。

   ```
   aws cloudformation create-stack --region $region_code --stack-name
    $vpc_stack_name \
     --template-url https://s3.us-west-2.amazonaws.com/amazon-
   eks/cloudformation/2020-10-29/amazon-eks-ipv6-vpc-public-private-
   subnets.yaml
   ```

スタックが作成されるまで数分かかります。以下のコマンドを実行します。コマンドの出力が「CREATE_COMPLETE」になるまで、次のステップに進まないでください。

```
aws cloudformation describe-stacks --region $region_code --stack-name
 $vpc_stack_name --query Stacks[].StackStatus --output text
```

c. 作成されたパブリックサブネットの ID を取得します。

```
aws cloudformation describe-stacks --region $region_code --stack-name
 $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SubnetsPublic`].OutputValue' --
output text
```

出力例は次のとおりです。

```
subnet-0a1a56c486EXAMPLE,subnet-099e6ca77aEXAMPLE
```

d. 作成されたパブリックサブネットで、IPv6 アドレスの自動割り当てオプションを有効にします。

```
aws ec2 modify-subnet-attribute --region $region_code --
subnet-id subnet-0a1a56c486EXAMPLE --assign-ipv6-address-on-
creation
aws ec2 modify-subnet-attribute --region $region_code --subnet-id
 subnet-099e6ca77aEXAMPLE --assign-ipv6-address-on-creation
```

e. テンプレートによって作成されたサブネットおよびセキュリティグループの名前を、デプロイされた AWS CloudFormation スタックから取得します。これらの名前は、後のステップで使用するために変数に格納しておきます。

```
security_groups=$(aws cloudformation describe-stacks --region $region_code
 --stack-name $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SecurityGroups`].OutputValue' --
output text)

public_subnets=$(aws cloudformation describe-stacks --region $region_code --
stack-name $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SubnetsPublic`].OutputValue' --
output text)
```

```
private_subnets=$(aws cloudformation describe-stacks --region $region_code
 --stack-name $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SubnetsPrivate`].OutputValue' --
output text)

subnets=${public_subnets},${private_subnets}
```

3. クラスター IAM ロールを作成し、必要な Amazon EKS IAM マネージドポリシーをそれにアタッチします。Amazon EKS によって管理される Kubernetes クラスターは、サービスで使用するリソースを管理するために、ユーザーに代わって他の AWS サービスを呼び出します。

   a. 次のコマンドを実行して、eks-cluster-role-trust-policy.json ファイルを作成します。

   ```
   cat >eks-cluster-role-trust-policy.json <<EOF
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
           "Service": "eks.amazonaws.com"
         },
         "Action": "sts:AssumeRole"
       }
     ]
   }
   EOF
   ```

   b. 次のコマンドを実行して、ロール名の変数を設定します。*myAmazonEKSClusterRole* は、好みの任意の名前で置き換えることができます。

   ```
   export cluster_role_name=myAmazonEKSClusterRole
   ```

   c. ロールを作成します。

   ```
   aws iam create-role --role-name $cluster_role_name --assume-role-policy-
   document file://"eks-cluster-role-trust-policy.json"
   ```

   d. IAM ロールの ARN を取得し、後のステップで使用するために変数に格納します。

```
cluster_iam_role=$(aws iam get-role --role-name $cluster_role_name --
query="Role.Arn" --output text)
```

e. このロールに、必要な Amazon EKS 管理の IAM ポリシーをアタッチします。

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSClusterPolicy --role-name $cluster_role_name
```

4. クラスターを作成する

```
aws eks create-cluster --region $region_code --name $cluster_name --kubernetes-
version 1.XX \
   --role-arn $cluster_iam_role --resources-vpc-config subnetIds=
$subnets,securityGroupIds=$security_groups \
   --kubernetes-network-config ipFamily=ipv6
```

-

> **Note**
>
> リクエストで指定したアベイラビリティーゾーンのいずれかに、Amazon EKS クラスターの作成に十分な容量がない場合には、エラーが表示されることがあります。このエラー出力には、新しいクラスターをサポートできるアベイラビリティーゾーンが表示されます。アカウント向けにサポートされているアベイラビリティーゾーンにある 2 つ以上のサブネットを使用して、クラスターを作成します。詳細については、「容量不足」を参照してください。

クラスターが作成されるまでに数分かかります。以下のコマンドを実行します。コマンドの出力が「ACTIVE」になるまで、次のステップに進まないでください。

```
aws eks describe-cluster --region $region_code --name $cluster_name --query
 cluster.status
```

5. クラスターの kubeconfig ファイルを作成または更新し、そのクラスターと通信できるようにします。

```
aws eks update-kubeconfig --region $region_code --name $cluster_name
```

デフォルトでは、config ファイルが ~/.kube に作成されるか、config ファイルが既に ~/.kube に存在する場合には、その中に新しいクラスター設定が追加されます。

6. ノードの IAM ロールを作成します。

   a. 次のコマンドを実行して、vpc-cni-ipv6-policy.json ファイルを作成します。

```
cat >vpc-cni-ipv6-policy <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AssignIpv6Addresses",
                "ec2:DescribeInstances",
                "ec2:DescribeTags",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeInstanceTypes"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*"
            ]
        }
    ]
}
EOF
```

   b. IAM ポリシーを作成します。

```
aws iam create-policy --policy-name AmazonEKS_CNI_IPv6_Policy --policy-
document file://vpc-cni-ipv6-policy.json
```

   c. 次のコマンドを実行して、node-role-trust-relationship.json ファイルを作成します。

```
cat >node-role-trust-relationship.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

d. 次のコマンドを実行して、ロール名の変数を設定します。*AmazonEKSNodeRole* は、好みの任意の名前で置き換えることができます。

```
export node_role_name=AmazonEKSNodeRole
```

e. IAM ロールを作成します。

```
aws iam create-role --role-name $node_role_name --assume-role-policy-
document file://"node-role-trust-relationship.json"
```

f. IAM ロールに IAM ポリシーをアタッチします。

```
aws iam attach-role-policy --policy-arn arn:aws:iam::
$account_id:policy/AmazonEKS_CNI_IPv6_Policy \
    --role-name $node_role_name
```

> **Important**
>
> 簡単にするため、このチュートリアルでは、ポリシーが以下の IAM ロールにアタッチされています。ただし、本番向けクラスターの場合は、異なる IAM ロールにポリシーをアタッチすることをお勧めします。詳細については、「サービスアカウントの IAM ロールを使用する Amazon VPC CNI plugin for Kubernetes の設定」を参照してください。

g. IAM ロールに、2 つの必須なマネージド IAM ポリシーをアタッチします。

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSWorkerNodePolicy \
  --role-name $node_role_name
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEC2ContainerRegistryReadOnly \
  --role-name $node_role_name
```

h. IAM ロールの ARN を取得し、後のステップで使用するために変数に格納します。

```
node_iam_role=$(aws iam get-role --role-name $node_role_name --
query="Role.Arn" --output text)
```

7. マネージド型ノードグループを作成する

a. 前のステップで作成したサブネットの ID を表示します。

```
echo $subnets
```

出力例は次のとおりです。

```
subnet-0a1a56c486EXAMPLE,subnet-099e6ca77aEXAMPLE,subnet-
0377963d69EXAMPLE,subnet-0c05f819d5EXAMPLE
```

b. ノードグループを作成します。*0a1a56c486EXAMPLE*、*099e6ca77aEXAMPLE*、*0377963d69EXAMPLE*、および *0c05f819d5EXAMPLE* を、前のステップの出力で返された値に置き換えます。次のコマンドでは、前の出力の中でサブネット ID の間にあるカンマを必ず削除して使用します。*t3.medium* は、任意の AWS Nitro System インスタンスタイプ に置き換えることができます。

```
aws eks create-nodegroup --region $region_code --cluster-name $cluster_name
 --nodegroup-name $nodegroup_name \
    --subnets subnet-0a1a56c486EXAMPLE subnet-099e6ca77aEXAMPLE
 subnet-0377963d69EXAMPLE subnet-0c05f819d5EXAMPLE \
    --instance-types t3.medium --node-role $node_iam_role
```

ノードグループの作成には数分かかります。以下のコマンドを実行します。出力が「ACTIVE」を返すまで、次のステップに進まないでください。

```
aws eks describe-nodegroup --region $region_code --cluster-name
 $cluster_name --nodegroup-name $nodegroup_name \
    --query nodegroup.status --output text
```

8. デフォルトの Pods に IPv6 アドレスが割り当てられていることを、IP 列で確認します。

```
kubectl get pods -n kube-system -o wide
```

出力例は次のとおりです。

```
NAME                       READY   STATUS    RESTARTS   AGE     IP
                             NODE
 NOMINATED NODE   READINESS GATES
aws-node-rslts             1/1     Running   1          5m36s
  2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
aws-node-t74jh             1/1     Running   0          5m32s
  2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-cw7w2   1/1     Running   0          56m
  2600:1f13:b66:8203:34e5::                ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-tx6n8   1/1     Running   0          56m
  2600:1f13:b66:8203:34e5::1               ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
kube-proxy-btpbk           1/1     Running   0          5m36s
  2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
kube-proxy-jjk2g           1/1     Running   0          5m33s
  2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
```

9. デフォルトのサービスに IPv6 アドレスが割り当てられていることを、IP 列で確認します。

```
kubectl get services -n kube-system -o wide
```

出力例は次のとおりです。

```
NAME       TYPE        CLUSTER-IP          EXTERNAL-IP   PORT(S)         AGE
 SELECTOR
```

```
kube-dns   ClusterIP   fd30:3087:b6c2::a   <none>        53/UDP,53/TCP   57m
 k8s-app=kube-dns
```

10. (オプション) サンプルアプリケーションをデプロイする か、ロードバランサーアプリケーション、または IPv6 Pods へのネットワークトラフィックのための、AWS Load Balancer Controller とサンプルアプリケーションをデプロイします。
11. このチュートリアルのために作成したクラスターとノードの使用が終了したら、以下のコマンドにより、作成したリソースのクリーンアップを行う必要があります。削除を行う前に、これらのリソースが、このチュートリアルの外部で使用されていないことを確認します。

    a. 前のステップを完了したのとは異なるシェルでこのステップを完了する場合は、前のステップで使用したすべての変数の値を設定し、*example values* を前のステップを完了したときに指定した値に置き換えます。前の手順を完了したのと同じシェルでこのステップを完了する場合は、このまま次のステップに進みます。

    ```
    export region_code=region-code
    export vpc_stack_name=my-eks-ipv6-vpc
    export cluster_name=my-cluster
    export nodegroup_name=my-nodegroup
    export account_id=111122223333
    export node_role_name=AmazonEKSNodeRole
    export cluster_role_name=myAmazonEKSClusterRole
    ```

    b. ノードグループを削除します。

    ```
    aws eks delete-nodegroup --region $region_code --cluster-name $cluster_name
     --nodegroup-name $nodegroup_name
    ```

    削除には数分かかります。以下のコマンドを実行します。何らかの出力が返された場合は、次のステップに進まないでください。

    ```
    aws eks list-nodegroups --region $region_code --cluster-name $cluster_name
     --query nodegroups --output text
    ```

    c. クラスターを削除します。

    ```
    aws eks delete-cluster --region $region_code --name $cluster_name
    ```

    クラスターの削除には数分かかります。次に進む前に、下記のコマンドを使用してクラスターが削除されていることを確認します。

```
aws eks describe-cluster --region $region_code --name $cluster_name
```

下記のような出力が返されるまで、次のステップに進まないでください。

```
An error occurred (ResourceNotFoundException) when calling the
 DescribeCluster operation: No cluster found for name: my-cluster.
```

d. 作成した IAM リソースを削除します。前のステップで使用した名前とは異なる名前を選択した場合は、*AmazonEKS_CNI_IPv6_Policy* を選択した名前に置き換えます。

```
aws iam detach-role-policy --role-name $cluster_role_name --policy-arn
 arn:aws:iam::aws:policy/AmazonEKSClusterPolicy
aws iam detach-role-policy --role-name $node_role_name --policy-arn
 arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
aws iam detach-role-policy --role-name $node_role_name --policy-arn
 arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
aws iam detach-role-policy --role-name $node_role_name --policy-arn
 arn:aws:iam::$account_id:policy/AmazonEKS_CNI_IPv6_Policy
aws iam delete-policy --policy-arn arn:aws:iam::
$account_id:policy/AmazonEKS_CNI_IPv6_Policy
aws iam delete-role --role-name $cluster_role_name
aws iam delete-role --role-name $node_role_name
```

e. VPC を作成した AWS CloudFormation スタックを削除します。

```
aws cloudformation delete-stack --region $region_code --stack-name
 $vpc_stack_name
```

Pods の SNAT

IPv6 ファミリーを使用してクラスターをデプロイした場合、このトピックの情報はクラスターに適用されません。IPv6 アドレスがネットワーク変換されないからです。クラスターでの IPv6 使用の詳細については、「クラスター、Pods、services 用の IPv6 アドレス」を参照してください。

デフォルトでは、クラスター内の各 Pod に、Pod がデプロイされた VPC に関連付けられた Classless Inter-Domain Routing (CIDR) ブロックから、プライベート IPv4 アドレスが割り当てられます。同じ VPC 内の Pods は、これらのプライベート IP アドレスをエンドポイントとして使用して相互に通信します。Pod が VPC に関連付けられている CIDR ブロック外の IPv4 アドレスと通信する場合、(Linux または Windows の両方用の) Amazon VPC CNI プラグインが Pod's IPv4 アドレス

を、デフォルトで `*-`、Pod が実行されているノードのプライマリ Elastic Network Interface のプライマリプライベート `IPv4` アドレスに変換します。

> **Note**
>
> Windows ノードについては、他にも考慮すべき事項があります。デフォルトでは、Windows の VPC CNI プラグインは、同じ VPC 内の宛先へのトラフィックを SNAT から除外するネットワーク設定で定義されます。つまり、内部 VPC 通信では SNAT が無効になっており、Pod に割り当てられた IP アドレスは VPC 内でルーティング可能です。ただし、VPC 外の宛先へのトラフィックでは、送信元 Pod IP がインスタンスの ENI のプライマリ IP アドレスに SNAT されます。Windows のこのデフォルト設定により、ポッドはホストインスタンスと同じ方法で VPC 外部のネットワークにアクセスできます。

この動作によって、次の現象が起こります。

- Pods は、実行しているノードにパブリック IP アドレスまたは Elastic IP アドレスが割り当てられており、パブリックサブネット内にある場合にのみ、インターネットリソースと通信できます。パブリックサブネットに関連付けられているルートテーブルには、インターネットゲートウェイへのルートが含まれています。可能であれば、プライベートサブネットにノードをデプロイすることをお勧めします。
- `1.8.0` よりも古いバージョンのプラグインの場合、VPC ピアリング、トランジット VPC、または AWS Direct Connect を使用してクラスター VPC に接続されているネットワークまたは VPC 内にあるリソースは、セカンダリ Elastic Network Interface の背後にある Pods との通信を開始できません。ただし、Pods はこれらのリソースとの通信を開始し、リソースから応答を受け取ることができます。

ご使用の環境で次のいずれかが当てはまる場合は、次のコマンドを使用してデフォルト設定を変更してください。

- VPC ピアリング、トランジット VPC、または AWS Direct Connect を使用してクラスター VPC に接続されているネットワークまたは VPC 内に、`IPv4` アドレスを使用して Pods との通信を開始する必要があるリソースがあり、プラグインのバージョンが `1.8.0` よりも前です。
- Pods がプライベートサブネットにあり、インターネットへのアウトバウンド通信を行う必要があります。サブネットには、NAT ゲートウェイへのルートがある。

```
kubectl set env daemonset -n kube-system aws-node AWS_VPC_K8S_CNI_EXTERNALSNAT=true
```

> **Note**
>
> AWS_VPC_K8S_CNI_EXTERNALSNAT および AWS_VPC_K8S_CNI_EXCLUDE_SNAT_CIDRS CNI の設定変数は Windows ノードに適用されません。SNAT の無効化は Windows でサポートされていません。IPv4 CIDR のリストを SNAT から除外する場合は、Windows ブートストラップスクリプトで ExcludedSnatCIDRs パラメータを指定して定義できます。このパラメータの使用に関する詳細については、「ブートストラップスクリプトの設定パラメータ」を参照してください。

*Pod's 仕様に hostNetwork=true が含まれている場合 (デフォルトは false)、その IP アドレスは別のアドレスに変換されません。これは、クラスター上で実行されている kube-proxy と Amazon VPC CNI plugin for Kubernetes Pods の場合のデフォルトです。これらの Pods の場合、IP アドレスはノードのプライマリ IP アドレスと同じであるため、Pod's IP アドレスは変換されません。Pod's hostNetwork の設定の詳細については、「Kubernetes API リファレンス」の「PodSpec v1 core」を参照してください。

Kubernetes ネットワークポリシーのクラスターを構成する

デフォルトでは、Kubernetes の IP アドレス、ポート、クラスター内の Pods 間の接続、または Pods と他のネットワークのリソースの接続に制限はありません。Kubernetes ネットワークポリシーを使用して、お客様の Pods 間で送受信されるネットワークトラフィックを制限できます。詳細については、「Kubernetes のドキュメント」の「ネットワークポリシー」を参照してください。

クラスターの Amazon VPC CNI plugin for Kubernetes のバージョンが 1.13 以前の場合、Kubernetes ネットワークポリシーをクラスターに適用するには、サードパーティーソリューションを実装する必要があります。バージョン 1.14 以降のプラグインでは、ネットワークポリシーを実装できるため、サードパーティソリューションを使用する必要はありません。このトピックでは、サードパーティのアドオンを使用せずにクラスターの Kubernetes ネットワークポリシーを使用するようにクラスターを構成する方法を説明します。

Amazon VPC CNI plugin for Kubernetes のネットワークポリシーは、次の設定でサポートされています。

- バージョン 1.25 以降の Amazon EKS クラスター。
- クラスターのバージョン 1.14 以降の Amazon VPC CNI plugin for Kubernetes。

- IPv4 または IPv6 アドレス用に設定されたクラスター。
- Pods 用のセキュリティグループでネットワークポリシーを使用できます。ネットワークポリシーを使用すると、クラスター内の通信をすべて制御できます。Pods 用のセキュリティグループを使用すると、Pod 内のアプリケーションから AWS のサービスへのアクセスを制御できます。
- カスタムネットワークおよびプレフィクス委任でネットワークポリシーを使用できます。

考慮事項

- Amazon VPC CNI plugin for Kubernetes を含むクラスターに Amazon VPC CNI plugin for Kubernetes ネットワークポリシーを適用する場合、Amazon EC2 Linux ノードにのみポリシーを適用できます。Fargate または Windows ノードにはポリシーを適用できません。
- クラスターが現在サードパーティーソリューションを使用して Kubernetes ネットワークポリシーを管理している場合、同じポリシーを Amazon VPC CNI plugin for Kubernetes で使用できます。ただし、同じポリシーを管理しないように、既存のソリューションを削除する必要があります。
- 同じPodに複数のネットワークポリシーを適用できます。同じPodを選択するポリシーが 2 つ以上設定されている場合、すべてのポリシーがPodに適用されます。
- ネットワークポリシーの各 `ingress:` または `egress:` セレクターの各プロトコルにおけるポートの一意の組み合わせの最大数は 24 です。
- どの Kubernetes サービスでも、サービスポートはコンテナポートと同じでなければなりません。名前付きポートを使用している場合は、サービス仕様でも同じ名前を使用してください。
- Pod 起動時のポリシーの適用

  Amazon VPC CNI plugin for Kubernetes はポッドのプロビジョニングと並行して、ポッドのネットワークポリシーを設定します。新しいポッドにすべてのポリシーが設定されるまで、新しいポッドのコンテナはデフォルトの許可ポリシーで起動します。これは標準モードと呼ばれます。デフォルトの許可ポリシーは、すべての ingress トラフィックと egress トラフィックが新しいポッドとの間で許可されることを意味します。

  このデフォルトのネットワークポリシーを変更するには、VPC CNI `aws-node` の `strict` コンテナで環境変数 `NETWORK_POLICY_ENFORCING_MODE` を `DaemonSet` に設定します。

```
env:
  - name: NETWORK_POLICY_ENFORCING_MODE
    value: "strict"
```

変数 NETWORK_POLICY_ENFORCING_MODE を strict に設定すると、VPC CNI を使用するポッドはデフォルトの拒否ポリシーで起動し、その後ポリシーが設定されます。これは Strict モードと呼ばれます。Strict モードでは、ポッドがクラスター内でアクセスする必要があるすべてのエンドポイントにネットワークポリシーが必要です。この要件は、CoreDNS ポッドに適用されることに注意してください。デフォルトの拒否ポリシーは、ホストネットワークを使用するポッドには設定されていません。

- ネットワークポリシー機能では、policyendpoints.networking.k8s.aws と呼ばれる PolicyEndpoint カスタムリソース定義 (CRD) が作成され、必要になります。カスタムリソースの PolicyEndpoint オブジェクトは Amazon EKS によって管理されます。これらのリソースを変更または削除しないでください。
- インスタンスロールの IAM 認証情報を使用するポッドを実行するか、EC2 IMDS に接続するポッドを実行する場合は、EC2 IMDS へのアクセスをブロックするネットワークポリシーがないか慎重に確認してください。EC2 IMDS へのアクセスを許可するネットワークポリシーを追加する必要がある場合があります。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスメタデータとユーザーデータ」を参照してください。

  サービスアカウントの IAM ロールを使用するポッドが、EC2 IMDS にアクセスすることはありません。
- Amazon VPC CNI plugin for Kubernetes は、各ポッドの追加のネットワークインターフェイスにはネットワークポリシーを適用せず、各ポッドのプライマリインターフェイス (eth0) のみにネットワークポリシーを適用します。これは以下のアーキテクチャに影響します。
  - ENABLE_V4_EGRESS 変数が true に設定された IPv6 ポッド。この変数により、IPv4 エグレス機能が IPv6 ポッドをクラスター外のエンドポイントなどの IPv4 エンドポイントに接続できるようになります。IPv4 エグレス機能は、ローカルループバック IPv4 アドレスを持つ追加のネットワークインターフェースを作成することで機能します。
  - Multus などのチェーンネットワークプラグインを使用する場合。これらのプラグインは各ポッドにネットワークインターフェースを追加するため、ネットワークポリシーはチェーンネットワークプラグインには適用されません。
- ネットワークポリシー機能は、デフォルトでノード上のポート 8162 をメトリクスに使用します。また、この機能はヘルスプローブにポート 8163 を使用していました。そのノード上、またはこれらのポートを使用する必要があるポッド内で他のアプリケーションを実行すると、そのアプリは実行できません。VPC CNI バージョン v1.14.1 以降では、これらのポートを次の場所で変更できます。

AWS Management Console

1. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。
2. 左のナビゲーションペインで、[クラスター] を選択し、 Amazon VPC CNI アドオンを設定するクラスターの名前を選択します。
3. [アドオン] タブを選択します。
4. アドオンボックスの右上にあるボックスを選択し、次に [編集] を選択します。
5. [*******の設定] ページで次のことを行います。

   a. [バージョン] ドロップダウンリストで `v1.14.0-eksbuild.3` 以降のバージョンを選択します。

   b. [オプションの構成設定] を展開します。

   c. [設定値] に JSON キー `"enableNetworkPolicy":` と値 `"true"` を入力します。結果のテキストは有効な JSON オブジェクトでなければなりません。このキーと値だけがテキストボックス内のデータである場合は、キーと値を中括弧 {} で囲みます。

      次の例では、ネットワークポリシー機能が有効になっており、ネットワークポリシーログも有効で、そのネットワークポリシーログが Amazon CloudWatch Logs に送信され、メトリクスとヘルスプローブがデフォルトのポート番号に設定されています。

```
{
    "enableNetworkPolicy": "true",
    "nodeAgent": {
        "enablePolicyEventLogs": "true",
        "enableCloudWatchLogs": "true",
        "healthProbeBindAddr": "8163",
        "metricsBindAddr": "8162"
    }
}
```

Helm

`helm` を通して Amazon VPC CNI plugin for Kubernetes をインストールしている場合、設定を更新してポートを変更できます。

- 次に、以下のコマンドを実行して、ポートを変更します。キー nodeAgent.metricsBindAddr または キー nodeAgent.healthProbeBindAddr の値にそれぞれポート番号を設定します。

```
helm upgrade --set nodeAgent.metricsBindAddr=8162 --set
 nodeAgent.healthProbeBindAddr=8163 aws-vpc-cni --namespace kube-system eks/
aws-vpc-cni
```

kubectl

1. エディターで aws-nodeDaemonSet を開きます。

```
kubectl edit daemonset -n kube-system aws-node
```

2. VPC CNI aws-node デーモンセットマニフェストの aws-network-policy-agent コンテナで、以下の args: のコマンド引数でポート番号を置き換えます。

```
    - args:
            - --metrics-bind-addr=:8162
            - --health-probe-bind-addr=:8163
```

前提条件

- 最小クラスターバージョン

既存の Amazon EKS クラスター。デプロイするには、「Amazon EKS の使用開始」を参照してください。クラスターは Kubernetes バージョン 1.25 以降である必要があります。クラスターは、次の表に示す Kubernetes バージョンとプラットフォームバージョンのいずれかを実行している必要があります。一覧にあるバージョンより後の Kubernetes とプラットフォームのバージョンもサポートされることにご注意ください。現在の Kubernetes バージョンを確認するには、次のコマンドの *my-cluster* をクラスターの名前に置き換えて、変更したコマンドを実行します。

```
aws eks describe-cluster
              --name my-cluster --query cluster.version --output
              text
```

Kubernetes バージョン	プラットフォームバージョン
1.27.4	eks.5
1.26.7	eks.6
1.25.12	eks.7

- 最小 VPC CNI バージョン

  クラスターのバージョン 1.14 以降の Amazon VPC CNI plugin for Kubernetes。現在のバージョンは、次のコマンドで確認できます。

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
 cut -d : -f 3
```

  バージョンが 1.14 より前の場合は、Amazon EKS アドオンの更新 を確認してバージョン 1.14 以降にアップグレードしてください。

- 最小 Linux カーネルバージョン

  ノードにはバージョン 5.10 以降の Linux カーネルが必要です。カーネルのバージョンは、uname -r で確認できます。Amazon EKS の最適化された Amazon Linux AMI と Bottlerocket AMI の最新バージョンを使用している場合、既に必要なカーネルバージョンがインストールされています。

  Amazon EKS 最適化高速 Amazon Linux AMI v20231116 バージョン以降には、カーネルバージョン 5.10 があります。

Kubernetes ネットワークポリシーを使用するようにクラスターを設定する

1. BPF ファイルシステムのマウント

> **Note**
>
> クラスターがバージョン 1.27 以降の場合、Amazon EKS 最適化された 1.27 以降の Amazon Linux AMI と Bottlerocket AMI にはすべてこの機能が既に実装されているため、このステップをスキップできます。
> その他のすべてのクラスターバージョンでは、Amazon EKS 最適化された Amazon Linux をバージョン v20230703 以降にアップグレードした場合、または Bottlerocket

AMI をバージョン `v1.0.2` にアップグレードした場合に。このステップをスキップできます。

a. バークレーパケットフィルタ (BPF) ファイルシステムを各ノードにマウントします。

```
sudo mount -t bpf bpffs /sys/fs/bpf
```

b. 次に、Amazon EC2 Auto Scaling グループの起動テンプレートのユーザーデータに同じコマンドを追加します。

2. VPC CNI でネットワークポリシーを有効にします。

a. クラスターにインストールされているアドオンのタイプを確認します。クラスターを作成するために使用したツールによっては、現在クラスターに Amazon EKS アドオンタイプがインストールされていない場合があります。*my-cluster* の部分は、自分のクラスター名に置き換えます。

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 addon.addonVersion --output text
```

バージョン番号が返された場合、Amazon EKS タイプのアドオンがクラスターにインストールされているため、このステップの残りのステップを完了する必要はありません。エラーが返された場合、クラスターに Amazon EKS タイプのアドオンがインストールされていません。

b. • Amazon EKS アドオン

AWS Management Console

a. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。

b. 左のナビゲーションペインで、[クラスター] を選択し、 Amazon VPC CNI アドオンを設定するクラスターの名前を選択します。

c. [アドオン] タブを選択します。

d. アドオンボックスの右上にあるボックスを選択し、次に [編集] を選択します。

e. [*#######*の設定] ページで次のことを行います。

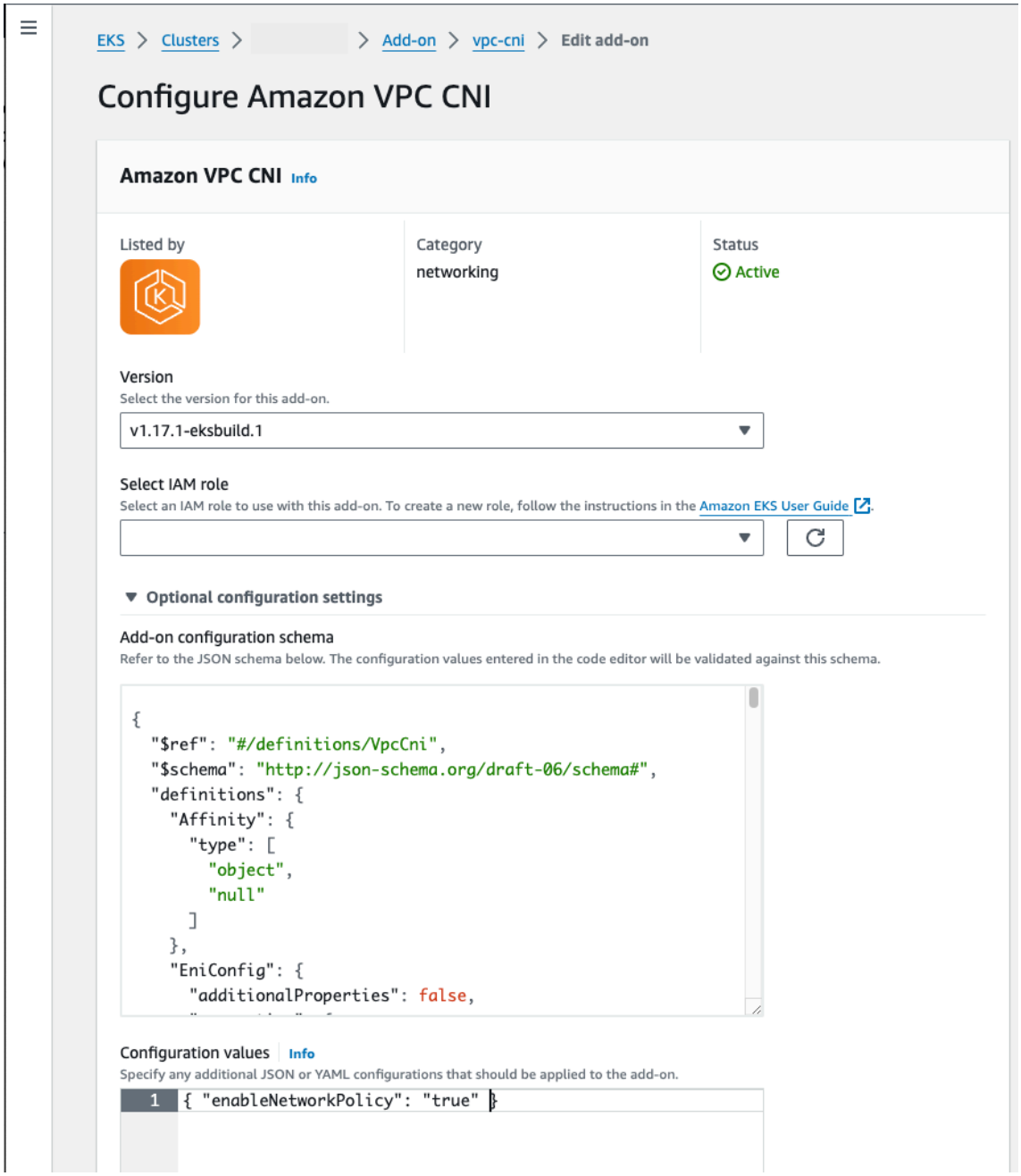
i. [バージョン] ドロップダウンリストで v1.14.0-eksbuild.3 以降のバージョンを選択します。

ii. [オプションの構成設定] を展開します。

iii. [設定値] に JSON キー "enableNetworkPolicy": と値 "true" を入力します。結果のテキストは有効な JSON オブジェクトでなければなりません。このキーと値だけがテキストボックス内のデータである場合は、キーと値を中括弧 {} で囲みます。次の例は、ネットワークポリシーが有効になっていることを示しています。

```
{ "enableNetworkPolicy": "true" }
```

次のスクリーンショットは、このシナリオの例を示しています。



AWS CLI

- 次の AWS CLI コマンドを実行します。`my-cluster` をクラスターの名前に置き換え、IAM ロール ARN を使用するロールに置き換えます。

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni
 --addon-version v1.14.0-eksbuild.3 \
```

```
    --service-account-role-arn arn:aws:iam::123456789012:role/
AmazonEKSVPCCNIRole \
    --resolve-conflicts PRESERVE --configuration-values
 '{"enableNetworkPolicy": "true"}'
```

- セルフマネージド型アドオン

  Helm

  helm を通して Amazon VPC CNI plugin for Kubernetes をインストールしている場合、設定を更新してネットワークポリシーを有効にできます。

  - 次のコマンドを実行してネットワークポリシーを有効にします。

    ```
    helm upgrade --set enableNetworkPolicy=true aws-vpc-cni --namespace
     kube-system eks/aws-vpc-cni
    ```

  kubectl

  a. エディターで amazon-vpc-cniConfigMap を開きます。

     ```
     kubectl edit configmap -n kube-system amazon-vpc-cni -o yaml
     ```

  b. 次の行を ConfigMap のdataに追加します。

     ```
     enable-network-policy-controller: "true"
     ```

     行を追加すると、ConfigMap は次の例のようになるはずです。

     ```
     apiVersion: v1
      kind: ConfigMap
      metadata:
       name: amazon-vpc-cni
       namespace: kube-system
      data:
       enable-network-policy-controller: "true"
     ```

  c. エディターで aws-nodeDaemonSet を開きます。

```
kubectl edit daemonset -n kube-system aws-node
```

d. VPC CNI aws-node デーモンセットマニフェストの aws-network-policy-agent コンテナで、args: のコマンド引数 --enable-network-policy=false の false を true に置き換えます。

```
    - args:
       - --enable-network-policy=true
```

3. aws-node ポッドがクラスター上で実行されていることを確認してください。

```
kubectl get pods -n kube-system | grep 'aws-node\|amazon'
```

出力例は次のとおりです。

```
aws-node-gmqp7                                          2/2     Running   1 (24h
 ago)   24h
aws-node-prnsh                                          2/2     Running   1 (24h
 ago)   24h
```

ネットワークポリシーが有効になっている場合、aws-node ポッドには 2 つのコンテナがあります。以前のバージョンでは、ネットワークポリシーが無効になっている場合、aws-node ポッドにコンテナは 1 つしか存在しません。

これで、Kubernetes ネットワークポリシーをクラスターにデプロイできます。詳細については、「Kubernetes ネットワークポリシー」を参照してください。

ネットワークポリシーの Stars デモ

このデモでは、Amazon EKS クラスターにフロントエンド、バックエンド、およびクライアントサービスを作成します。また、このデモでは、各サービス間で利用可能な入力および出力のパスを示す管理グラフィカルユーザーインターフェースが作成されます。実稼働ワークロードを実行しないクラスターでデモを完了することをお勧めします。

ネットワークポリシーを作成する前に、サービスはすべて、双方向に通信できます。ネットワークポリシーを適用すると、クライアントはフロントエンドサービスとのみ通信でき、バックエンドはフロントエンドからのみトラフィックを受け付けます。

Stars Policy デモを実行するには

1. フロントエンド、バックエンド、クライアント、および管理ユーザーインターフェイスサービスを適用します。

```
kubectl apply -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/namespace.yaml
kubectl apply -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/management-ui.yaml
kubectl apply -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/backend.yaml
kubectl apply -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/frontend.yaml
kubectl apply -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/client.yaml
```

2. クラスター上のすべての Pods を表示します。

```
kubectl get pods -A
```

出力例は次のとおりです。

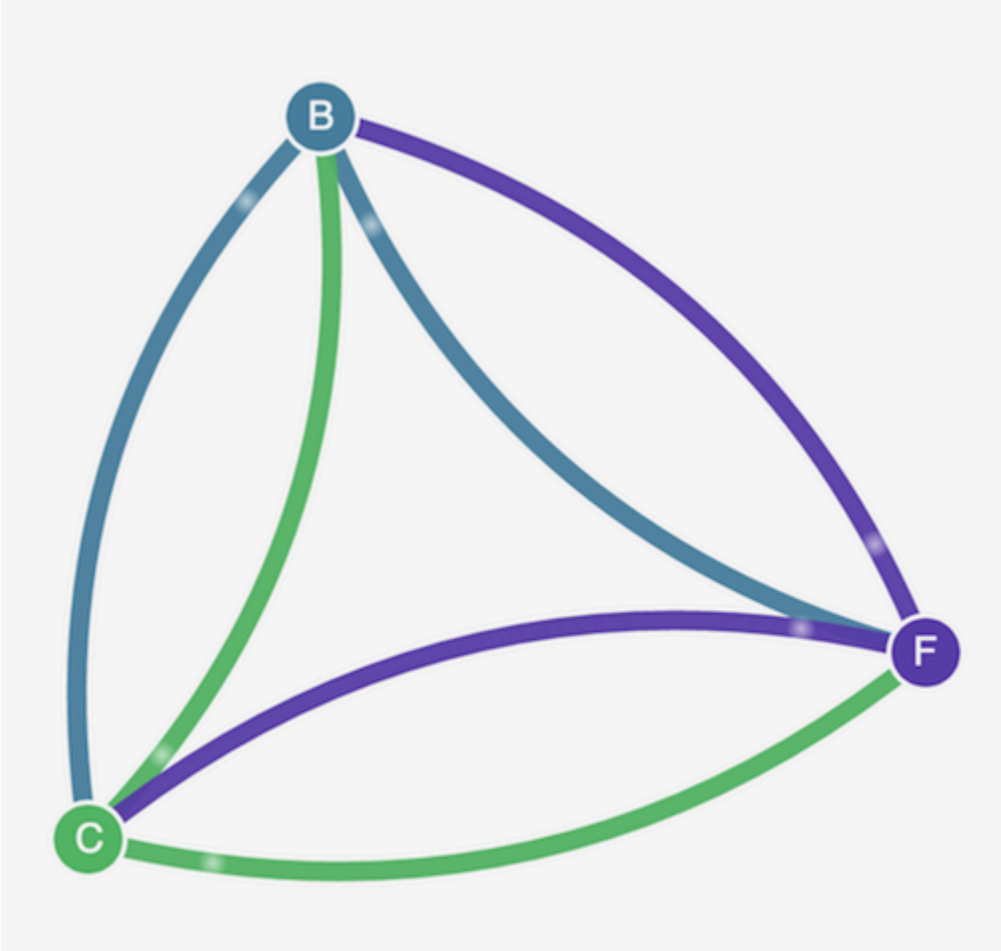
出力では、次の出力に示す名前空間にポッドが表示されます。READY 列内のポッドの *NAMES* とポッド数は、次の出力とは異なります。似たような名前のポッドが見えて、それらがすべて STATUS 列内に Running を持つまで続けないでください。

```
NAMESPACE         NAME                                       READY   STATUS
 RESTARTS   AGE
[...]
client            client-xlffc                               1/1     Running   0
       5m19s
[...]
management-ui     management-ui-qrb2g                        1/1     Running   0
       5m24s
stars             backend-sz87q                              1/1     Running   0
       5m23s
stars             frontend-cscnf                             1/1     Running   0
       5m21s
[...]
```

3. 管理ユーザーインターフェイスに接続するには、クラスターで実行されているサービスの EXTERNAL-IP に接続します。

```
kubectl get service/management-ui -n management-ui
```

4. ブラウザーを開いて、前のステップの場所に移動します。管理ユーザーインターフェイスが表示されます。[C] ノードはクライアントサービス、[F] ノードはフロントエンドサービス、[B] ノードはバックエンドサービスを表します。各ノードには、太字の色付きの行で示されるように、他のすべてのノードへの完全な通信アクセスが含まれます。



5. 相互にサービスを分離するには、次のネットワークポリシーを `stars` および `client` の名前空間の両方に適用します。

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: default-deny
spec:
  podSelector:
    matchLabels: {}
```

次のコマンドを使用して、ポリシーを両方の名前空間に適用できます。

```
kubectl apply -n stars -f https://eksworkshop.com/beginner/120_network-policies/
calico/stars_policy_demo/apply_network_policies.files/default-deny.yaml
kubectl apply -n client -f https://eksworkshop.com/beginner/120_network-policies/
calico/stars_policy_demo/apply_network_policies.files/default-deny.yaml
```

6. ブラウザを更新します。管理ユーザーインターフェイスはノードに到達できなくなるため、ユーザーインターフェイスに表示されないことがわかります。
7. 管理ユーザーインターフェイスがサービスにアクセスできるように、次の別のネットワークポリシーを適用します。このポリシーを適用して UI を許可します。

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: stars
  name: allow-ui
spec:
  podSelector:
    matchLabels: {}
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              role: management-ui
```

このポリシーを適用してクライアントを許可します。

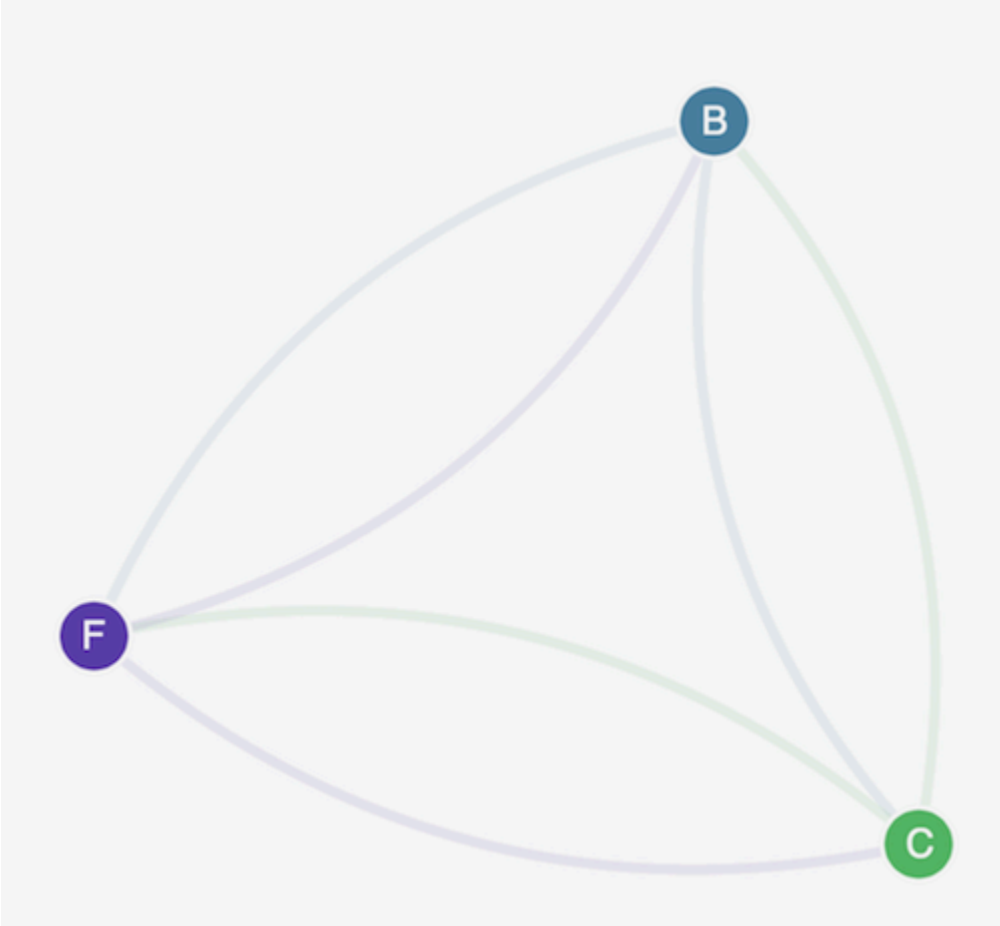
```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: client
  name: allow-ui
spec:
  podSelector:
    matchLabels: {}
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
```

```
              role: management-ui
```

次のコマンドを使用して両方のポリシーを適用できます。

```
kubectl apply -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/apply_network_policies.files/allow-ui.yaml
kubectl apply -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/apply_network_policies.files/allow-ui-client.yaml
```

8. ブラウザを更新します。管理ユーザーインターフェイスは再びノードにアクセスできますが、ノードは相互に通信できないことがわかります。

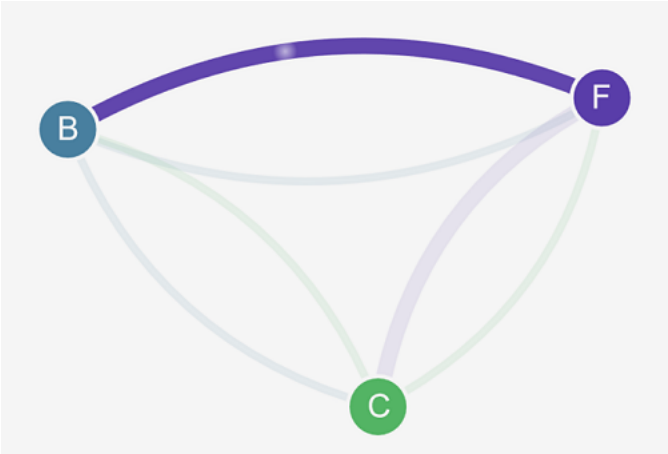


9. フロントエンドサービスからバックエンドサービスへのトラフィックを許可するには、次のネットワークポリシーを適用します。

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: stars
```

```
  name: backend-policy
spec:
  podSelector:
    matchLabels:
      role: backend
  ingress:
    - from:
        - podSelector:
            matchLabels:
              role: frontend
      ports:
        - protocol: TCP
          port: 6379
```

10. ブラウザを更新します。フロントエンドがバックエンドと通信できることがわかります。

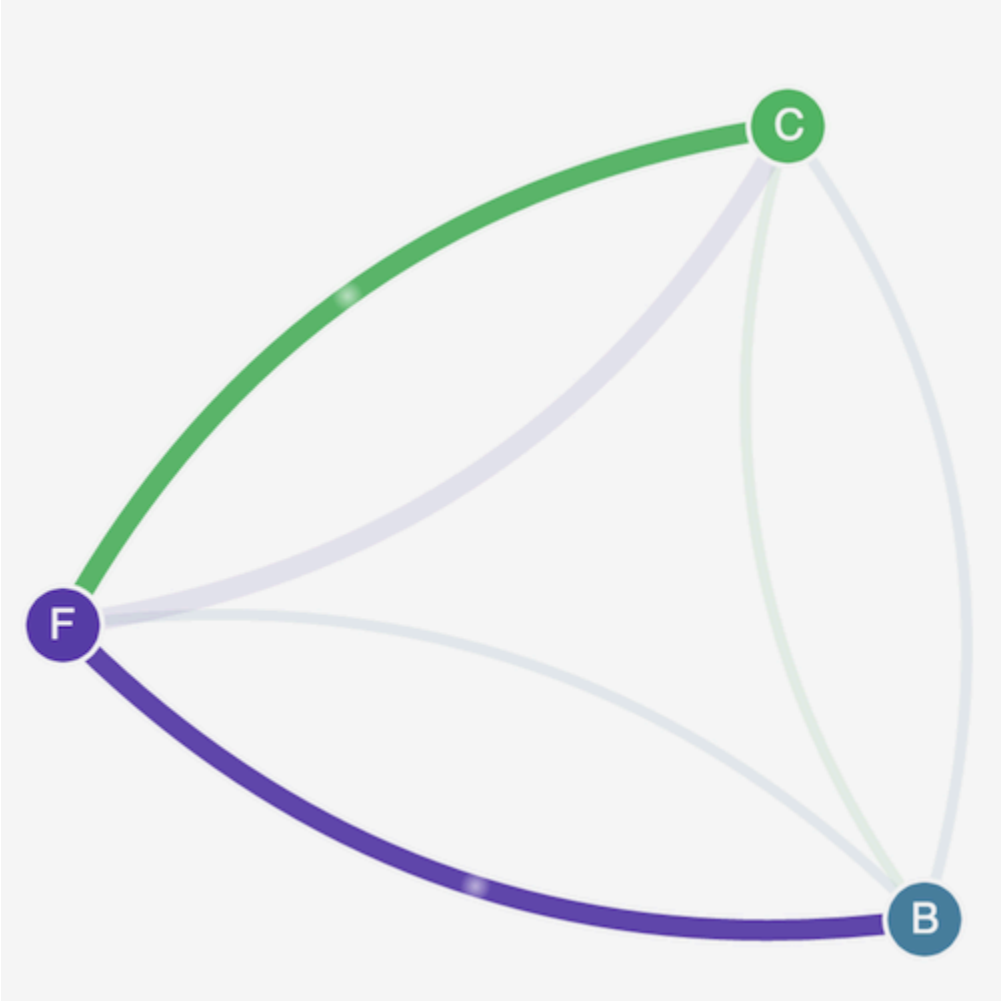


11. クライアントからフロントエンドサービスへのトラフィックを許可するには、次のネットワークポリシーを適用します。

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: stars
  name: frontend-policy
spec:
  podSelector:
    matchLabels:
      role: frontend
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
```

```
            role: client
    ports:
      - protocol: TCP
        port: 80
```

12. ブラウザを更新します。クライアントがフロントエンドサービスと通信できることがわかります。フロントエンドサービスは、バックエンドサービスと引き続き通信できます。



13. (オプション) デモが完了したら、そのリソースを削除することができます。

```
kubectl delete -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/client.yaml
kubectl delete -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/frontend.yaml
kubectl delete -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/backend.yaml
kubectl delete -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/management-ui.yaml
```

```
kubectl delete -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/namespace.yaml
```

リソースを削除した後でも、ノード上にネットワークポリシーエンドポイントが残っている場合があり、それがクラスター内のネットワークを予期しない方法で妨げることがあります。これらのルールを削除する唯一の確実な方法は、ノードを再起動するか、すべてのノードを終了して、リサイクルすることです。すべてのノードを終了するには、Auto Scaling グループの必要数を 0 に設定してから目的数にバックアップするか、単にノードを終了します。

ネットワークポリシーのトラブルシューティング

ネットワークポリシーのログ を読み、eBPF SDK のツールを実行することにより、ネットワークポリシーを使用するネットワーク接続をトラブルシューティングおよび調査できます。

ネットワークポリシーのログ

ネットワークポリシーによって接続が許可されているか拒否されているかは、フローログに記録されています。各ノードのネットワークポリシーログには、ネットワークポリシーが設定されているすべてのポッドのフローログが含まれます。ネットワークポリシーログは `/var/log/aws-routed-eni/network-policy-agent.log` に保存されます。次の例は `network-policy-agent.log` ファイルからのものです。

```
{"level":"info","timestamp":"2023-05-30T16:05:32.573Z","logger":"ebpf-
client","msg":"Flow Info: ","Src
IP":"192.168.87.155","Src Port":38971,"Dest IP":"64.6.160","Dest
Port":53,"Proto":"UDP","Verdict":"ACCEPT"}
```

ネットワークポリシーログはデフォルトで無効になっています。ネットワークポリシーログを有効にするには、次の手順に従います。

> **Note**
>
> ネットワークポリシーログには、VPC CNI `aws-node` デーモンセットマニフェストの `aws-network-policy-agent` コンテナ用に 1 つの vCPU を追加する必要があります。

Amazon EKS アドオン

AWS Management Console

1. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。
2. 左のナビゲーションペインで、[クラスター] を選択し、 Amazon VPC CNI アドオンを設定するクラスターの名前を選択します。
3. [アドオン] タブを選択します。
4. アドオンボックスの右上にあるボックスを選択し、次に [編集] を選択します。
5. [#######の設定] ページで次のことを行います。

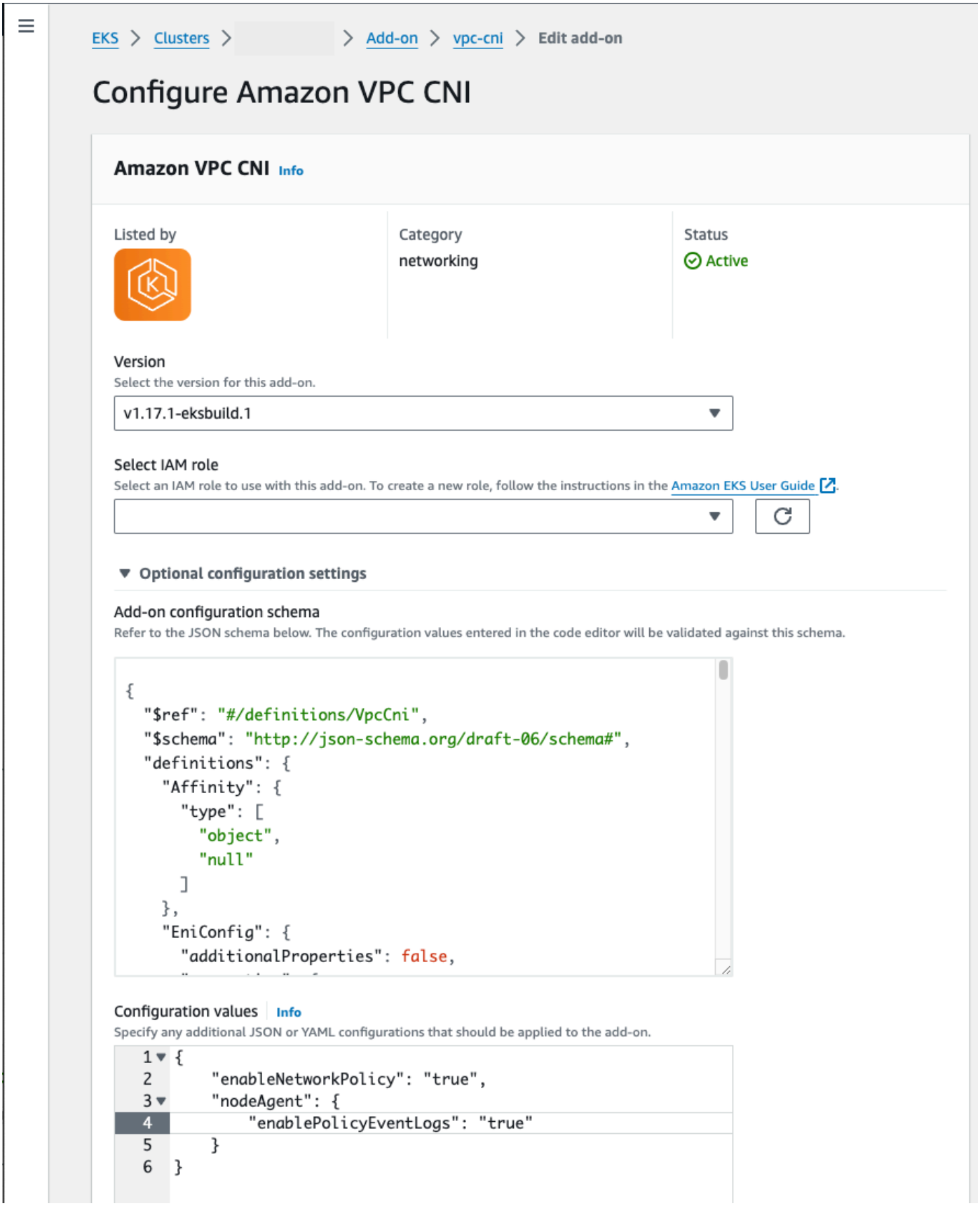
   a. [バージョン] ドロップダウンリストで `v1.14.0-eksbuild.3` 以降のバージョンを選択します。

   b. [オプションの構成設定] を展開します。

   c. 最上位の JSON キー `"nodeAgent":` を入力します。値は [設定値] に キー `"enablePolicyEventLogs":` と `"true"` の値を持つオブジェクトです。結果のテキストは有効な JSON オブジェクトでなければなりません。次の例は、ネットワークポリシーとネットワークポリシーログが有効になっており、ネットワークポリシーログが CloudWatch Logs に送信されていることを示しています。

   ```
   {
       "enableNetworkPolicy": "true",
       "nodeAgent": {
           "enablePolicyEventLogs": "true"
       }
   }
   ```

次のスクリーンショットは、このシナリオの例を示しています。


EKS > Clusters > > Add-on > vpc-cni > Edit add-on
Configure Amazon VPC CNI
Amazon VPC CNI Info
Listed by
Category
networking
Status
Active
Version
Select the version for this add-on.
v1.17.1-eksbuild.1
Select IAM role
Select an IAM role to use with this add-on. To create a new role, follow the instructions in the Amazon EKS User Guide.
Optional configuration settings
Add-on configuration schema
Refer to the JSON schema below. The configuration values entered in the code editor will be validated against this schema.
{
"$ref": "#/definitions/VpcCni",
"$schema": "http://json-schema.org/draft-06/schema#",
"definitions": {
"Affinity": {
"type": [
"object",
"null"
]
},
"EniConfig": {
"additionalProperties": false,
Configuration values Info
Specify any additional JSON or YAML configurations that should be applied to the add-on.
1 {
2 "enableNetworkPolicy": "true",
3 "nodeAgent": {
4 "enablePolicyEventLogs": "true"
5 }
6 }

AWS CLI

- 次の AWS CLI コマンドを実行します。`my-cluster` をクラスターの名前に置き換え、IAM ロール ARN を使用するロールに置き換えます。

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version v1.14.0-eksbuild.3 \
    --service-account-role-arn arn:aws:iam::123456789012:role/
AmazonEKSVPCCNIRole \
    --resolve-conflicts PRESERVE --configuration-values '{"nodeAgent":
 {"enablePolicyEventLogs": "true"}}'
```

セルフマネージド型アドオン

Helm

`helm` を通して Amazon VPC CNI plugin for Kubernetes をインストールしている場合、設定を更新してネットワークポリシーを記述できます。

- 次のコマンドを実行してネットワークポリシーを有効にします。

```
helm upgrade --set nodeAgent.enablePolicyEventLogs=true aws-vpc-cni --namespace
 kube-system eks/aws-vpc-cni
```

kubectl

`kubectl` を通して Amazon VPC CNI plugin for Kubernetes をインストールしている場合、設定を更新してネットワークポリシーを記述できます。

1. エディターで `aws-nodeDaemonSet` を開きます。

```
kubectl edit daemonset -n kube-system aws-node
```

2. VPC CNI `aws-node` デーモンセットマニフェストの `aws-network-policy-agent` コンテナで、`args:` のコマンド引数 `--enable-policy-event-logs=false` の `false` を `true` に置き換えます。

```
    - args:
```

```
        - --enable-policy-event-logs=true
```

ネットワークポリシーログを Amazon CloudWatch Logs に送信する

Amazon CloudWatch Logs などのサービスを使用して、ネットワークポリシーログをモニタリングできます。次の方法を使用して、ネットワークポリシーログを CloudWatch Logs に送信できます。

EKS クラスターの場合、ポリシーログは /aws/eks/*cluster-name*/cluster/ に配置され、セルフマネージド型 K8S クラスターの場合、ログは /aws/k8s-cluster/cluster/ に配置されます。

ネットワークポリシーログを Amazon VPC CNI plugin for Kubernetes で送信する

ネットワークポリシーを有効にすると、2 つ目のコンテナがノードエージェントの aws-node ポッドに追加されます。このノードエージェントは、ネットワークポリシーログを CloudWatch Logs に送信できます。

> **Note**
>
> ノードエージェントはネットワークポリシーログのみを送信します。VPC CNI によって作成された他のログは含まれません。

前提条件

- VPC CNI に使用している IAM ロールに、次の権限をスタンザまたは個別のポリシーとして追加します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
```

```
        }
    ]
}
```

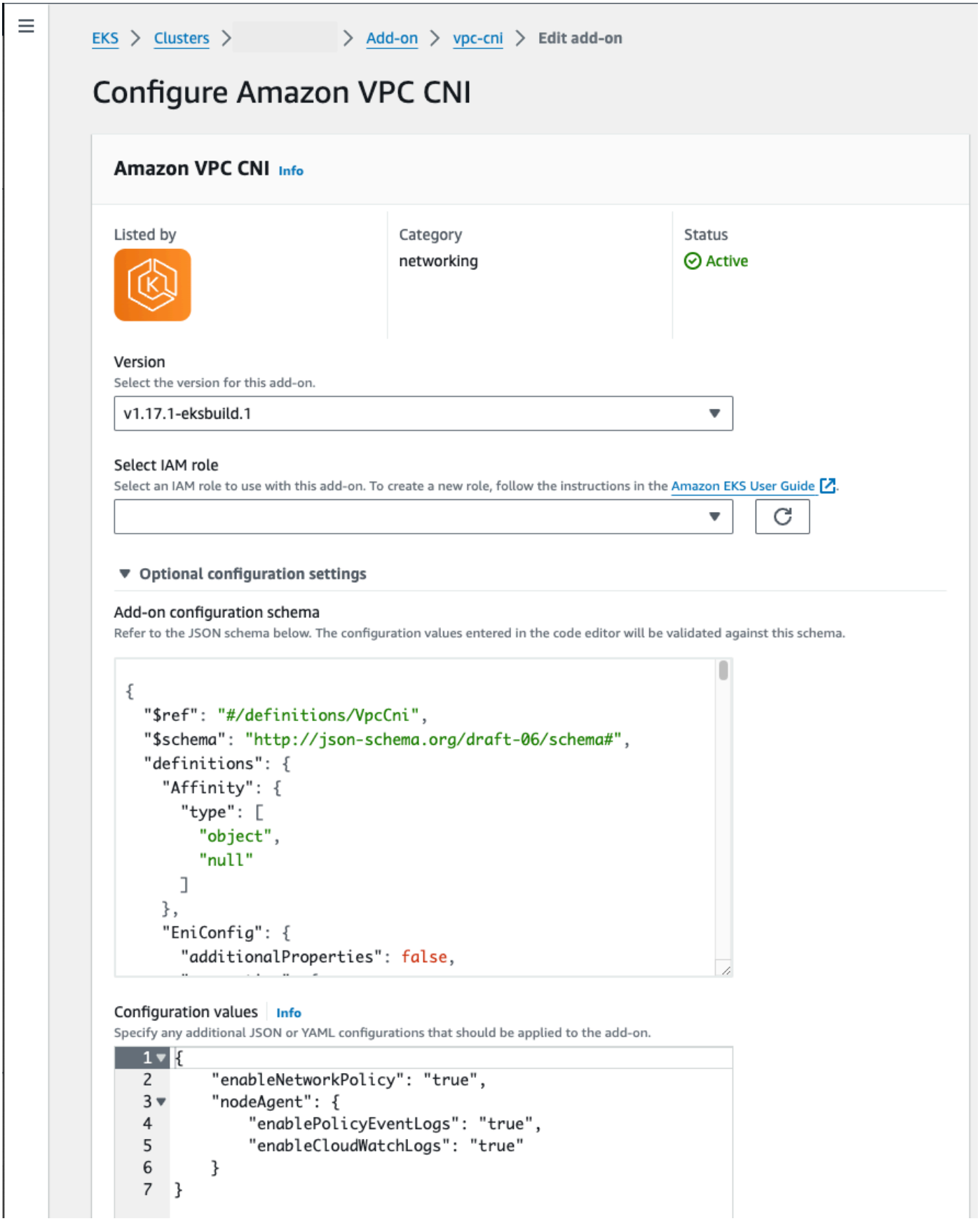
Amazon EKS アドオン

AWS Management Console

1. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。
2. 左のナビゲーションペインで、[クラスター] を選択し、 Amazon VPC CNI アドオンを設定するクラスターの名前を選択します。
3. [アドオン] タブを選択します。
4. アドオンボックスの右上にあるボックスを選択し、次に [編集] を選択します。
5. [#######の設定] ページで次のことを行います。
   a. [バージョン] ドロップダウンリストで v1.14.0-eksbuild.3 以降のバージョンを選択します。
   b. [オプションの構成設定] を展開します。
   c. 最上位の JSON キー "nodeAgent": を入力します。値は [設定値] にキー "enableCloudWatchLogs": と "true" の値を持つオブジェクトです。結果のテキストは有効な JSON オブジェクトでなければなりません。次の例は、ネットワークポリシーとネットワークポリシーログが有効になっており、ログが CloudWatch Logs に送信されていることを示しています。

```
{
    "enableNetworkPolicy": "true",
    "nodeAgent": {
        "enablePolicyEventLogs": "true",
        "enableCloudWatchLogs": "true",
    }
}
```

次のスクリーンショットは、このシナリオの例を示しています。


EKS > Clusters > > Add-on > vpc-cni > Edit add-on
Configure Amazon VPC CNI
Amazon VPC CNI Info
Listed by
Category
networking
Status
Active
Version
Select the version for this add-on.
v1.17.1-eksbuild.1
Select IAM role
Select an IAM role to use with this add-on. To create a new role, follow the instructions in the Amazon EKS User Guide.
Optional configuration settings
Add-on configuration schema
Refer to the JSON schema below. The configuration values entered in the code editor will be validated against this schema.
{
"$ref": "#/definitions/VpcCni",
"$schema": "http://json-schema.org/draft-06/schema#",
"definitions": {
"Affinity": {
"type": [
"object",
"null"
]
},
"EniConfig": {
"additionalProperties": false,
Configuration values Info
Specify any additional JSON or YAML configurations that should be applied to the add-on.
1 {
2 "enableNetworkPolicy": "true",
3 "nodeAgent": {
4 "enablePolicyEventLogs": "true",
5 "enableCloudWatchLogs": "true"
6 }
7 }

AWS CLI

- 次の AWS CLI コマンドを実行します。`my-cluster` をクラスターの名前に置き換え、IAM ロール ARN を使用するロールに置き換えます。

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version v1.14.0-eksbuild.3 \
    --service-account-role-arn arn:aws:iam::123456789012:role/
AmazonEKSVPCCNIRole \
    --resolve-conflicts PRESERVE --configuration-values '{"nodeAgent":
 {"enablePolicyEventLogs": "true", "enableCloudWatchLogs": "true"}}'
```

セルフマネージド型アドオン

Helm

`helm` を通して Amazon VPC CNI plugin for Kubernetes をインストールしている場合、設定を更新してネットワークポリシーログを CloudWatch Logs に送信できます。

- 次のコマンドを実行してネットワークポリシーログを有効にし、CloudWatch Logs に送信します。

```
helm upgrade --set nodeAgent.enablePolicyEventLogs=true --set
 nodeAgent.enableCloudWatchLogs=true aws-vpc-cni --namespace kube-system eks/
aws-vpc-cni
```

kubectl

1. エディターで `aws-nodeDaemonSet` を開きます。

```
kubectl edit daemonset -n kube-system aws-node
```

2. VPC CNI `aws-node` デーモンセットマニフェストの `aws-network-policy-agent` コンテナで、`args:` の 2 つのコマンド引数 `--enable-policy-event-logs=false` と `--enable-cloudwatch-logs=false` の `false` を `true` に置き換えます。

```
    - args:
       - --enable-policy-event-logs=true
```

```
        - --enable-cloudwatch-logs=true
```

ネットワークポリシーログを Fluent Bit デーモンセットで送信する

ノードからログを送信するためにデーモンセットの Fluent Bit を使用している場合、ネットワークポリシーのネットワークポリシーログを含めるように設定を追加できます。次の設定例を使用できます。

```
    [INPUT]
        Name              tail
        Tag               eksnp.*
        Path              /var/log/aws-routed-eni/network-policy-agent*.log
        Parser            json
        DB                /var/log/aws-routed-eni/flb_npagent.db
        Mem_Buf_Limit     5MB
        Skip_Long_Lines   On
        Refresh_Interval  10
```

eBPF SDK を含む

Amazon VPC CNI plugin for Kubernetes は、ノードに eBPF SDK ツールのコレクションをインストールします。eBPF SDK ツールを使用して、ネットワークポリシーの問題を特定できます。例えば、次のコマンドはノードで実行されているプログラムを一覧表示します。

```
sudo /opt/cni/bin/aws-eks-na-cli ebpf progs
```

このコマンドを実行するために、任意の方法を使用してノードに接続できます。

Kubernetes ネットワークポリシー

Kubernetes ネットワークポリシーを実装するには、Kubernetes `NetworkPolicy` オブジェクトを作成して、クラスターにデプロイします。`NetworkPolicy` オブジェクトは名前空間に限定されます。ラベルセレクター、名前空間、および IP アドレス範囲に基づいて、Pods 間のトラフィックを許可または拒否するポリシーを実装します。`NetworkPolicy` オブジェクトの作成の詳細については、Kubernetes ドキュメントの「ネットワークポリシー」を参照してください。

Kubernetes `NetworkPolicy` オブジェクトの実施は、Extended Berkeley Packet Filter (eBPF) を使用して実装されます。`iptables` をベースにした実装に関連して、CPU 使用率の低下やシーケンシャルルックアップの回避など、低レイテンシーを実現し、パフォーマンス特性を発揮します。さらに、eBPF プローブは、複雑なカーネルレベルの問題のデバッグやオブザーバビリティの向上に役立

つ、コンテキスト豊富なデータへのアクセスを提供します。Amazon EKS は、プローブを利用して各ノードのポリシー結果をログに記録し、そのデータを外部のログコレクターにエクスポートしてデバッグに役立てる eBPF ベースのエクスポーターをサポートしています。詳細については、eBPF ドキュメントを参照してください。

ポッド用のカスタムネットワーク

Amazon VPC CNI plugin for Kubernetes が、Amazon EC2 ノード用にセカンダリ Elastic Network Interface (ネットワークインターフェイス) を作成する際には、デフォルトで、ノードのプライマリネットワークインターフェイスと同じサブネットがその格納先になります。また、プライマリネットワークインターフェイスに関連付けられているものと同じセキュリティグループが、セカンダリネットワークインターフェイスにも関連付けられます。以下の 1 つ以上の理由により、プラグインを使用して異なるサブネットにセカンダリネットワークインターフェイスを作成させたり、セカンダリネットワークインターフェイスに異なるセキュリティグループを関連付けたり、また、その両方を行ったりする必要性が生じることがあります。

- プライマリネットワークインターフェイスが存在するサブネットで使用可能な IPv4 アドレスには、数の上で制限があります。これにより、サブネット内に作成できる Pods の数に制限が生じる可能性があります。セカンダリネットワークインターフェイス用に別のサブネットを指定すると、Pods のために使用可能な IPv4 アドレスの数を増やすことができます。
- セキュリティ上の理由からも、ノードのプライマリネットワークインターフェイスで使用するものとは異なるセキュリティグループまたはサブネットが、Pods に必要となる場合があります。
- ノードをパブリックサブネット内に構成した場合、Pods はプライベートサブネット内に配置します。パブリックサブネットに関連付けられているルートテーブルには、インターネットゲートウェイへのルートが含まれています。プライベートサブネットに関連付けられているルートテーブルには、インターネットゲートウェイへのルートは含まれません。

考慮事項

- カスタムネットワークを有効にすると、プライマリネットワークインターフェイスに割り当てられた IP アドレスは Pods に割り当てられません。Pods には、セカンダリネットワークインターフェイスからの IP アドレスのみが割り当てられます。
- クラスターで IPv6 ファミリーを使用している場合は、カスタムネットワーキングを使用することはできません。
- IPv4 アドレスの枯渇を軽減するために、カスタムネットワーキングを使用する予定の場合は、IPv6 ファミリーを使用してクラスターを作成することもできます。詳細については、「クラスター、Pods、services 用の IPv6 アドレス」を参照してください。

- サブネットにデプロイされたセカンダリネットワークインターフェイス用の Pods が、ノードのプライマリネットワークインターフェイスとは異なるサブネットおよびセキュリティグループを使用できるように設定されていても、そのサブネットとセキュリティグループは、ノードと同じ VPC 内に配置される必要があります。

前提条件

- Amazon VPC CNI plugin for Kubernetes がセカンダリネットワークインターフェイスを作成し、IP アドレスを Pods に割り当てる方法に精通していること。詳細については、GitHub の「ENI Allocation」(ENI 割り当て) を参照してください。
- ご使用のデバイスまたは AWS CloudShell で、バージョン `2.12.3` 以降、または AWS Command Line Interface (AWS CLI) のバージョン `1.27.160` 以降がインストールおよび設定されていること。現在のバージョンを確認するには、「**`aws --version | cut -d / -f2 | cut -d ' ' -f1`**」を参照してください。macOS の `yum`、`apt-get`、または Homebrew などのパッケージマネージャは、AWS CLI の最新バージョンより数バージョン遅れることがあります。最新バージョンをインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI のインストール、更新、およびアンインストール」と「aws configure でのクイック設定」を参照してください。AWS CloudShell にインストールされている AWS CLI バージョンは、最新バージョンより数バージョン遅れている可能性もあります。更新するには、「AWS CloudShell ユーザーガイド」の「ホームディレクトリへの AWS CLI のインストール」を参照してください。
- デバイスまたは AWS CloudShell に、`kubectl` コマンドラインツールがインストールされていること。バージョンは、ご使用のクラスターの Kubernetes バージョンと同じか、1 つ前のマイナーバージョン以前、あるいはそれより新しいバージョンが使用できます。例えば、クラスターのバージョンが `1.29` である場合、`kubectl` のバージョン `1.28`、`1.29`、または `1.30` が使用できます。`kubectl` をインストールまたはアップグレードする方法については、「kubectl のインストールまたは更新」を参照してください。
- このトピック内の手順は、 Bash シェル内で実行することが推奨されます。Bash シェルを使用していない場合、行継続文字や、変数の設定と使用に関する方法など、一部のスクリプトコマンドのためにシェルの調整が必要となります。さらに、シェルの引用規則とエスケープ規則は異なる場合があります。詳細については、「AWS Command Line Interface ユーザーガイド」の「AWS CLI での文字列への引用符の使用」を参照してください。

このチュートリアルでは、*`example values`* を使用することをお勧めしますが、置き換えるように書かれている箇所はその限りではありません。本番稼働用クラスター向けに手順を完了するときには、どの *`example value`* も置き換えることができます。すべての手順は、同一のターミナルで実

行することをお勧めします。設定された変数はステップ全体で使用され、また、これらは異なるターミナルには存在しないためです。

このトピックのコマンドは、「AWS CLI 使用例を使用する」に記載されている規則に従ってフォーマットされています。使用している AWS CLI プロファイルで定義されているデフォルトの AWS リージョン とは異なる AWS リージョン にあるリソースに対してコマンドラインからコマンドを実行する場合は、コマンドに **--region *region-code*** を追加する必要があります。

カスタムネットワーキングを本番用クラスターにデプロイする場合は、ステップ 2: VPC を設定するにスキップします。

ステップ 1: テスト VPC およびクラスターを作成する

クラスターを作成するには

次の手順により、テスト VPC とクラスターを作成し、そのクラスターのためにカスタムネットワークを構成できます。本番向けのクラスターで必要になるいくつかの機能で、このトピックに関連のないものについては説明していません。そのため、本番ワークロード向けに、このテストクラスターを使用することはお勧めしません。詳細については、「Amazon EKS クラスターの作成」を参照してください。

1. 残りのステップで使用する変数をいくつか定義します。

```
export cluster_name=my-custom-networking-cluster
account_id=$(aws sts get-caller-identity --query Account --output text)
```

2. VPC を作成します。

   1. Amazon EKS の AWS CloudFormation テンプレートを使用して VPC を作成します。

```
aws cloudformation create-stack --stack-name my-eks-custom-networking-vpc \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-
eks/cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml \
  --parameters ParameterKey=VpcBlock,ParameterValue=192.168.0.0/24 \
  ParameterKey=PrivateSubnet01Block,ParameterValue=192.168.0.64/27 \
  ParameterKey=PrivateSubnet02Block,ParameterValue=192.168.0.96/27 \
  ParameterKey=PublicSubnet01Block,ParameterValue=192.168.0.0/27 \
  ParameterKey=PublicSubnet02Block,ParameterValue=192.168.0.32/27
```

      AWS CloudFormation スタックが作成されるまで数分かかります。次のコマンドを実行して、スタックのデプロイステータスを確認します。

```
aws cloudformation describe-stacks --stack-name my-eks-custom-networking-vpc --
query Stacks\[\].StackStatus  --output text
```

コマンドの出力が「CREATE_COMPLETE」になるまで、次のステップに進まないでください。

2. テンプレートによって作成された、プライベートサブネット ID の値により変数を定義します。

```
subnet_id_1=$(aws cloudformation describe-stack-resources --stack-name my-eks-
custom-networking-vpc \
    --query "StackResources[?
LogicalResourceId=='PrivateSubnet01'].PhysicalResourceId" --output text)
subnet_id_2=$(aws cloudformation describe-stack-resources --stack-name my-eks-
custom-networking-vpc \
    --query "StackResources[?
LogicalResourceId=='PrivateSubnet02'].PhysicalResourceId" --output text)
```

3. 前の手順で取得した、サブネットのアベイラビリティーゾーンを使用して変数を定義します。

```
az_1=$(aws ec2 describe-subnets --subnet-ids $subnet_id_1 --query
 'Subnets[*].AvailabilityZone' --output text)
az_2=$(aws ec2 describe-subnets --subnet-ids $subnet_id_2 --query
 'Subnets[*].AvailabilityZone' --output text)
```

3. クラスターの IAM ロールを作成します。

   a. IAM 信頼ポリシー用の JSON ファイルを作成するには、次のコマンドを実行します。

```
cat >eks-cluster-role-trust-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
```

```
  ]
}
EOF
```

b. Amazon EKS クラスター の IAM ロールを作成します。必要に応じて、`eks-cluster-role-trust-policy.json` の前に、手順でファイルの書き込み先となったコンピュータ上のパスを追加します。このコマンドは、前のステップで作成した信頼ポリシーをロールに関連付けます。IAM ロールを作成するには、ロールを作成する IAM プリンシパルに `iam:CreateRole` アクション (許可) を割り当てる必要があります。

```
aws iam create-role --role-name myCustomNetworkingAmazonEKSClusterRole --
assume-role-policy-document file://"eks-cluster-role-trust-policy.json"
```

c. このロールに、Amazon EKS 管理の IAM ポリシー (AmazonEKSClusterPolicy) をアタッチします。IAM ポリシーを IAM プリンシパルにアタッチするには、ポリシーのアタッチを行っているプリンシパルに、次のいずれかの IAM アクション (許可) を割り当てる必要があります: `iam:AttachUserPolicy` または `iam:AttachRolePolicy`。

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSClusterPolicy --role-name myCustomNetworkingAmazonEKSClusterRole
```

4. Amazon EKS クラスターを作成し、このクラスターとデバイスの間の通信を設定します。

a. クラスターを作成する。

```
aws eks create-cluster --name my-custom-networking-cluster \
   --role-arn arn:aws:iam::$account_id:role/
myCustomNetworkingAmazonEKSClusterRole \
   --resources-vpc-config subnetIds=$subnet_id_1","$subnet_id_2
```

> **Note**
>
> リクエストで指定したアベイラビリティーゾーンのいずれかに、Amazon EKS クラスターの作成に十分な容量がない場合には、エラーが表示されることがあります。このエラー出力には、新しいクラスターをサポートできるアベイラビリティーゾーンが表示されます。アカウント向けにサポートされているアベイラビリティーゾーンにある 2 つ以上のサブネットを使用して、クラスターを作成します。詳細については、「容量不足」を参照してください。

b. クラスターが作成されるまでに数分かかります。次のコマンドを実行して、クラスターのデプロイステータスを確認します。

```
aws eks describe-cluster --name my-custom-networking-cluster --query
 cluster.status
```

コマンドの出力が「"ACTIVE"」になるまで、次のステップに進まないでください。

c. クラスターと通信するために kubectl を設定します。

```
aws eks update-kubeconfig --name my-custom-networking-cluster
```

ステップ 2: VPC を設定する

このチュートリアルでは、ステップ 1: テスト VPC およびクラスターを作成する で作成した VPC が必要です。本番クラスターの場合は、すべての *example values* を独自のものに置き換えて、VPC に応じて手順を調整します。

1. 現在インストールされている Amazon VPC CNI plugin for Kubernetes のバージョンが最新であることを確認します。Amazon EKS アドオンタイプの最新バージョンを確認し、そのバージョンに更新するには、「アドオンの更新」を参照してください。セルフマネージドアドオンタイプの最新バージョンを確認し、そのバージョンに更新するには、「Amazon VPC CNI plugin for Kubernetes Amazon EKS アドオンの使用」を参照してください。
2. クラスター VPC の ID を取得し、後の手順で使用するために変数に格納します。本番向けクラスターの場合は、*my-custom-networking-cluster* を自分のクラスター名に置き換えます

```
vpc_id=$(aws eks describe-cluster --name my-custom-networking-cluster --query
 "cluster.resourcesVpcConfig.vpcId" --output text)
```

3. 追加の Classless Inter-Domain Routing (CIDR) ブロックを、クラスターの VPC に関連付けます。CIDR ブロックは、既存の関連付けられた CIDR ブロックと重複することはできません。

   1. 現在 VPC に関連付けられている CIDR ブロックを表示します。

```
aws ec2 describe-vpcs --vpc-ids $vpc_id \
    --query 'Vpcs[*].CidrBlockAssociationSet[*].{CIDRBlock: CidrBlock, State:
 CidrBlockState.State}' --out table
```

出力例は次のとおりです。

```
---------------------------------
|          DescribeVpcs          |
+-----------------+--------------+
|    CIDRBlock    |    State     |
+-----------------+--------------+
|  192.168.0.0/24 |  associated  |
+-----------------+--------------+
```

2. 追加の CIDR ブロックを VPC に関連付けます。詳細については、「Amazon VPC ユーザーガイド」の「Associate additional IPv4 CIDR blocks with your VPC」(VPC とセカンダリ IP アドレス CIDR ブロックを関連付ける) を参照してください。

```
aws ec2 associate-vpc-cidr-block --vpc-id $vpc_id --cidr-block 192.168.1.0/24
```

3. 新しいブロックが関連付けられていることを確認します。

```
aws ec2 describe-vpcs --vpc-ids $vpc_id --query
 'Vpcs[*].CidrBlockAssociationSet[*].{CIDRBlock: CidrBlock, State:
 CidrBlockState.State}' --out table
```

出力例は次のとおりです。

```
---------------------------------
|          DescribeVpcs          |
+-----------------+--------------+
|    CIDRBlock    |    State     |
+-----------------+--------------+
|  192.168.0.0/24 |  associated  |
|  192.168.1.0/24 |  associated  |
+-----------------+--------------+
```

新しい CIDR ブロックの State が associated に遷移するまで、次のステップには進まないでください。

4. 既存のサブネットが存在する各アベイラビリティーゾーン内で、使用するサブネットを必要な数だけ作成します。前のステップで VPC に関連付けた、CIDR ブロック内にある CIDR ブロックを指定します。

1. 新しいサブネットを作成します。サブネットは、既存のサブネットが置かれているものとは異なる VPC CIDR ブロックの中に作成する必要があり、作成するアベイラビリティーゾーンは、既存のサブネットと同じにする必要があります。この例では、現在のプライベートサブネットが存在する各アベイラビリティーゾーンにある新しい CIDR ブロックの中に、1 つのサブネットが作成されます。作成されたサブネットの ID は、後のステップで使用するために変数に格納されます。Name の値は、前のステップで Amazon EKS VPC テンプレートを使用して作成されたサブネットに、割り当てられている値と一致します。名前は必須ではありません。名前には異なる値を使用できます。

```
new_subnet_id_1=$(aws ec2 create-subnet --vpc-id $vpc_id --availability-zone
 $az_1 --cidr-block 192.168.1.0/27 \
    --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=my-eks-
custom-networking-vpc-PrivateSubnet01},{Key=kubernetes.io/role/internal-
elb,Value=1}]' \
    --query Subnet.SubnetId --output text)
new_subnet_id_2=$(aws ec2 create-subnet --vpc-id $vpc_id --availability-zone
 $az_2 --cidr-block 192.168.1.32/27 \
    --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=my-eks-
custom-networking-vpc-PrivateSubnet02},{Key=kubernetes.io/role/internal-
elb,Value=1}]' \
    --query Subnet.SubnetId --output text)
```

> **Important**
>
> デフォルトで新しいサブネットは、VPC のメインルートテーブルに暗黙的に関連付けられます。このルートテーブルにより、その VPC にデプロイされているすべてのリソース間の通信が可能になります。ただし、VPC に関連付けられた CIDR ブロックの外部にある IP アドレスが割り当てられたリソースとの通信は許可されません。この動作を変更するには、独自のルートテーブルをサブネットに関連付けます。詳細については、「Amazon VPC ユーザーガイド」の「サブネットルートテーブル」を参照してください。

2. 使用している VPC の現在のサブネットを表示します。

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=$vpc_id" \
    --query 'Subnets[*].{SubnetId: SubnetId,AvailabilityZone:
 AvailabilityZone,CidrBlock: CidrBlock}' \
    --output table
```

出力例は次のとおりです。

```
-------------------------------------------------------------------
|                          DescribeSubnets                        |
+------------------+--------------------+---------------------------+
| AvailabilityZone |     CidrBlock      |         SubnetId          |
+------------------+--------------------+---------------------------+
|  us-west-2d      |  192.168.0.0/27    |     subnet-example1       |
|  us-west-2a      |  192.168.0.32/27   |     subnet-example2       |
|  us-west-2a      |  192.168.0.64/27   |     subnet-example3       |
|  us-west-2d      |  192.168.0.96/27   |     subnet-example4       |
|  us-west-2a      |  192.168.1.0/27    |     subnet-example5       |
|  us-west-2d      |  192.168.1.32/27   |     subnet-example6       |
+------------------+--------------------+---------------------------+
```

作成した 192.168.1.0 CIDR ブロック内のサブネットが、192.168.0.0 CIDR ブロック内のサブネットと同じアベイラビリティーゾーンに置かれていることが確認できます。

ステップ 3: Kubernetes リソースを設定する

Kubernetes リソースを設定するには

1. aws-node DaemonSet で AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG 環境変数を true に設定します。

```
kubectl set env daemonset aws-node -n kube-system
 AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG=true
```

2. クラスターのセキュリティグループの ID を取得し、次のステップで使用するために変数に格納します。このセキュリティグループは、クラスターの作成時に Amazon EKS により自動的に作成されます。

```
cluster_security_group_id=$(aws eks describe-cluster --name $cluster_name --query
 cluster.resourcesVpcConfig.clusterSecurityGroupId --output text)
```

3. Pods をデプロイするサブネットごとに ENIConfig カスタムリソースを作成します。

   a. 各ネットワークインターフェイス設定に固有のファイルを作成します。

次のコマンドでは、前のステップで作成した 2 つのサブネットのそれぞれに、個別の ENIConfig ファイルが作成されます。name の値は一意である必要があります。これらの名前は、サブネットが存在するアベイラビリティーゾーンと同じものが付けられます。クラスターセキュリティグループは、ENIConfig に割り当てられます。

```
cat >$az_1.yaml <<EOF
apiVersion: crd.k8s.amazonaws.com/v1alpha1
kind: ENIConfig
metadata:
  name: $az_1
spec:
  securityGroups:
    - $cluster_security_group_id
  subnet: $new_subnet_id_1
EOF
```

```
cat >$az_2.yaml <<EOF
apiVersion: crd.k8s.amazonaws.com/v1alpha1
kind: ENIConfig
metadata:
  name: $az_2
spec:
  securityGroups:
    - $cluster_security_group_id
  subnet: $new_subnet_id_2
EOF
```

本番向けのクラスターでは、前のコマンドに次の変更を行います。

- *$cluster_security_group_id* は、ENIConfig で使用する (既存の) セキュリティグループの ID に置き換えます。
- 可能な限り、ENIConfig が使用されるアベイラビリティーゾーンと同じ名前を、ENIConfigs に付けることをお勧めします。いくつかの理由から、ENIConfigs に対して、アベイラビリティーゾーンとは異なる名前を使用する必要が生じることがあります。例えば、同じアベイラビリティーゾーンに 3 つ以上のサブネットがあり、そのすべてでカスタムネットワーキングを使用したい場合は、同一のアベイラビリティーゾーンに複数の ENIConfigs が必要になります。各 ENIConfig には一意の名前が必要であるため、複数の ENIConfigs に対しアベイラビリティーゾーンの名前を使用することはできません。

ENIConfig 名がすべてアベイラビリティーゾーン名と同じでない場合は、前のコマンドで *$az_1* と *$az_2* を独自の名前に置き換え、このチュートリアルの後半でノードに ENIConfig の注釈を付けます。

> **Note**
>
> 本番向けクラスターで使用するために有効なセキュリティグループを指定しておらず、かつ:
>
> - Amazon VPC CNI plugin for Kubernetes のバージョン 1.8.0 以降を使用する場合は、ノードのプライマリ Elastic Network Interface に関連付けられたセキュリティグループが使用されます。
> - 1.8.0 より前のバージョンの Amazon VPC CNI plugin for Kubernetes を使用している場合は、VPC のデフォルトのセキュリティグループが、セカンダリネットワークインターフェースに割り当てられます。

> **Important**
>
> - AWS_VPC_K8S_CNI_EXTERNALSNAT=false は、Kubernetes 用の Amazon VPC CNI プラグインの設定のデフォルト設定です。デフォルト設定を使用している場合、VPC に関連付けられた CIDR ブロックのいずれにもない IP アドレスに向けて送信されたトラフィックには、ノードのプライマリネットワークインターフェイスでの、セキュリティグループとサブネットが使用されます。セカンダリネットワークインターフェイスを作成するために使用された、ENIConfigs で定義されているサブネットとセキュリティグループは、このトラフィックでは使用されません。この設定の詳細については、「Pods の SNAT」を参照してください。
> - また、Pods でもセキュリティグループ使用をする場合、そのセキュリティグループは SecurityGroupPolicy で指定されたものとなり、ENIConfigs で指定されたものは使用されません。詳細については、「Pods のセキュリティグループ」を参照してください。

b. 以下のコマンドを使用して、作成したカスタムリソースファイルをそれぞれクラスターに適用します。

```
kubectl apply -f $az_1.yaml
kubectl apply -f $az_2.yaml
```

4. ENIConfigs が作成されたことを確認します。

```
kubectl get ENIConfigs
```

出力例は次のとおりです。

```
NAME         AGE
us-west-2a   117s
us-west-2d   105s
```

5. 本番向けのクラスターで、カスタムネットワーキングを有効にしており、ENIConfigs の名前を使用されているアベイラビリティーゾーンとは異なるものにした場合は、次のステップに移り、Amazon EC2 ノードをデプロイします。

クラスターで作成された任意の新しい Amazon EC2 ノードに対し、Kubernetes がアベイラビリティーゾーン用の ENIConfig を自動的に適用することを許可します。

1. このチュートリアルでのテスト用クラスターの場合は、このまま次のステップに移動します。

実稼働クラスターの場合は、ENI_CONFIG_ANNOTATION_DEF 環境変数用のキー k8s.amazonaws.com/eniConfig を使用する annotation が、aws-node DaemonSet のためのコンテナ仕様内に存在することを確認します。

```
kubectl describe daemonset aws-node -n kube-system | grep
 ENI_CONFIG_ANNOTATION_DEF
```

出力が返される場合は、アノテーションが存在します。出力が返されない場合、その変数は設定されていません。本番向けクラスターの場合には、ここでの設定と、以降のステップで示す設定のどちらも使用可能です。ここでの設定を使用すると、以降のステップの設定は上書きされます。このチュートリアルでは、この後のステップでの設定を使用します。

2. クラスター内に作成された任意の新しい Amazon EC2 ノードに対し、自動的にアベイラビリティーゾーンのための ENIConfig を適用するように、aws-node DaemonSet を更新します。

```
kubectl set env daemonset aws-node -n kube-system
 ENI_CONFIG_LABEL_DEF=topology.kubernetes.io/zone
```

ステップ 4: Amazon EC2 ノードをデプロイする

Amazon EC2 ノードをデプロイするには

1. ノードの IAM ロールを作成します。

   a. IAM 信頼ポリシー用の JSON ファイルを作成するには、次のコマンドを実行します。

```
cat >node-role-trust-relationship.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

   b. 次のコマンドを実行して、ロール名の変数を設定します。*myCustomNetworkingAmazonEKSNodeRole* は、任意の名前に置き換えることができます。

```
export node_role_name=myCustomNetworkingAmazonEKSNodeRole
```

   c. IAM ロールを作成し、返された Amazon リソースネーム (ARN) を、後のステップで使用するために変数に格納します。

```
node_role_arn=$(aws iam create-role --role-name $node_role_name --assume-role-
policy-document file://"node-role-trust-relationship.json" \
    --query Role.Arn --output text)
```

   d. IAM ロールに、3 つの必須な IAM マネージドポリシーをアタッチします。

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name $node_role_name
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name $node_role_name
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
    --role-name $node_role_name
```

> **Important**
>
> 分かりやすくするため、このチュートリアルでは、AmazonEKS_CNI_Policy ポリシーはノードの IAM ロールにアタッチされています。ただし、本番向けクラスターの場合は、Amazon VPC CNI plugin for Kubernetes のみで使用される IAM ロールに、個別にポリシーをアタッチすることをお勧めします。詳細については、「サービスアカウントの IAM ロールを使用する Amazon VPC CNI plugin for Kubernetes の設定」を参照してください。

2. 次のいずれかのタイプのノードグループを作成します。デプロイするインスタンスタイプを確認するには、「Amazon EC2 インスタンスタイプを選択する」を参照してください。このチュートリアルのために、マネージド型、および起動テンプレートを使用しない、または AMI ID が指定されていない起動テンプレートを使用するオプションを完了します。本番ワークロードでノードグループを使用する場合は、そのグループをデプロイする前に、マネージド型およびセルフマネージド型の、すべてのノードグループオプションについて理解しておくことをお勧めします。

   - マネージド型 — 次のいずれかのオプションを使用して、ノードグループをデプロイします。
     - 起動テンプレートを使用しない、または AMI ID が指定されていない起動テンプレートを使用する — 次のコマンドを実行します。このチュートリアルでは、*example values* を使用します。本番稼働用のノードグループの場合は、すべての *example values* を実際の値に置き換えます。ノードグループ名は 63 文字以下である必要があります。先頭は文字または数字でなければなりませんが、残りの文字にはハイフンおよびアンダースコアを含めることもできます。

```
aws eks create-nodegroup --cluster-name $cluster_name --nodegroup-name my-
nodegroup \
```

```
    --subnets $subnet_id_1 $subnet_id_2 --instance-types t3.medium --node-role
$node_role_arn
```

- 指定された AMI ID を持つ起動テンプレートを使用

  1. Amazon EKS で推奨される、ノードでの Pods の最大数を決定します。各 Amazon EC2 インスタンスタイプの Amazon EKS 推奨最大 Pods 数 の手順に従います (**--cni-custom-networking-enabled** をステップ 3 に追加してください)。後のステップで使用するために、この出力を書き留めます。
  2. 起動テンプレートで、Amazon EKS 最適化 AMI ID を指定するか、Amazon EKS 最適化 AMI から構築されたカスタム AMI を指定します。その後、起動テンプレートを使用してノードグループをデプロイし、さらに、起動テンプレートにある次のユーザーデータを指定します。このユーザーデータは、引数を bootstrap.sh ファイルに渡します。ブートストラップファイルの詳細については、「GitHub」の「bootstrap.sh」を参照してください。*20* は、前のステップで使用した値 (推奨) か、または独自の値に置き換えることができます。

     ```
     /etc/eks/bootstrap.sh my-cluster --use-max-pods false --kubelet-extra-args
      '--max-pods=20'
     ```

     Amazon EKS 最適化 AMI から構築されていないカスタム AMI を作成した場合は、自分で設定をカスタム作成する必要があります。

- セルフマネージド型

  1. Amazon EKS で推奨される、ノードでの Pods の最大数を決定します。各 Amazon EC2 インスタンスタイプの Amazon EKS 推奨最大 Pods 数 の手順に従います (**--cni-custom-networking-enabled** をステップ 3 に追加してください)。後のステップで使用するために、この出力を書き留めます。
  2. セルフマネージド型の Amazon Linux ノードの起動 の手順に従ってノードグループをデプロイします。BootstrapArguments パラメータに、以下のテキストを指定します。*20* は、前のステップで使用した値 (推奨) か、または独自の値に置き換えることができます。

     ```
     --use-max-pods false --kubelet-extra-args '--max-pods=20'
     ```

> **Note**
>
> 本番クラスター内のノードで、非常に大量の Pods をサポートさせる場合には、各 Amazon EC2 インスタンスタイプの Amazon EKS 推奨最大 Pods 数 のスクリプトを再

度実行します。また、このコマンドには **--cni-prefix-delegation-enabled** オプションを追加します。例えば、m5.large インスタンスタイプの場合は *110* が返されます。この機能を有効化する方法については、「Amazon EC2 ノードで使用可能な IP アドレスの量を増やす」を参照してください。この機能は、カスタムネットワーキングで使用できます。

ノードグループの作成には数分かかります。次のコマンドを使用すると、マネージド型のノードグループ作成のステータスを確認できます。

```
aws eks describe-nodegroup --cluster-name $cluster_name --nodegroup-name my-nodegroup --query nodegroup.status --output text
```

出力が「ACTIVE」を返すまで、次のステップに進まないでください。

3. このチュートリアルでは、このステップをスキップできます。

本番向けのクラスターにおいて、ENIConfigs の名前を、それが使用されているアベイラビリティーゾーンと同にしなかった場合には、ノードに対し、そのノードで使用すべき ENIConfig の名前をアノテーションします。各アベイラビリティーゾーンにサブネットが 1 つしかなく、ENIConfigs にアベイラビリティーゾーンと同じ名前を付けている場合には、このステップは必要ありません。前のステップで機能を有効化してあるのなら、Amazon VPC CNI plugin for Kubernetes が適切な ENIConfig をノードに対し自動的に関連付けるためです。

a. クラスター内のノードのリストを取得します。

```
kubectl get nodes
```

出力例は次のとおりです。

```
NAME                                           STATUS   ROLES    AGE     VERSION
ip-192-168-0-126.us-west-2.compute.internal   Ready    <none>   8m49s
 v1.22.9-eks-810597c
ip-192-168-0-92.us-west-2.compute.internal    Ready    <none>   8m34s
 v1.22.9-eks-810597c
```

b. 各ノードが属するアベイラビリティーゾーンを決定します。前のステップで返された各ノードに対し 次のコマンドを実行します。

```
aws ec2 describe-instances --filters Name=network-interface.private-dns-
name,Values=ip-192-168-0-126.us-west-2.compute.internal \
--query 'Reservations[].Instances[].{AvailabilityZone:
 Placement.AvailabilityZone, SubnetId: SubnetId}'
```

出力例は次のとおりです。

```
[
    {
        "AvailabilityZone": "us-west-2d",
        "SubnetId": "subnet-Example5"
    }
]
```

c. 各ノードを、サブネット ID とアベイラビリティーゾーンのために作成した ENIConfig でアノテーションします。各ノードのアノテーションに使用できる ENIConfig は 1 つだけです。ただし、複数のノードに対し同じ ENIConfig でアノテーションすることは可能です。*example values* を自分の値に置き換えます。

```
kubectl annotate node ip-192-168-0-126.us-west-2.compute.internal
 k8s.amazonaws.com/eniConfig=EniConfigName1
kubectl annotate node ip-192-168-0-92.us-west-2.compute.internal
 k8s.amazonaws.com/eniConfig=EniConfigName2
```

4. カスタムネットワーキング機能の使用に切り替える前に、ノードを使用している本番向けクラスター内で Pods を実行していた場合には、以下のタスクを完了してください。

   a. カスタムネットワーク機能を使用しているノードが、使用可能であることを確認します。

   b. ノードを遮断およびドレインして、Pods をスムーズにシャットダウンします。詳細については、「Kubernetes ドキュメント」の「Safely Drain a Node」(ノードを安全にドレインする) を参照してください。

   c. ノードを終了します。ノードが既存のマネージド型ノードグループ内にある場合は、そのノードグループを削除できます。デバイスに沿ったコマンドをコピーします。必要に応じてコマンドに次の変更を加え、変更したコマンドを実行します。

      - *my-cluster* をクラスターの名前に置き換えます。
      - *my-nodegroup* をノードグループの名前に置き換えます。

```
aws eks delete-nodegroup --cluster-name my-cluster --nodegroup-name my-
nodegroup
```

カスタムネットワーク機能は、k8s.amazonaws.com/eniConfig ラベルで登録されている新しいノードでのみ使用されます。

5. Pods に、前の手順で作成したサブネットの 1 つに関連付けられた CIDR ブロックから、 IP アドレスが割り当てられていることを確認します。

```
kubectl get pods -A -o wide
```

出力例は次のとおりです。

```
NAMESPACE     NAME                       READY   STATUS    RESTARTS   AGE     IP
           NODE                                            NOMINATED NODE   READINESS
 GATES
kube-system   aws-node-2rkn4             1/1     Running   0          7m19s
 192.168.0.92    ip-192-168-0-92.us-west-2.compute.internal    <none>
 <none>
kube-system   aws-node-k96wp             1/1     Running   0          7m15s
 192.168.0.126   ip-192-168-0-126.us-west-2.compute.internal   <none>
 <none>
kube-system   coredns-657694c6f4-smcgr   1/1     Running   0          56m
 192.168.1.23    ip-192-168-0-92.us-west-2.compute.internal    <none>
 <none>
kube-system   coredns-657694c6f4-stwv9   1/1     Running   0          56m
 192.168.1.28    ip-192-168-0-92.us-west-2.compute.internal    <none>
 <none>
kube-system   kube-proxy-jgshq           1/1     Running   0          7m19s
 192.168.0.92    ip-192-168-0-92.us-west-2.compute.internal    <none>
 <none>
kube-system   kube-proxy-wx9vk           1/1     Running   0          7m15s
 192.168.0.126   ip-192-168-0-126.us-west-2.compute.internal   <none>
 <none>
```

VPC に追加した 192.168.1.0 CIDR ブロックからの IP アドレスが、coredns Pods に割り当てられていることが確認できます。カスタムネットワーキングを使用していない場合は、ここ

に 192.168.0.0 CIDR ブロックからのアドレスが割り当てられています。この CIDR ブロックが、元々 VPC に関連付けられている唯一のブロックであるためです。

Pod's spec が hostNetwork=true を含む場合には、ノードのプライマリ IP アドレスが割り当てられます。追加したサブネットのアドレスは割り当てられません。デフォルトでは、この値は false に設定されます。この値は、クラスターで実行されている kube-proxy および Amazon VPC CNI plugin for Kubernetes(aws-node)Pods に対して true に設定されます。これが、kube-proxy とプラグインの aws-node Pods に対し、前の出力にある 192.168.1.*x* アドレスが割り当てられない理由です。Pod's hostNetwork の設定の詳細については、「Kubernetes API リファレンス」の「PodSpec v1 core」を参照してください。

ステップ 5: チュートリアルでのリソースを削除する

このチュートリアル完了後は、作成したリソースを削除することをお勧めします。その後、ここでの手順を調整して、本番向けクラスターのカスタムネットワーキングを有効化することができます。

チュートリアルでのリソースを削除するには

1. 作成したノードグループが完全にテスト向けである場合は、それを削除してください。

```
aws eks delete-nodegroup --cluster-name $cluster_name --nodegroup-name my-nodegroup
```

AWS CLI から、クラスターが削除済みだと出力された場合でも、実際には、その削除プロセスが完了していない場合があります。削除のプロセスには数分かかります。次のコマンドを実行して、処理の完了を確認します。

```
aws eks describe-nodegroup --cluster-name $cluster_name --nodegroup-name my-
nodegroup --query nodegroup.status --output text
```

次のように出力されるまで、次に進まないでください。

```
An error occurred (ResourceNotFoundException) when calling the DescribeNodegroup
 operation: No node group found for name: my-nodegroup.
```

2. 作成したノードグループが完全にテスト向けである場合は、ノードの IAM ロールを削除します。

   a. ロールからポリシーをデタッチします。

```
aws iam detach-role-policy --role-name myCustomNetworkingAmazonEKSNodeRole --
policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
aws iam detach-role-policy --role-name myCustomNetworkingAmazonEKSNodeRole --
policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
aws iam detach-role-policy --role-name myCustomNetworkingAmazonEKSNodeRole --
policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
```

b. ロールを削除します。

```
aws iam delete-role --role-name myCustomNetworkingAmazonEKSNodeRole
```

3. クラスターを削除します。

```
aws eks delete-cluster --name $cluster_name
```

以下のコマンドを使用して、クラスターが削除されたことを確認します。

```
aws eks describe-cluster --name $cluster_name --query cluster.status --output text
```

次のような出力が返された場合は、クラスターが正常に削除されています。

```
An error occurred (ResourceNotFoundException) when calling the DescribeCluster
 operation: No cluster found for name: my-cluster.
```

4. クラスターの IAM ロールを削除します。

a. ロールからポリシーをデタッチします。

```
aws iam detach-role-policy --role-name myCustomNetworkingAmazonEKSClusterRole
 --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy
```

b. ロールを削除します。

```
aws iam delete-role --role-name myCustomNetworkingAmazonEKSClusterRole
```

5. 前のステップで作成したサブネットを削除します。

```
aws ec2 delete-subnet --subnet-id $new_subnet_id_1
aws ec2 delete-subnet --subnet-id $new_subnet_id_2
```

6. 作成した VPC を削除します。

```
aws cloudformation delete-stack --stack-name my-eks-custom-networking-vpc
```

Amazon EC2 ノードで使用可能な IP アドレスの量を増やす

各 Amazon EC2 インスタンスでは、Elastic Network Interface の最大数と、各ネットワークインターフェイスに割り当て可能な IP アドレスの最大数がサポートされています。各ノードには、ネットワークインターフェースごとに 1 つの IP アドレスが必要です。その他の使用可能な IP アドレスはすべて Pods に割り当てることができます。Pod それぞれに固有の IP アドレスが必要です。その結果、使用可能なコンピューティングリソースとメモリリソースはあるが、Pods に割り当てる IP アドレスが不足しているために追加の Pods に対応できないノードが存在する可能性があります。

このトピックでは、個別のセカンダリ IP アドレスではなく、IP プレフィックスをノードに割り当てて、ノードで Pods に割り当て可能な IP アドレスの数を大幅に増やす方法について説明します。各プレフィックスには複数の IP アドレスが含まれます。IP プレフィックスが割り当てられるようにクラスターを設定しない場合、クラスターは Pod 接続に必要なネットワークインターフェイスと IP アドレスを設定するために Amazon EC2 アプリケーションプログラミングインターフェイス (API) の呼び出しをさらに実行する必要があります。クラスターのサイズが大きくなるにつれて、これらの API コールの頻度が高くなるため、Pod とインスタンスの起動時間が長くなる場合があります。これにより、大規模でスパイク的なワークロードの需要を満たすためにスケーリングの遅延が発生します。また、スケーリング要件を満たすためには、追加のクラスターと VPC をプロビジョニングする必要があるため、コストと管理オーバーヘッドも増加します。詳細については、「GitHub」の「Kubernetes スケーラビリティ閾値」を参照してください。

考慮事項

- 各 Amazon EC2 インスタンスタイプでは、Pods の最大数がサポートされています。マネージドノードグループが複数のインスタンスタイプで構成されている場合、クラスターのインスタンスで指定する Pods の最大数の内、最小の値がクラスター内のすべてのノードに適用されます。
- デフォルトでは、1 つのノードで実行できる Pods の最大数は 110 ですが、この値は変更できます。数値を変更し、既存のマネージド型ノードグループがある場合、ノードグループの AMI または起動テンプレートを次に更新するときに、新しいノードで変更済みの値が適用されるようになります。
- IP アドレスの割り当てから IP プレフィックスの割り当てに移行する場合は、既存のノードにローリング置換を実行するのではなく、新しいノードグループを作成して使用可能な IP アドレスの数を増やすことをお勧めします。IP アドレスと IP プレフィックスの両方が割り当てられているノー

ドで Pods を実行すると、アドバタイズされた IP アドレスの容量に不一致が生じ、ノード上の将来のワークロードに影響する可能性があります。推奨される移行方法については、「Amazon EKS ベストプラクティスガイド」の「セカンダリ IP モードからプレフィックス委任モード、またはその逆への移行中にすべてのノードを置換する」を参照してください。

- Linux ノードのみを含むクラスターの場合。
  - ネットワークインターフェイスにプレフィックスを割り当てるようにアドオンを設定すると、クラスター内のすべてのノードグループ内のすべてのノードを削除しない限り、Amazon VPC CNI plugin for Kubernetes アドオンを `1.9.0` (あるいは `1.10.1`) より前のバージョンにダウングレードすることはできません。
  - Pods のセキュリティグループ (`POD_SECURITY_GROUP_ENFORCING_MODE=standard` および `AWS_VPC_K8S_CNI_EXTERNALSNAT=false` に設定) も使用している場合、Pods が VPC の外部のエンドポイントと通信するときに、Pods に割り当てたセキュリティグループではなく、ノードのセキュリティグループが使用されます。

    Pods のセキュリティグループ (`POD_SECURITY_GROUP_ENFORCING_MODE=strict` に設定) も使用している場合、Pods が VPC の外部のエンドポイントと通信するときに、`Pod's` セキュリティグループが使用されます。

前提条件

- 既存のクラスター。デプロイするには、「Amazon EKS クラスターの作成」を参照してください。
- Amazon EKS ノードが配置されているサブネットでは、`/28` 個の連続ブロック (`IPv4` クラスターの場合)、または `/80` 個の Classless Inter-Domain Routing (CIDR) ブロック (`IPv6` クラスターの場合) が十分な数として必要になります。`IPv6` クラスターに含めることができるのは Linux ノードだけです。IP アドレスがサブネット CIDR 全体に分散している場合、IP プレフィックスを使用すると失敗する可能性があります。次のことを推奨します。
  - サブネット CIDR 予約を使用すると、予約された範囲内の IP アドレスがまだ使用されている場合でも、解放時に IP アドレスが再割り当てされません。これにより、セグメンテーションなしでプレフィックスを割り当てることができます。
  - IP プレフィックスが割り当てられているワークロードの実行に特に使用される新しいサブネットを使用してください。IP プレフィックスを割り当てると、Windows と Linux 両方のワークロードを同じサブネットで実行できます。
- ノードに IP プレフィックスを割り当てるには、ノードが AWS Nitro ベースである必要があります。Nitro ベースではないインスタンスでは、引き続き個別のセカンダリ IP アドレスが割り当てら

れますが、Pods に割り当てることのできる IP アドレスの数は Nitro-based のインスタンスよりも大幅に少なくなります。

- Linux ノードのみのクラスターの場合 – クラスターが `IPv4` ファミリーで設定されている場合は、バージョン `1.9.0` 以降の Amazon VPC CNI plugin for Kubernetes アドオンがインストールされている必要があります。現在のバージョンは、次のコマンドで確認できます。

```
kubectl describe daemonset aws-node --namespace kube-system | grep Image | cut -d "/"
 -f 2
```

  クラスターが `IPv6` ファミリーで設定されている場合は、バージョン `1.10.1` のアドオンがインストールされている必要があります。プラグインバージョンが必要なバージョンよりも古い場合は、更新する必要があります。詳細については、更新された「Amazon VPC CNI plugin for Kubernetes Amazon EKS アドオンの使用」セクションを参照してください。

- Windows ノードのみのクラスターの場合
  - クラスターとそのプラットフォームバージョンは、次の表のバージョン以降である必要があります。クラスターバージョンをアップグレードするには、「Amazon EKS クラスターの Kubernetes バージョンの更新」を参照してください。クラスターがプラットフォームの最小バージョンでない場合、Amazon EKS がプラットフォームバージョンを更新するまで、ノードに IP プレフィックスを割り当てることはできません。

Kubernetes バージョン	プラットフォームバージョン
`1.27`	`eks.3`
`1.26`	`eks.4`
`1.25`	`eks.5`

    現在の Kubernetes バージョンとプラットフォームバージョンを確認するには、次のコマンドの *`my-cluster`* をクラスターの名前に置き換えて、変更したコマンド **`aws eks describe-cluster --name my-cluster --query 'cluster.{"Kubernetes Version": version, "Platform Version": platformVersion}'`** を実行します。
  - クラスターで Windows サポートが有効になっています。詳細については、「Amazon EKS クラスター の Windows サポートの有効化」を参照してください。

Amazon EC2 ノードで使用可能な IP アドレスの量を増やす

1. ノードに IP アドレスプレフィックスを割り当てるようにクラスターを設定します。ノードのオペレーティングシステムに対応するタブで手順を完了します。

   Linux

   1. パラメータを有効にして、Amazon VPC CNI DaemonSet のネットワークインターフェイスにプレフィックスを割り当てます。1.21 以降のクラスターをデプロイすると、バージョン 1.10.1 以降の Amazon VPC CNI plugin for Kubernetes アドオンも同時にデプロイされます。IPv6 ファミリーを使用してクラスターを作成している場合、この設定はデフォルトで true に設定されています。IPv4 ファミリーを使用してクラスターを作成している場合、この設定はデフォルトで false に設定されています。

      ```
      kubectl set env daemonset aws-node -n kube-system
       ENABLE_PREFIX_DELEGATION=true
      ```

      Important

      サブネットに使用可能な IP アドレスがある場合でも、サブネットに使用可能な /28 個の連続したブロックがない場合には、Amazon VPC CNI plugin for Kubernetes ログに次のエラーが記述されます。

      ```
      InsufficientCidrBlocks: The specified subnet does not have enough free
       cidr blocks to satisfy the request
      ```

      これは、サブネット全体に広がる既存のセカンダリ IP アドレスの断片化が原因で発生する可能性があります。このエラーを解決するには、新しいサブネットを作成してそこに Pods を起動するか、Amazon EC2 サブネットの CIDR 予約を使用して、プレフィックス割り当てで使用するためのサブネット内の領域を予約します。詳細については、Amazon VPC ユーザーガイドの「サブネット CIDR の予約」を参照してください。

   2. 前提条件のリストに記載されたバージョン以降の Amazon VPC CNI plugin for Kubernetes を使用しながら、起動テンプレートなしでマネージド型ノードグループをデプロイする場合、または AMI ID を指定していない起動テンプレートを使用してデプロイする場合には、このまま次のステップに進みます。マネージド型ノードグループが、Pods の最大数を自動的に計算します。

AMI ID を指定した起動テンプレートを使用して、セルフマネージド型ノードグループまたはマネージド型ノードグループをデプロイする場合は、Amazon EKS でノードのために推奨される Pods の最大数を特定する必要があります。各 Amazon EC2 インスタンスタイプの Amazon EKS 推奨最大 Pods 数 の指示に従って、**--cni-prefix-delegation-enabled** をステップ 3に追加します。後のステップで使用するために、出力に注意してください。

> **Important**
>
> マネージド型ノードグループは、maxPods の値に最大数を適用します。vCPUs が 30 未満のインスタンスの場合、最大数は 110 で、その他すべてのインスタンスの最大数は 250 です。この最大数は、プレフィックス委任が有効かどうかにかかわらず適用されます。

3. IPv6 用に設定された 1.21 以降のクラスターを使用している場合は、このまま次のステップに進みます。

   以下のオプションの内の 1 つでパラメータを指定します。どのオプションが適切で、どの値を提供するかを決定するには、GitHub のWARM_PREFIX_TARGET、WARM_IP_TARGET、および MINIMUM_IP_TARGET を参照してください。

   example values をゼロより大きい値に置き換えることができます。

   - WARM_PREFIX_TARGET

     ```
     kubectl set env ds aws-node -n kube-system WARM_PREFIX_TARGET=1
     ```

   - WARM_IP_TARGET または MINIMUM_IP_TARGET – この値を設定した場合、WARM_PREFIX_TARGET に対し設定されている値はすべて上書きされます。

     ```
     kubectl set env ds aws-node -n kube-system WARM_IP_TARGET=5
     ```

     ```
     kubectl set env ds aws-node -n kube-system MINIMUM_IP_TARGET=2
     ```

4. 少なくとも 1 つの Amazon EC2 Nitro Amazon Linux 2 インスタンスタイプを使用して、次のいずれかのタイプのノードグループを作成します。Nitro インスタンスタイプのリストについては、「Amazon EC2 ユーザーガイド」の「Instances built on the

Nitro System」を参照してください。この機能は Windows ではサポートされていません。*110* が含まれているオプションの場合は、ステップ 3 の値 (推奨) または独自の値に置き換えてください。

- セルフマネージド – セルフマネージド型の Amazon Linux ノードの起動 の手順に従ってノードグループをデプロイします。BootstrapArguments パラメータに、以下のテキストを指定します。

```
--use-max-pods false --kubelet-extra-args '--max-pods=110'
```

eksctl を使用してノードグループを作成する場合は、次のコマンドを使用できます。

```
eksctl create nodegroup --cluster my-cluster --managed=false --max-pods-per-node 110
```

- マネージド型 — 次のいずれかのオプションを使用して、ノードグループをデプロイします。
  - 起動テンプレートがない場合、または AMI ID が指定されていない起動テンプレートを使用する場合 —「マネージド型ノードグループの作成」の手順を完了します。マネージド型ノードグループは、Amazon EKS で推奨される max-pods の値を自動的に計算します。
  - 指定した AMI ID を持つ起動テンプレート — 起動テンプレートで Amazon EKS 最適化 AMI ID を指定するか、Amazon EKS 最適化 AMI から構築されたカスタム AMI を指定してから、起動テンプレートを使用してノードグループをデプロイ起動テンプレートでは、次のユーザーデータを指定します。このユーザーデータは、引数を bootstrap.sh ファイルに渡します。ブートストラップファイルの詳細については、「GitHub」の「bootstrap.sh」を参照してください。

```
/etc/eks/bootstrap.sh my-cluster \
  --use-max-pods false \
  --kubelet-extra-args '--max-pods=110'
```

eksctl を使用してノードグループを作成する場合は、次のコマンドを使用できます。

```
eksctl create nodegroup --cluster my-cluster --max-pods-per-node 110
```

Amazon EKS 最適化 AMI から構築されていないカスタム AMI を作成した場合は、自分で設定をカスタム作成する必要があります。

> **Note**
>
> また、インスタンスとは異なるサブネットの Pods に IP アドレスを割り当てる場合は、このステップで機能を有効化する必要があります。詳細については、「ポッド用のカスタムネットワーク」を参照してください。

Windows

1. IP プレフィックスの割り当てを有効にします。

   a. 編集する `amazon-vpc-cni ConfigMap` を開きます。

   ```
   kubectl edit configmap -n kube-system amazon-vpc-cni -o yaml
   ```

   b. `data` セクションに次の行を追加します。

   ```
     enable-windows-prefix-delegation: "true"
   ```

   c. ファイルを保存し、エディタを閉じます。

   d. `ConfigMap` に行が追加されたことを確認します。

   ```
   kubectl get configmap -n kube-system amazon-vpc-cni -o
    "jsonpath={.data.enable-windows-prefix-delegation}"
   ```

   返された出力が `true` でない場合は、エラーが発生した可能性があります。ステップをもう一度実行してみてください。

   > **Important**
   >
   > サブネットに使用可能な IP アドレスがある場合でも、サブネットに使用可能な /28 個の連続したブロックがない場合には、ノードイベントに次のエラーが表示されます。

```
"failed to allocate a private IP/Prefix address:
 InsufficientCidrBlocks: The specified subnet does not have enough
 free cidr blocks to satisfy the request"
```

これは、サブネット全体に広がる既存のセカンダリ IP アドレスの断片化が原因で発生する可能性があります。このエラーを解決するには、新しいサブネットを作成してそこに Pods を起動するか、Amazon EC2 サブネットの CIDR 予約を使用して、プレフィックス割り当てで使用するためのサブネット内の領域を予約します。詳細については、Amazon VPC ユーザーガイドの「サブネット CIDR の予約」を参照してください。

2. (オプション) クラスターの事前スケーリング動作と動的スケーリング動作を制御するための追加設定を指定します。詳細については、「GitHub」の「Windows でのプレフィックス委任モードの設定オプション」を参照してください。

   a. 編集する amazon-vpc-cni ConfigMap を開きます。

   ```
   kubectl edit configmap -n kube-system amazon-vpc-cni -o yaml
   ```

   b. *example values* を 0 より大きい値に置き換え、必要なエントリを ConfigMap の data セクションに追加します。warm-ip-target または minimum-ip-target のいずれかに値を設定した場合、その値は warm-prefix-target に設定された値をオーバーライドします。

   ```
   warm-prefix-target: "1"
   warm-ip-target: "5"
   minimum-ip-target: "2"
   ```

   c. ファイルを保存し、エディタを閉じます。

3. 少なくとも 1 つの Amazon EC2 Nitro インスタンスタイプで Windows ノードグループを作成します。Nitro インスタンスタイプのリストについては、「Amazon EC2 ユーザーガイド」の「Instances built on the Nitro System」を参照してください。デフォルトでは、1 つのノードにデプロイできる Pods の最大数は 110 です。この数値を増減させる場合は、ブートストラップ設定のユーザーデータに以下を指定します。*max-pods-quantity* を最大ポッド値に置き換えます。

   ```
   -KubeletExtraArgs '--max-pods=max-pods-quantity'
   ```

マネージド型ノードグループをデプロイする場合は、この設定を起動テンプレートに追加する必要があります。詳細については、「起動テンプレートを使用したマネージドノードのカスタマイズ」を参照してください。Windows ブートストラップスクリプトの設定パラメータの詳細については、「ブートストラップスクリプトの設定パラメータ」を参照してください。

2. ノードがデプロイされると、クラスター内のノードの表示が可能になります。

```
kubectl get nodes
```

出力例は次のとおりです。

```
NAME                                               STATUS     ROLES    AGE   VERSION
ip-192-168-22-103.region-code.compute.internal   Ready      <none>   19m   v1.XX.X-eks-6b7464
ip-192-168-97-94.region-code.compute.internal    Ready      <none>   19m   v1.XX.X-eks-6b7464
```

3. いずれかのノードを記述して、そのノードの max-pods 値と使用可能な IP アドレスの数を決定します。前の出力で返されたいずれかのノードの名前の *192.168.30.193* を IPv4 アドレスに置き換えます。

```
kubectl describe node ip-192-168-30-193.region-code.compute.internal | grep 'pods\|PrivateIPv4Address'
```

出力例は次のとおりです。

```
pods:                                       110
vpc.amazonaws.com/PrivateIPv4Address:  144
```

前の出力にある 110 は、*144 #*の IP アドレスが使用可能であるにもかかわらず、Kubernetes がノードにデプロイする Pods の最大数です。

Pods のセキュリティグループ

Pods のセキュリティグループは、Amazon EC2 セキュリティグループを Kubernetes Pods と統合します。Amazon EC2 セキュリティグループを使用して、多くの Amazon EC2 インスタンスタイプと Fargate で実行されているノードにデプロイする Pods との間のインバウンドおよびアウトバウンド

のネットワークトラフィックを許可するルールを定義できます。この機能の詳細な説明については、「Introducing security groups for Pods」(ポッドのセキュリティグループの紹介) のブログを参照してください。

考慮事項

- Pods のセキュリティグループをデプロイする前に、次の制限事項と条件を考慮してください。
- Pods のセキュリティグループは、Windows ノードでは使用できません。
- Pods のセキュリティグループは、バージョン 1.16.0 以降の、Amazon VPC CNI プラグインを使用して、Amazon EC2 ノードを含む IPv6 ファミリー用に設定されたクラスターで使用できます。バージョン 1.7.7 以降の Amazon VPC CNI プラグインを使用すると、Fargate ノードのみを含むクラスター設定の IPv6 ファミリーで Pods のセキュリティグループを使用できます。詳細については、「クラスター、Pods、services 用の IPv6 アドレス」を参照してください。
- Pods のセキュリティグループは、ほとんどの Nitro ベースの Amazon EC2 インスタンスファミリーでサポートされていますが、ファミリーの全世代がサポートしているわけではありません。例えば、m5、c5、r5、m6g、、c6g、r6g インスタンスファミリーと世代ではサポートされています。t ファミリーのインスタンスタイプは一切サポートされていません。サポートされているインスタンスの詳細なリストについては、GitHub で limits.go ファイルを参照してください。ノードは、そのファイルで一覧表示されているインスタンスタイプのうち、IsTrunkingCompatible: true を含むものである必要があります。
- Pod のセキュリティポリシーを使用して Pod 変更へのアクセスを制限している場合、eks:vpc-resource-controller Kubernetes ユーザーは、psp が割り当てられた role に対して Kubernetes ClusterRoleBinding で指定する必要があります。デフォルトの Amazon EKS の psp、role および ClusterRoleBinding を使用している場合、これは eks:podsecuritypolicy:authenticated ClusterRoleBinding です。例えば、次の例に示すように、ユーザーを subjects: セクションに追加します。

```
[...]
subjects:
  - kind: Group
    apiGroup: rbac.authorization.k8s.io
    name: system:authenticated
  - apiGroup: rbac.authorization.k8s.io
    kind: User
    name: eks:vpc-resource-controller
  - kind: ServiceAccount
    name: eks-vpc-resource-controller
```

- カスタムネットワークと Pods のセキュリティグループを組み合わせて使用している場合、ENIConfig で指定されたセキュリティグループではなく、Pods のセキュリティグループによって指定されたセキュリティグループが使用されます。
- Amazon VPC CNI プラグインのバージョン 1.10.2 以前を使用していて、Pod 仕様に terminationGracePeriodSeconds 設定を含める場合は、設定の値を「0」にすることはできません。
- Amazon VPC CNI プラグインのバージョン 1.10 以前、またはデフォルト設定が 1.11=POD_SECURITY_GROUP_ENFORCING_MODE のバージョン strict を使用している場合、Kubernetes が NodePort に設定されたインスタンスターゲットを使用するタイプ LoadBalancer および externalTrafficPolicy の Local サービスは、セキュリティグループを割り当てる Pods ではサポートされていません。インスタンスターゲットでのロードバランサーの使用の詳細については、「Amazon EKS でのネットワーク負荷分散」を参照してください
- Amazon VPC CNI プラグインのバージョン 1.10 以降、またはデフォルト設定が POD_SECURITY_GROUP_ENFORCING_MODE=strict のバージョン 1.11 を使用している場合、セキュリティグループが割り当てられた Pods からのアウトバウンドトラフィックに対してソース NAT が無効になり、その結果アウトバウンドセキュリティグループルールが適用されます。インターネットにアクセスするには、セキュリティグループが割り当てられた Pods を、NAT ゲートウェイまたはインスタンスで構成されたプライベートサブネットにデプロイされたノードで起動する必要があります。パブリックサブネットにデプロイされたセキュリティグループが割り当てられた Pods は、インターネットにアクセスできません。

  POD_SECURITY_GROUP_ENFORCING_MODE=standard に設定されたプラグインのバージョン 1.11 以降を使用している場合、VPC の外部を送信先とする Pod トラフィックは、インスタンスのプライマリネットワークインターフェイスの IP アドレスに変換されます。このトラフィックには、Pod's のセキュリティグループ内のルールではなく、プライマリネットワークインターフェイスのセキュリティグループ内のルールが適用されます。
- セキュリティグループが関連付けられている Pods で Calico ネットワークポリシーを使用するには、Amazon VPC CNI プラグインのバージョン 1.11.0 以降を使用し、かつ POD_SECURITY_GROUP_ENFORCING_MODE=standard に設定することが必要になります。それ以外の場合、関連付けられたセキュリティグループを持つ Pods との間のトラフィックフローに、Calico ネットワークポリシー は適用されず、Amazon EC2 セキュリティグループの適用のみに限定されます。Amazon VPC CNI バージョンを更新するには、「Amazon VPC CNI plugin for Kubernetes Amazon EKS アドオンの使用」を参照してください
- Nodelocal DNSCache を使用するクラスターのセキュリティグループを使用する Amazon EC2 ノードで実行されている Pods は、Amazon VPC CNI プラグインのバージョンが 1.11.0 以降で POD_SECURITY_GROUP_ENFORCING_MODE=standard に設定されている場合にのみサポートさ

れます。Amazon VPC CNI プラグインのバージョンを更新するには、「Amazon VPC CNI plugin for Kubernetes Amazon EKS アドオンの使用」を参照してください

- Pods のセキュリティグループは、解約率が高い Pods の場合、Pod の起動レイテンシーより高くなる可能性があります。これは、リソースコントローラーのレート制限によるものです。

Pods のセキュリティグループに Amazon VPC CNI plugin for Kubernetes を設定します

Pods のセキュリティグループをデプロイするには

Fargate Pods のセキュリティグループのみを使用し、クラスターに Amazon EC2 ノードがない場合は、サンプルアプリケーションをデプロイする に進みます。

1. 次のコマンドで、現在の Amazon VPC CNI plugin for Kubernetes のバージョンを確認します。

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni:
 | cut -d : -f 3
```

出力例は次のとおりです。

```
v1.7.6
```

Amazon VPC CNI plugin for Kubernetes バージョンが 1.7.7 より前の場合は、プラグインを 1.7.7 以降に更新してください。詳細については、「Amazon VPC CNI plugin for Kubernetes Amazon EKS アドオンの使用」を参照してください。

2. AmazonEKSVPCResourceController マネージド IAM ポリシーを Amazon EKS クラスターに関連付けられているクラスターのロールに追加します。ポリシーにより、ロールはネットワークインターフェイス、プライベート IP アドレス、およびネットワークインスタンスへのアタッチとデタッチを管理できます。

   a. クラスター IAM ロールの名前を取得し、変数に格納します。*my-cluster* を自分のクラスター名に置き換えます。

```
cluster_role=$(aws eks describe-cluster --name my-cluster --query
 cluster.roleArn --output text | cut -d / -f 2)
```

   b. ロールへのポリシーの付与

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSVPCResourceController --role-name $cluster_role
```

3. Amazon VPC CNI アドオンを有効にして Pods のネットワークインターフェイスを管理するには、aws-node DaemonSet で ENABLE_POD_ENI 変数を true に設定します。この設定が true になると、クラスター内の各ノードについて、アドオンはCNinode カスタムリソースを作成します。VPC リソースコントローラーは、1 つの特別なネットワークインターフェイスを作成してアタッチします。これは、トランクネットワークインターフェイスと呼ばれ、説明は aws-k8s-trunk-eni です。

```
kubectl set env daemonset aws-node -n kube-system ENABLE_POD_ENI=true
```

> **Note**
>
> トランクネットワークインターフェイスは、インスタンスタイプでサポートされているネットワークインターフェイスの最大数に含まれます。各インスタンスタイプによりサポートされるネットワークインターフェイスの最大数のリストについては、「Amazon EC2 ユーザーガイド」の「各インスタンスタイプのネットワークインターフェイスあたりの IP アドレス数」を参照してください。ノードにすでに最大数の標準ネットワークインターフェイスがアタッチされている場合、VPC リソースコントローラーはスペースを予約します。コントローラーが標準ネットワークインターフェイスをデタッチして削除し、トランクネットワークインターフェイスを作成し、インスタンスにアタッチできるように、実行中の Pods をスケールダウンする必要があります。

4. CNINode カスタムリソースがどのノードにあるか、次のコマンドで確認できます。[No resources found] が返された場合、数秒待ってから、もう一度試してください。前のステップで、Amazon VPC CNI plugin for Kubernetes Pods を再起動する必要があります。再起動には数秒かかります。

```
$ kubectl get cninode -A
     NAME FEATURES
     ip-192-168-64-141.us-west-2.compute.internal
 [{"name":"SecurityGroupsForPods"}]
     ip-192-168-7-203.us-west-2.compute.internal [{"name":"SecurityGroupsForPods"}]
```

1.15 より古いバージョンの VPC CNI バージョンを使用している場合は、CNINode カスタムリソースの代わりにノードラベルが使用されていました。true に設定されているノード

ラベル aws-k8s-trunk-eni がどのノードにあるか、次のコマンドで確認できます。[No resources found] が返された場合、数秒待ってから、もう一度試してください。前のステップで、Amazon VPC CNI plugin for Kubernetes Pods を再起動する必要があります。再起動には数秒かかります。

```
kubectl get nodes -o wide -l vpc.amazonaws.com/has-trunk-attached=true
-
```

トランクネットワークインターフェイスが作成されると、Pods にはトランクネットワークインターフェイスまたは標準ネットワークインターフェイスのセカンダリ IP アドレスが割り当てられます。ノードが削除されると、トランクインターフェイスは自動的に削除されます。

後のステップで Pod のセキュリティグループをデプロイすると、VPC リソースコントローラーは、ブランチネットワークインターフェイスと呼ばれる特別なネットワークインターフェイスを aws-k8s-branch-eni の説明とともに作成し、セキュリティグループを関連付けます。ノードにアタッチされた標準ネットワークインターフェイスとトランクネットワークインターフェイスに加えて、ブランチネットワークインターフェイスが作成されます。

Liveness プローブまたは Readiness プローブを使用している場合は、TCP Early Demux も無効にする必要があります。これにより、kubelet は TCP を使用してブランチネットワークインターフェイス上の Pods に接続できます。TCP Early Demux を無効にするには、次のコマンドを実行します。

```
kubectl patch daemonset aws-node -n kube-system \
  -p '{"spec": {"template": {"spec": {"initContainers": [{"env":
[{"name":"DISABLE_TCP_EARLY_DEMUX","value":"true"}],"name":"aws-vpc-cni-
init"}]}}}}'
```

> **Note**
>
> Amazon VPC CNI plugin for Kubernetes アドオンのバージョン 1.11.0 以降を使用し、かつ POD_SECURITY_GROUP_ENFORCING_MODE=standard に設定している場合、次のステップで説明されているように、前のコマンドを実行する必要はありません。

5. クラスターで NodeLocal DNSCache が使用されている場合、独自のセキュリティグループを持つ Pods で Calico ネットワークポリシーを使用する場合、または割り当てたい Pods に対して externalTrafficPolicy を Local に設定したインスタンス ターゲットを使用するタイプ NodePort および LoadBalancer の Kubernetes サービスがある場合、セキュリティ グループ

を追加するには、バージョン 1.11.0 以降の Amazon VPC CNI plugin for Kubernetes アドオンを使用する必要があり、次の設定を有効にする必要があります。

```
kubectl set env daemonset aws-node -n kube-system
 POD_SECURITY_GROUP_ENFORCING_MODE=standard
```

> **Important**
>
> - Pod セキュリティグループルールは、kubelet や nodeLocalDNS など、同じノードにある Pods 間のトラフィックまたは Pods と services 間のトラフィックには適用されません。同じノードで異なるセキュリティグループを使用するポッドは、異なるサブネットに設定されているため通信できません。また、これらのサブネット間のルーティングは無効になっています。
> - Pods から VPC の外部のアドレスへのアウトバウンドトラフィックは、インスタンスのプライマリネットワークインターフェイスの IP アドレスに変換されたネットワークアドレスです (AWS_VPC_K8S_CNI_EXTERNALSNAT=true に設定していない場合)。このトラフィックには、Pod's のセキュリティグループ内のルールではなく、プライマリネットワークインターフェイスのセキュリティグループ内のルールが適用されます。
> - この設定を既存の Pods に適用するには、Pods または Pods が実行されているノードを再起動する必要があります。

サンプルアプリケーションをデプロイする

Pods にセキュリティグループを使用するには、既存のセキュリティグループがあり、かつ次の手順で説明するように、クラスターに Amazon EKS SecurityGroupPolicy をデプロイする必要があります。次のステップは、Pod に対してセキュリティグループポリシーを使用する方法を示しています。次のステップではターミナル間で保持されない変数が使用されるため、特に明記されていない限り、同じターミナルからすべてのステップを完了してください。

セキュリティグループでサンプル Pod をデプロイするには

1. リソースをデプロイする Kubernetes 名前空間を作成します。*my-namespace* は、使用する名前空間の名前に置き換えることができます。

```
kubectl create namespace my-namespace
```

2. Amazon EKS SecurityGroupPolicy をクラスターにデプロイします。

   a. 次のコンテンツをデバイスにコピーします。サービスアカウントラベルに基づいて Pods を選択したい場合は、*podSelector* を **serviceAccountSelector** で置き換えることができます。セレクターをどちらか 1 つ指定する必要があります。podSelector が空 (例: podSelector: {}) であると、名前空間内のすべての Pods が選択されます。*my-role* は自分のロール名に変更できます。serviceAccountSelector が空であると、名前空間内のすべてのサービスアカウントが選択されます。*my-security-group-policy* を自分の SecurityGroupPolicy の名前に置き換え、*my-namespace* は SecurityGroupPolicy を作成する名前空間に置き換えることができます。

   my_pod_security_group_id を既存のセキュリティグループの ID に置き換える必要があります。既存のセキュリティグループがない場合は、作成する必要があります。詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EC2 security groups for Linux instances」(Linux インスタンス用の Amazon EC2 セキュリティグループ) を参照してください。1 ~ 5 個のセキュリティグループ ID を指定できます。複数の ID を指定した場合、すべてのセキュリティグループ内のすべてのルールの組み合わせが、選択した Pods に対して有効になります。

```
cat >my-security-group-policy.yaml <<EOF
apiVersion: vpcresources.k8s.aws/v1beta1
kind: SecurityGroupPolicy
metadata:
  name: my-security-group-policy
  namespace: my-namespace
spec:
  podSelector:
    matchLabels:
      role: my-role
  securityGroups:
    groupIds:
      - my_pod_security_group_id
EOF
```

> **Important**
>
> Pod に対して指定した 1 つまたは複数のセキュリティグループは、次の基準を満たす必要があります。

- 存在している必要があります。セキュリティグループが存在しない場合、セレクターに一致する Pod をデプロイすると、Pod は作成プロセスでスタックしたままになります。Pod を記述すると、`An error occurred (InvalidSecurityGroupID.NotFound) when calling the CreateNetworkInterface operation: The securityGroup ID 'sg-05b1d815d1EXAMPLE' does not exist` のようなエラーメッセージが表示されます。
- プローブを設定した任意のポートポートを介して、ノードに適用されたセキュリティ グループ (kubelet の場合) からのインバウンド通信を許可する必要があります。
- TCP および UDP ポート 53 番経由のアウトバウンド通信を、CoreDNS を実行している Pods (または Pods が実行されるノード) に割り当てられたセキュリティグループへ許可する必要があります。CoreDNS Pods のセキュリティグループは、指定したセキュリティグループからのインバウンド TCP および UDP ポート 53 番トラフィックを許可する必要があります。
- セキュリティグループには、通信を行う必要がある他の Pods との通信に必要なインバウンドルールとアウトバウンドルールを含める必要があります。
- Fargate でセキュリティグループを使用している場合は、セキュリティグループに、Pods と Kubernetes コントロールプレーンとの通信を許可するルールを含める必要があります。最も簡単な方法は、クラスターセキュリティグループをセキュリティグループの 1 つとして指定することです。

セキュリティグループポリシーは、新しくスケジュールされた Pods にのみ適用されます。実行中の Pods には影響しません。

b. ポリシーをデプロイします。

```
kubectl apply -f my-security-group-policy.yaml
```

3. 前のステップで指定した *podSelector* の *my-role* の値に一致するラベルを持つサンプルアプリケーションをデプロイします。

a. 次のコンテンツをデバイスにコピーします。*#####*を独自のものに置き換えて、変更したコマンドを実行します。*my-role* を置き換える場合は、前のステップでセレクターに指定した値と同じであることを確認してください。

```
cat >sample-application.yaml <<EOF
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deployment
  namespace: my-namespace
  labels:
    app: my-app
spec:
  replicas: 4
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
        role: my-role
    spec:
      terminationGracePeriodSeconds: 120
      containers:
      - name: nginx
        image: public.ecr.aws/nginx/nginx:1.23
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: my-app
  namespace: my-namespace
  labels:
    app: my-app
spec:
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
EOF
```

b. 次のコマンドを使用して、アプリケーションをデプロイします。アプリケーションをデプロイすると、Amazon VPC CNI plugin for Kubernetes は `role` ラベルと一致し、前のステップで指定したセキュリティグループが Pod に適用されます。

```
kubectl apply -f sample-application.yaml
```

4. サンプルアプリケーションでデプロイされた Pods を表示します。このトピックの以降の説明では、このターミナルを `TerminalA` と呼びます。

```
kubectl get pods -n my-namespace -o wide
```

出力例は次のとおりです。

```
NAME                             READY   STATUS    RESTARTS   AGE     IP
       NODE                                            NOMINATED NODE   READINESS
 GATES
my-deployment-5df6f7687b-4fbjm   1/1     Running   0          7m51s   192.168.53.48
    ip-192-168-33-28.region-code.compute.internal   <none>           <none>
my-deployment-5df6f7687b-j9fl4   1/1     Running   0          7m51s
  192.168.70.145   ip-192-168-92-33.region-code.compute.internal   <none>
 <none>
my-deployment-5df6f7687b-rjxcz   1/1     Running   0          7m51s
  192.168.73.207   ip-192-168-92-33.region-code.compute.internal   <none>
 <none>
my-deployment-5df6f7687b-zmb42   1/1     Running   0          7m51s   192.168.63.27
    ip-192-168-33-28.region-code.compute.internal   <none>           <none>
```

> **Note**
>
> - `Waiting` 状態でスタックした Pods があれば、`kubectl describe pod my-deployment-xxxxxxxxxx-xxxxx -n my-namespace` を実行します。`Insufficient permissions: Unable to create Elastic Network Interface.` が表示された場合、前のステップで IAM クラスターロールに IAM ポリシーを追加したことを確認します。
> - Pods が `Pending` 状態でスタックした場合、ノードのインスタンスタイプが `limits.go` のリストに含まれていることを確認します。そのうえで、インスタンスタイプでサポートされるブランチネットワークインターフェイスの最大製品数とノードグループ内のノード数を乗算した数にまだ達していないことを確認します。例え

ば、m5.large インスタンスでは、9 つのブランチネットワークインターフェイスがサポートされています。ノードグループに 5 つのノードがある場合、ノードグループに対して最大 45 のブランチネットワークインターフェイスを作成できます。46 個目の Pod をデプロイしようとすると、セキュリティグループに関連付けられた別の Pod が削除されるまで Pending 状態のままになります。

kubectl describe pod ***my-deployment-xxxxxxxxxx-xxxxx*** **-n** ***my-namespace*** を実行したときに次のようなメッセージが表示されている場合は、無視しても問題ありません。このメッセージは、Amazon VPC CNI plugin for Kubernetes がホストネットワークの設定を試み、ネットワークインターフェイスの作成中に失敗したときに表示される場合があります。プラグインは、ネットワークインターフェイスが作成されるまで、このイベントをログに記録します。

```
Failed to create Pod sandbox: rpc error: code = Unknown desc = failed to set up
 sandbox container
 "e24268322e55c8185721f52df6493684f6c2c3bf4fd59c9c121fd4cdc894579f" network for Pod
 "my-deployment-5df6f7687b-4fbjm": networkPlugin
cni failed to set up Pod "my-deployment-5df6f7687b-4fbjm-c89wx_my-namespace"
 network: add cmd: failed to assign an IP address to container
```

インスタンスタイプで実行できる Pods の最大数を超えることはできません。各インスタンスタイプで実行できる Pods の最大数の一覧については、GitHub の「eni-max-pods.txt」を参照してください。セキュリティグループが関連付けられている Pod を削除するか、Pod が実行されているノードを削除すると、VPC リソースコントローラーによってブランチネットワークインターフェイスが削除されます。セキュリティグループ用に Pods を含むクラスターを Pods で削除する場合、コントローラーがブランチネットワークインターフェイスを削除しないため、自分で削除する必要があります。ネットワークインターフェイスの削除方法の詳細については、「Amazon EC2 ユーザーガイド」の「ネットワークインターフェイスの削除」を参照してください。

5. 別のターミナルから Pods のいずれかをシェルで操作します。このトピックの以降の説明では、このターミナルを TerminalB と呼びます。*5df6f7687b-4fbjm* を、前のステップの出力で返されたいずれかの Pods の ID で置き換えます。

```
kubectl exec -it -n my-namespace my-deployment-5df6f7687b-4fbjm -- /bin/bash
```

6. TerminalB のシェルから、サンプルアプリケーションが動作することを確認します。

```
curl my-app
```

出力例は次のとおりです。

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
[...]
```

アプリケーションを実行しているすべての Pods が、作成したセキュリティグループに関連付けられているため、出力を受信できます。そのグループには、セキュリティグループが関連付けられているすべての Pods 間のすべてのトラフィックを許可するルールが含まれています。DNS トラフィックは、そのセキュリティグループからノードに関連付けられているクラスターセキュリティグループへのアウトバウンドで許可されます。ノードは、Pods が名前のルックアップを実行した CoreDNS Podsを実行しています。

7. TerminalA で、クラスターセキュリティグループへの DNS 通信を許可するセキュリティグループルールを、セキュリティグループから削除します。前のステップで DNS ルールをクラスターセキュリティグループに追加しなかった場合は、*$my_cluster_security_group_id* をルールを作成したセキュリティグループの ID で置き換えます。

```
aws ec2 revoke-security-group-ingress --group-id $my_cluster_security_group_id --
security-group-rule-ids $my_tcp_rule_id
aws ec2 revoke-security-group-ingress --group-id $my_cluster_security_group_id --
security-group-rule-ids $my_udp_rule_id
```

8. TerminalB で、アプリケーションにもう一度アクセスしてみます。

```
curl my-app
```

出力例は次のとおりです。

```
curl: (6) Could not resolve host: my-app
```

クラスターセキュリティグループが関連付けられている CoreDNS Pods に Pod がアクセスできなくなったため、この試行は失敗します。クラスターセキュリティグループから、Pod に関連

付けられたセキュリティグループからの DNS 通信を許可するセキュリティグループルールがなくなりました。

前のステップで、いずれかの Pods に返された IP アドレスを使用してアプリケーションにアクセスしようとすると、セキュリティグループが関連付けられており、名前のルックアップが不要な Pods 間ですべてのポートが許可されているため、引き続き応答を受信します。

9. 試行し終えたら、作成したサンプルのセキュリティグループポリシー、アプリケーション、およびセキュリティグループを削除できます。TerminalA から、次のコマンドを実行します。

```
kubectl delete namespace my-namespace
aws ec2 revoke-security-group-ingress --group-id $my_pod_security_group_id --
security-group-rule-ids $my_inbound_self_rule_id
wait
sleep 45s
aws ec2 delete-security-group --group-id $my_pod_security_group_id
```

Pods 用の複数のネットワークインターフェイス

Multus CNI は Amazon EKS 用のコンテナネットワークインターフェイス (CNI) プラグインです。これにより、Pod に複数のネットワークインターフェイスをアタッチできるようになります。詳細については、「GitHub」の「Multus-CNI」を参照してください。

Amazon EKS では、各 Pod には、Amazon VPC CNI プラグインによって割り当てられたネットワークインターフェイスが 1 つあります。Multus を使用すると、複数のインターフェイスを備えたマルチホーム Pod を作成できます。これは、Multusが「meta-plugin」(他の複数の CNI プラグインを呼び出すことができる CNI プラグイン) として機能することによって達成されます。AWS では、Amazon VPC CNI プラグインをデフォルトのデリゲートプラグインとして設定することで、Multus をサポートしています。

考慮事項

- Amazon EKS では、シングルルートの I/O 仮想化 (SR-IOV)、およびデータプレーン開発キット (DPDK) のCNI プラグインを構築および公開することはありません。ただし、Multus で管理されたホストデバイスおよび ipvlan プラグインを介して、Amazon EC2 Elastic Network Adapters (ENA) と直接接続することで、パケットアクセラレーションを実現することができます。
- Amazon EKS では Multus をサポートすることで、追加の CNI プラグインの簡単な連鎖を可能にする汎用プロセスを提供します。AWS は、Multus とその連鎖プロセスをサポートしています。ただ

し、連鎖可能な互換性のある CNI プラグインすべてについてのサポートや、連鎖設定とは無関係な CNI プラグインで発生する可能性のある問題のサポートは行っていません。

- Amazon EKS は Multus プラグインのサポートとライフサイクル管理を提供していますが、追加のネットワークインターフェイスに関連付けられた IP アドレスや他の管理については責任を負いません。Amazon VPC CNI プラグインを使用する、デフォルトのネットワークインターフェイスの IP アドレスと管理に変更はありません。
- デフォルトのデリゲートプラグインとして正式にサポートされるのは、Amazon VPC CNI プラグインのみです。Amazon VPC CNI プラグインをプライマリネットワークに使用しないことを選択した場合は、公開されている Multus インストールマニフェストを変更して、デフォルトのデリゲートプラグインを代替の CNI に再設定する必要があります。
- Multus は、Amazon VPC CNI をプライマリ CNI として使用する場合にのみサポートされます。セカンダリなど、高次のインターフェイスに使用する場合、Amazon VPC CNI はサポートされません。
- Pods に割り当てられた追加のネットワークインターフェイスに対し、Amazon VPC CNI プラグインによる管理が行われないようにするには、ネットワークインターフェイスに次のタグを追加します。

  キー: `node.k8s.amazonaws.com/no_manage`

  値: `true`
- Multus はネットワークポリシーと互換性がありますが、Pods に接続された追加のネットワークインターフェイスの一部である可能性のあるポートと IP アドレスを含めるようにポリシーを強化する必要があります。

実装のチュートリアルについては、「GitHub」の「Multus セットアップガイド」を参照してください。

互換性のある代替 CNI プラグイン

Amazon VPC CNI plugin for Kubernetes は、Amazon EKS でサポートされている唯一の CNI プラグインです。Amazon EKS では、アップストリーム Kubernetes が実行されており、クラスター内の Amazon EC2 ノードに互換性のある CNI プラグインをインストールできます。クラスターに Fargate ノードがある場合、Amazon VPC CNI plugin for Kubernetes は既に Fargate ノードに存在します。これは Fargate ノードで使用できる唯一の CNI プラグインです。Fargate ノードに代替 CNI プラグインをインストールしようとすると失敗します。

Amazon EC2 ノードで代替 CNI プラグインを使用する予定であれば、当該プラグインの商用サポートを受けるか、社内のエキスパートによるトラブルシューティングを行い、解決策を CNI プラグインプロジェクトに提供することをお勧めします。

Amazon EKS は、互換性のある代替 CNI プラグインのサポートを提供するパートナーネットワークと連携しています。バージョンと認定、および実行されたテストの詳細については、次のパートナーのドキュメントを参照してください。

パートナー	製品	ドキュメント
Tigera	Calico	インストール手順
Isovalent	Cilium	インストール手順
ジュニパー	クラウドネイティブ Contrail Networking (CN2)	インストール手順
VMware	Antrea	インストール手順

Amazon EKS では、すべてのユースケースをカバーする幅広いオプションの提供を目指しています。

互換性のある代替ネットワークポリシープラグイン

Calico は、コンテナネットワークとセキュリティのために広く採用されているソリューションです。Calico on EKS を使用すると、EKS クラスターに完全準拠のネットワークポリシーが適用されます。さらに、基盤となる VPC の IP アドレスを節約する Calico のネットワークを使用することもできます。Calico Cloud は Calico Open Source の機能を強化するとともに、高度なセキュリティとオブザーバビリティ機能を提供します。

AWS Load Balancer Controller とは

AWS Load Balancer Controller によって Kubernetes クラスター向けの AWS Elastic Load Balancer が管理されます。コントローラーを使用すると、クラスターアプリケーションをインターネットに公開できます。コントローラーは、クラスターサービスまたは Ingress リソースを指す AWS ロードバランサーをプロビジョニングします。つまり、コントローラーはクラスター内の複数のポッドを指す単一の IP アドレスまたは DNS 名を作成します。

コントローラーは、Kubernetes Ingress リソースまたは Service リソースを監視します。これに応じて、適切な AWS Elastic Load Balancing リソースが作成されます。Kubernetes リソースに注釈を適用することで、ロードバランサーの特定の動作を設定できます。例えば、注釈を使用してロードバランサーに AWS セキュリティグループをアタッチできます。

コントローラは、以下のリソースをプロビジョニングします。

Kubernetes `Ingress`

Kubernetes `Ingress` を作成すると、LBC は AWS Application Load Balancer (ALB) を作成します。Ingress リソースに適用できる注釈を確認してください。

`LoadBalancer` タイプの Kubernetes サービス

`LoadBalancer` タイプの Kubernetes サービスを作成すると、LBC は AWS Network Load Balancer (NLB) を作成します。サービスリソースに適用できる注釈を確認してください。

以前は、インスタンスターゲットには Kubernetes Network Load Balancer が使用されていましたが、IP ターゲットには LBC が使用されていました。AWS Load Balancer Controller バージョン `2.3.0` 以降では、いずれかのターゲットタイプを使用して NLBs を作成できます。NLB ターゲットタイプの詳細については、Network Load Balancer のユーザーガイドの 「ターゲットタイプ」を参照してください。

コントローラーは GitHub で管理されるオープンソースプロジェクトです。

コントローラをデプロイする前に、Amazon EKS でのアプリケーション負荷分散 および Amazon EKS でのネットワーク負荷分散 についての前提条件と考慮事項を、確認しておくことをお勧めします。これらのトピックでは、AWS ロードバランサーを含むサンプルアプリケーションをデプロイしています。

Controller をインストールする

- 「the section called “Helm を使用してインストールする”」では、この手順を説明しています。Amazon EKS を初めて使用する場合は、これを使用してください。この手順では、Kubernetes のパッケージマネージャーである Helm と `eksctl` を使用して、LBC のインストールを簡素化します。
- または、「the section called “マニフェストを使用してインストールする”」の手順を使用します。これは、高度なクラスター設定に適しており、パブリックコンテナレジストリへのネットワークアクセスが制限されているクラスターなどが対象です。

非推奨のコントローラーバージョンから移行する

- 非推奨バージョンの AWS Load Balancer Controller がインストールされている場合は、「the section called “非推奨のコントローラーから移行する”」の手順を確認してください。
- 非推奨バージョンはアップグレードできません。このバージョンを削除し、AWS Load Balancer Controller の最新バージョンをインストールする必要があります。
- 非推奨バージョンには以下が含まれます。
  - AWS Load Balancer Controller の前身である AWS ALB Ingress Controller for Kubernetes (「Ingress Controller」)
  - AWS Load Balancer Controller の `0.1.x` バージョンすべて

レガシークラウドプロバイダー

Kubernetes には、AWS のレガシークラウドプロバイダーが含まれています。レガシークラウドプロバイダーは AWS Load Balancer Controller と同様に、AWS ロードバランサーをプロビジョニングできます。レガシークラウドプロバイダーは Classic Load Balancer を作成します。AWS Load Balancer Controller をインストールしない場合、Kubernetes はデフォルトでレガシークラウドプロバイダーを使用します。AWS Load Balancer Controller をインストールして、レガシークラウドプロバイダーを使用しないようにしてください。

> **Important**
>
> バージョン 2.5 以降では、AWS Load Balancer Controller は、`type: LoadBalancer` を持つ Kubernetes サービスリソースのデフォルトコントローラーとなり、サービスごとに AWS Network Load Balancer (NLB) を作成します。これは、サービスの変更ウェブフックを変化することで実行され、これにより `type: LoadBalancer` の新しいサービスの `spec.loadBalancerClass` フィールドが `service.k8s.aws/nlb` に設定されます。Helm チャートの値 `enableServiceMutatorWebhook` を `false` に設定すると、この機能をオフにして、レガシークラウドプロバイダーをデフォルトのコントローラーとして使用するように戻すことができます。この機能をオフにしない限り、クラスターはサービスに新しい Classic Load Balancer をプロビジョニングしません。既存の Classic Load Balancer は従来どおり機能します。

Helm を使用して AWS Load Balancer Controller をインストールする

このトピックでは、Kubernetes のパッケージマネージャーである Helm と `eksctl` を使用して AWS Load Balancer Controller をインストールする方法について説明します。コントローラーはデフォルトのオプションでインストールされます。注釈を使用した設定など、コントローラーの詳細については、GitHub の「AWS Load Balancer Controller ドキュメント」を参照してください。

以下のステップでは、 *`example values`* を独自の値に置き換えます。

前提条件

このチュートリアルを開始する前に、Amazon EKS クラスターの作成と管理に必要な次のツールとリソースを、インストールおよび設定しておく必要があります。

- 既存の Amazon EKS クラスター。デプロイするには、「Amazon EKS の使用開始」を参照してください。
- クラスター用の既存 AWS Identity and Access Management IAM OpenID Connect (OIDC) プロバイダー。既に存在しているかどうかを確認する、または作成するには「クラスターの IAM OIDC プロバイダーを作成する」を参照してください。
- Amazon VPC CNI plugin for Kubernetes、`kube-proxy` および CoreDNS アドオンが、サービスアカウントトークンに記載されている最小のバージョンであることを確認してください。
- AWS Elastic Load Balancing に関する知識。詳細については、Elastic Load Balancing ユーザーガイドを参照してください。
- Kubernetes サービスとイングレスリソースに関する知識。

- Helm はローカルにインストールされています。

ステップ 1: **`eksctl`** を使用して IAM ロールを作成する

> **Note**
>
> AWS Load Balancer Controller の IAM ロールを AWS アカウントごとに作成するだけで済みます。`AmazonEKSLoadBalancerControllerRole` が IAM コンソール に存在するかどうかを確認します。このロールが存在する場合は、「the section called “ステップ 2: AWS Load Balancer Controller をインストールする”」に進んでください。

IAM ポリシーを作成します。

1. ユーザーに代わって AWS API を呼び出すことを許可する、AWS Load Balancer Controller 用の IAM ポリシーをダウンロードします。

   AWS

   ```
   $ curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.7.2/docs/install/iam_policy.json
   ```

   AWS GovCloud (US)

   ```
   $ curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.7.2/docs/install/iam_policy_us-gov.json
   ```

   ```
   $ mv iam_policy_us-gov.json iam_policy.json
   ```

2. 前のステップでダウンロードしたポリシー を使用して、IAM ポリシーを作成します。

   ```
   $ aws iam create-policy \
       --policy-name AWSLoadBalancerControllerIAMPolicy \
       --policy-document file://iam_policy.json
   ```

   > **Note**
   >
   > AWS Management Console でポリシーを確認すると、コンソールには [ELB] サービスに関する警告が表示されますが、[ELB v2] サービスに関する警告は表示されません。これは、ポリシー内のアクションの一部が [ELB v2] には存在するが、[ELB] には存在しないために起こります。[ELB] に関する警告は無視できます。

`eksctl` を使用して IAM ロールを作成する

- *`my-cluster`* をクラスターの名前に置き換え、*`111122223333`* をアカウント ID に置き換えてから、コマンドを実行します。クラスターが AWS GovCloud (米国東部) または AWS GovCloud (米国東部) の AWS リージョン にある場合は、`arn:aws:` を `arn:aws-us-gov:` に置き換えます。

  ```
  $ eksctl create iamserviceaccount \
  ```

```
  --cluster=my-cluster \
  --namespace=kube-system \
  --name=aws-load-balancer-controller \
  --role-name AmazonEKSLoadBalancerControllerRole \
  --attach-policy-
arn=arn:aws:iam::111122223333:policy/AWSLoadBalancerControllerIAMPolicy \
  --approve
```

ステップ 2: AWS Load Balancer Controller をインストールする

Helm V3 を使用して AWS Load Balancer Controller をインストールする

1. eks-charts Helm チャートリポジトリを追加します。AWS はこのリポジトリを GitHub で管理しています。

```
$ helm repo add eks https://aws.github.io/eks-charts
```

2. ローカルリポジトリを更新して、最新のグラフがあることを確認します。

```
$ helm repo update eks
```

3. AWS Load Balancer Controller をインストールします。

   my-cluster を自分のクラスター名に置き換えます。次のコマンドでは、aws-load-balancer-controller は前のステップで作成した Kubernetes サービスアカウントです。

   Helm チャートの設定の詳細については、GitHub の values.yaml を参照してください。

```
$ helm install aws-load-balancer-controller eks/aws-load-balancer-controller \
  -n kube-system \
  --set clusterName=my-cluster \
  --set serviceAccount.create=false \
  --set serviceAccount.name=aws-load-balancer-controller
```

   a. Amazon EC2 インスタンスメタデータサービス (IMDS) に対するアクセスが制限されている Amazon EC2 ノードにコントローラーをデプロイする場合、または Fargate にデプロイする場合には、次の helm コマンドに次のフラグを追加します。

      - **--set region=*region-code***
      - **--set vpcId=*vpc-xxxxxxxx***

b. Helm チャートと Load Balancer Controller の利用可能なバージョンを表示するには、次のコマンドを使用します。

```
helm search repo eks/aws-load-balancer-controller --versions
```

> **Important**
>
> デプロイされたグラフは、セキュリティに関する更新を自動的に受信しません。この更新が利用可能になったら、手動で新しいグラフにアップグレードする必要があります。アップグレードする場合は、前のコマンドで *install* を **upgrade** に変更します。helm install コマンドでは、コントローラーのカスタムリソース定義 (CRDs) が自動的にインストールされます。一方、helm upgrade コマンドでは自動的にインストールされません。helm upgrade, コマンドを使用する場合は、CRDs を手動でインストールする必要があります。次のコマンドを実行して、CRDs をインストールします。
>
> ```
> wget https://raw.githubusercontent.com/aws/eks-charts/master/stable/aws-load-balancer-controller/crds/crds.yaml
> kubectl apply -f crds.yaml
> ```

ステップ 3: コントローラーがインストールされていることを確認する

1. コントローラがインストールされていることを確認します。

```
$ kubectl get deployment -n kube-system aws-load-balancer-controller
```

出力例は次のとおりです。

```
NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
aws-load-balancer-controller   2/2     2            2           84s
```

Helm を使用してデプロイした場合は、前の出力を受け取ります。Kubernetes マニフェストを使用してデプロイした場合、レプリカは 1 つのみとなります。

2. コントローラーを使用して AWS リソースをプロビジョニングする場合には、クラスターは特定の要件を満たしている必要があります。詳細については、Amazon EKS でのアプリケーション負荷分散およびAmazon EKS でのネットワーク負荷分散を参照してください。

Kubernetes マニフェストを使用して AWS Load Balancer Controller アドオンをインストールする

このトピックでは、Kubernetes マニフェストをダウンロードして適用し、コントローラーをインストールする方法について説明します。GitHub でコントローラーの完全な ドキュメント を表示できます。

以下のステップでは、 *example values* を独自の値に置き換えます。

前提条件

このチュートリアルを開始する前に、Amazon EKS クラスターの作成と管理に必要な次のツールとリソースを、インストールおよび設定しておく必要があります。

- 既存の Amazon EKS クラスター。デプロイするには、「Amazon EKS の使用開始」を参照してください。
- クラスター用の既存 AWS Identity and Access Management IAM OpenID Connect (OIDC) プロバイダー。既に存在しているかどうかを確認する、または作成するには「クラスターの IAM OIDC プロバイダーを作成する」を参照してください。
- Amazon VPC CNI plugin for Kubernetes、kube-proxy および CoreDNS アドオンが、サービスアカウントトークンに記載されている最小のバージョンであることを確認してください。
- AWS Elastic Load Balancing に関する知識。詳細については、Elastic Load Balancing ユーザーガイドを参照してください。
- Kubernetes サービスとイングレスリソースに関する知識。

ステップ 1: IAM を設定する

> **Note**
>
> AWS Load Balancer Controller の IAM ロールを AWS アカウントごとに作成するだけで済みます。AmazonEKSLoadBalancerControllerRole が IAM コンソール に存在するかどうかを確認します。このロールが存在する場合は、「the section called “ステップ 2: cert-manager をインストールする”」に進んでください。

IAM ポリシーを作成します。

1. ユーザーに代わって AWS API を呼び出すことを許可する、AWS Load Balancer Controller 用の IAM ポリシーをダウンロードします。

   AWS

   ```
   $ curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.7.2/docs/install/iam_policy.json
   ```

   AWS GovCloud (US)

   ```
   $ curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.7.2/docs/install/iam_policy_us-gov.json
   ```

   ```
   $ mv iam_policy_us-gov.json iam_policy.json
   ```

2. 前のステップでダウンロードしたポリシー を使用して、IAM ポリシーを作成します。

   ```
   $ aws iam create-policy \
       --policy-name AWSLoadBalancerControllerIAMPolicy \
       --policy-document file://iam_policy.json
   ```

   > **Note**
   >
   > AWS Management Console でポリシーを確認すると、コンソールには [ELB] サービスに関する警告が表示されますが、[ELB v2] サービスに関する警告は表示されません。これは、ポリシー内のアクションの一部が [ELB v2] には存在するが、[ELB] には存在しないために起こります。[ELB] に関する警告は無視できます。

eksctl

`eksctl` を使用して IAM ロールを作成する

- *`my-cluster`* をクラスターの名前に置き換え、*`111122223333`* をアカウント ID に置き換えてから、コマンドを実行します。クラスターが AWS GovCloud (米国東部) または AWS GovCloud (米国東部) の AWS リージョン にある場合は、`arn:aws:` を `arn:aws-us-gov:` に置き換えます。

```
$ eksctl create iamserviceaccount \
  --cluster=my-cluster \
  --namespace=kube-system \
  --name=aws-load-balancer-controller \
  --role-name AmazonEKSLoadBalancerControllerRole \
  --attach-policy-
arn=arn:aws:iam::111122223333:policy/AWSLoadBalancerControllerIAMPolicy \
  --approve
```

AWS CLI and kubectl

AWS CLI と **kubectl** を使用して IAM ロールを作成する

1. クラスターの OIDC プロバイダー ID を取得し、変数に格納します。

```
oidc_id=$(aws eks describe-cluster --name my-cluster --query
 "cluster.identity.oidc.issuer" --output text | cut -d '/' -f 5)
```

2. クラスターの ID を持つ IAM OIDC プロバイダーが既にアカウントにあるかどうかを確認します。クラスターと IAM の両方に OIDC を設定する必要があります。

```
aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
```

出力が返された場合は、クラスター用の IAM OIDC プロバイダーがすでに存在します。出力が返されない場合は、クラスター用の IAM OIDC プロバイダーを作成する必要があります。詳細については、「クラスターの IAM OIDC プロバイダーを作成する」を参照してください。

3. 次のコンテンツをデバイスにコピーします。*111122223333* をアカウントID に置き換えます。*region-code* をクラスターのある AWS リージョン に置き換えます。*EXAMPLED539D4633E53DE1B71EXAMPLE* を、前のステップで返された出力に置き換えます。クラスターが AWS GovCloud (米国東部) または AWS GovCloud (米国東部) の AWS リージョン にある場合は、`arn:aws:` を `arn:aws-us-gov:` に置き換えます。テキストを置き換えたら、変更したコマンドを実行して `load-balancer-role-trust-policy.json` ファイルを作成します。

```
cat >load-balancer-role-trust-policy.json <<EOF
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::111122223333:oidc-provider/
oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
                    "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-
system:aws-load-balancer-controller"
                }
            }
        }
    ]
}
EOF
```

4. IAM ロールを作成します。

```
aws iam create-role \
  --role-name AmazonEKSLoadBalancerControllerRole \
  --assume-role-policy-document file://"load-balancer-role-trust-policy.json"
```

5. IAM ロールに、必要な Amazon EKS 管理の IAM ポリシーをアタッチします。*111122223333* をアカウントID に置き換えます。

```
aws iam attach-role-policy \
  --policy-arn
 arn:aws:iam::111122223333:policy/AWSLoadBalancerControllerIAMPolicy \
  --role-name AmazonEKSLoadBalancerControllerRole
```

6. 次のコンテンツをデバイスにコピーします。*111122223333* をアカウントID に置き換えます。クラスターが AWS GovCloud (米国東部) または AWS GovCloud (米国東部) の AWS リージョン にある場合は、`arn:aws:` を `arn:aws-us-gov:` に置き換えます。テキストを置き換えたら、変更したコマンドを実行して `aws-load-balancer-controller-service-account.yaml` ファイルを作成します。

```
cat >aws-load-balancer-controller-service-account.yaml <<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/component: controller
    app.kubernetes.io/name: aws-load-balancer-controller
  name: aws-load-balancer-controller
  namespace: kube-system
  annotations:
    eks.amazonaws.com/role-arn:
 arn:aws:iam::111122223333:role/AmazonEKSLoadBalancerControllerRole
EOF
```

7. クラスター上で Kubernetes サービスアカウントを作成します。aws-load-balancer-controller という名前の Kubernetes サービスアカウントは、*AmazonEKSLoadBalancerControllerRole* という名前で作成した IAM ロールで注釈が付けられています。

```
$ kubectl apply -f aws-load-balancer-controller-service-account.yaml
```

ステップ 2: **cert-manager** をインストールする

以下のいずれかの方法で cert-manager をインストールして、Webhook に証明書設定を導入します。詳細については、「cert-manager ドキュメント」の「開始方法」を参照してください。

cert-manager をインストールするには、quay.io コンテナレジストリを使用することをお勧めします。ノードが quay.io コンテナレジストリにアクセスできない場合は、Amazon ECR を使用して cert-manager をインストールします (以下を参照)。

Quay.io

Quay.io を使用して **cert-manager** をインストールする

- ノードが quay.io コンテナレジストリにアクセスできる場合は、cert-manager をインストールして、Webhook に証明書設定を挿入します。

```
$ kubectl apply \
    --validate=false \
```

```
    -f https://github.com/jetstack/cert-manager/releases/download/v1.13.5/cert-
manager.yaml
```

Amazon ECR

Amazon ECR を使用して **cert-manager** をインストールする

1. 以下のいずれかの方法で cert-manager をインストールして、Webhook に証明書設定を導入します。詳細については、「cert-manager ドキュメント」の「開始方法」を参照してください。
2. マニフェストをダウンロードします。

```
curl -Lo cert-manager.yaml https://github.com/jetstack/cert-manager/releases/
download/v1.13.5/cert-manager.yaml
```

3. 次のイメージをプルして、ノードがアクセスできるリポジトリにプッシュします。イメージをプルし、タグ付けして独自のリポジトリにプッシュする方法の詳細については、あるリポジトリから別のリポジトリにコンテナイメージをコピーする を参照してください。

```
quay.io/jetstack/cert-manager-cainjector:v1.13.5
quay.io/jetstack/cert-manager-controller:v1.13.5
quay.io/jetstack/cert-manager-webhook:v1.13.5
```

4. 三つのイメージのマニフェストの quay.io を、独自のレジストリ名に置き換えます。次のコマンドは、プライベートリポジトリの名前がソースリポジトリと同じであることを前提としています。*111122223333.dkr.ecr.region-code.amazonaws.com* をプライベートレジストリに置き換えます。

```
$ sed -i.bak -e 's|quay.io|111122223333.dkr.ecr.region-code.amazonaws.com|' ./
cert-manager.yaml
```

5. マニフェストを適用します。

```
$ kubectl apply \
    --validate=false \
    -f ./cert-manager.yaml
```

ステップ 3: AWS Load Balancer Controller をインストールする

Kubernetes マニフェストを使用して AWS Load Balancer Controller をインストールする

1. Controller の詳細をダウンロードします。コントローラーの詳細については、「GitHub」の ドキュメント を参照してください。

```
curl -Lo v2_7_2_full.yaml https://github.com/kubernetes-sigs/aws-load-balancer-controller/releases/download/v2.7.2/v2_7_2_full.yaml
```

2. ファイルに以下の編集を行います。

   a. v2_7_2_full.yaml ファイルをダウンロードした場合は、次のコマンドを実行してマニフェストの ServiceAccount セクションを削除します。このセクションを削除しないと、前のステップでサービスアカウントに追加した必須の注釈が上書きされます。コントローラーを削除した場合、このセクションの削除により、前のステップで作成したサービスアカウントも保持されます。

```
$ sed -i.bak -e '596,604d' ./v2_7_2_full.yaml
```

   別のバージョンのファイルをダウンロードした場合は、エディタでファイルを開き、次の行を削除します。

```
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/component: controller
    app.kubernetes.io/name: aws-load-balancer-controller
  name: aws-load-balancer-controller
  namespace: kube-system
---
```

   b. *my-cluster* をユーザーのクラスター名で置き換えて、ファイルの Deployment spec セクションの your-cluster-name をクラスター名に置き換えます。

```
$ sed -i.bak -e 's|your-cluster-name|my-cluster|' ./v2_7_2_full.yaml
```

   c. ノードが Amazon EKS Amazon ECR イメージリポジトリにアクセスできない場合は、次のイメージをプルして、ノードがアクセスできるリポジトリにプッシュする必要がありま

す。イメージをプルし、タグ付けして独自のリポジトリにプッシュする方法の詳細については、あるリポジトリから別のリポジトリにコンテナイメージをコピーする を参照してください。

```
public.ecr.aws/eks/aws-load-balancer-controller:v2.7.2
```

マニフェストにレジストリの名前を追加します。次のコマンドは、プライベートリポジトリの名前がソースリポジトリと同じであると仮定し、プライベートレジストリの名前をファイルに追加します。*111122223333.dkr.ecr.region-code.amazonaws.com* は、実際のレジストリに置き換えます。この行は、プライベートリポジトリにソースリポジトリと同じ名前を付けたことを前提としています。そうでない場合は、プライベートレジストリ名の後の `eks/aws-load-balancer-controller` テキストを、リポジトリ名に変更します。

```
$ sed -i.bak -e 's|public.ecr.aws/eks/aws-load-balancer-
controller|111122223333.dkr.ecr.region-code.amazonaws.com/eks/aws-load-
balancer-controller|' ./v2_7_2_full.yaml
```

d. (Fargate または制限付き IMDS の場合にのみ必須)

Amazon EC2 インスタンスメタデータサービス (IMDS) に対するアクセスが制限されている Amazon EC2 ノードにコントローラをデプロイする場合、または Fargate にデプロイする場合には、**following parameters** の下に `- args:` を追加します。

```
[...]
spec:
      containers:
        - args:
            - --cluster-name=your-cluster-name
            - --ingress-class=alb
            - --aws-vpc-id=vpc-xxxxxxxx
            - --aws-region=region-code


[...]
```

3. ファイルを適用します。

```
$ kubectl apply -f v2_7_2_full.yaml
```

4. IngressClass および IngressClassParams マニフェストをクラスターにダウンロードします。

```
$ curl -Lo v2_7_2_ingclass.yaml https://github.com/kubernetes-sigs/aws-load-balancer-controller/releases/download/v2.7.2/v2_7_2_ingclass.yaml
```

5. マニフェストをクラスターに適用します。

```
$ kubectl apply -f v2_7_2_ingclass.yaml
```

ステップ 4: コントローラーがインストールされていることを確認する

1. コントローラがインストールされていることを確認します。

```
$ kubectl get deployment -n kube-system aws-load-balancer-controller
```

出力例は次のとおりです。

```
NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
aws-load-balancer-controller   2/2     2            2           84s
```

Helm を使用してデプロイした場合は、前の出力を受け取ります。Kubernetes マニフェストを使用してデプロイした場合、レプリカは 1 つのみとなります。

2. コントローラーを使用して AWS リソースをプロビジョニングする場合には、クラスターは特定の要件を満たしている必要があります。詳細については、Amazon EKS でのアプリケーション負荷分散およびAmazon EKS でのネットワーク負荷分散を参照してください。

非推奨のコントローラーから移行する

このトピックでは、非推奨のコントローラーバージョンから移行する方法を説明します。具体的には、AWS Load Balancer Controller の非推奨バージョンを削除する方法について説明します。

- 非推奨バージョンはアップグレードできません。このバージョンを削除し、LBC の最新バージョンをインストールする必要があります。
- 非推奨バージョンには以下が含まれます。
  - AWS Load Balancer Controller の前身である AWS ALB Ingress Controller for Kubernetes (「Ingress Controller」)

- AWS Load Balancer Controller の `0.1.x` バージョンすべて

非推奨のコントローラーバージョンを削除する

> **Note**
>
> 非推奨バージョンのインストールは、Helm を使用して行われた、あるいは Kubernetes マニフェストを使用して手動で行われた可能性があります。この手順は、元々インストールしてあるツールを使用して実行します。

Helm を使用して Ingress Controller を削除する

1. `incubator/aws-alb-ingress-controller` Helm チャートをインストールしてある場合は、これをアンインストールします。

   ```
   $ helm delete aws-alb-ingress-controller -n kube-system
   ```

2. `eks-charts/aws-load-balancer-controller` のバージョン `0.1.x` をインストールしている場合は、これをアンインストールします。`0.1.x` からバージョン `1.0.0` へのアップグレードは、Webhook API のバージョンとの互換性がないため動作しません。

   ```
   $ helm delete aws-load-balancer-controller -n kube-system
   ```

Kubernetes マニフェストを使用して Ingress Controller を削除する

1. Ingress Controller がインストール済みであるかどうかを確認します。

   ```
   $ kubectl get deployment -n kube-system alb-ingress-controller
   ```

   これは、コントローラが取り付けられていない場合の出力です。

   サーバーからのエラー (NotFound): deployments.apps「alb-ingress-controller」が見つかりません

   これは、コントローラが取り付けられている場合の出力です。

   ```
   NAME                     READY UP-TO-DATE AVAILABLE AGE
   ```

```
alb-ingress-controller 1/1   1          1         122d
```

2. 次のコマンドを入力してコントローラを削除します。

```
$ kubectl delete -f https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-
ingress-controller/v1.1.8/docs/examples/alb-ingress-controller.yaml
kubectl delete -f https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-
ingress-controller/v1.1.8/docs/examples/rbac-role.yaml
```

AWS Load Balancer Controller に移行する

ALB Ingress Controller for Kubernetes から AWS Load Balancer Controller に移行するには、以下を実行する必要があります。

1. ALB Ingress Controller を削除します (上記を参照)。
2. AWS Load Balancer Controller をインストールします。
3. LBC で使用される IAM ロールにポリシーを追加します。このポリシーは、ALB Ingress Controller for Kubernetes によって作成されたリソースを LBC が管理することを許可します。

AWS Load Balancer Controller の IAM ロールに移行ポリシーを追加する

1. IAM ポリシーをダウンロードします。このポリシーは、ALB Ingress Controller for Kubernetes によって作成されたリソースを LBC が管理することを許可します。ポリシーを表示することもできます。

```
$ curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
controller/v2.7.2/docs/install/iam_policy_v1_to_v2_additional.json
```

2. クラスターが AWS GovCloud (米国東部) または AWS GovCloud (米国東部) の AWS リージョンにある場合は、`arn:aws:` を `arn:aws-us-gov:` に置き換えます。。

```
$ sed -i.bak -e 's|arn:aws:|arn:aws-us-gov:|' iam_policy_v1_to_v2_additional.json
```

3. IAM ポリシーを作成し、返された ARN を書き留めます。

```
$ aws iam create-policy \
  --policy-name AWSLoadBalancerControllerAdditionalIAMPolicy \
  --policy-document file://iam_policy_v1_to_v2_additional.json
```

4. LBC で使用される IAM ロールに IAM ポリシーをアタッチします。*your-role-name* を、AmazonEKSLoadBalancerControllerRole などのロールの名前に置き換えます。

   eksctl を使用してロールを作成している場合、作成されたロール名を見つけるには、AWS CloudFormation コンソールを開き、eksctl-***my-cluster***-addon-iamserviceaccount-kube-system-aws-load-balancer-controller スタックを選択します。[Resources (リソース)] タブを選択します。ロール名は、[Physical ID (物理 ID)] 列で見つかります。クラスターが AWS GovCloud (米国東部) または AWS GovCloud (米国東部) の AWS リージョン にある場合は、arn:aws: を arn:aws-us-gov: に置き換えます。

   ```
   $ aws iam attach-role-policy \
     --role-name your-role-name \
     --policy-arn
    arn:aws:iam::111122223333:policy/AWSLoadBalancerControllerAdditionalIAMPolicy
   ```

CoreDNS Amazon EKS アドオンの使用

CoreDNS は、Kubernetes クラスター DNS として機能できる柔軟で拡張可能な DNS サーバーです。1 つ以上のノードで Amazon EKS クラスターを起動すると、クラスターにデプロイされたノード数に関係なく、デフォルトで CoreDNS イメージの 2 つのレプリカがデプロイされます。CoreDNS の Pods は、クラスター内のすべての Pods の名前解決を行います。クラスターに AWS Fargate プロファイル が含まれ、名前空間が CoreDNS deployment の名前空間と一致している場合、CoreDNS Pods を Fargate ノードにデプロイできます。CoreDNS の詳細については、「Kubernetes ドキュメント」の「サービスディスカバリーに CoreDNS を使用する」を参照してください。

次の表は、各 Kubernetes バージョンの Amazon EKS アドオンタイプの利用可能な最新バージョンを示しています。

Kubernetes バージョン	1.30	1.29	1.28	1.27	1.26	1.25	1.24	1.23
	v1.11.1-eksbuild.1	v1.11.1-eksbuild.1	v1.10.1-eksbuild.1	v1.10.1-eksbuild.1	v1.9.3-eksbuild.1	v1.9.3-eksbuild.1	v1.9.3-eksbuild.1	v1.8.7-eksbuild.10

> **Important**
>
> このアドオンを自己管理している場合、表のバージョンは、利用可能なセルフマネージドバージョンと同じではない可能性があります。このアドオンのセルフマネージドタイプの更新の詳細については、「セルフマネージド型アドオンを更新する」を参照してください。

CoreDNS アップグレードに関する重要な考慮事項

- CoreDNS Deployment の安定性と可用性を向上させるために、PodDisruptionBudget を使用してバージョン v1.9.3-eksbuild.6 以降および v1.10.1-eksbuild.3 がデプロイされます。既存の PodDisruptionBudget をデプロイした場合、これらのバージョンへのアップグレードは失敗する可能性があります。アップグレードが失敗した場合は、次のいずれかのタスクを実行することで問題が解決します。
  - Amazon EKS アドオンをアップグレードする際は、競合解決のオプションとして既存の設定を上書きすることを選択してください。Deployment に他のカスタム設定を行った場合は、アップグレード後に他のカスタム設定を再適用できるように、アップグレード前に必ず設定をバックアップしてください。
  - 既存の PodDisruptionBudget を削除して、アップグレードを再試行してください。
- EKS アドオンバージョン v1.9.3-eksbuild.3 以降および v1.10.1-eksbuild.6 以降では、CoreDNS Deployment は readinessProbe を /ready エンドポイントを使用するように設定します。このエンドポイントは CoreDNS の Corefile 設定ファイルで有効になっています。

  カスタム Corefile を使用する場合は、ready プラグインを設定に追加して、プローブが使用できるように /ready エンドポイントを CoreDNS でアクティブにする必要があります。
- EKS アドオンバージョン v1.9.3-eksbuild.7 以降および v1.10.1-eksbuild.4 以降では、PodDisruptionBudget を変更できます。次の例では、フィールドを使用して、アドオンを編集したり、[任意の設定項目]でこれらの設定を変更したりできます。この例はデフォルトの PodDisruptionBudget を示しています。

```
{
    "podDisruptionBudget": {
        "enabled": true,
        "maxUnavailable": 1
        }
}
```

maxUnavailable または minAvailable を設定できますが、両方を単一の PodDisruptionBudget で設定することはできません。PodDisruptionBudgets の詳細については、「Kubernetes ドキュメント」の「PodDisruptionBudget の指定」を参照してください。

enabled を false に設定しても、PodDisruptionBudget は削除されないことに注意してください。このフィールドを false に設定後、PodDisruptionBudget オブジェクトを削除する必要があります。同様に、PodDisruptionBudget のあるバージョンにアップグレードした後、古いバージョンのアドオンを使用するようにアドオンを編集した場合 (アドオンをダウングレード)、PodDisruptionBudget は削除されません。次のコマンドを実行して、PodDisruptionBudget を削除します。

```
kubectl delete poddisruptionbudget coredns -n kube-system
```

- EKS アドオンバージョン v1.10.1-eksbuild.5 以降では、デフォルトの許容値を KEP 2067 に準拠するように node-role.kubernetes.io/master:NoSchedule から node-role.kubernetes.io/control-plane:NoSchedule に変更してください。KEP 2067 の詳細については、GitHub の Kubernetes 拡張プロポーザル (KEP) の「KEP-2067: kubeadm の「マスター」ラベルとテイントの名前を変更する」を参照してください。

  EKS アドオンバージョン v1.8.7-eksbuild.8 以降、および v1.9.3-eksbuild.9 以降では、両方の許容範囲がすべての Kubernetes バージョンと互換性があるように設定されています。

- EKS アドオンバージョン v1.9.3-eksbuild.11 および v1.10.1-eksbuild.7 以降では、CoreDNS Deployment は topologySpreadConstraints のデフォルト値を設定します。デフォルト値により、複数のアベイラビリティーゾーンに使用可能なノードがある場合に、CoreDNS Pods がアベイラビリティーゾーン全体に分散されます。デフォルト値の代わりに使用するカスタム値を設定できます。デフォルト値は次のとおりです。

```
topologySpreadConstraints:
  - maxSkew: 1
    topologyKey: topology.kubernetes.io/zone
    whenUnsatisfiable: ScheduleAnyway
    labelSelector:
      matchLabels:
        k8s-app: kube-dns
```

CoreDNS v**1.11** アップグレードに関する考慮事項

- EKS アドオン バージョン `v1.11.1-eksbuild.4` 以降では、コンテナイメージは、Amazon EKS Distro によって維持される最小ベースイメージに基づいています。これには最小限のパッケージが含まれ、シェルはありません。詳細については、「Amazon EKS Distro」を参照してください。CoreDNS イメージの使用方法とトラブルシューティングは変わりません。

Amazon EKS アドオンの作成

Amazon EKS タイプのアドオンを作成します。チェック

前提条件

- 既存の Amazon EKS クラスター。デプロイするには、「Amazon EKS の使用開始」を参照してください。

1. クラスターにインストールされているアドオンのバージョンを確認します。

   ```
   kubectl describe deployment coredns --namespace kube-system | grep coredns: | cut -d : -f 3
   ```

   出力例は次のとおりです。

   ```
   v1.10.1-eksbuild.11
   ```

2. クラスターにインストールされているアドオンのタイプを確認します。クラスターを作成するために使用したツールによっては、現在クラスターに Amazon EKS アドオンタイプがインストールされていない場合があります。*my-cluster* の部分は、自分のクラスター名に置き換えます。

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query addon.addonVersion --output text
   ```

   バージョン番号が返された場合、Amazon EKS タイプのアドオンがクラスターにインストールされているため、このステップの残りのステップを完了する必要はありません。エラーが返された場合、クラスターに Amazon EKS タイプのアドオンがインストールされていません。インストールするには、この手順の残りのステップを完了します。

3. 現在インストールされているアドオンの設定を保存します。

```
kubectl get deployment coredns -n kube-system -o yaml > aws-k8s-coredns-old.yaml
```

4. AWS CLI を使用してアドオンを作成します。AWS Management Console または `eksctl` を使用してアドオンを作成する場合は、「アドオンの作成」を参照して、アドオン名の `coredns` を指定します。デバイスに沿ったコマンドをコピーします。必要に応じてコマンドに次の変更を加え、変更したコマンドを実行します。

   - *my-cluster* を自分のクラスター名に置き換えます。
   - *v1.11.1-eksbuild.9* を、クラスターバージョンの最新バージョンの表に表示されている最新バージョンに置き換えます。

   ```
   aws eks create-addon --cluster-name my-cluster --addon-name coredns --addon-
   version v1.11.1-eksbuild.9
   ```

   Amazon EKS アドオンのデフォルト設定と競合するカスタム設定を現在のアドオンに適用した場合、作成が失敗する可能性があります。作成が失敗した場合、エラーが表示されます。これは、問題の解決に役立てることができます。または、前のコマンドに **--resolve-conflicts OVERWRITE** を追加することもできます。これにより、アドオンは既存のカスタム設定を上書きできます。アドオンを作成したら、カスタム設定で更新できます。

5. クラスターの Kubernetes バージョン用のアドオンの最新バージョンがクラスターに追加されたことを確認します。*my-cluster* を自分のクラスター名に置き換えます。

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
    addon.addonVersion --output text
   ```

   アドオンの作成が完了するまでに数秒かかる場合があります。

   出力例は次のとおりです。

   ```
   v1.11.1-eksbuild.9
   ```

6. 元のアドオンに対してカスタム設定を行った場合は、Amazon EKS アドオンを作成する前に、前のステップで保存した設定を使用して、カスタム設定で Amazon EKS アドオンを更新します。

Amazon EKS アドオンの更新

Amazon EKS タイプのアドオンを更新します。Amazon EKS タイプのアドオンをクラスターに追加していない場合は、この手順を完了する代わりに、アドオンを追加するか、「セルフマネージド型アドオンを更新する」を参照してください。

1. クラスターにインストールされているアドオンのバージョンを確認します。*my-cluster* をクラスター名に置き換えます。

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
 "addon.addonVersion" --output text
```

出力例は次のとおりです。

```
v1.10.1-eksbuild.11
```

返されたバージョンが、最新バージョンの表にあるクラスターの Kubernetes バージョンのバージョンと同じである場合は、既に最新バージョンがクラスターにインストールされているため、この手順の残りを完了する必要はありません。出力にバージョン番号ではなくエラーが表示される場合は、Amazon EKS タイプのアドオンがクラスターにインストールされていません。この手順でアドオンを更新する前に、アドオンを作成する必要があります。

2. 現在インストールされているアドオンの設定を保存します。

```
kubectl get deployment coredns -n kube-system -o yaml > aws-k8s-coredns-old.yaml
```

3. AWS CLI を使用してアドオンを更新します。AWS Management Console または `eksctl` を使用してアドオンを更新する場合は、「アドオンの更新」を参照してください。デバイスに沿ったコマンドをコピーします。必要に応じてコマンドに次の変更を加え、変更したコマンドを実行します。

   - *my-cluster* を自分のクラスター名に置き換えます。
   - *v1.11.1-eksbuild.9* を、クラスターバージョンの最新バージョンの表に表示されている最新バージョンに置き換えます。
   - **`--resolve-conflicts`** *PRESERVE* オプションはアドオンの既存の設定値を保存します。アドオン設定にカスタム値を設定していて、このオプションを使用しない場合、Amazon EKS は値をデフォルト値で上書きします。このオプションを使用する場合、実稼働クラスターのアドオンを更新する前に、非稼動クラスターのフィールドおよび値変更をテストすることをお勧めします。この値を `OVERWRITE` に変更する場合、すべての設定が Amazon EKS のデフォル

ト値に変更されます。いずれかの設定にカスタム値を設定した場合、Amazon EKS のデフォルト値で上書きされる可能性があります。この値を none に変更した場合、Amazon EKS は設定の値を一切変更しませんが、更新が失敗する可能性があります。更新に失敗した場合、競合の解決に役立つエラーメッセージが返されます。

- 構成設定を更新しない場合は、コマンドから **--configuration-values '{*"replicaCount":3*}'** を削除します。構成設定を更新する場合は、*#ReplicaCount#:3* を設定したい設定に置き換えてください。この例では、CoreDNS3のレプリカの数はに設定されています。指定する値は、設定スキーマに対して有効である必要があります。設定スキーマがわからない場合は、**aws eks describe-addon-configuration --addon-name coredns --addon-version *v1.11.1-eksbuild.9*** を実行します。その際に、*v1.11.1-eksbuild.9* は、設定を表示する対象のアドオンのバージョン番号に置き換えてください。出力でスキーマが返されます。既存のカスタム設定があり、それをすべて削除してすべての設定の値を Amazon EKS のデフォルトに戻したい場合は、コマンドから *"replicaCount":3* を削除して、**{}** が空になるようにします。CoreDNS設定の詳細については、Kubernetesドキュメントの「DNS サービスのカスタマイズ」を参照してください。

```
aws eks update-addon --cluster-name my-cluster --addon-name coredns --addon-
version v1.11.1-eksbuild.9 \
    --resolve-conflicts PRESERVE --configuration-values '{"replicaCount":3}'
```

更新が完了するまでに数秒かかる場合があります。

4. アドオンのバージョンが更新されたことを確認します。*my-cluster* を自分のクラスター名に置き換えます。

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns
```

更新が完了するまでに数秒かかる場合があります。

出力例は次のとおりです。

```
{
    "addon": {
        "addonName": "coredns",
        "clusterName": "my-cluster",
        "status": "ACTIVE",
        "addonVersion": "v1.11.1-eksbuild.9",
        "health": {
            "issues": []
```

```
        },
        "addonArn": "arn:aws:eks:region:111122223333:addon/my-cluster/coredns/
d2c34f06-1111-2222-1eb0-24f64ce37fa4",
        "createdAt": "2023-03-01T16:41:32.442000+00:00",
        "modifiedAt": "2023-03-01T18:16:54.332000+00:00",
        "tags": {},
        "configurationValues": "{\"replicaCount\":3}"
    }
}
```

セルフマネージド型アドオンを更新する

> **Important**
>
> セルフマネージド型のアドオンを使用する代わりに、Amazon EKS タイプのアドオンをクラスターに追加することをお勧めします。タイプの違いがよくわからない場合は、「the section called “Amazon EKS アドオン”」を参照してください。Amazon EKS アドオンをクラスターに追加する方法については、「the section called “アドオンの作成”」を参照してください。Amazon EKS アドオンを使用できない場合は、その理由に関する問題をコンテナロードマップの GitHub リポジトリに送信することをお勧めします。

1. クラスターにインストールされているアドオンがセルフマネージド型であることを確認します。*my-cluster* の部分は、自分のクラスター名に置き換えます。

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
    addon.addonVersion --output text
   ```

   エラーメッセージが返された場合、クラスターにセルフマネージド型のアドオンがインストールされています。インストールするには、この手順の残りのステップを完了します。バージョン番号が返された場合、クラスターに Amazon EKS タイプのアドオンがインストールされています。Amazon EKS タイプのアドオンを更新するには、この手順を使用するのではなく、「Amazon EKS アドオンの更新」の手順を使用してください。アドオンタイプの違いがよくわからない場合は、「Amazon EKS アドオン」を参照してください。
2. クラスターに現在インストールされているコンテナイメージのバージョンを確認します。

   ```
   kubectl describe deployment coredns -n kube-system | grep Image | cut -d ":" -f 3
   ```

出力例は次のとおりです。

```
v1.8.7-eksbuild.2
```

3. 現在の CoreDNS のバージョンが v1.5.0 以降で、CoreDNS バージョン表に記載されるバージョンよりも前の場合、この手順はスキップしてください。現在のバージョンが 1.5.0 より前の場合、プロキシアドオンではなく進んだアドオンを使用するためには、CoreDNS の ConfigMap を修正する必要があります。

   1. 次のコマンドを使用して ConfigMap ファイルを開きます。

      ```
      kubectl edit configmap coredns -n kube-system
      ```

   2. 次の行の proxy を forward に置き換えます。ファイルを保存し、エディタを終了します。

      ```
      proxy . /etc/resolv.conf
      ```

4. Kubernetes 1.17 以前にクラスターを最初にデプロイした場合、CoreDNS マニフェストから廃止された行の削除が必要な場合がります。

   > **Important**
   >
   > CoreDNS バージョン 1.7.0 に更新する前に、この手順を完了する必要があります。以前のバージョンに更新する場合でも、この手順を完了することをお勧めします。

   1. CoreDNS マニフェストにその行があるかどうかを確認します。

      ```
      kubectl get configmap coredns -n kube-system -o jsonpath='{$.data.Corefile}' |
       grep upstream
      ```

      出力が返されない場合、マニフェストにその行がないため、次の手順に進み、CoreDNS を更新できます。出力が返された場合は、その行を削除します。

   2. 以下のコマンドを使用して ConfigMapを編集し、ファイル内の upstream という単語がある行を削除します。このファイル内の他の部分は変更しないでください。行を削除したら、変更を保存します。

      ```
      kubectl edit configmap coredns -n kube-system -o yaml
      ```

5. 現在の CoreDNS イメージバージョンを取得します。

```
kubectl describe deployment coredns -n kube-system | grep Image
```

出力例は次のとおりです。

```
602401143452.dkr.ecr.region-code.amazonaws.com/eks/coredns:v1.8.7-eksbuild.2
```

6. CoreDNS 1.8.3 以降に更新する場合は、endpointslices のアクセス許可を system:coredns Kubernetes clusterrole に追加する必要があります。

```
kubectl edit clusterrole system:coredns -n kube-system
```

ファイルの rules セクション内の既存の権限行の下に次の行を追加します。

```
[...]
- apiGroups:
  - discovery.k8s.io
  resources:
  - endpointslices
  verbs:
  - list
  - watch
[...]
```

7. *602401143452* と *region-code* を前のステップで返された出力の値に置き換えて、CoreDNS アドオンを更新します。*v1.11.1-eksbuild.9* を、ご利用の Kubernetes バージョンの最新バージョンの表に記載されている CoreDNS バージョンに置き換えます。

```
kubectl set image deployment.apps/coredns -n kube-system
 coredns=602401143452.dkr.ecr.region-code.amazonaws.com/eks/coredns:v1.11.1-
eksbuild.9
```

出力例は次のとおりです。

```
deployment.apps/coredns image updated
```

8. コンテナイメージのバージョンをもう一度チェックして、前のステップで指定したバージョンに更新されたことを確認します。

```
kubectl describe deployment coredns -n kube-system | grep Image | cut -d ":" -f 3
```

出力例は次のとおりです。

```
v1.11.1-eksbuild.9
```

CoreDNS の自動スケーリング

1 つ以上のノードで Amazon EKS クラスターを起動すると、クラスターにデプロイされたノード数に関係なく、デフォルトで CoreDNS イメージの 2 つのレプリカの Deployment がデプロイされます。これらの CoreDNS ポッドは、クラスター内のすべてのポッドの名前解決を行います。アプリケーションは名前解決を使用して、クラスター内のポッドとサービスに接続し、同様にクラスター外のサービスに接続します。ポッドからの名前解決 (クエリ) のリクエスト数が増えると、CoreDNS ポッドが圧倒されて速度が低下し、ポッドが処理できないリクエストが拒否される可能性があります。

CoreDNS ポッドの負荷の増加を処理するには、CoreDNS の自動スケーリングシステムを検討してください。Amazon EKS は、CoreDNS の EKS アドオンバージョンで CoreDNS デプロイの自動スケーリングを管理できます。この CoreDNS オートスケーラーは、ノード数や CPU コア数など、クラスターの状態を継続的にモニタリングします。この情報に基づいて、コントローラーは EKS クラスター内の CoreDNS デプロイのレプリカ数を動的に調整します。この機能は、CoreDNS v1.9 および EKS リリースバージョン 1.25 以降で機能します。CoreDNS 自動スケーリングと互換性のあるバージョンの詳細については、次のセクションを参照してください。

この機能は、アプリケーションの全体的な可用性とクラスターのスケーラビリティを向上させるために、他の EKS クラスターの自動スケーリングのベストプラクティスと組み合わせて使用することをお勧めします。

前提条件

Amazon EKS が CoreDNS デプロイをスケーリングするには、次の 3 つの前提条件があります。

- CoreDNS の EKS アドオンバージョンを使用する必要があります。
- クラスターは、少なくとも最小のクラスターバージョンとプラットフォームバージョンを実行している必要があります。
- クラスターは、少なくとも CoreDNS の EKS アドオンの最小バージョンを実行している必要があります。

最小クラスターバージョン

CoreDNS の自動スケーリングは、Amazon EKS によって管理されるクラスターコントロールプレーンの新しいコンポーネントによって行われます。このため、新しいコンポーネントを持つ最小プラットフォームバージョンをサポートする EKS リリースにクラスターをアップグレードする必要があります。

新しい Amazon EKS クラスター。デプロイするには、「Amazon EKS の使用開始」を参照してください。クラスターは Kubernetes バージョン `1.25` 以降である必要があります。クラスターは、次の表に示す Kubernetes バージョンとプラットフォームバージョン、またはそれ以降のいずれかを実行している必要があります。一覧にあるバージョンより後の Kubernetes とプラットフォームのバージョンもサポートされることにご注意ください。現在の Kubernetes バージョンを確認するには、次のコマンドの *my-cluster* をクラスターの名前に置き換えて、変更したコマンドを実行します。

```
aws eks describe-cluster
              --name my-cluster --query cluster.version --output
              text
```

Kubernetes バージョン	プラットフォームバージョン
1.29.3	eks.7
1.28.8	eks.13
1.27.12	eks.17
1.26.15	eks.18
1.25.16	eks.19

> **Note**
>
> 以降の Kubernetes バージョンのすべてのプラットフォームバージョンもサポートされています。例えば、Kubernetes バージョン `1.30` の `eks.1` 以降のバージョンなどです。

EKS アドオンの最小バージョン

Kubernetes バージョン	1.29	1.28	1.27	1.26	1.25
	v1.11.1-eksbuild.9	v1.10.1-eksbuild.11	v1.10.1-eksbuild.11	v1.9.3-eksbuild.15	v1.9.3-eksbuild.15

AWS Management Console での CoreDNS 自動スケーリングの設定

1. クラスターが最小クラスターバージョン以上であることを確認します。

   Amazon EKS は、同じ Kubernetes バージョンのプラットフォームバージョン間でクラスターを自動的にアップグレードするため、このプロセスを自分で開始することはできません。代わりに、クラスターを次の Kubernetes バージョンにアップグレードすると、クラスターはその K8s バージョンと最新のプラットフォームバージョンにアップグレードされます。例えば、1.25 から 1.26 にアップグレードすると、クラスターは 1.26.15 eks.18 にアップグレードされます。

   新しい Kubernetes バージョンでは、大幅な変更が加えられている場合があります。このため、本稼働用クラスターで更新を行う前に、新しい Kubernetes バージョンのクラスターを別に用意し、アプリケーションの動作をテストしておくことをお勧めします。

   クラスターを新しい Kubernetes バージョンにアップグレードするには、「Amazon EKS クラスターの Kubernetes バージョンの更新」の手順に従います。
2. セルフマネージド型の CoreDNS デプロイではなく、CoreDNS 用の EKS アドオンがあることを確認します。

   クラスターを作成するために使用したツールによっては、現在クラスターに Amazon EKS アドオンタイプがインストールされていない場合があります。クラスターにインストールされているアドオンのタイプを確認するには、次のコマンドを実行します。my-cluster を自分のクラスター名に置き換えます。

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
    addon.addonVersion --output text
   ```

バージョン番号が返された場合、クラスターに Amazon EKS タイプのアドオンがインストールされ、次のステップに進むことができます。エラーが返された場合、クラスターに Amazon EKS タイプのアドオンがインストールされていません。手順 Amazon EKS アドオンの作成 の残りのステップを完了して、セルフマネージドバージョンを Amazon EKS アドオンに置き換えます。

3. CoreDNS の EKS アドオンが、最小 EKS アドオンバージョンと同じかそれ以上のバージョンであることを確認します。

   クラスターにインストールされているアドオンのバージョンを確認します。AWS Management Consoleで確認するか、以下のコマンドを実行できます。

   ```
   kubectl describe deployment coredns --namespace kube-system | grep coredns: | cut -d : -f 3
   ```

   出力例は次のとおりです。

   ```
   v1.10.1-eksbuild.11
   ```

   このバージョンを前のセクションの最小 EKS アドオンバージョンと比較します。必要に応じて、「Amazon EKS アドオンの更新」の手順に従って EKS アドオンを上位のバージョンにアップグレードします。

4. 自動スケーリング設定を EKS アドオンの [オプションの設定] に追加します。

   a. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。

   b. 左のナビゲーションペインで、[クラスター] を選択し、次にアドオンを設定するクラスター名を選択します。

   c. [アドオン] タブを選択します。

   d. CoreDNS アドオンボックスの右上にあるボックスを選択し、次に [編集] を選択します。

   e. CoreDNS を設定ページで、次の操作を行います。

      i. 使用する [バージョン] を選択します。前のステップと同じバージョンを保持し、別のアクションでバージョンと設定を更新することをお勧めします。

      ii. [オプションの構成設定] を展開します。

iii. [設定値] で JSON キー "autoscaling": とネストした JSON オブジェクトの値に、キー "enabled": と値 true を指定します。結果のテキストは有効な JSON オブジェクトでなければなりません。このキーと値だけがテキストボックス内のデータである場合は、キーと値を中括弧 {} で囲みます。次の例は、自動スケーリングが有効になっていることを示しています。

```
{
  "autoScaling": {
    "enabled": true
  }
}
```

iv. (オプション) 自動スケーリングが CoreDNS ポッド数をスケールできる最小値と最大値を指定できます。

次の例は、自動スケーリングが有効で、すべてのオプションキーに値があることを示しています。クラスター内の DNS サービスに回復力を提供できるように、CoreDNS ポッドの最小数は常に 2 より大きくすることをお勧めします。

```
{
  "autoScaling": {
    "enabled": true,
    "minReplicas": 2,
    "maxReplicas": 10
  }
}
```

f. CoreDNS ポッドを置き換えて新しい設定を適用するには、[変更を保存] を選択します。

Amazon EKS は、Kubernetes Deployment for CoreDNS のロールアウトを使用して EKS アドオンに変更を適用します。ロールアウトのステータスは、AWS Management Console のアドオン [更新履歴] と `kubectl rollout status deployment/coredns --namespace kube-system` で追跡できます。

`kubectl rollout` は以下のコマンドを実行します。

```
$ kubectl rollout

history  -- View rollout history
pause    -- Mark the provided resource as paused
restart  -- Restart a resource
```

```
resume   -- Resume a paused resource
status   -- Show the status of the rollout
undo     -- Undo a previous rollout
```

ロールアウトに時間がかかりすぎる場合、Amazon EKS はロールアウトを取り消し、[アドオン更新] のタイプと [失敗] のステータスのメッセージがアドオンの [更新履歴] に追加されます。問題を調査するには、ロールアウトの履歴から開始し、CoreDNS のポッドで `kubectl logs` を実行して CoreDNS のログを確認します。

5. [更新履歴] の新しいエントリのステータスが [成功] の場合、ロールアウトが完了し、アドオンはすべての CoreDNS のポッドで新しい設定を使用していることを意味します。クラスター内のノード数とノードの CPU コア数を変更すると、Amazon EKS は CoreDNS デプロイのレプリカ数をスケールします。

AWS Command Line Interface での CoreDNS 自動スケーリングの設定

1. クラスターが最小クラスターバージョン以上であることを確認します。

   Amazon EKS は、同じ Kubernetes バージョンのプラットフォームバージョン間でクラスターを自動的にアップグレードするため、このプロセスを自分で開始することはできません。代わりに、クラスターを次の Kubernetes バージョンにアップグレードすると、クラスターはその K8s バージョンと最新のプラットフォームバージョンにアップグレードされます。例えば、`1.25` から `1.26` にアップグレードすると、クラスターは `1.26.15 eks.18` にアップグレードされます。

   新しい Kubernetes バージョンでは、大幅な変更が加えられている場合があります。このため、本稼働用クラスターで更新を行う前に、新しい Kubernetes バージョンのクラスターを別に用意し、アプリケーションの動作をテストしておくことをお勧めします。

   クラスターを新しい Kubernetes バージョンにアップグレードするには、「Amazon EKS クラスターの Kubernetes バージョンの更新」の手順に従います。

2. セルフマネージド型の CoreDNS デプロイではなく、CoreDNS 用の EKS アドオンがあることを確認します。

   クラスターを作成するために使用したツールによっては、現在クラスターに Amazon EKS アドオンタイプがインストールされていない場合があります。クラスターにインストールされているアドオンのタイプを確認するには、次のコマンドを実行します。`my-cluster` を自分のクラスター名に置き換えます。

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
 addon.addonVersion --output text
```

バージョン番号が返された場合、クラスターに Amazon EKS タイプのアドオンがインストールされています。エラーが返された場合、クラスターに Amazon EKS タイプのアドオンがインストールされていません。手順 Amazon EKS アドオンの作成 の残りのステップを完了して、セルフマネージドバージョンを Amazon EKS アドオンに置き換えます。

3. CoreDNS の EKS アドオンが、最小 EKS アドオンバージョンと同じかそれ以上のバージョンであることを確認します。

   クラスターにインストールされているアドオンのバージョンを確認します。AWS Management Consoleで確認するか、以下のコマンドを実行できます。

   ```
   kubectl describe deployment coredns --namespace kube-system | grep coredns: | cut -
   d : -f 3
   ```

   出力例は次のとおりです。

   ```
   v1.10.1-eksbuild.11
   ```

   このバージョンを前のセクションの最小 EKS アドオンバージョンと比較します。必要に応じて、「Amazon EKS アドオンの更新」の手順に従って EKS アドオンを上位のバージョンにアップグレードします。

4. 自動スケーリング設定を EKS アドオンの [オプションの設定] に追加します。

   次の AWS CLI コマンドを実行します。`my-cluster` をクラスターの名前に置き換え、IAM ロール ARN を使用するロールに置き換えます。

   ```
   aws eks update-addon --cluster-name my-cluster --addon-name coredns \
       --resolve-conflicts PRESERVE --configuration-values '{"autoScaling":
   {"enabled":true}}'
   ```

   Amazon EKS は、Kubernetes Deployment for CoreDNS のロールアウトを使用して EKS アドオンに変更を適用します。ロールアウトのステータスは、AWS Management Console のアドオ

ン [更新履歴] と `kubectl rollout status deployment/coredns --namespace kube-system` で追跡できます。

`kubectl rollout` は以下のコマンドを実行します。

```
kubectl rollout

history  -- View rollout history
pause    -- Mark the provided resource as paused
restart  -- Restart a resource
resume   -- Resume a paused resource
status   -- Show the status of the rollout
undo     -- Undo a previous rollout
```

ロールアウトに時間がかかりすぎる場合、Amazon EKS はロールアウトを取り消し、[アドオン更新] のタイプと [失敗] のステータスのメッセージがアドオンの [更新履歴] に追加されます。問題を調査するには、ロールアウトの履歴から開始し、CoreDNS のポッドで `kubectl logs` を実行して CoreDNS のログを確認します。

5. (オプション) 自動スケーリングが CoreDNS ポッド数をスケールできる最小値と最大値を指定できます。

   次の例は、自動スケーリングが有効で、すべてのオプションキーに値があることを示しています。クラスター内の DNS サービスに回復力を提供できるように、CoreDNS ポッドの最小数は常に 2 より大きくすることをお勧めします。

   ```
   aws eks update-addon --cluster-name my-cluster --addon-name coredns \
       --resolve-conflicts PRESERVE --configuration-values '{"autoScaling":
   {"enabled":true}, "minReplicas": 2, "maxReplicas": 10}'
   ```

6. 次のコマンドを実行して、アドオンの更新のステータスを確認します。

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name coredns \
   ```

   `"status": "ACTIVE"` という行が表示された場合、ロールアウトが完了し、アドオンはすべての CoreDNS ポッドで新しい設定を使用していることを意味します。クラスター内のノード

数とノードの CPU コア数を変更すると、Amazon EKS は CoreDNS デプロイのレプリカ数をスケールします。

CoreDNS のメトリクス

CoreDNSEKS アドオンとして、CoreDNS9153ポートからのメトリクスをサービス内の Prometheus 形式で公開します。kube-dnsPrometheus、Amazon CloudWatch エージェント、またはその他の互換性のあるシステムを使用して、これらのメトリックスをスクレイピング (収集) できます。

Prometheus と CloudWatch エージェントの両方と互換性のあるscrape configuration(スクレイプ設定) の例については、Amazon CloudWatch ユーザーガイドの「Prometheus 用の CloudWatch エージェント設定」を参照してください。

Kubernetes **kube-proxy** アドオンの使用

> **Important**
>
> セルフマネージド型のアドオンを使用する代わりに、Amazon EKS タイプのアドオンをクラスターに追加することをお勧めします。タイプの違いがよくわからない場合は、「the section called “Amazon EKS アドオン”」を参照してください。Amazon EKS アドオンをクラスターに追加する方法については、「the section called “アドオンの作成”」を参照してください。Amazon EKS アドオンを使用できない場合は、その理由に関する問題をコンテナロードマップの GitHub リポジトリに送信することをお勧めします。

kube-proxy アドオンは Amazon EKS クラスター内の各 Amazon EC2 ノードにデプロイされます。これはノード上のネットワークルールを維持し、Pods へのネットワーク通信を有効化します。アドオンは、クラスター内の Fargate ノードにはデプロイされません。詳細については、Kubernetes ドキュメントの「kube-proxy」を参照してください。

次の表は、各 Kubernetes バージョンの Amazon EKS アドオンタイプの利用可能な最新バージョンを示しています。

Kubernetes バージョン	1.30	1.29	1.28	1.27	1.26	1.25	1.24	1.23
	v1.30.0-eksbuild	v1.29.3-eksbuild	v1.28.8-eksbuild	v1.27.16-eksbuild.5	v1.26.15-eksbuild.5	v1.25.16-eksbuild.8	v1.24.17-eksbuild.8	v1.23.17-eksbuild.9

> **Important**
>
> 以前のバージョンのドキュメントには誤りが含まれていました。kube-proxy バージョン v1.28.5、v1.27.9、および v1.26.12 は利用できません。
> このアドオンを自己管理している場合、表のバージョンは、利用可能なセルフマネージドバージョンと同じではない可能性があります。

各 Amazon EKS クラスターバージョン用に利用可能な kube-proxy コンテナイメージは 2 タイプあります。

- デフォルト - このイメージタイプは Debian ベースの Docker イメージに基づいており、Kubernetes アップストリームコミュニティによって維持されています。
- 最小限 - このイメージタイプは、最小限のベースイメージに基づいており、シェルを持たない Amazon EKS Distro によって維持され、最小限のパッケージを含んでシェルはありません。詳細については、「Amazon EKS Distro」を参照してください。

Amazon EKS クラスターの各バージョンで使用可能な最新のセルフマネージド **kube-proxy** コンテナイメージバージョン

[イメージタイプ]	1.30	1.29	1.28	1.27	1.26	1.25	1.24	1.23
kube-proxy (デフォルトタイプ)	最小限のタイプのみ使用可能	最小限のタイプのみ使用可能	最小限のタイプのみ使用可能	最小限のタイプのみ使用可能	最小限のタイプのみ使用可能	最小限のタイプのみ使用可能	v1.24.17-eksbuild.2	v1.23.16-eksbuild.2

[イメージタイプ]	1.30	1.29	1.28	1.27	1.26	1.25	1.24	1.23
kube-proxy (最小タイプ)	v1.30.0-eksbuild.3-minimal	v1.29.3-eksbuild.5-minimal	v1.28.8-eksbuild.5-minimal	v1.27.16-minimal-eksbuild.5	v1.26.15-minimal-eksbuild.5	v1.25.16-minimal-eksbuild.5	v1.24.17-minimal-eksbuild.5	v1.23.17-minimal-eksbuild.5

> **Important**
>
> - デフォルトの画像タイプは、Kubernetes バージョン 1.25 およびそれ以降では使用できません。最小限のイメージタイプを使用する必要があります。
> - Amazon EKS アドオンタイプを更新するときは、有効な Amazon EKS アドオンバージョンを指定しますが、この表に記載されているバージョンではない可能性があります。これは、Amazon EKS アドオンのバージョンが、このアドオンのセルフマネージドタイプを更新するときに指定されるコンテナイメージのバージョンと常に一致するとは限らないためです。このアドオンのセルフマネージドタイプを更新するときは、この表に記載されている有効なコンテナイメージのバージョンを指定します。

前提条件

- 既存の Amazon EKS クラスター。デプロイするには、「Amazon EKS の使用開始」を参照してください。

考慮事項

- Amazon EKS クラスターの Kube-proxy は、Kubernetes と同じ互換性およびスキューポリシーが適用されます。「アドオンバージョンの互換性を取得する」ではその方法を説明しています。
- Kube-proxy は、Amazon EC2 ノードの kubelet と同じマイナーバージョンである必要があります。
- Kube-proxy は、クラスターのコントロールプレーンのマイナーバージョンよりも新しいものにすることはできません。

- 最近クラスターを新しい Kubernetes マイナーバージョンに更新した場合、Amazon EC2 ノードを同じマイナーバージョンに更新した後で、kube-proxy をノードと同じマイナーバージョンに更新してください。

kube-proxy セルフマネージド型アドオンを更新するには

1. クラスターにインストールされているアドオンがセルフマネージド型であることを確認します。*my-cluster* の部分は、自分のクラスター名に置き換えます。

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name kube-proxy --query
    addon.addonVersion --output text
   ```

   エラーメッセージが返された場合、クラスターにセルフマネージド型のアドオンがインストールされています。このトピックの残りの手順は、セルフマネージド型のアドオンを更新することです。バージョン番号が返された場合、クラスターに Amazon EKS タイプのアドオンがインストールされています。更新するには、このトピックの手順ではなく「アドオンの更新」の手順を使用してください。アドオンタイプの違いがよくわからない場合は、「Amazon EKS アドオン」を参照してください。
2. クラスターに現在インストールされているコンテナイメージのバージョンを確認します。

   ```
   kubectl describe daemonset kube-proxy -n kube-system | grep Image
   ```

   出力例は次のとおりです。

   ```
   Image:    602401143452.dkr.ecr.region-code.amazonaws.com/eks/kube-proxy:v1.29.1-
   eksbuild.2
   ```

   出力例では、*v1.29.1-eksbuild.2* がクラスターにインストールされているバージョンです。
3. *602401143452* と *region-code* を前のステップの出力の値と置き換えて kube-proxy アドオンを更新します*v1.30.0-eksbuild.3* を、各 Amazon EKS クラスターバージョンで利用可能な最新のセルフマネージド kube-proxy コンテナイメージバージョン表に記載された kube-proxy バージョンと置き換えます。デフォルトまたは最小のイメージタイプのバージョン番号を指定できます。

```
kubectl set image daemonset.apps/kube-proxy -n kube-system kube-
proxy=602401143452.dkr.ecr.region-code.amazonaws.com/eks/kube-proxy:v1.30.0-
eksbuild.3
```

出力例は次のとおりです。

```
daemonset.apps/kube-proxy image updated
```

4. 新しいバージョンがクラスターにインストールされたことを確認します。

```
kubectl describe daemonset kube-proxy -n kube-system | grep Image | cut -d ":" -f 3
```

出力例は次のとおりです。

```
v1.30.0-eksbuild.3
```

5. 同じクラスターで x86 ノードと Arm ノードを使用しており、クラスターが 2020 年 8 月 17 日より前にデプロイされている場合。以下のコマンドにより kube-proxy マニフェストを編集して、複数のハードウェアアーキテクチャにノードセレクターを含めます。このオペレーションを行うのは 1 回限りです。マニフェストにセレクタを追加した後、アドオンを更新するたびに追加する必要はありません。クラスターが 2020 年 8 月 17 日以降にデプロイされている場合、kube-proxy は既にマルチアーキテクチャに対応しています。

```
kubectl edit -n kube-system daemonset/kube-proxy
```

エディタを使用して、次のノードセレクターをファイルに追加し、その内容を保存します。エディタ上で、このテキストを挿入する場所の例については、「GitHub」の CNI マニフェストファイル をご覧ください。これにより、Kubernetes はノードのハードウェアアーキテクチャに基づいて、正しいハードウェアイメージを取得できるようになります。

```
- key: "kubernetes.io/arch"
  operator: In
  values:
  - amd64
  - arm64
```

6. クラスターが最初に Kubernetes バージョン 1.14 以降で作成されている場合は、kube-proxy には既にこの Affinity Rule が含まれているため、このステップはスキップできます。最初

に Kubernetes バージョン 1.13 以前のバージョンで Amazon EKS クラスターを作成し、クラスターに Fargate ノードを使用することを考えている場合、kube-proxy マニフェストを編集して NodeAffinity ルールを含め、kube-proxy Pods が Fargate ノードでスケジュール設定しないようにしてください。この編集を行うのは 1 回限りです。Affinity Rule をマニフェストに一度追加したら、アドオンを更新するたびに追加する必要はありません。kube-proxy DaemonSet を編集します。

```
kubectl edit -n kube-system daemonset/kube-proxy
```

エディタで、次の Affinity Rule をファイルの DaemonSet spec セクションに追加した後、変更内容を保存します。エディタ上で、このテキストを挿入する場所の例については、「GitHub」の CNI マニフェストファイル をご覧ください。

```
- key: eks.amazonaws.com/compute-type
  operator: NotIn
  values:
  - fargate
```

インターフェイスエンドポイント (AWS PrivateLink) を使用して Amazon Elastic Kubernetes Service にアクセス

AWS PrivateLink を使用して、VPC と Amazon Elastic Kubernetes Service 間にプライベート接続ができます。インターネットゲートウェイ、NAT デバイス、VPN 接続、AWS Direct Connect 接続のいずれかを使用せずに、VPC 内にあるかのように Amazon EKS にアクセスできます。VPC のインスタンスは、パブリック IP アドレスがなくても Amazon EKS にアクセスできます。

AWS PrivateLink が電源を供給するインターフェイスエンドポイントを作成することにより、このプライベート接続を確立します。インターフェイスエンドポイントに対して有効にする各サブネットにエンドポイントネットワークインターフェイスを作成します。これらは、Amazon EKS 宛てのトラフィックのエントリポイントとして機能するリクエスタ管理型ネットワークインターフェイスです。

詳細については、『AWS PrivateLink ガイド』の「AWS のサービス でアクセスする」を参照してください。

Amazon EKS の考慮事項

- Amazon EKS のインターフェイスエンドポイントを設定する前に、「AWS PrivateLink ガイド」の「考慮事項」を確認してください。
- Amazon EKS は、インターフェイスエンドポイントを介してすべての API アクションの呼び出しをサポートしていますが、Kubernetes API の呼び出しはサポートしていません。Kubernetes API サーバーはすでにプライベートエンドポイントをサポートしています。Kubernetes API サーバーのプライベートエンドポイントは、クラスターとの通信に使用する Kubernetes API サーバー用のプライベートエンドポイントを作成します (`kubectl` などの Kubernetes 管理ツールを使用)。Kubernetes API サーバーへのプライベートアクセスを有効にすると、ノードと API サーバー間のすべての通信が VPC 内で行われるようにできます。Amazon EKS API 用 AWS PrivateLink を使用すると、トラフィックをパブリックインターネットに公開せずに VPC から Amazon EKS API を呼び出すことができます。
- Amazon EKS をインターフェイスエンドポイントを介してのみアクセスできるように設定することはできません。
- Amazon EKS のインターフェイスエンドポイントには AWS PrivateLink の標準料金が適用されます。各アベイラビリティーゾーンでインターフェイスエンドポイントがプロビジョニングされる 1 時間ごと、ならびにインターフェイスエンドポイントを介して処理されたデータに対して請求されます。詳細については、「AWS PrivateLink 料金表」を参照してください。
- VPC エンドポイントポリシーは Amazon EKS をサポートしていません。デフォルトで、インターフェイスエンドポイント経由で Amazon EKS への完全なアクセスが許可されます。または、セキュリティグループをエンドポイントのネットワークインターフェイスに関連付けて、インターフェイスエンドポイントを介して Amazon EKS へのトラフィックを制御することもできます。
- VPC フローログを使用して、インターフェイスエンドポイントを含めたネットワークインターフェイス間で送受信される IP トラフィックに関する情報を取得できます。フローログデータは Amazon CloudWatch または Amazon S3 に発行できます。詳細については、「Amazon VPC ユーザーガイド」の「VPC フローログを使用した IP トラフィックのログ記録」を参照してください。
- Amazon EKS API は、インターフェイスエンドポイントがある VPC に接続することで、オンプレミスのデータセンターからアクセスできます。AWS Direct Connect または AWS Site-to-Site VPN を使用してオンプレミスサイトを VPC に接続できます。
- 他の VPC は、AWS Transit Gateway または VPC ピアリング接続を使用してインターフェイスエンドポイントを備えた VPC に接続できます。VPC ピアリングは、2 つの VPC 間のネットワーク接続です。VPC 間または他のアカウントで VPC を使用して VPC ピアリング接続を確立できます。VPC は異なる AWS リージョン の間で使用できます。ピア接続された VPC 間のトラフィックは AWS ネットワーク上に留まります。トラフィックは公共インターネットを経由しま

せん。Transit Gateway は、VPC 間で相互接続するために使用できるネットワークの中継ハブです。VPC と Transit Gateway 間のトラフィックは AWS グローバルプライベートネットワークに残ります。トラフィックは公共インターネットに公開されません。

- Amazon EKS の VPC インターフェイスエンドポイントは IPv4 のみでアクセスできます。IPv6 はサポートされていません。
- AWS PrivateLink サポートは、アジアパシフィック (ハイデラバード)、アジアパシフィック (メルボルン)、アジアパシフィック (大阪)、カナダ西部 (カルガリー)、欧州 (スペイン)、欧州 (チューリッヒ)、中東 (アラブ首長国連邦) の AWS リージョン ではご利用いただけません。

Amazon EKS 用のインターフェイスエンドポイントを作成します

Amazon VPC コンソールまたは AWS Command Line Interface (AWS CLI) を使用して、Amazon EKS のインターフェイスエンドポイントを作成できます。詳細については、『AWS PrivateLink ガイド』の「Create a VPC endpoint (VPC エンドポイントを作成)を参照してください。

以下のサービス名を使用して Amazon EKS のインターフェイスエンドポイントを作成します。

```
com.amazonaws.region-code.eks
```

プライベート DNS 機能は、Amazon EKS や他の AWS のサービス のインターフェイスエンドポイントを作成するときにデフォルトで有効になります。ただし、次の VPC 属性が true に設定されていることを確認する必要があります。enableDnsHostnames および enableDnsSupport。詳細については、「Amazon VPC ユーザーガイド」の「VPC の DNS 属性の表示と更新」を参照してください。インターフェイスエンドポイントでプライベート DNS 機能を有効にすると、次のことができます。

- デフォルトのリージョン DNS 名を使用して Amazon EKS にあらゆる API リクエストを行うことができます。例えば、eks.*region*.amazonaws.com と指定します。API のリストについては、「Amazon EKS API リファレンス」の「アクション」を参照してください。
- EKS API を呼び出すアプリケーションに変更を加える必要はありません。
- Amazon EKS のデフォルトサービスエンドポイントへの呼び出しは、プライベート AWS ネットワーク経由でインターフェイスエンドポイントを介して自動的にルーティングされます。

ワークロード

ワークロードは、Kubernetes の Pods にデプロイされるコンテナにデプロイされます。Pod には、1 つ以上のコンテナが含まれます。通常、同じサービスを提供する 1 つ以上の Pods が Kubernetes サービスにデプロイされます。同じサービスを提供する複数の Pods をデプロイすると、次の操作を実行できます。

- AWS Management Consoleを使用して、各クラスターでワークロードに関する情報の表示を実行できます。
- Kubernetes Vertical Pod Autoscaler を使用して、Pods を垂直方向にスケールアップまたはスケールダウンできます。
- Kubernetes Horizontal Pod Autoscaler で需要を満たすために必要な Pods の数を水平方向にスケーリングします。
- Pods 全体でネットワークトラフィックのバランスをとるために、外部 (インターネットアクセス可能な Pods の場合) または内部 (プライベート Pods の場合) の Network Load Balancer を作成します。ロードバランサーは、OSI モデルのレイヤ 4 でトラフィックをルーティングします。
- Amazon EKS でのアプリケーション負荷分散 を作成して、Pods 間でアプリケーショントラフィックを分散できます。アプリケーションロードバランサーは、OSI モデルのレイヤ 7 でトラフィックをルーティングします。
- Kubernetes を初めて利用する場合は、こちらのトピック「サンプルアプリケーションをデプロイする」が役立ちます。
- `externalIPs` を使用して、サービスに割り当てることができる IP アドレスを制限できます。

サンプルアプリケーションをデプロイする

このトピックでは、サンプルアプリケーションをクラスターにデプロイします。

前提条件

- 既存の少なくとも 1 つのノードがある Kubernetes クラスター。既存の Amazon EKS クラスターがない場合は、Amazon EKS の使用開始 のガイドの 1 つを使用して Amazon EKS クラスターをデプロイできます。Windows アプリケーションをデプロイする場合は、クラスターと少なくとも 1 つの Amazon EC2 Windows ノードで Windows サポート を有効にする必要があります。
- コンピュータに `kubectl` がインストールされている。詳細については、「`kubectl` のインストールまたは更新」を参照してください。

- クラスターと通信できるように Kubectl が設定されている。詳細については、「Amazon EKS クラスターの kubeconfig ファイルを作成または更新する」を参照してください。
- サンプルワークロードを Fargate にデプロイする予定の場合は、名前を変更しない限り、このチュートリアルで作成したのと同じ名前空間 eks-sample-app を含む既存の Fargate プロファイルが必要です。入門ガイドのいずれかを使用してクラスターを作成した場合は、新しいプロファイルを作成するか、既存のプロファイルに名前空間を追加する必要があります。これは、入門ガイドで作成されたプロファイルでは、このチュートリアルで使用される名前空間が指定されていないためです。VPC には、少なくとも 1 つのプライベートサブネットも必要です。

サンプルアプリケーションをデプロイするには

多くの変数は次のステップで変更できますが、変数値は、指定された場合のみ変更することをお勧めします。KubernetesPods、デプロイ、およびサービスについて理解を深めたら、他の値を変更して試すことができます。

1. 名前空間を作成します。名前空間を使用すると、Kubernetes 内のリソースをグループ化できます。詳細については、「Kubernetes ドキュメント」の「名前空間」を参照してください。サンプルアプリケーションを AWS Fargate にデプロイする予定の場合、AWS Fargate プロファイル内の namespace の値が eks-sample-app であることを確認します。

   ```
   kubectl create namespace eks-sample-app
   ```

2. Kubernetes デプロイを作成する。このサンプルのデプロイでは、パブリックリポジトリからコンテナイメージをプルし、その 3 つのレプリカ (個別の Pods) をクラスターにデプロイします。詳細については、「Kubernetes ドキュメント」の「デプロイ」を参照してください。アプリケーションを Linux ノードまたは Windows ノードにデプロイできます。Fargate にデプロイする場合は、Linux アプリケーションのみデプロイできます。

   a. 次の内容を eks-sample-deployment.yaml という名前のファイルに保存します。サンプルアプリケーションのコンテナではネットワークストレージを使用しませんが、それを使用する必要があるアプリケーションが存在する場合があります。詳細については、「ストレージ」を参照してください。

      Linux

      kubernetes.io/arch キーの下の amd64 または arm64 values は、アプリケーションをいずれかのハードウェアアーキテクチャにデプロイできることを意味します (クラスター内に両方がある場合)。これは、このイメージがマルチアーキテクチャイメージで

あるため可能ですが、すべてがそうであるわけではありません。イメージを取得するリポジトリ内のイメージの詳細を表示することで、イメージがサポートされているハードウェアアーキテクチャを判別できます。ハードウェアアーキテクチャのタイプをサポートしないイメージをデプロイする場合、またはイメージをデプロイしない場合は、そのタイプをマニフェストから削除します。詳細については、「Kubernetes ドキュメント」の「よく知られているラベル、注釈、テイント」を参照してください。

kubernetes.io/os: linux nodeSelector は、クラスターに、例えば、Linux ノードと Windows ノードがある場合、イメージは Linux ノードにのみデプロイされることを意味します。詳細については、「Kubernetes ドキュメント」の「よく知られているラベル、注釈、テイント」を参照してください。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: eks-sample-linux-deployment
  namespace: eks-sample-app
  labels:
    app: eks-sample-linux-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: eks-sample-linux-app
  template:
    metadata:
      labels:
        app: eks-sample-linux-app
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: kubernetes.io/arch
                operator: In
                values:
                - amd64
                - arm64
      containers:
      - name: nginx
        image: public.ecr.aws/nginx/nginx:1.23
```

```
        ports:
        - name: http
          containerPort: 80
        imagePullPolicy: IfNotPresent
      nodeSelector:
        kubernetes.io/os: linux
```

Windows

kubernetes.io/os: windows nodeSelector は、クラスターに、例えば、Windows ノードと Linux ノードがある場合、イメージは Windows ノードにのみデプロイされることを意味します。詳細については、「Kubernetes ドキュメント」の「よく知られているラベル、注釈、テイント」を参照してください。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: eks-sample-windows-deployment
  namespace: eks-sample-app
  labels:
    app: eks-sample-windows-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: eks-sample-windows-app
  template:
    metadata:
      labels:
        app: eks-sample-windows-app
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: beta.kubernetes.io/arch
                operator: In
                values:
                - amd64
      containers:
      - name: windows-server-iis
        image: mcr.microsoft.com/windows/servercore:ltsc2019
```

```
        ports:
        - name: http
          containerPort: 80
        imagePullPolicy: IfNotPresent
        command:
        - powershell.exe
        - -command
        - "Add-WindowsFeature Web-Server; Invoke-WebRequest -UseBasicParsing
 -Uri 'https://dotnetbinaries.blob.core.windows.net/servicemonitor/2.0.1.6/
ServiceMonitor.exe' -OutFile 'C:\\ServiceMonitor.exe'; echo
 '<html><body><br/><br/><marquee><H1>Hello EKS!!!<H1><marquee></body><html>'
 > C:\\inetpub\\wwwroot\\default.html; C:\\ServiceMonitor.exe 'w3svc'; "
      nodeSelector:
        kubernetes.io/os: windows
```

b. デプロイマニフェストをクラスターに適用します。

```
kubectl apply -f eks-sample-deployment.yaml
```

3. サービスを作成します。サービスを利用すると、単一の IP アドレスまたは名前を使用して、すべてのレプリカにアクセスできます。詳細については、「Kubernetes ドキュメント」の「サービス」を参照してください。サンプルアプリケーションには実装されていませんが、他の AWS サービスと対話する必要があるアプリケーションがある場合、Pods の Kubernetes サービスアカウントを作成し、AWS IAM アカウントに関連付けることをお勧めします。サービスアカウントを指定すると、Pods には、他のサービスとのやり取りのために指定した最小限のアクセス許可だけが与えられます。詳細については、「サービスアカウントの IAM ロール」を参照してください。

   a. 次の内容を eks-sample-service.yaml という名前のファイルに保存します。Kubernetes は、クラスター内からのみアクセスできる独自の IP アドレスをサービスに割り当てます。クラスターの外部からサービスにアクセスするには、AWS Load Balancer Controller をデプロイして、サービスに対してアプリケーションやネットワークトラフィックの負荷分散を行います。

      Linux

```
apiVersion: v1
kind: Service
metadata:
  name: eks-sample-linux-service
  namespace: eks-sample-app
```

```
  labels:
    app: eks-sample-linux-app
spec:
  selector:
    app: eks-sample-linux-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

Windows

```
apiVersion: v1
kind: Service
metadata:
  name: eks-sample-windows-service
  namespace: eks-sample-app
  labels:
    app: eks-sample-windows-app
spec:
  selector:
    app: eks-sample-windows-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

b. サービスマニフェストをクラスターに適用します。

```
kubectl apply -f eks-sample-service.yaml
```

4. eks-sample-app 名前空間内に存在するすべてのリソースを表示します。

```
kubectl get all -n eks-sample-app
```

出力例は次のとおりです。

Windows リソースをデプロイした場合は、次の出力の *linux* のすべてのインスタンスは windows です。他の *###* は、実際の出力とは異なる場合があります。

```
NAME                                                READY   STATUS    RESTARTS   AGE
pod/eks-sample-linux-deployment-65b7669776-m6qxz    1/1     Running   0          27m
```

```
pod/eks-sample-linux-deployment-65b7669776-mmxvd   1/1     Running   0          27m
pod/eks-sample-linux-deployment-65b7669776-qzn22   1/1     Running   0          27m

NAME                                 TYPE         CLUSTER-IP      EXTERNAL-IP
 PORT(S)   AGE
service/eks-sample-linux-service   ClusterIP   10.100.74.8     <none>        80/
TCP    32m

NAME                                         READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/eks-sample-linux-deployment 3/3     3            3           27m

NAME                                                    DESIRED   CURRENT   READY
   AGE
replicaset.apps/eks-sample-linux-deployment-776d8f8fd8    3         3         3
   27m
```

出力では、前の手順でデプロイしたサンプルマニフェストで指定されたサービスとデプロイが表示されます。また、3 つの Pods も表示されます。これは、サンプルマニフェストで 3 `replicas` が指定されたためです。Pods についての詳細は、「Kubernetesドキュメント」の「ポッド」を参照してください。Kubernetes は、サンプルマニフェストで指定されていない場合でも、自動的に `replicaset` リソースを作成します。ReplicaSets の詳細については、「Kubernetesドキュメント」の「ReplicaSet」を参照してください。

> **Note**
>
> Kubernetes は、マニフェストで指定したレプリカの数を維持します。本稼働デプロイで、Kubernetes を使用してレプリカ数を水平にスケーリングしたり、Pods のコンピューティングリソースを垂直にスケーリングしたりする場合は、Horizontal Pod Autoscaler と Vertical Pod Autoscaler を使用します。

5. デプロイされたサービスの詳細を表示します。Windows サービスをデプロイした場合は、*linux* を **windows** で置き換えます。

```
kubectl -n eks-sample-app describe service eks-sample-linux-service
```

出力例は次のとおりです。

Windows リソースをデプロイした場合は、次の出力の *linux* のすべてのインスタンスは windows です。他の *###* は、実際の出力とは異なる場合があります。

```
Name:              eks-sample-linux-service
Namespace:         eks-sample-app
Labels:            app=eks-sample-linux-app
Annotations:       <none>
Selector:          app=eks-sample-linux-app
Type:              ClusterIP
IP Families:       <none>
IP:                10.100.74.8
IPs:               10.100.74.8
Port:              <unset>  80/TCP
TargetPort:        80/TCP
Endpoints:         192.168.24.212:80,192.168.50.185:80,192.168.63.93:80
Session Affinity:  None
Events:            <none>
```

前の出力では、IP: の値は、クラスター内のどのノードまたは Pod からもアクセスできる一意の IP アドレスですが、クラスターの外からはアクセスできません。Endpoints の値は、VPC 内からサービスの一部である Pods に割り当てられる IP アドレスです。

6. 前のステップで 名前空間を表示 したときに、出力にリストされた Pods のうち、1 つの詳細を表示します。Windows アプリケーションをデプロイした場合は、*linux* を **windows** に置き換え、*776d8f8fd8-78w66* をいずれかの Pods に対して返された値に置き換えます。

```
kubectl -n eks-sample-app describe pod eks-sample-linux-deployment-65b7669776-m6qxz
```

省略された出力

Windows リソースをデプロイした場合は、次の出力の *linux* のすべてのインスタンスは windows です。他の*example values* は、実際の出力とは異なる場合があります。

```
Name:         eks-sample-linux-deployment-65b7669776-m6qxz
Namespace:    eks-sample-app
Priority:     0
Node:         ip-192-168-45-132.us-west-2.compute.internal/192.168.45.132
[...]
IP:           192.168.63.93
IPs:
  IP:           192.168.63.93
Controlled By:  ReplicaSet/eks-sample-linux-deployment-65b7669776
[...]
Conditions:
```

```
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
[...]
Events:
  Type    Reason     Age    From
 Message
  ----    ------     ----   ----
 -------
  Normal  Scheduled  3m20s  default-scheduler
 Successfully assigned eks-sample-app/eks-sample-linux-deployment-65b7669776-m6qxz
 to ip-192-168-45-132.us-west-2.compute.internal
[...]
```

前の出力では、IP: の値は CIDR ブロックから Pod に割り当てられる一意の IP です。CIDR ブロックは、ノードが存在するサブネットに割り当てられています。異なる CIDR ブロックから Pods の IP アドレスを割り当てたい場合は、デフォルトの動作を変更できます。詳細については、「ポッド用のカスタムネットワーク」を参照してください。また、Kubernetes スケジューラーが IP アドレス *192.168.45.132* を使用して Node 上の Pod をスケジュールしたことを確認できます。

> **Tip**
>
> コマンドラインを使用する代わりに、Pods、サービス、デプロイ、その他の Kubernetes リソースに関する多くの詳細を AWS Management Console に表示できます。詳細については、「Kubernetes リソースを表示する」を参照してください。

7. 前の手順で説明した Pod でシェルを実行し、*65b7669776-m6qxz* をいずれかの Pods の ID に置き換えます。

   Linux

   ```
   kubectl exec -it eks-sample-linux-deployment-65b7669776-m6qxz -n eks-sample-app
    -- /bin/bash
   ```

Windows

```
kubectl exec -it eks-sample-windows-deployment-65b7669776-m6qxz -n eks-sample-app -- powershell.exe
```

8. Pod シェルから、前のステップでデプロイと共にインストールされたウェブサーバーからの出力を表示します。サービス名のみ指定する必要があります。デフォルトでは、Amazon EKS クラスターとともにデプロイされる CoreDNS によって、サービスの IP アドレスに解決されます。

   Linux

   ```
   curl eks-sample-linux-service
   ```

   出力例は次のとおりです。

   ```
   <!DOCTYPE html>
   <html>
   <head>
   <title>Welcome to nginx!</title>
   [...]
   ```

   Windows

   ```
   Invoke-WebRequest -uri eks-sample-windows-service/default.html -UseBasicParsing
   ```

   出力例は次のとおりです。

   ```
   StatusCode        : 200
   StatusDescription : OK
   Content           : < h t m l > < b o d y > < b r / > < b r / > < m a r q u e e
    > < H 1 > H e l l o
                       E K S ! ! ! < H 1 > < m a r q u e e > < / b o d y > < h t
    m l >
   ```

9. Pod シェルから、Pod の DNS サーバーを表示します。

Linux

```
cat /etc/resolv.conf
```

出力例は次のとおりです。

```
nameserver 10.100.0.10
search eks-sample-app.svc.cluster.local svc.cluster.local cluster.local us-west-2.compute.internal
options ndots:5
```

前の出力では、クラスターにデプロイされたすべての Pods に対して、10.100.0.10 が nameserver として自動的に割り当てられます。

Windows

```
Get-NetIPConfiguration
```

省略された出力

```
InterfaceAlias       : vEthernet
[...]
IPv4Address          : 192.168.63.14
[...]
DNSServer            : 10.100.0.10
```

前の出力では、クラスターにデプロイされたすべての Pods に対して、10.100.0.10 が DNS サーバーとして自動的に割り当てられます。

10. exit を入力して、Pod の接続を切断します。
11. サンプルアプリケーションの使用が終了したら、次のコマンドを使用して、サンプルの名前空間、サービス、デプロイを削除できます。

```
kubectl delete namespace eks-sample-app
```

次のステップ

サンプルアプリケーションをデプロイしたら、次の演習の一部を試すことができます。

- the section called “アプリケーション負荷分散”
- the section called “ネットワーク負荷分散”

Vertical Pod Autoscaler

Kubernetes Vertical Pod Autoscaler は、Pods の CPU とメモリの予約を自動的に調整し、アプリケーションを「適切なサイズ」にするのに役立ちます。この調整により、クラスターリソースの使用率が向上し、他の Pods の CPU とメモリが解放されます。このトピックでは、Vertical Pod Autoscaler をクラスターにデプロイし、動作していることを確認するのに役立ちます。

前提条件

- 既存の Amazon EKS クラスターがあります。そうでない場合は、「Amazon EKS の使用開始」を参照してください。
- Kubernetes メトリクスサーバーがインストールされています。詳細については、「Kubernetes メトリクスサーバーのインストール」を参照してください。
- Amazon EKS クラスターと通信するように設定された `kubectl` クライアントを使用しています。
- デバイスに、OpenSSL `1.1.1` 以降がインストールされています。

Vertical Pod Autoscaler をデプロイする

このセクションでは、Vertical Pod Autoscaler をクラスターにデプロイします。

Vertical Pod Autoscaler をデプロイするには

1. ターミナルウィンドウを開き、Vertical Pod Autoscaler ソースコードをダウンロードするディレクトリに移動します。
2. kubernetes/autoscaler GitHub リポジトリのクローンを作成します。

```
git clone https://github.com/kubernetes/autoscaler.git
```

3. `vertical-pod-autoscaler` ディレクトリを変更します。

```
cd autoscaler/vertical-pod-autoscaler/
```

4. (オプション) Vertical Pod Autoscaler の別のバージョンをすでにデプロイしている場合は、次のコマンドを使用して削除します。

```
./hack/vpa-down.sh
```

5. ノードが registry.k8s.io コンテナレジストリに対してインターネットからアクセスできない場合は、次のイメージをプルして、独自のプライベートリポジトリにプッシュする必要があります。イメージをプルして独自のプライベートリポジトリにプッシュする方法の詳細については、「あるリポジトリから別のリポジトリにコンテナイメージをコピーする」 を参照してください。

```
registry.k8s.io/autoscaling/vpa-admission-controller:0.10.0
registry.k8s.io/autoscaling/vpa-recommender:0.10.0
registry.k8s.io/autoscaling/vpa-updater:0.10.0
```

プライベートの Amazon ECR リポジトリにイメージをプッシュする場合は、マニフェストの registry.k8s.io をレジストリに置き換えます。*111122223333* をアカウントID に置き換えます。*region-code* をクラスターのある AWS リージョン に置き換えます。次のコマンドは、お使いのリポジトリにマニフェストのリポジトリ名と同じ名前を付けていることを前提としています。リポジトリに別名を付けた場合も、同様に変更する必要があります。

```
sed -i.bak -e 's/registry.k8s.io/111122223333.dkr.ecr.region-
code.amazonaws.com/' ./deploy/admission-controller-deployment.yaml
sed -i.bak -e 's/registry.k8s.io/111122223333.dkr.ecr.region-
code.amazonaws.com/' ./deploy/recommender-deployment.yaml
sed -i.bak -e 's/registry.k8s.io/111122223333.dkr.ecr.region-
code.amazonaws.com/' ./deploy/updater-deployment.yaml
```

6. 次のコマンドを使用して、Vertical Pod Autoscaler をクラスターにデプロイします。

```
./hack/vpa-up.sh
```

7. Vertical Pod Autoscaler Pods が正常に作成されたことを確認します。

```
kubectl get pods -n kube-system
```

出力例は次のとおりです。

```
NAME                                        READY   STATUS    RESTARTS   AGE
[...]
metrics-server-8459fc497-kfj8w              1/1     Running   0          83m
vpa-admission-controller-68c748777d-ppspd   1/1     Running   0          7s
```

```
vpa-recommender-6fc8c67d85-gljpl            1/1     Running   0          8s
vpa-updater-786b96955c-bgp9d                1/1     Running   0          8s
```

Vertical Pod Autoscaler のインストールをテストします

このセクションでは、サンプルアプリケーションをデプロイして、Vertical Pod Autoscaler が動作していることを確認します。

Vertical Pod Autoscaler のインストールをテストするには

1. 次のコマンドを使用して hamster.yaml Vertical Pod Autoscaler の例をデプロイします。

```
kubectl apply -f examples/hamster.yaml
```

2. hamster アプリケーション例から Pods を取得します。

```
kubectl get pods -l app=hamster
```

出力例は次のとおりです。

```
hamster-c7d89d6db-rglf5   1/1     Running   0          48s
hamster-c7d89d6db-znvz5   1/1     Running   0          48s
```

3. cpu と memory の予約を表示する Pods の 1 つを説明します。*c7d89d6db-rglf5* を前のステップの出力で返された ID の 1 つに置き換えます。

```
kubectl describe pod hamster-c7d89d6db-rglf5
```

出力例は次のとおりです。

```
[...]
Containers:
  hamster:
    Container ID:  docker://
e76c2413fc720ac395c33b64588c82094fc8e5d590e373d5f818f3978f577e24
    Image:         registry.k8s.io/ubuntu-slim:0.1
    Image ID:      docker-pullable://registry.k8s.io/ubuntu-
slim@sha256:b6f8c3885f5880a4f1a7cf717c07242eb4858fdd5a84b5ffe35b1cf680ea17b1
    Port:          <none>
    Host Port:     <none>
```

```
    Command:
      /bin/sh
    Args:
      -c
      while true; do timeout 0.5s yes >/dev/null; sleep 0.5s; done
    State:          Running
      Started:      Fri, 27 Sep 2019 10:35:16 -0700
    Ready:          True
    Restart Count:  0
    Requests:
      cpu:        100m
      memory:     50Mi
[...]
```

元の Pod は 100 millicpu の CPU と 50 メビバイトのメモリを予約していることがわかります。このアプリケーション例では、100 millicpu は Pod の実行に必要なものより少ないため、CPU に制約があります。また、必要とするよりもはるかに少ないメモリを予約します。Vertical Pod Autoscaler `vpa-recommender` デプロイでは、`hamster` Pods を分析して、CPU とメモリの要件が適切かどうかを確認します。調整が必要な場合、`vpa-updater` は更新された値で Pods を再起動します。

4. `vpa-updater` が新しい `hamster` Pod を起動するまで待ちます。これには 1~2 分かかります。次のコマンドを使用して、Pods をモニタリングできます。

> **Note**
>
> 新しい Pod が起動されたかどうかが不明な場合は、Pod 名を前のリストと比較します。新しい Pod が起動すると、新しい Pod 名が表示されます。

```
kubectl get --watch Pods -l app=hamster
```

5. 新しい `hamster` Pod が開始されたら、それについて説明し、更新された CPU とメモリの予約を表示します。

```
kubectl describe pod hamster-c7d89d6db-jxgfv
```

出力例は次のとおりです。

```
[...]
```

```
Containers:
  hamster:
    Container ID:
 docker://2c3e7b6fb7ce0d8c86444334df654af6fb3fc88aad4c5d710eac3b1e7c58f7db
    Image:         registry.k8s.io/ubuntu-slim:0.1
    Image ID:      docker-pullable://registry.k8s.io/ubuntu-
slim@sha256:b6f8c3885f5880a4f1a7cf717c07242eb4858fdd5a84b5ffe35b1cf680ea17b1
    Port:          <none>
    Host Port:     <none>
    Command:
      /bin/sh
    Args:
      -c
      while true; do timeout 0.5s yes >/dev/null; sleep 0.5s; done
    State:          Running
      Started:      Fri, 27 Sep 2019 10:37:08 -0700
    Ready:          True
    Restart Count:  0
    Requests:
      cpu:        587m
      memory:     262144k
[...]
```

以前の出力では、cpu 予約が元の値の 5 倍以上である 587 millicpu に増加したことがわかります。memory は 262,144 キロバイト (約 250 メビバイト、つまり元の値の 5 倍) に増加しました。この Pod はリソース不足であり、Vertical Pod Autoscaler は見積りをより適切な値で修正しました。

6. hamster-vpa リソースの詳細を表示して、新しい推奨事項を表示します。

```
kubectl describe vpa/hamster-vpa
```

出力例は次のとおりです。

```
Name:         hamster-vpa
Namespace:    default
Labels:       <none>
Annotations:  kubectl.kubernetes.io/last-applied-configuration:
                {"apiVersion":"autoscaling.k8s.io/
v1beta2","kind":"VerticalPodAutoscaler","metadata":{"annotations":
{},"name":"hamster-vpa","namespace":"d...
API Version:  autoscaling.k8s.io/v1beta2
```

```
Kind:         VerticalPodAutoscaler
Metadata:
  Creation Timestamp:  2019-09-27T18:22:51Z
  Generation:          23
  Resource Version:    14411
  Self Link:           /apis/autoscaling.k8s.io/v1beta2/namespaces/default/
verticalpodautoscalers/hamster-vpa
  UID:                 d0d85fb9-e153-11e9-ae53-0205785d75b0
Spec:
  Target Ref:
    API Version:  apps/v1
    Kind:         Deployment
    Name:         hamster
Status:
  Conditions:
    Last Transition Time:  2019-09-27T18:23:28Z
    Status:                True
    Type:                  RecommendationProvided
  Recommendation:
    Container Recommendations:
      Container Name:  hamster
      Lower Bound:
        Cpu:     550m
        Memory:  262144k
      Target:
        Cpu:     587m
        Memory:  262144k
      Uncapped Target:
        Cpu:     587m
        Memory:  262144k
      Upper Bound:
        Cpu:     21147m
        Memory:  387863636
Events:          <none>
```

7. アプリケーション例の試用が終了したら、次のコマンドで削除します。

```
kubectl delete -f examples/hamster.yaml
```

Horizontal Pod Autoscaler

Kubernetes Horizontal Pod Autoscaler は、そのリソースの CPU 使用率に基づいて設定されたデプロイ、レプリケーションコントローラー、またはレプリカセット内の Pods の数を自動的にスケーリングします。これにより、アプリケーションは需要の増加に合わせてスケールアウトしたり、リソースが不要になったときにスケールインしたりできるため、ノードを他のアプリケーションに解放できます。ターゲットの CPU 使用率を設定すると、Horizontal Pod Autoscaler はターゲットを満たすようにアプリケーションをスケールインまたはスケールアウトします。

Horizontal Pod Autoscaler は、Kubernetes の標準 API リソースであり、動作するには、メトリクスソース (Kubernetes メトリクスサーバーなど) が Amazon EKS クラスターにインストールされている必要があります。アプリケーションのスケーリングを開始するために、クラスターに Horizontal Pod Autoscaler をデプロイまたはインストールする必要はありません。詳細については、Kubernetes ドキュメントの「Horizontal Pod Autoscaler」を参照してください。

このトピックでは、Amazon EKS クラスターの Horizontal Pod Autoscaler の準備や、サンプルアプリケーションで動作することの確認方法を説明しています。

> **Note**
>
> このトピックは、「Kubernetes ドキュメント」の「Horizontal Pod autoscaler ウォークスルー」に基づいています。

前提条件

- 既存の Amazon EKS クラスターがあります。そうでない場合は、「Amazon EKS の使用開始」を参照してください。
- Kubernetes メトリクスサーバーがインストールされています。詳細については、「Kubernetes メトリクスサーバーのインストール」を参照してください。
- Amazon EKS クラスターと通信するように設定された `kubectl` クライアントを使用しています。

Horizontal Pod Autoscaler テストアプリケーションを実行する

このセクションでは、サンプルアプリケーションをデプロイして、Horizontal Pod Autoscaler が動作していることを確認します。

> **Note**
>
> この例は、「Kubernetes ドキュメント」の「Horizontal Pod Autoscaler Walkthrough」に基づいています。

Horizontal Pod Autoscaler のインストールをテストするには

1. 次のコマンドを使用して、シンプルな Apache ウェブサーバーアプリケーションをデプロイします。

   ```
   kubectl apply -f https://k8s.io/examples/application/php-apache.yaml
   ```

   この Apache ウェブサーバー Pod には 500 millicpu の CPU 制限が与えられ、ポート 80 で提供されています。
2. php-apache デプロイ用の Horizontal Pod Autoscaler リソースを作成します。

   ```
   kubectl autoscale deployment php-apache --cpu-percent=50 --min=1 --max=10
   ```

   このコマンドは、デプロイの CPU 使用率が 50% で、最小 1 個の Pod、最大 10 個 の Pods を使用するオートスケーラーを作成します。CPU の平均負荷が 50% を下回ると、オートスケーラーはデプロイの Pods 数を最小の 1 に減らそうとします。負荷が 50% を超えると、オートスケーラーは展開内の Pods の数を最大 10 まで増やしようとします。詳細については、Kubernetes ドキュメントの「How does a HorizontalPodAutoscaler work?」(HorizontalPodAutoscaler の仕組み) を参照してください。
3. 詳細を表示するには、次のコマンドで Autoscaler を説明します。

   ```
   kubectl get hpa
   ```

   出力例は次のとおりです。

   ```
   NAME         REFERENCE                TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
   php-apache   Deployment/php-apache   0%/50%    1         10        1          51s
   ```

   ご覧のように、現在の CPU 負荷は 0% です。サーバーにまだ負荷がないためです。Pod 数は既に最低の境界 (1) にあるため、スケールインすることはできません。
4. コンテナを実行して、ウェブサーバーのロードを作成します。

```
kubectl run -i \
    --tty load-generator \
    --rm --image=busybox \
    --restart=Never \
    -- /bin/sh -c "while sleep 0.01; do wget -q -O- http://php-apache; done"
```

5. デプロイのスケールアウトを監視するには、前のステップを実行したターミナルとは別のターミナルで次のコマンドを定期的に実行します。

```
kubectl get hpa php-apache
```

出力例は次のとおりです。

```
NAME         REFERENCE                  TARGETS    MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache   250%/50%   1         10        5
 4m44s
```

レプリカ数が増えるまで、1 分以上かかる場合があります。実際の CPU のパーセンテージが目標のパーセンテージよりも高い限り、レプリカの数は最大 10 まで増加します。この場合は 250% のため、REPLICAS の数は増え続けます。

> **Note**
>
> レプリカの数が最大値に達するまで、数分かかる場合があります。例えば、CPU 負荷が 50% 以下になるために必要なレプリカの数が 6 だけの場合、負荷はレプリカの数 6 を超えてスケールすることはありません。

6. 負荷を停止します。負荷を生成しているターミナルウィンドウで、Ctrl+C キーを押して負荷を停止します。スケールインを監視しているターミナルで次のコマンドを再度実行することで、レプリカの数が 1 にスケールバックされるのを確認できます。

```
kubectl get hpa
```

出力例は次のとおりです。

```
NAME         REFERENCE                  TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache   0%/50%    1         10        1          25m
```

> **Note**
>
> 現在の CPU の割合が 0% であっても、スケールダウンのデフォルトの時間枠は 5 分であるため、レプリカの数が 1 に再び達するまでには時間がかかります。時間枠は変更可能です。詳細については、「Kubernetes ドキュメント」の「Horizontal Pod Autoscaler」を参照してください。

7. サンプルアプリケーションのテストが終了したら、php-apache リソースを削除します。

```
kubectl delete deployment.apps/php-apache service/php-apache
 horizontalpodautoscaler.autoscaling/php-apache
```

Amazon EKS でのネットワーク負荷分散

ネットワークトラフィックは、OSI モデルの L4 で負荷分散されます。L7 でアプリケーショントラフィックの負荷を分散するには、Kubernetes `ingress` をデプロイし、これによって AWS Application Load Balancer をプロビジョニングします。詳細については、「Amazon EKS でのアプリケーション負荷分散」を参照してください。2 種類の負荷分散の違いについては、AWS ウェブサイトの「Elastic Load Balancing の特徴」を参照してください。

タイプ `LoadBalancer` の Kubernetes `Service` を作成する際、デフォルトでは AWS クラウドプロバイダーロードバランサーコントローラーにより AWS Classic Load Balancer が作成されますが、AWS Network Load Balancer も作成できます。このコントローラーは、将来の重大なバグ修正のみを受けています。AWS クラウドプロバイダーのロードバランサーの使用に関する情報は、「Kubernetes ドキュメント」の「AWS cloud provider load balancer controller」(クラウドプロバイダーのロードバランサーコントローラー) を参照してください。使用方法については、このトピックでは説明しません。

`2.7.2` クラウドプロバイダーロードバランサーコントローラーの代わりに、AWS Load Balancer Controller のバージョン AWS 以降を使用することをお勧めします。AWS Load Balancer Controller では、AWS Network Load Balancer は作成できますが、AWS Classic Load Balancer は作成できません。このトピックの残りの部分では、AWS Load Balancer Controller の使用について説明します。

AWS Network Load Balancer では、Amazon EC2 IP およびインスタンスターゲットや AWS Fargate IP ターゲットにデプロイされた Pods へのネットワークトラフィックの負荷分散を行うことができます。詳細については、「GitHub」の「AWS Load Balancer Controller」を参照してください。

前提条件

AWS Load Balancer Controller を使用してネットワークトラフィックの負荷分散を行う前に、次の要件を満たす必要があります。

- 既存のクラスターがある。既存のクラスターがない場合は、「Amazon EKS の使用開始」を参照してください。既存のクラスターのバージョンを更新する必要がある場合は、Amazon EKS クラスターの Kubernetes バージョンの更新 を参照して下さい。
- クラスターにデプロイ済みの AWS Load Balancer Controller があること。詳細については、「AWS Load Balancer Controller とは」を参照してください。バージョン `2.7.2` 以降をお勧めします。
- 少なくとも 1 つのサブネット。アベイラビリティーゾーンで複数のタグ付きサブネットが見つかった場合、コントローラーは、サブネット ID が辞書式順序で最初に来る最初のサブネットを選択します。サブネットには、利用可能な IP アドレスが最低 8 個必要です。
- AWS Load Balancer Controller バージョン `2.1.1` 以前を使用している場合、サブネットに次のようにタグ付けする必要があります。バージョン `2.1.2` 以降を使用している場合、このタグはオプションです。同じ VPC 内で複数のクラスターを実行していたり、VPC 内で複数の AWS のサービスがサブネットを共有していたりして、クラスターごとにロードバランサーがプロビジョニングされる場所をより詳細に制御したい場合は、サブネットにタグを付けることができます。サービスオブジェクトのアノテーションとしてサブネット ID を明示的に指定すると、Kubernetes と AWS Load Balancer Controller は、これらのサブネットを直接使用してロードバランサーを作成します。ロードバランサーのプロビジョニングにこの方法を使用することを選択した場合、サブネットのタグ付けは不要です。次のプライベートサブネットおよびパブリックサブネットのタグ付け要件は省略できます。*`my-cluster`* をクラスター名に置き換えます。
  - キー – `kubernetes.io/cluster/`*`my-cluster`*
  - 値 – `shared` または `owned`
- サービスオブジェクトまたは Ingress オブジェクトのアノテーションとしてサブネット ID を明示的に指定しない場合、パブリックサブネットとプライベートサブネットは次の要件を満たしている必要があります。サービスまたはイングレス オブジェクトのアノテーションとしてサブネット ID を明示的に指定してロードバランサーをプロビジョニングする場合、Kubernetes と AWS Load Balancer Controller はそれらのサブネットを直接使用してロードバランサーを作成するため、次のタグは必要ありません。
  - プライベートサブネット — 次の形式でタグ付けする必要があります。これは、サブネットを内部ロードバランサーに使用できることを、Kubernetes と AWS ロードバランサーコントローラーが認識できるようにするためです。`eksctl` または Amazon EKS AWS AWS CloudFormation テンプレートを使用して、2020 年 3 月 26 日以降に VPC を作成する場合、サ

ブネットは、作成時に適切にタグ付けされます。Amazon EKS AWS AWS CloudFormation VPC テンプレートの詳細については、「Amazon EKS クラスター VPC の作成」を参照してください。

- キー – kubernetes.io/role/internal-elb
- 値 – 1

- パブリックサブネット — 次の形式でタグ付けする必要があります。これは、Kubernetes が (サブネットIDの辞書式順序に基づいて) 各アベイラビリティーゾーンでパブリックサブネットを選択する代わりに、外部ロードバランサーにこれらのサブネットのみを使用することを認識できるようにするためです。eksctl または Amazon EKS AWS CloudFormation テンプレートを使用して、2020 年 3 月 26 日以降に VPC を作成する場合、サブネットは、作成時に適切にタグ付けされます。Amazon EKS AWS CloudFormation VPC テンプレートの詳細については、「Amazon EKS クラスター VPC の作成」を参照してください。
  - キー – kubernetes.io/role/elb
  - 値 – 1

サブネットロールタグが明示的に追加されていない場合、Kubernetes サービスコントローラーはクラスター VPC サブネットのルートテーブルを調べて、サブネットがプライベートかパブリックかを判断します。この動作に頼らず、プライベートまたはパブリックロールタグを明示的に追加することをお勧めします。AWS Load Balancer Controller は、ルートテーブルを検査しないため、自動検出を正常に実行するには、プライベートタグとパブリックタグが必要です。

考慮事項

- ロードバランサーの設定は、サービス用のマニフェストに追加された注釈により制御されます。AWS Load Balancer Controller を使用する場合のサービスアノテーションは、AWS クラウドプロバイダーのロードバランサーコントローラーを使用する場合のサービスアノテーションとは異なります。サービスをデプロイする前に、必ず AWS Load Balancer Controller のアノテーションを確認してください。
- Amazon VPC CNI plugin for Kubernetes を使用する場合、AWS Load Balancer Controller は、Amazon EC2 IP またはインスタンスターゲットおよび Fargate IP ターゲットに負荷分散を行います。互換性のある代替 CNI プラグインを使用する場合、コントローラーはインスタンスターゲットにのみ負荷分散できます。Network Load Balancer のターゲットタイプの詳細については、「Network Load Balancer のユーザーガイド」の「Target type」(ターゲットタイプ) を参照してください。

- ロードバランサーの作成時 (または作成後) にタグを追加する場合は、サービス仕様に次のアノテーションを追加します。詳細については、「AWS Load Balancer Controller ドキュメント」の「AWS リソースタグ」を参照してください。

```
service.beta.kubernetes.io/aws-load-balancer-additional-resource-tags
```

- Elastic IP アドレスを Network Load Balancer に割り当てるには、次のアノテーションを追加します。*example values* を Elastic IP アドレスの `Allocation IDs` に置き換えます。`Allocation IDs` の数は、ロードバランサーに使用されるサブネットの数に一致する必要があります。詳細については、AWS Load Balancer Controller ドキュメントを参照してください。

```
service.beta.kubernetes.io/aws-load-balancer-eip-allocations:
 eipalloc-xxxxxxxxxxxxxxxxx,eipalloc-yyyyyyyyyyyyyyyyy
```

- Amazon EKS は、クライアントトラフィック用のインバウンドルールをノードのセキュリティグループに 1 つ追加し、VPC 内のロードバランサーのサブネットごとに、作成する各 Network Load Balancer のヘルスチェック用のルールを 1 つずつ追加します。タイプ `LoadBalancer` のサービスの展開は、Amazon EKS が、セキュリティグループに許可されているルールの最大数に対するクォータを超えるルールを作成しようとした場合に失敗することがあります。詳細については、Amazon VPC ユーザーガイドの「セキュリティグループ」を参照してください。セキュリティグループのルールの最大数を超える可能性を最小限に抑えるために、次のオプションを検討します。
  - セキュリティグループクォータあたりのルールの増加をリクエストする。詳細については、Service Quotas ユーザーガイドの「Requesting a quota increase (クォータの引き上げのリクエスト)」を参照してください。
  - インスタンスターゲットではなく IP ターゲットを使用する。IP ターゲットを使用すると、同じターゲットポートでルールを共有できます。ロードバランサーのサブネットは、アノテーションを使用して手動で指定できます。詳細については、「GitHub」の「アノテーション」を参照してください。
  - タイプ `LoadBalancer` のサービスの代わりにイングレスを使用して、サービスにトラフィックを送信します。AWS Application Load Balancer では、Network Load Balancer よりも必要なルールは少なくなります。複数のイングレス間で ALB を共有できます。詳細については、「Amazon EKS でのアプリケーション負荷分散」を参照してください。複数のサービス間で Network Load Balancer を共有することはできません。
  - クラスターを複数のアカウントにデプロイする。

- Pods が Amazon EKS クラスター内の Windows で実行されている場合、ロードバランサーを使用する 1 つのサービスで最大 1,024 個のバックエンド Pods をサポートできます。各 Pod には固有の IP アドレスがあります。
- 新しい Network Load Balancer の作成には、AWS Load Balancer Controller を使用することをお勧めします。AWS クラウドプロバイダーロードバランサーコントローラーで作成された既存の Network Load Balancer を置き換えようとすると、複数の Network Load Balancer でアプリケーションのダウンタイムを引き起こす可能性があります。

ネットワークロードバランサーを作成する

IP またはインスタンスのターゲットを含むネットワークロードバランサーを作成できます。

IP targets

IP ターゲットは、Amazon EC2 ノードまたは Fargate にデプロイされた Pods で使用できます。Kubernetes サービスはタイプ `LoadBalancer` として作成する必要があります。詳細については、「Kubernetes ドキュメント」の「Type Load Balancers」を参照してください。

IP ターゲットを使用するロードバランサーを作成するには、次のアノテーションをサービスマニフェストに追加し、サービスをデプロイします。`aws-load-balancer-type` の `external` 値により、AWS クラウドプロバイダーのロードバランサーコントローラーではなく、AWS Load Balancer Controller によって Network Load Balancer が作成されます。アノテーション付きのサンプルサービスマニフェストを表示できます。

```
service.beta.kubernetes.io/aws-load-balancer-type: "external"
service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: "ip"
```

> **Note**
>
> IPv6 Pods への負荷分散を行う場合は、次のアノテーションを追加します。IPv6 を使用して負荷分散を行えるのは、IP ターゲットにのみです。インスタンスターゲットには行えません。このアノテーションを使用しない場合、負荷分散には IPv4 が使用されます。
>
> ```
> service.beta.kubernetes.io/aws-load-balancer-ip-address-type: dualstack
> ```

デフォルトでは、Network Load Balancer は `internal` `aws-load-balancer-scheme` を使用して作成されます。Network Load Balancer は、クラスターの VPC 内にある任意のサブネットで起動できます。これには、クラスターの作成時に指定されていないサブネットも含まれます。

Kubernetes は、サブネットのルートテーブルを調べて、サブネットがパブリックかプライベートかどうかを特定します。パブリックサブネットには、インターネットゲートウェイを使ったインターネットへの直接の経路がありますが、プライベートサブネットにはありません。

パブリックサブネットに Network Load Balancer を作成して Amazon EC2 ノードへの負荷分散を行う場合 (Fargate はプライベートのみ可能)、次のアノテーションを付けて `internet-facing` を指定します。

```
service.beta.kubernetes.io/aws-load-balancer-scheme: "internet-facing"
```

> **Note**
>
> `service.beta.kubernetes.io/aws-load-balancer-type: "nlb-ip"` アノテーションは、下位互換性のために引き続きサポートされています。ただし、新しいロードバランサーには、`service.beta.kubernetes.io/aws-load-balancer-type: "nlb-ip"` の代わりに以前のアノテーションを使用することをお勧めします。

> **Important**
>
> サービスの作成後は、このアノテーションを編集しないでください。変更する必要がある場合は、サービスオブジェクトを削除し、このアノテーションを希望する値にして再度作成します。

Instance targets

AWS クラウドプロバイダーロードバランサーコントローラーでは、インスタンスターゲットのみを使用して Network Load Balancer を作成します。バージョン `2.2.0` 以降の AWS Load Balancer Controller でも同様に、インスタンスターゲットを使用して Network Load Balancer を作成します。新しい Network Load Balancer の作成には、AWS クラウドプロバイダーロードバランサーコントローラーではなく、こちらを使用することをお勧めします。Network Load Balancer のインスタンスターゲットは、Amazon EC2 ノードにデプロイされた Pods で使用でき

ますが、Fargate にデプロイされたポッドでは使用できません。Fargate にデプロイされた Pods 間でネットワークトラフィックを負荷分散するには、IP ターゲットを使用する必要があります。

Network Load Balancer をプライベートサブネットにデプロイするには、サービスの仕様に次のアノテーションが必要です。アノテーション付きのサンプルサービスマニフェストを表示できます。`aws-load-balancer-type` の `external` 値により、AWS クラウドプロバイダーロードバランサーコントローラーではなく、AWS Load Balancer Controller によって Network Load Balancer が作成されます。

```
service.beta.kubernetes.io/aws-load-balancer-type: "external"
service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: "instance"
```

デフォルトでは、Network Load Balancer は `internal` `aws-load-balancer-scheme` を使用して作成されます。内部 Network Load Balancer の場合、Amazon EKS クラスターは、VPC 内で少なくとも 1 つのプライベートサブネットを使用するように設定されている必要があります。Kubernetesはルートテーブルを調べサブネットがパブリックかプライベートかを特定します。パブリックサブネットには、インターネットゲートウェイを使ったインターネットへの直接の経路がありますが、プライベートサブネットにはありません。

パブリックサブネットに Network Load Balancer を作成して Amazon EC2 ノードへの負荷分散を行う場合、次のアノテーションを付けて `internet-facing` を指定します。

```
service.beta.kubernetes.io/aws-load-balancer-scheme: "internet-facing"
```

> **Important**
>
> サービスの作成後は、このアノテーションを編集しないでください。変更する必要がある場合は、サービスオブジェクトを削除し、このアノテーションを希望する値にして再度作成します。

(オプション) サンプルアプリケーションをデプロイする

前提条件

- クラスター VPC に少なくとも 1 つのパブリックサブネットまたはプライベートサブネットが存在する。

- クラスターにデプロイ済みの AWS Load Balancer Controller があること。詳細については、「AWS Load Balancer Controller とは」を参照してください。バージョン 2.7.2 以降をお勧めします。

サンプルアプリケーションをデプロイするには

1. Fargate にデプロイする場合は、VPC 内に利用可能なプライベートサブネットがあることを確認し、Fargate プロファイルを作成します。Fargate にデプロイしない場合は、このステップを省略してください。次のコマンドを実行するか、コマンドにある AWS Management Console と name に同じ値を使用して、namespace でプロファイルを作成できます。*example values* を自分の値に置き換えます。

```
eksctl create fargateprofile \
    --cluster my-cluster \
    --region region-code \
    --name nlb-sample-app \
    --namespace nlb-sample-app
```

2. サンプルアプリケーションをデプロイします。

   a. アプリケーションの名前空間を作成します。

```
kubectl create namespace nlb-sample-app
```

   b. 次の内容をコンピュータ上の *sample-deployment*.yaml という名前のファイルに保存します。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nlb-sample-app
  namespace: nlb-sample-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
```

```
    spec:
      containers:
        - name: nginx
          image: public.ecr.aws/nginx/nginx:1.23
          ports:
            - name: tcp
              containerPort: 80
```

c. マニフェストをクラスターに適用します。

```
kubectl apply -f sample-deployment.yaml
```

3. IP ターゲットに負荷分散を行うインターネット向け Network Load Balancer を使用して、サービスを作成します。

a. 次の内容をコンピュータ上の *sample-service*.yaml という名前のファイルに保存します。Fargate ノードにデプロイする場合は、service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing 行を削除します。

```
apiVersion: v1
kind: Service
metadata:
  name: nlb-sample-service
  namespace: nlb-sample-app
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-type: external
    service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: ip
    service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing
spec:
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
  type: LoadBalancer
  selector:
    app: nginx
```

b. マニフェストをクラスターに適用します。

```
kubectl apply -f sample-service.yaml
```

4. サービスがデプロイされたことを確認します。

```
kubectl get svc nlb-sample-service -n nlb-sample-app
```

出力例は次のとおりです。

```
NAME             TYPE           CLUSTER-IP         EXTERNAL-IP
                                                 PORT(S)        AGE
sample-service   LoadBalancer   10.100.240.137
 k8s-nlbsampl-nlbsampl-xxxxxxxxxx-xxxxxxxxxxxxxxxx.elb.region-code.amazonaws.com
  80:32400/TCP   16h
```

> **Note**
>
> *10.100.240.137* および *xxxxxxxxxx-[xxxxxxxxxxxxxxxx]* の値は出力例とは異なります (ロードバランサーに固有のものになります)。*[us-west-2]* は、クラスターがどの AWS リージョン にあるかによって異なる場合があります。

5. Amazon EC2 AWS Management Console を開きます。左側のナビゲーションペインで、([Load Balancing] (ロードバランシング) の下の) [Target Groups] (ターゲットグループ) を選択します。[名前] 列で、前のステップの出力である EXTERNAL-IP 列に入っている名前の一部と [Load balancer] 列の値が一致するターゲットグループの名前を選択します 。例えば、出力が以前の出力と同じ場合は、k8s-default-samplese-*xxxxxxxxxx* という名前のターゲットグループを選択します。[Target type] (ターゲットタイプ) は IP です。これは、サービスのサンプルマニフェストで指定されているためです。
6. [ターゲットグループ] を選択し、[ターゲット] タブを選択します。[登録済みターゲット] の下に、前のステップでデプロイされた 3 つのレプリカの 3 つの IP アドレスが表示されます。すべてのターゲットのステータスが [正常] になってから次に進みます。すべてのターゲットが healthy になるまで数分かかることがあります。ターゲットは、healthy 状態に変更される前の unhealthy 状態である可能性があります。
7. *xxxxxxxxxx-xxxxxxxxxxxxxxxx* および *us-west-2* を EXTERNAL-IP の前のステップの出力で返された値に置き換え、サービスにトラフィックを送信します。プライベートサブネットにデプロイした場合は、踏み台ホストなどの VPC 内のデバイスからページを表示する必要があります。詳細については、「AWS の Linux 踏み台ホスト」を参照してください。

```
curl k8s-default-samplese-xxxxxxxxxx-xxxxxxxxxxxxxxxx.elb.region-code.amazonaws.com
```

出力例は次のとおりです。

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
[...]
```

8. サンプルのデプロイ、サービス、および名前空間の使用が終了したら、それらを削除します。

```
kubectl delete namespace nlb-sample-app
```

Amazon EKS でのアプリケーション負荷分散

Kubernetes `ingress` を作成するとき、アプリケーショントラフィックの負荷分散を行う AWS Application Load Balancer (ALB) がプロビジョニングされます。詳細については、「Application Load Balancer ユーザーガイド」の「Application Load Balancer とは？」および「Kubernetes ドキュメント」の「イングレス」を参照してください。ALB は、ノードまたは AWS Fargate にデプロイされた Pods で使用できます。ALB をパブリックサブネットまたはプライベートサブネットにデプロイできます。

アプリケーショントラフィックは、OSI モデルの L7 で負荷分散されます。L4 でネットワークトラフィックの負荷分散をするには、`LoadBalancer` タイプの Kubernetes `service` をデプロイします。このタイプは、 AWS Network Load Balancer をプロビジョニングします。詳細については、「Amazon EKS でのネットワーク負荷分散」を参照してください。2 種類の負荷分散の違いについては、AWS ウェブサイトの「Elastic Load Balancing の特徴」を参照してください。

前提条件

あるアプリケーションに対してアプリケーショントラフィックの負荷分散を行うには、次の要件を満たす必要があります。

- 既存のクラスターがある。既存のクラスターがない場合は、「Amazon EKS の使用開始」を参照してください。既存のクラスターのバージョンを更新する必要がある場合は、Amazon EKS クラスターの Kubernetes バージョンの更新 を参照して下さい。
- クラスターにデプロイ済みの AWS Load Balancer Controller があること。詳細については、「AWS Load Balancer Controller とは」を参照してください。バージョン `2.7.2` 以降をお勧めします。

- 少なくとも 2 つのサブネットが異なるアベイラビリティーゾーンに存在している。AWS Load Balancer Controller は、各アベイラビリティーゾーンから 1 つのサブネットを選択します。タグ付けされたサブネットがアベイラビリティーゾーンで複数見つかった場合、コントローラーは、サブネット ID が辞書式順序で最初に来るサブネットを選択します。各サブネットには最低 8 個の利用可能な IP アドレスが必要です。

  ワーカーノードにアタッチされた複数のセキュリティグループを使用している場合は、次のように 1 つのセキュリティグループにタグを付ける必要があります。*my-cluster* をクラスター名に置き換えます。

  - キー – `kubernetes.io/cluster/`*my-cluster*
  - 値 – `shared` または `owned`

- AWS Load Balancer Controller のバージョン `2.1.1` 以前を使用している場合、サブネットに次のようにタグ付けする必要があります。バージョン `2.1.2` 以降を使用している場合、このタグ付けはオプションです。ただし、次のいずれかの場合は、サブネットにタグ付けることをお勧めします。同じVPCで実行されている複数のクラスターがあるか、VPCでサブネットを共有する複数の AWS サービスがあります。または、各クラスターに対してロードバランサーをプロビジョニングする場所を詳細に制御する必要があります。*my-cluster* をクラスター名に置き換えます。

  - キー – `kubernetes.io/cluster/`*my-cluster*
  - 値 – `shared` または `owned`

- パブリックサブネットとプライベートサブネットは次の要件を満たしている必要があります。サービスオブジェクトまたは Ingress オブジェクトのアノテーションとしてサブネット ID を明示的に指定しない限り、これは例外です。サービスオブジェクトまたは Ingress オブジェクトのアノテーションとしてサブネット ID を明示的に指定して、ロードバランサーをプロビジョニングすることを想定しています。このような状況では、Kubernetes と AWS ロードバランサーコントローラーは、これらのサブネットを直接使用してロードバランサーを作成します。次のタグは必要ありません。

  - プライベートサブネット — 次の形式でタグ付けする必要があります。これは、サブネットを内部ロードバランサーに使用できることを、Kubernetes と AWS ロードバランサーコントローラーが認識できるようにするためです。`eksctl` または Amazon EKS AWS CloudFormation テンプレートを使用して、2020 年 3 月 26 日以降に VPC を作成する場合、サブネットは、作成時に適切にタグ付けされます。Amazon EKS AWS CloudFormation VPC テンプレートの詳細については、「Amazon EKS クラスター VPC の作成」を参照してください。

    - キー – `kubernetes.io/role/internal-elb`
    - 値 – `1`

- パブリックサブネット — 次の形式でタグ付けする必要があります。これは、外部ロードバランサー用に指定されたサブネットのみが使用できることを、Kubernetes が認識できるようにするためです。この方法では、Kubernetes は各アベイラビリティーゾーンでパブリックサブネットを (サブネット ID に基づいて辞書順に) 選択しません。`eksctl` または Amazon EKS AWS CloudFormation テンプレートを使用して、2020 年 3 月 26 日以降に VPC を作成する場合、サブネットは、作成時に適切にタグ付けされます。Amazon EKS AWS CloudFormation VPC テンプレートの詳細については、「Amazon EKS クラスター VPC の作成」を参照してください。
  - キー – `kubernetes.io/role/elb`
  - 値 – `1`

サブネットロールタグが明示的に追加されていない場合、Kubernetes サービスコントローラーはクラスター VPC サブネットのルートテーブルを調べます。これは、サブネットがプライベートかパブリックかを判断するためです。この動作に頼らないことをお勧めします。代わりに、プライベートまたはパブリックロールタグを明示的に追加します。AWS Load Balancer Controller は、ルートテーブルを精査しません。また、自動検出を正常に実行するには、プライベートタグとパブリックタグが必要です。

考慮事項

- Kubernetes のイングレスリソースが `kubernetes.io/ingress.class: alb` アノテーションを使用してクラスターで作成されるたびに、AWS Load Balancer Controller により ALB およびサポートされる必要な AWS リソースが作成されます。イングレスリソースは、HTTP または HTTPS トラフィックをクラスター内の異なる Pods にルーティングするように ALB を設定します。イングレスオブジェクトに AWS Load Balancer Controller を確実に使用させるには、Kubernetes イングレス仕様に次のアノテーションを追加します。詳細については、GitHub の「[Ingress specification]」(イングレス仕様) を参照してください。

```
annotations:
    kubernetes.io/ingress.class: alb
```

> **Note**
>
> IPv6 Pods へのロードバランサーを行う場合は、イングレス仕様に次のアノテーションを追加します。IPv6 を使用して負荷分散を行えるのは、IP ターゲットにのみです。インスタンスターゲットには行えません。このアノテーションを使用しない場合、負荷分散には IPv4 が使用されます。

```
alb.ingress.kubernetes.io/ip-address-type: dualstack
```

- AWS Load Balancer Controller は、以下のトラフィックモードをサポートしています。
  - インスタンス – クラスター内のノードを ALB のターゲットとして登録します。ALB に到達するトラフィックは、サービスの NodePort にルーティングされてから、Pods にプロキシされます。これがデフォルトのトラフィックモードです。alb.ingress.kubernetes.io/target-type: instance の注釈を使用して明示的に指定することもできます。

    > **Note**
    >
    > このトラフィックモードを使用するには、Kubernetes サービスで NodePort または「LoadBalancer」 タイプを指定する必要があります。

  - IP - Pods を ALB のターゲットとして登録します。ALB に到達するトラフィックは、サービスの Pods に直接ルーティングされます。このトラフィックモードを使用するには、alb.ingress.kubernetes.io/target-type: ip の注釈を指定する必要があります。IP ターゲットタイプは、ターゲットの Pods が Fargate で実行されている場合に必要です。
- コントローラーによって作成された ALB にタグを付けるには、次のアノテーションをコントローラーに追加します: alb.ingress.kubernetes.io/tags。AWS Load Balancer Controller でサポートされるすべての使用可能なアノテーションのリストについては、「GitHub」の「[Ingress annotations]」(イングレスアノテーション) を参照してください。
- ALB コントローラーのバージョンをアップグレードまたはダウングレードすると、ALB コントローラーのバージョンに依存する機能が大きく変更される可能性があります。各リリースで導入された新しい変更の詳細については、「GitHub」の「ALB controller release noted」(ALB コントローラーリリースノート) を参照してください。

IngressGroups を使用して複数のサービスのリソース間でアプリケーションロードバランサーを共有するには

イングレスをグループに参加させるには、Kubernetes のイングレスリソース仕様に次のアノテーションを追加します。

```
alb.ingress.kubernetes.io/group.name: my-group
```

グループ名は、次のようにする必要があります。

- 長さが 63 文字以下。
- 名前は、小文字の英文字、-、および . で構成されます。
- 数字または文字で始めます。

コントローラーは、同じイングレスグループ内のすべてのイングレスのイングレスルールを自動的にマージします。これは、単一のALBでそれらをサポートしています。イングレスで定義されたほとんどのアノテーションは、そのイングレスで定義されたパスにのみ適用されます。デフォルトでは、イングレスリソースはどのイングレスグループにも属していません。

> **Warning**
>
> 潜在的なセキュリティリスク: イングレスリソースを作成または変更する RBAC 権限を持つすべての Kubernetes ユーザーが同じ信頼境界内にいる場合にのみ、イングレスにイングレスグループを指定します。アノテーションをグループ名で追加すると、他の Kubernetes ユーザーが、同じイングレスグループに属するイングレスを作成または変更する可能性があります。これにより、優先度の高いルールで既存のルールを上書きするなど、望ましくない動作が発生する可能性があります。

イングレスリソースの順序番号を追加できます。

```
alb.ingress.kubernetes.io/group.order: '10'
```

番号は 1 ~ 1000 の範囲で指定できます。同じイングレスグループ内のすべてのイングレスについて、最小の番号を持つものが最初に評価されます。このアノテーションがないイングレスはすべて、値 0 で評価されます。番号が小さいルールと番号が大きいルールが重複すると、番号が小さいルールが上書きされる可能性があります。デフォルトでは、同じイングレスグループ内のイングレス間のルールの順序は、名前空間と名前に基づき辞書式順序で決定されます。

> **Important**
>
> 同じイングレスグループの各イングレスに、一意のプライオリティ番号があることを確認します。　イングレス間で重複する順序番号を持つことはできません。

(オプション) サンプルアプリケーションをデプロイする

前提条件

- クラスター VPC に少なくとも 1 つのパブリックサブネットまたはプライベートサブネットが存在する。
- クラスターにデプロイ済みの AWS Load Balancer Controller があること。詳細については、「AWS Load Balancer Controller とは」を参照してください。バージョン 2.7.2 以降をお勧めします。

サンプルアプリケーションをデプロイするには

Amazon EC2 ノード、Fargate Pods、またはその両方を持つクラスターで、サンプルアプリケーションを実行できます。

1. Fargate にデプロイしない場合は、このステップを省略してください。Fargate にデプロイする場合は、Fargate プロファイルを作成します。次のコマンドを実行するか、コマンドにある AWS Management Console と `name` に同じ値を使用して、`namespace` でプロファイルを作成できます。*`example values`* を自分の値に置き換えます。

   ```
   eksctl create fargateprofile \
       --cluster my-cluster \
       --region region-code \
       --name alb-sample-app \
       --namespace game-2048
   ```

2. サンプルアプリケーションとして、ゲーム 2048 をデプロイし、AWS Load Balancer Controller が Ingress オブジェクトの結果として AWS ALB を作成することを確認します。デプロイ先のサブネットのタイプに関する手順を実行します。

   a. IPv6 ファミリーを使用して作成したクラスター内の Pods にデプロイしている場合には、次のステップに進みます。

      - Public

        ```
        kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.7.2/docs/examples/2048/2048_full.yaml
        ```

      - プライベート

1. マニフェストをダウンロードします。

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.7.2/docs/examples/2048/2048_full.yaml
```

2. ファイルを編集して、`alb.ingress.kubernetes.io/scheme: internet-facing` と記述された行を探します。
3. *`internet-facing`* を **`internal`** に変更し、ファイルを保存します。
4. マニフェストをクラスターに適用します。

```
kubectl apply -f 2048_full.yaml
```

b. IPv6 ファミリー を使用して作成したクラスター内の Pods にデプロイする場合は、次のステップを完了します。

1. マニフェストをダウンロードします。

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.7.2/docs/examples/2048/2048_full.yaml
```

2. エディタでファイルを開き、イングレス仕様のアノテーションに次の行を追加します。

```
alb.ingress.kubernetes.io/ip-address-type: dualstack
```

3. Pods に向いているインターネットではなく、内部の Pods へのロードバランサーを行う場合は、`alb.ingress.kubernetes.io/scheme: *internet-facing*` と書いてある行を `alb.ingress.kubernetes.io/scheme: **internal**` に変更します。
4. ファイルを保存します。
5. マニフェストをクラスターに適用します。

```
kubectl apply -f 2048_full.yaml
```

3. 数分後、次のコマンドを使用して、イングレスリソースが作成されたことを確認します。

```
kubectl get ingress/ingress-2048 -n game-2048
```

出力例は次のとおりです。

```
NAME           CLASS    HOSTS   ADDRESS
                       PORTS   AGE
ingress-2048   <none>   *       k8s-game2048-ingress2-xxxxxxxxxx-yyyyyyyyyy.region-
code.elb.amazonaws.com   80      2m32s
```

> **Note**
>
> プライベートサブネットでロードバランサーを作成した場合、前の出力の ADDRESS の下の値の前に `internal-` が付けられます。
> 数分後、イングレスが正常に作成されていない場合は、次のコマンドを実行して AWS Load Balancer Controller のログを表示します。これらのログには、デプロイに関する問題の診断に使用できるエラーメッセージが含まれている場合があります。
>
> ```
> kubectl logs -f -n kube-system -l app.kubernetes.io/instance=aws-load-
> balancer-controller
> ```

4. パブリックサブネットにデプロイした場合は、ブラウザを開き、前のコマンドの出力から ADDRESS URL に移動してサンプルアプリケーションを表示します。何も表示されない場合は、ブラウザを更新してからもう一度試してください。プライベートサブネットにデプロイした場合は、踏み台ホストなどの VPC 内のデバイスからページを表示する必要があります。詳細については、「AWS の Linux 踏み台ホスト」を参照してください。
5. サンプルアプリケーションの試用が終了したら、次のコマンドのいずれかを実行して削除します。

   - ダウンロードしたコピーを適用するのではなくマニフェストを適用した場合は、次のコマンドを使用します。

     ```
     kubectl delete -f https://raw.githubusercontent.com/kubernetes-sigs/aws-load-
     balancer-controller/v2.7.2/docs/examples/2048/2048_full.yaml
     ```

   - マニフェストをダウンロードして編集した場合は、次のコマンドを使用します。

     ```
     kubectl delete -f 2048_full.yaml
     ```

サービスに割り当てることができる外部 IP アドレスを制限する

Kubernetes サービスには、クラスターの内部から次を経由することで到達できます。

- Kubernetes によって自動的に割り当てられるクラスター IP アドレス
- サービス仕様で `externalIPs` プロパティに指定した任意の IP アドレス 外部IPアドレスは Kubernetes によって管理されておらず、クラスター管理者の責任です。`externalIPs` で指定される外部 IP アドレスは、クラウドプロバイダーによって提供される `LoadBalancer` タイプのサービスに割り当てられる外部 IP アドレスとは異なります。

Kubernetes サービスの詳細については、「Kubernetes ドキュメント」の「サービス」を参照してください。サービス仕様で `externalIPs` に指定できる IP アドレスを制限できます。

サービス仕様で **`externalIPs`** に指定できる IP アドレスを制限するには

1. webhook の証明書を管理するには、`cert-manager` をデプロイします。詳細については、`cert-manager` ドキュメントを参照してください。

   ```
   kubectl apply -f https://github.com/jetstack/cert-manager/releases/download/v1.5.4/cert-manager.yaml
   ```

2. `cert-manager` Pods が実行中であることを確認します。

   ```
   kubectl get pods -n cert-manager
   ```

   出力例は次のとおりです。

   ```
   NAME                                       READY   STATUS    RESTARTS   AGE
   cert-manager-58c8844bb8-nlx7q              1/1     Running   0          15s
   cert-manager-cainjector-745768f6ff-696h5   1/1     Running   0          15s
   cert-manager-webhook-67cc76975b-4v4nk      1/1     Running   0          14s
   ```

3. 既存のサービスを確認して、アドレスを制限する CIDR ブロックに含まれない外部 IP アドレスが割り当てられていないことを確認します。

   ```
   kubectl get services -A
   ```

   出力例は次のとおりです。

```
NAMESPACE                       NAME                                        TYPE
   CLUSTER-IP       EXTERNAL-IP     PORT(S)         AGE
cert-manager                    cert-manager                                ClusterIP
   10.100.102.137   <none>          9402/TCP        20m
cert-manager                    cert-manager-webhook                        ClusterIP
   10.100.6.136     <none>          443/TCP         20m
default                         kubernetes                                  ClusterIP
   10.100.0.1       <none>          443/TCP         2d1h
externalip-validation-system    externalip-validation-webhook-service       ClusterIP
   10.100.234.179   <none>          443/TCP         16s
kube-system                     kube-dns                                    ClusterIP
   10.100.0.10      <none>          53/UDP,53/TCP   2d1h
my-namespace                    my-service                                  ClusterIP
   10.100.128.10    192.168.1.1     80/TCP          149m
```

いずれかの値が、アクセスを制限するブロック内にない IP アドレスである場合は、ブロック内に存在するアドレスを変更し、サービスを再びデプロイする必要があります。例えば、前の出力の `my-service` サービスには、ステップ 5 の CIDR ブロックの例に含まれていない外部 IP アドレスが割り当てられています。

4. 外部 IP のウェブフックマニフェストをダウンロードします。また、GitHub で ウェブフックのソースコード を確認できます。

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/docs/externalip-webhook.yaml
```

5. CIDR ブロックを指定します。ダウンロードしたファイルを エディタで開き、次の行の最初の # を削除します。

```
#args:
#- --allowed-external-ip-cidrs=10.0.0.0/8
```

`10.0.0.0/8` を独自の CIDR ブロックに置き換えます。ブロックはいくつでも指定できます。複数のブロックを指定する場合は、ブロック間にコンマを追加します。

6. クラスターが `us-west-2` AWS リージョン に存在しない場合、ファイル内の `us-west-2`、`602401143452`、および `amazonaws.com` を次のコマンドに。コマンドを実行する前に、*`region-code`* と *`111122223333`* を Amazon コンテナイメージレジストリ のリストにある AWS リージョン の値に置き換えてください。

```
sed -i.bak -e 's|602401143452|111122223333|' externalip-webhook.yaml
sed -i.bak -e 's|us-west-2|region-code|' externalip-webhook.yaml
sed -i.bak -e 's|amazonaws.com||' externalip-webhook.yaml
```

7. マニフェストをクラスターに適用します。

```
kubectl apply -f externalip-webhook.yaml
```

externalIPs 用に指定された IP アドレスを使用して、サービスをクラスターにデプロイしようとするとき、それが「CIDR ブロックを指定します」ステップ で指定したブロックに含まれていない場合は失敗します。

あるリポジトリから別のリポジトリにコンテナイメージをコピーする

このトピックでは、ノードがアクセスできないリポジトリからコンテナイメージをプルし、ノードがアクセスできるリポジトリにイメージをプッシュする方法について説明します。イメージを、Amazon ECR またはノードがアクセスできる代替リポジトリにプッシュできます。

前提条件

- コンピューターにインストールおよび設定された Docker エンジン。詳細については、Docker ドキュメントの「Docker エンジンのインストール」を参照してください。
- ご使用のデバイスまたは AWS CloudShell で、バージョン 2.12.3 以降、または AWS Command Line Interface (AWS CLI) のバージョン 1.27.160 以降がインストールおよび設定されていること。現在のバージョンを確認するには、「**aws --version | cut -d / -f2 | cut -d ' ' -f1**」を参照してください。macOS の yum、apt-get、または Homebrew などのパッケージマネージャは、AWS CLI の最新バージョンより数バージョン遅れることがあります。最新バージョンをインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI のインストール、更新、およびアンインストール」と「aws configure でのクイック設定」を参照してください。AWS CloudShell にインストールされている AWS CLI バージョンは、最新バージョンより数バージョン遅れている可能性もあります。更新するには、「AWS CloudShellユーザーガイド」の「ホームディレクトリへの AWS CLI のインストール」を参照してください。
- ノードでAmazon のネットワークを介してプライベート Amazon ECR リポジトリからコンテナイメージのプルを行ったり、コンテナイメージをプッシュする場合は、Amazon ECR のインター

フェイス VPC エンドポイント。詳細については、Amazon Elastic コンテナレジストリ ユーザーガイドの「Amazon ECR 用の VPC エンドポイントを作成する」を参照してください。

リポジトリからコンテナイメージをプルし、独自のリポジトリにプッシュする場合は、次の手順を完了してください。このトピックで紹介する次の例では、Amazon VPC CNI plugin for Kubernetes メトリクスヘルパー のイメージがプルされます。これらのステップを実行する場合は、必ず *example values* をユーザー自身の値に置き換えてください。

あるリポジトリから別のリポジトリにコンテナイメージをコピーする

1. Amazon ECR リポジトリまたは別のリポジトリがまだ作成されていない場合は、ノードがアクセスできるリポジトリを作成します。次のコマンドは、Amazon ECR プライベートリポジトリを作成します。Amazon ECR プライベートリポジトリ名は文字で始まる必要があります。小文字、数字、ハイフン (-)、アンダースコア (_)、フォワードスラッシュ (/) のみ含むことができます。詳細については、Amazon Elastic Container Registry ユーザーガイドの「プライベートリポジトリの作成」を参照してください。

   cni-metrics-helper は、ユーザーが任意で選択する名前に置き換えることができます。ベストプラクティスとして、各イメージに個別のリポジトリを作成します。イメージタグはリポジトリ内で一意である必要があるため、この方法をお勧めします。*region-code* を Amazon ECR でサポートされている AWS リージョン に置き換えます。

   ```
   aws ecr create-repository --region region-code --repository-name cni-metrics-helper
   ```

2. ノードがプルする必要があるイメージのレジストリ、リポジトリ、およびタグ (オプション) を決めます。この情報は `registry/repository[:tag]` 形式です。

   Amazon EKS のイメージのインストールに関するトピックの多くは、マニフェストファイルを適用するか、Helm チャートを使用してイメージをインストールする必要があります。ただし、マニフェストファイルの適用や、Helm チャートをインストールする前に、まずマニフェストやチャートの `values.yaml` ファイルの内容を確認してください。そうすることで、プルするレジストリ、リポジトリ、およびタグを決定することができます。

   例えば、Amazon VPC CNI plugin for Kubernetes メトリクスヘルパーのマニフェストファイルに次の行があります。`602401143452.dkr.ecr.us-west-2.amazonaws.com` がレジストリで、これは Amazon ECR のプライベートレジストリです。`cni-metrics-helper` が、レポジトリです。

```
image: "602401143452.dkr.ecr.us-west-2.amazonaws.com/cni-metrics-helper:v1.12.6"
```

イメージの場所として、次のバリエーションがあります：

- `repository-name:tag` のみ。この場合、`docker.io` は通常レジストリですが、レジストリが指定されていない場合、デフォルトで Kubernetes がリポジトリ名の前に追加するため、指定されていません。
- `repository-name/repository-namespace/repository:tag`. リポジトリの名前空間は任意ですが、イメージを分類するためにリポジトリ所有者が指定される場合があります。例えば、Amazon ECR Public Gallery にあるすべての Amazon EC2 イメージは、`aws-ec2` 名前空間を使用します。

Helm を使用してイメージをインストールする前に、Helm `values.yaml` ファイルを表示してイメージの場所を決定します。例えば、Amazon VPC CNI plugin for Kubernetes メトリクスヘルパーの `values.yaml` ファイルには次の行が含まれています。

```
image:
  region: us-west-2
  tag: v1.12.6
  account: "602401143452"
  domain: "amazonaws.com"
```

3. マニフェストファイルで指定されたコンテナイメージをプルします。

   a. Amazon ECR Public Gallery など、パブリックレジストリからプルする場合は、認証が不要なため、次のサブステップにスキップします。この例では、CNI メトリクスヘルパーイメージのリポジトリを含む Amazon ECR プライベートレジストリに対して認証します。Amazon EKS は、Amazon コンテナイメージレジストリ に記載されている各レジストリにイメージを保持します。*602401143452* と *region-code* を別のレジストリの情報に置き換えることで、任意のレジストリに対して認証が可能です。Amazon EKS がサポートされている AWS リージョン には、それぞれに個別のレジストリが存在します。

   ```
   aws ecr get-login-password --region region-code | docker login --username AWS --
   password-stdin 602401143452.dkr.ecr.region-code.amazonaws.com
   ```

   b. イメージをプルする。この例では、前のサブステップで認証したレジストリからプルします。*602401143452* と *region-code* を前のサブステップで指定した情報に置き換えます。

```
docker pull 602401143452.dkr.ecr.region-code.amazonaws.com/cni-metrics-
helper:v1.12.6
```

4. プルしたイメージに、レジストリ、リポジトリ、タグでタグ付けします。次の例では、マニフェストファイルからイメージをプルし、最初のステップで作成した Amazon ECR プライベートリポジトリにイメージをプッシュすることを前提としています。*111122223333* をアカウントIDに置き換えます。*region-code* を、Amazon ECR プライベートリポジトリを作成した AWS リージョン に置き換えます。

```
docker tag cni-metrics-helper:v1.12.6 111122223333.dkr.ecr.region-
code.amazonaws.com/cni-metrics-helper:v1.12.6
```

5. レジストリに対して認証します。この例では、最初のステップで作成した Amazon ECR プライベートレジストリに対して認証します。詳細については、Amazon Elastic Container Registry ユーザーガイドの「レジストリの認証」を参照してください。

```
aws ecr get-login-password --region region-code | docker login --username AWS --
password-stdin 111122223333.dkr.ecr.region-code.amazonaws.com
```

6. リポジトリにプッシュするイメージを付けます。この例では、最初のステップで作成した Amazon ECR プライベートリポジトリにイメージをプッシュします。詳細については、「Amazon Elastic Container Registry ユーザーガイド」の「Docker イメージのプッシュ」を参照してください。

```
docker push 111122223333.dkr.ecr.region-code.amazonaws.com/cni-metrics-
helper:v1.12.6
```

7. プッシュしたイメージ用の `registry/repository:tag` を使用して前のステップでイメージの決定に使用したマニフェストファイルを更新します。Helm チャートを使用してインストールする場合、`registry/repository:tag` を指定するオプションが多くあります。チャートをインストールする場合、リポジトリにプッシュしたイメージの `registry/repository:tag` を指定します。

Amazon コンテナイメージレジストリ

AWS Amazon EKS アドオンをクラスターにデプロイすると、ノードはアドオンのインストールメカニズムで指定されたレジストリ (インストールマニフェストや Helm `values.yaml` ファイルなど) から必要なコンテナイメージを取得します。画像は、Amazon EKS Amazon ECR プライベートリポ

ジトリからプルします。Amazon EKS は、Amazon EKSがサポートする各 AWS リージョン のリポジトリにイメージをレプリケートします。ノードは、次のレジストリのいずれかからインターネット経由でコンテナイメージをプルできます。または、VPC 内に Amazon ECR (AWS PrivateLink) 用のインターフェイス VPC エンドポイントを作成した場合、ノードは Amazon のネットワーク経由でイメージをプルできます。レジストリには、AWS IAM アカウントでの認証が必要です。ノードは、Amazon EKS ノードの IAM ロールを使用して認証します。このロールには、関連付けられた AmazonEC2ContainerRegistryReadOnly で管理される IAM ポリシーのアクセス許可があります。

AWS リージョン	レジストリ
af-south-1	877085696533.dkr.ecr.af-south-1.amazonaws.com
ap-east-1	800184023465.dkr.ecr.ap-east-1.amazonaws.com
ap-northeast-1	602401143452.dkr.ecr.ap-northeast-1.amazonaws.com
ap-northeast-2	602401143452.dkr.ecr.ap-northeast-2.amazonaws.com
ap-northeast-3	602401143452.dkr.ecr.ap-northeast-3.amazonaws.com
ap-south-1	602401143452.dkr.ecr.ap-south-1.amazonaws.com
ap-south-2	900889452093.dkr.ecr.ap-south-2.amazonaws.com
ap-southeast-1	602401143452.dkr.ecr.ap-southeast-1.amazonaws.com
ap-southeast-2	602401143452.dkr.ecr.ap-southeast-2.amazonaws.com
ap-southeast-3	296578399912.dkr.ecr.ap-southeast-3.amazonaws.com

AWS リージョン	レジストリ
ap-southeast-4	491585149902.dkr.ecr.ap-southeast-4.amazonaws.com
ca-central-1	602401143452.dkr.ecr.ca-central-1.amazonaws.com
ca-west-1	761377655185.dkr.ecr.ca-west-1.amazonaws.com
cn-north-1	918309763551.dkr.ecr.cn-north-1.amazonaws.com.cn
cn-northwest-1	961992271922.dkr.ecr.cn-northwest-1.amazonaws.com.cn
eu-central-1	602401143452.dkr.ecr.eu-central-1.amazonaws.com
eu-central-2	900612956339.dkr.ecr.eu-central-2.amazonaws.com
eu-north-1	602401143452.dkr.ecr.eu-north-1.amazonaws.com
eu-south-1	590381155156.dkr.ecr.eu-south-1.amazonaws.com
eu-south-2	455263428931.dkr.ecr.eu-south-2.amazonaws.com
eu-west-1	602401143452.dkr.ecr.eu-west-1.amazonaws.com
eu-west-2	602401143452.dkr.ecr.eu-west-2.amazonaws.com
eu-west-3	602401143452.dkr.ecr.eu-west-3.amazonaws.com

AWS リージョン	レジストリ
il-central-1	066635153087.dkr.ecr.il-central-1.amazonaws.com
me-south-1	558608220178.dkr.ecr.me-south-1.amazonaws.com
me-central-1	759879836304.dkr.ecr.me-central-1.amazonaws.com
sa-east-1	602401143452.dkr.ecr.sa-east-1.amazonaws.com
us-east-1	602401143452.dkr.ecr.us-east-1.amazonaws.com
us-east-2	602401143452.dkr.ecr.us-east-2.amazonaws.com
us-gov-east-1	151742754352.dkr.ecr.us-gov-east-1.amazonaws.com
us-gov-west-1	013241004608.dkr.ecr.us-gov-west-1.amazonaws.com
us-west-1	602401143452.dkr.ecr.us-west-1.amazonaws.com
us-west-2	602401143452.dkr.ecr.us-west-2.amazonaws.com

Amazon EKS アドオン

アドオンは、Kubernetes アプリケーションにサポート操作機能を提供するソフトウェアですが、アプリケーションに固有のものではありません。これには、クラスターがネットワーク、コンピューティング、およびストレージの基盤となる AWS リソースと対話できるようにするオブザーバビリティエージェントや Kubernetes ドライバーなどのソフトウェアが含まれます。アドオンソフト

ウェアは、通常、Kubernetes コミュニティや AWS のようなクラウドプロバイダー、またはサードパーティーベンダーによって構築、保持されます。Amazon EKS は、Amazon VPC CNI plugin for Kubernetes、kube-proxy、および CoreDNS などのセルフマネージド型アドオンを、すべてのクラスターで自動的にインストールします。必要に応じて、アドオンのデフォルト設定は変更と更新が可能です。

Amazon EKS アドオンは、Amazon EKS クラスター用に厳選されたアドオンセットのインストールと管理を実行します。Amazon EKS アドオンには、最新のセキュリティパッチ、バグ修正が含まれており、また Amazon EKS と連携するために AWS により検証されています。Amazon EKS アドオンを使用すると、Amazon EKS クラスターの安全性と安定性を一貫して保証でき、アドオンのインストール、設定、および更新に必要となる作業量を削減できます。セルフマネージド型のアドオン (kube-proxy など) がすでにクラスターで実行されており、Amazon EKS アドオンとして利用可能な場合、kube-proxy Amazon EKS アドオンをインストールすれば、その機能からのメリットを活用できるようになります。

Amazon EKS API を使用することで、Amazon EKS アドオンのための、Amazon EKS 管理の特定の設定フィールドを更新できます。Amazon EKS で管理されていない設定フィールドは、アドオンの起動後に Kubernetes クラスター内で直接変更することもできます。これには、アドオン向けの特定の設定フィールドの定義が含まれます (該当する場合)。これらの変更は、一度実行されると Amazon EKS によって上書きされることはありません。これは、Kubernetes によるサーバー側の適用機能を使用して実現しています。詳細については、「Kubernetes フィールド管理」を参照してください。

Amazon EKS アドオンは、どの Amazon EKS ノードタイプでも使用できます。

考慮事項

- クラスターでアドオンの設定を行うには、アドオンを操作するための IAM 許可を、IAM プリンシパルが持っている必要があります。詳細については、「Amazon Elastic Kubernetes Service で定義されるアクション」で、Addon の名前ごとにアクションを参照してください。
- Amazon EKS アドオンは、クラスター用にプロビジョニングまたは設定するノードで実行されます。ノードタイプには、Amazon EC2 インスタンスと Fargate が含まれます。
- Amazon EKS で管理されていないフィールドを変更して、Amazon EKS アドオンのインストールをカスタマイズできます。詳細については、「Kubernetes フィールド管理」を参照してください。
- AWS Management Console を使用してクラスターを作成すると、Amazon EKS kube-proxy、Amazon VPC CNI plugin for Kubernetes、および CoreDNS Amazon EKS アドオンが自動的にクラスターに追加されます。また、eksctl を使用して config ファイルでクラスターを作成する場合は、eksctl は Amazon EKS アドオンを持つクラスターを作成できます。config

ファイルなしで eksctl を使用して、または他のツールを使用して、クラスターを作成する場合は、Amazon EKS アドオンではなく、セルフマネージド型の kube-proxy、Amazon VPC CNI plugin for Kubernetes、および CoreDNS のアドオンがインストールされます。これらは自分で管理することが可能で、また、クラスターの作成後に Amazon EKS アドオンを手動で追加することもできます。

- eks:addon-cluster-admin ClusterRoleBinding は cluster-admin ClusterRole を eks:addon-manager Kubernetes ID にバインドします。このロールには、eks:addon-manager ID が Kubernetes 名前空間を作成し、アドオンを名前空間にインストールするために必要なアクセス許可があります。eks:addon-cluster-admin ClusterRoleBinding が削除されても、Amazon EKS クラスターは引き続き機能しますが、Amazon EKS はアドオンを管理できなくなります。次のプラットフォームバージョンから始まるすべてのクラスターは、新しい ClusterRoleBinding バージョンを使用します。

Kubernetes バージョン	プラットフォームのバージョン
1.28	eks.12
1.27	eks.14
1.26	eks.29
1.25	eks.35
1.24	eks.43

Amazon EKS アドオンは、Amazon EKS API、AWS Management ConsoleAWS CLI、および eksctl を使用して、追加、更新、または削除ができます。詳細については、「Amazon EKS アド

オンの管理」を参照してください。Amazon EKS アドオンは、AWS CloudFormation を使用しても作成できます。

Amazon EKS で利用可能な Amazon EKS アドオン

以下の Amazon EKS アドオンをクラスターで作成できます。`eksctl`、AWS Management Console、または AWS CLI を使用して、使用可能なアドオンの最新リストをいつでも表示できます。使用可能なすべてのアドオンを確認したり、アドオンをインストールしたりするには、アドオンの作成 を参照してください。アドオンが IAM アクセス権限を要求する場合、クラスター用に IAM OpenID Connect (OIDC) プロバイダーが必要です。既に存在しているかどうかを確認する、または作成するには「クラスターの IAM OIDC プロバイダーを作成する」を参照してください。アドオンはインストール後に、更新または削除できます。

アドオンを選択すると、そのアドオンとそのインストール要件について詳しく知ることができます。

Amazon VPC CNI plugin for Kubernetes

- 名前 – `vpc-cni`
- 説明 — クラスターにネイティブ VPC ネットワークを提供するKubernetes コンテナネットワークインターフェイス (CNI) プラグイン。このアドオンのセルフマネージドタイプまたはマネージドタイプが、デフォルトで各 Amazon EC2 ノードにインストールされます。
- 必要な IAM アクセス権限 — このアドオンは Amazon EKS の サービスアカウントの IAM ロール機能を利用します。クラスターが `IPv4` ファミリーを使用する場合は、AmazonEKS_CNI_Policy のアクセス権限が必要です。クラスターが `IPv6` ファミリーを使用する場合は、IPv6 モードのアクセス許可を持つ IAM ポリシーを作成する必要があります。次のコマンドで、IAM ロールを作成し、そのロールにポリシーの 1 つをアタッチして、アドオンが使用する Kubernetes サービスアカウントに注釈を付けることができます。

  `my-cluster` をクラスターの名前に、*`AmazonEKSVPCCNIRole`* をロールに使用する名前に置き換えます。クラスターが `IPv6` ファミリを使用する場合は、*`AmazonEKS_CNI_Policy`* を作成したポリシーの名前に置き換えます。このコマンドを使用するには、デバイスに `eksctl` がインストールされている必要があります。別のツールを使用してロールを作成し、ポリシーをアタッチして、Kubernetes サービスアカウントに注釈を付ける必要がある場合は、「IAM ロールを引き受けるように Kubernetes サービスアカウントを設定する」を参照してください。

```
eksctl create iamserviceaccount --name aws-node --namespace kube-system --cluster my-
cluster --role-name AmazonEKSVPCCNIRole \
```

```
    --role-only --attach-policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy --
approve
```

- 追加情報 — アドオンの構成可能な設定の詳細については、GitHub の「aws-vpc-cni-k8s」を参照してください。プラグインの詳細については、「提案: AWS VPC 経由の Kubernetes ネットワーク用 CNI プラグイン」を参照してください。アドオンの作成方法の詳細については、「Amazon EKS アドオンの作成」を参照してください。
- 更新情報 – 一度に更新できるマイナーバージョンは 1 つのみです。例えば、現在のバージョンが `1.28.x-eksbuild.y` で、これを `1.30.x-eksbuild.y` に更新する場合は、現在のバージョンを `1.29.x-eksbuild.y` に更新してから、再度 `1.30.x-eksbuild.y` に更新する必要があります。アドオンの更新の詳細については、「Amazon EKS アドオンの更新」を参照してください。

CoreDNS

- 名前 – `coredns`
- 説明 – Kubernetes クラスター DNS として機能する、柔軟で拡張可能な DNS サーバーです。このアドオンのセルフマネージドタイプまたはマネージドタイプが、クラスターの作成時にデフォルトでインストールされました。1 つ以上のノードで Amazon EKS クラスターを起動すると、クラスターにデプロイされたノード数に関係なく、デフォルトで CoreDNS イメージの 2 つのレプリカがデプロイされます。CoreDNS の Pods は、クラスター内のすべての Pods の名前解決を行います。クラスターに AWS Fargate プロファイル が含まれ、名前空間が CoreDNS `deployment` の名前空間と一致している場合、CoreDNS Pods を Fargate ノードにデプロイできます。
- 必要な IAM アクセス権限 — このアドオンにはアクセス権限は必要ありません。
- 追加情報 — CoreDNS の詳細については、Kubernetesドキュメントの「Using CoreDNS for Service」(サービス検出のための CoreDNS の使用) と「Customizing DNS Service」(DNS サービスのカスタマイズ) を参照してください。

`Kube-proxy`

- 名前 – `kube-proxy`
- 説明 – 各 Amazon EC2 ノードのネットワークルールを管理します。これにより、Pods とのネットワーク通信が可能になります。このアドオンのセルフマネージドタイプまたはマネージドタイプが、デフォルトでクラスター内の各 Amazon EC2 ノードにインストールされます。
- 必要な IAM アクセス権限 — このアドオンにはアクセス権限は必要ありません。
- 追加情報 — `kube-proxy` の詳細については、Kubernetes ドキュメントの `kube-proxy` を参照してください。

- 更新情報 — 現在のバージョンを更新する前に、次の要件を考慮してください。
  - Amazon EKS クラスターの Kube-proxy は、Kubernetes と同じ互換性およびスキューポリシーが適用されます。
  - Kube-proxy は、Amazon EC2 ノードの kubelet と同じマイナーバージョンである必要があります。
  - Kube-proxy は、クラスターのコントロールプレーンのマイナーバージョンよりも新しいものにすることはできません。
  - Amazon EC2 ノードの kube-proxy バージョンは、コントロールプレーンより 2 つ以上前のマイナーバージョンにすることはできません。例えば、コントロールプレーンが Kubernetes 1.30 を実行している場合、kube-proxy マイナーバージョンは 1.28 より前にすることはできません。
  - 最近クラスターを新しい Kubernetes マイナーバージョンに更新した場合、Amazon EC2 ノードを同じマイナーバージョンに更新した後で、kube-proxy をノードと同じマイナーバージョンに更新してください。

Amazon EBS CSI ドライバー

- 名前 – aws-ebs-csi-driver
- 説明 — クラスターに Amazon EBS ストレージを提供する、Kubernetes コンテナストレージインターフェイス (CSI) プラグイン。
- 必要な IAM アクセス権限 — このアドオンは Amazon EKS の サービスアカウントの IAM ロール機能を利用します。AmazonEBSCSIDriverPolicy AWS マネージドポリシーの許可が必要です。次のコマンドで IAM ロールを作成して、それにマネージドポリシーをアタッチできます。*my-cluster* をクラスターの名前に、*AmazonEKS_EBS_CSI_DriverRole* をロールに使用する名前に置き換えます。このコマンドを使用するには、デバイスに eksctl がインストールされている必要があります。別のツールを使用する必要がある場合や、暗号化にカスタム KMS キー を使用する必要がある場合は、Amazon EBS CSI ドライバー IAM ロールの作成 を参照してください。

```
eksctl create iamserviceaccount \
    --name ebs-csi-controller-sa \
    --namespace kube-system \
    --cluster my-cluster \
    --role-name AmazonEKS_EBS_CSI_DriverRole \
    --role-only \
    --attach-policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy
 \
```

```
    --approve
```

- 追加情報 – アドオンの詳細については、「Amazon EBS CSI ドライバー」を参照してください。

Amazon EFS CSI ドライバー

- 名前 – `aws-efs-csi-driver`
- 説明 — クラスターに Amazon EFS ストレージを提供する、Kubernetes コンテナストレージインターフェイス (CSI) プラグイン。
- 必要な IAM アクセス権限 — このアドオンは Amazon EKS の サービスアカウントの IAM ロール機能を利用します。`AmazonEFSCSIDriverPolicy` AWS マネージドポリシーの許可が必要です。次のコマンドで IAM ロールを作成して、そのロールにマネージドポリシーをアタッチできます。*`my-cluster`* をクラスターの名前に、*`AmazonEKS_EFS_CSI_DriverRole`* をロールに使用する名前に置き換えます。このコマンドを使用するには、デバイスに `eksctl` がインストールされている必要があります。別のツールを使用する必要がある場合は、IAM ロールの作成 を参照してください。

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
eksctl create iamserviceaccount \
    --name efs-csi-controller-sa \
    --namespace kube-system \
    --cluster $cluster_name \
    --role-name $role_name \
    --role-only \
    --attach-policy-arn arn:aws:iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy
 \
    --approve
TRUST_POLICY=$(aws iam get-role --role-name $role_name --query
 'Role.AssumeRolePolicyDocument' | \
    sed -e 's/efs-csi-controller-sa/efs-csi-*/' -e 's/StringEquals/StringLike/')
aws iam update-assume-role-policy --role-name $role_name --policy-document
 "$TRUST_POLICY"
```

- 追加情報 – アドオンの詳細については、「Amazon EFS CSI ドライバー」を参照してください。

Mountpoint for Amazon S3 CSI ドライバー

- 名前 – `aws-mountpoint-s3-csi-driver`

- 説明 – クラスターに Amazon S3 ストレージを提供する、Kubernetes コンテナストレージインターフェイス (CSI) プラグイン。
- 必要な IAM アクセス権限 — このアドオンは Amazon EKS の サービスアカウントの IAM ロール機能を利用します。作成される IAM ロールには S3 へのアクセスを許可するポリシーが必要です。ポリシーを作成するときは、MountpointIAM アクセス許可の推奨事項に従ってください。あるいは、AWS 管理ポリシー AmazonS3FullAccess を使用することもできますが、この管理ポリシーでは、Mountpoint に必要以上のアクセス権限が付与されます。

  次のコマンドで IAM ロールを作成して、そのロールにマネージドポリシーをアタッチできます。*my-cluster* をクラスターの名前、*region-code* を正しい AWS リージョン コードに、*AmazonEKS_S3_CSI_DriverRole* をロールの名前、*AmazonEKS_S3_CSI_DriverRole_ARN* をロール ARN に置き換えます。このコマンドを使用するには、デバイスに `eksctl` がインストールされている必要があります。IAM コンソールまたは AWS CLI を使用する手順については、「IAM ロールの作成」を参照してください。

  ```
  CLUSTER_NAME=my-cluster
  REGION=region-code
  ROLE_NAME=AmazonEKS_S3_CSI_DriverRole
  POLICY_ARN=AmazonEKS_S3_CSI_DriverRole_ARN
  eksctl create iamserviceaccount \
      --name s3-csi-driver-sa \
      --namespace kube-system \
      --cluster $CLUSTER_NAME \
      --attach-policy-arn $POLICY_ARN \
      --approve \
      --role-name $ROLE_NAME \
      --region $REGION \
      --role-only
  ```

- 追加情報 – アドオンの詳細については、「Mountpoint for Amazon S3 CSI driver」を参照してください。

CSI スナップショットコントローラー

- 名前 – `snapshot-controller`
- 説明 – コンテナストレージインターフェイス (CSI) スナップショットコントローラーを使用すると、Amazon EBS CSI ドライバーなどの互換性のある CSI ドライバーのスナップショット機能を使用できます。
- 必要な IAM アクセス権限 — このアドオンにはアクセス権限は必要ありません。

- 追加情報 – アドオンの詳細については、「CSI スナップショットコントローラー」を参照してください。

AWS Distro for OpenTelemetry

- 名前 – `adot`
- 説明 – AWS Distro for OpenTelemetry (ADOT) は、安全な本番環境対応の、AWS によってサポートされている OpenTelemetry プロジェクトのディストリビューションです。
- 必要な IAM アクセス許可 – このアドオンには、詳細設定でオプトインできる事前設定されたカスタムリソースの 1 つを使用している場合のみ IAM 権限が必要です。
- 追加情報 – 詳細については、AWS Distro for OpenTelemetry ドキュメントの「EKS アドオンを使用した AWS Distro for OpenTelemetry の開始方法」を参照してください。

  ADOT では、前提条件として `cert-managercert-manager` がクラスターにデプロイされている必要があります。 そうでなければ、Amazon EKS Terraform `cluster_addons`プロパティを使用して直接デプロイした場合、このアドオンは機能しません。その他の要件については、OpenTelemetry ドキュメント用の AWS Distro の「EKS アドオンを使用して AWS Distro for OpenTelemetry を始めるための要件」を参照してください。

Amazon GuardDuty エージェント

- 名前 – `aws-guardduty-agent`
- 説明 – Amazon GuardDuty は、AWS CloudTrail 管理イベントや Amazon VPC フローログなどの基本的なデータソースを分析および処理するセキュリティモニタリングサービスです。Amazon GuardDuty は、Kubernetes 監査ログやランタイムモニタリングなどの機能も処理します。
- 必要な IAM アクセス権限 — このアドオンにはアクセス権限は必要ありません。
- 追加情報 – 詳細については、「Runtime Monitoring for Amazon EKS clusters in Amazon GuardDuty」を参照してください。
  - Amazon EKS クラスターで潜在的なセキュリティの脅威を検出するには、Amazon GuardDuty ランタイムモニタリングを有効にし、GuardDuty セキュリティエージェントを Amazon EKS クラスターにデプロイします。

Amazon CloudWatch Observability Agent

- 名前 – `amazon-cloudwatch-observability`

- 説明 Amazon CloudWatch エージェントは、AWS が提供するモニタリングおよびオブザーバビリティサービスです。このアドオンは CloudWatch エージェントをインストールし、Amazon EKS のオブザーバビリティが強化された CloudWatch アプリケーションシグナルと CloudWatch コンテナインサイトの両方を有効にします。
- 必要な IAM アクセス権限 — このアドオンは Amazon EKS の サービスアカウントの IAM ロール機能を利用します。AWSXrayWriteOnlyAccess と CloudWatchAgentServerPolicy AWS の管理ポリシーのパーミッションが必要です。次のコマンドで、IAM ロールを作成し、それに管理ポリシーをアタッチして、アドオンが使用する Kubernetes サービスアカウントに注釈を付けることができます。*my-cluster* をクラスターの名前に、*AmazonEKS_Observability_role* をロールに使用する名前に置き換えます。このコマンドを使用するには、デバイスに `eksctl` がインストールされている必要があります。別のツールを使用してロールを作成し、ポリシーをアタッチして、Kubernetes サービスアカウントに注釈を付ける必要がある場合は、「IAM ロールを引き受けるように Kubernetes サービスアカウントを設定する」を参照してください。

```
eksctl create iamserviceaccount \
    --name cloudwatch-agent \
    --namespace amazon-cloudwatch \
    --cluster my-cluster \
    --role-name AmazonEKS_Observability_Role \
    --role-only \
    --attach-policy-arn arn:aws:iam::aws:policy/AWSXrayWriteOnlyAccess \
    --attach-policy-arn arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy \
    --approve
```

- 追加情報 – 詳細については、「CloudWatch エージェントのインストール」を参照してください。

Amazon EKS Pod Identity エージェント

- 名前 – `eks-pod-identity-agent`
- 説明 – Amazon EKS Pod Identityは、Amazon EC2 インスタンスプロファイルから Amazon EC2 インスタンスに認証情報を提供する場合と同じような方法で、アプリケーションの認証情報を管理する機能があります。
- 必要な IAM アクセス許可 – このアドオンは、Amazon EKS ノードの IAM ロール からのアクセス権限を使用します。
- 更新情報 – 一度に更新できるマイナーバージョンは 1 つのみです。例えば、現在のバージョンが `1.28.`*x*`-eksbuild.`*y* で、これを `1.30.`*x*`-eksbuild.`*y* に更新する場合は、現在のバージョン

を 1.29.*x*-eksbuild.*y* に更新してから、再度 1.30.*x*-eksbuild.*y* に更新する必要があります。アドオンの更新の詳細については、「Amazon EKS アドオンの更新」を参照してください。

独立系ソフトウェアベンダーによる追加の Amazon EKS アドオン

前述の Amazon EKS アドオンのリストに加えて、独立系ソフトウェアベンダーのさまざまな運用ソフトウェア Amazon EKS アドオンを追加することもできます。アドオンを選択すると、そのアドオンとそのインストール要件について詳しく知ることができます。

Find, procure and deploy add-ons from AWS Marketplace to Amazon EKS (YouTube)

Accuknox

- パブリッシャー – Accuknox
- 名前 – `accuknox_kubearmor`
- 名前空間 – `kubearmor`
- サービスアカウント名 – サービスアカウントはこのアドオンでは使用されません。
- AWS マネージド IAM ポリシー – このアドオンでは管理ポリシーは使用されません。
- カスタム IAM アクセス許可 – このアドオンではカスタムアクセス許可は使用されません。
- セットアップと使用方法 – KubeArmor ドキュメントの「KubeArmor 入門」を参照してください。

Akuity

- パブリッシャー – Akuity
- 名前 – `akuity_agent`
- 名前空間 – `akuity`
- サービスアカウント名 – サービスアカウントはこのアドオンでは使用されません。
- AWS マネージド IAM ポリシー – このアドオンでは管理ポリシーは使用されません。
- カスタム IAM アクセス許可 – このアドオンではカスタムアクセス許可は使用されません。
- セットアップと使用方法 – Akuity Platform ドキュメントの「Installing the Akuity Agent on Amazon EKS with the Akuity EKS add-on」を参照してください。

Calyptia

- パブリッシャー – Calyptia

- 名前 – `calyptia_fluent-bit`
- 名前空間 – `calytia-fluentbit`
- サービスアカウント名 – `clyptia-fluentbit`
- AWSマネージド IAM ポリシー – AWSMarketplaceMeteringRegisterUsage。
- 必要な IAM ロールを作成するコマンド – 次のコマンドでは、クラスターの IAM OpenID Connect (OIDC) プロバイダーが必要です。既に存在しているかどうかを確認する、または作成するには「クラスターの IAM OIDC プロバイダーを作成する」を参照してください。*my-cluster* をクラスターの名前に、*my-calyptia-role* をロールに使用する名前に置き換えます。このコマンドを使用するには、デバイスに `eksctl` がインストールされている必要があります。別のツールを使用してロールを作成し、Kubernetes サービスアカウントに注釈を付ける必要がある場合は、「IAM ロールを引き受けるように Kubernetes サービスアカウントを設定する」を参照してください。

```
eksctl create iamserviceaccount --name service-account-name  --namespace calyptia-
fluentbit --cluster my-cluster --role-name my-calyptia-role \
    --role-only --attach-policy-arn arn:aws:iam::aws:policy/
AWSMarketplaceMeteringRegisterUsage --approve
```

- セットアップと使用方法 – Calyptia ドキュメントの「Calyptia for Fluent Bit」を参照してください。

Cisco Observability Collector

- パブリッシャー – Cisco
- 名前 – `cisco_cisco-cloud-observability-collectors`
- 名前空間 – `appdynamics`
- サービスアカウント名 – サービスアカウントはこのアドオンでは使用されません。
- AWS マネージド IAM ポリシー – このアドオンでは管理ポリシーは使用されません。
- カスタム IAM アクセス許可 – このアドオンではカスタムアクセス許可は使用されません。
- セットアップと使用方法 – Cisco AppDynamics ドキュメントの「Use the Cisco Cloud Observability AWS Marketplace Add-Ons」を参照してください。

Cisco Observability Operator

- パブリッシャー – Cisco

- 名前 – `cisco_cisco-cloud-observability-operators`
- 名前空間 – `appdynamics`
- サービスアカウント名 – サービスアカウントはこのアドオンでは使用されません。
- AWS マネージド IAM ポリシー – このアドオンでは管理ポリシーは使用されません。
- カスタム IAM アクセス許可 – このアドオンではカスタムアクセス許可は使用されません。
- セットアップと使用方法 – Cisco AppDynamics ドキュメントの「Use the Cisco Cloud Observability AWS Marketplace Add-Ons」を参照してください。

CLOUDSOFT

- パブリッシャー – CLOUDSOFT
- 名前 – `cloudsoft_cloudsoft-amp`
- 名前空間 – `cloudsoft-amp`
- サービスアカウント名 – サービスアカウントはこのアドオンでは使用されません。
- AWS マネージド IAM ポリシー – このアドオンでは管理ポリシーは使用されません。
- カスタム IAM アクセス許可 – このアドオンではカスタムアクセス許可は使用されません。
- セットアップと使用方法 - CLOUDSOFT ドキュメントの「Amazon EKS ADDON」を参照してください。

Cribl

- パブリッシャー – Cribl
- 名前 – `cribl_cribledge`
- 名前空間 – `cribledge`
- サービスアカウント名 – サービスアカウントはこのアドオンでは使用されません。
- AWS マネージド IAM ポリシー – このアドオンでは管理ポリシーは使用されません。
- カスタム IAM アクセス許可 – このアドオンではカスタムアクセス許可は使用されません。
- セットアップと使用方法 – Cribl ドキュメントの「Edge 用 Cribl Amazon EKS アドオンのインストール」を参照してください。

Dynatrace

- パブリッシャー – Dynatrace

- 名前 – dynatrace_dynatrace-operator
- 名前空間 – dynatrace
- サービスアカウント名 – サービスアカウントはこのアドオンでは使用されません。
- AWS マネージド IAM ポリシー – このアドオンでは管理ポリシーは使用されません。
- カスタム IAM アクセス許可 - このアドオンではカスタムアクセス許可は使用されません。
- セットアップと使用方法 -「dynatrace ドキュメント」の「Kubernetes モニタリング」を参照してください。

Datree

- パブリッシャー - Datree
- 名前 – datree_engine-pro
- 名前空間 – datree
- サービスアカウント名 — datree-webhook-server-awsmp
- AWS マネージド IAM ポリシー – AWSLicenseManagerConsumptionPolicy
- 必要な IAM ロールを作成するコマンド – 次のコマンドでは、クラスターの IAM OpenID Connect (OIDC) プロバイダーが必要です。既に存在しているかどうかを確認する、または作成するには「クラスターの IAM OIDC プロバイダーを作成する」を参照してください。*my-cluster* をクラスターの名前に、*my-datree-role* をロールに使用する名前に置き換えます。このコマンドを使用するには、デバイスに eksctl がインストールされている必要があります。別のツールを使用してロールを作成し、Kubernetes サービスアカウントに注釈を付ける必要がある場合は、「IAM ロールを引き受けるように Kubernetes サービスアカウントを設定する」を参照してください。

```
eksctl create iamserviceaccount --name datree-webhook-server-awsmp --namespace datree
 --cluster my-cluster --role-name my-datree-role \
    --role-only --attach-policy-arn arn:aws:iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy --approve
```

- カスタム IAM アクセス許可 - このアドオンではカスタムアクセス許可は使用されません。
- セットアップと使用方法 - Datree ドキュメントの「Amazon EKS インテグレーション」を参照してください。

Datadog

- パブリッシャー – Datadog

- 名前 – `datadog_operator`
- 名前空間 – `datadog-agent`
- サービスアカウント名 – サービスアカウントはこのアドオンでは使用されません。
- AWS マネージド IAM ポリシー – このアドオンでは管理ポリシーは使用されません。
- カスタム IAM アクセス許可 – このアドオンではカスタムアクセス許可は使用されません。
- セットアップと使用方法 – Datadog ドキュメントの「Datadog Operator アドオンを使用して Amazon EKS に Datadog エージェントをインストールする」を参照してください。

Groundcover

- パブリッシャー – groundcover
- 名前 – `groundcover_agent`
- 名前空間 – `groundcover`
- サービスアカウント名 – サービスアカウントはこのアドオンでは使用されません。
- AWS マネージド IAM ポリシー – このアドオンでは管理ポリシーは使用されません。
- カスタム IAM アクセス許可 – このアドオンではカスタムアクセス許可は使用されません。
- セットアップと使用方法 – groundcover ドキュメントの「groundcover Amazon EKS アドオンのインストール」を参照してください。

Grafana Labs

- パブリッシャー – Grafana Labs
- 名前 – `grafana-labs_kubernetes-monitoring`
- 名前空間 – `monitoring`
- サービスアカウント名 – サービスアカウントはこのアドオンでは使用されません。
- AWS マネージド IAM ポリシー – このアドオンでは管理ポリシーは使用されません。
- カスタム IAM アクセス許可 – このアドオンではカスタムアクセス許可は使用されません。
- セットアップと使用方法 — Grafana Labs ドキュメントの「Kubernetes モニタリングを Amazon EKS のアドオンとして設定する」を参照してください。

HA Proxy

- パブリッシャー – HA Proxy

- 名前 – haproxy-technologies_kubernetes-ingress-ee
- 名前空間 – haproxy-controller
- サービスアカウント名 – customer defined
- AWS マネージド IAM ポリシー – AWSLicenseManagerConsumptionPolicy
- 必要な IAM ロールを作成するコマンド – 次のコマンドでは、クラスターの IAM OpenID Connect (OIDC) プロバイダーが必要です。既に存在しているかどうかを確認する、または作成するには「クラスターの IAM OIDC プロバイダーを作成する」を参照してください。*my-cluster* をクラスターの名前に、*my-haproxy-role* をロールに使用する名前に置き換えます。このコマンドを使用するには、デバイスに eksctl がインストールされている必要があります。別のツールを使用してロールを作成し、Kubernetes サービスアカウントに注釈を付ける必要がある場合は、「IAM ロールを引き受けるように Kubernetes サービスアカウントを設定する」を参照してください。

```
eksctl create iamserviceaccount --name service-account-name  --namespace haproxy-
controller --cluster my-cluster --role-name my-haproxy-role \
    --role-only --attach-policy-arn arn:aws:iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy --approve
```

- カスタム IAM アクセス許可 - このアドオンではカスタムアクセス許可は使用されません。
- セットアップと使用方法 — HAProxy のドキュメントの「AWS から Amazon EKS 上に HAProxy Enterprise Kubernetes Ingress Controller をインストールする」を参照してください。

Kpow

- パブリッシャー – Factorhouse
- 名前 – factorhouse_kpow
- 名前空間 – factorhouse
- サービスアカウント名 – kpow
- AWS マネージド IAM ポリシー - AWSLicenseManagerConsumptionPolicy
- 必要な IAM ロールを作成するコマンド – 次のコマンドでは、クラスターの IAM OpenID Connect (OIDC) プロバイダーが必要です。既に存在しているかどうかを確認する、または作成するには「クラスターの IAM OIDC プロバイダーを作成する」を参照してください。*my-cluster* をクラスターの名前に、*my-kpow-role* をロールに使用する名前に置き換えます。このコマンドを使用するには、デバイスに eksctl がインストールされている必要があります。別のツールを使用

してロールを作成し、Kubernetes サービスアカウントに注釈を付ける必要がある場合は、「IAM ロールを引き受けるように Kubernetes サービスアカウントを設定する」を参照してください。

```
eksctl create iamserviceaccount --name kpow --namespace factorhouse --cluster my-cluster --role-name my-kpow-role \
    --role-only --attach-policy-arn arn:aws:iam::aws:policy/service-role/AWSLicenseManagerConsumptionPolicy --approve
```

- カスタム IAM アクセス許可 - このアドオンではカスタムアクセス許可は使用されません。
- セットアップと使用方法 - 「Kpow ドキュメント」の「AWS Marketplace LM」を参照してください。

Kubecost

- パブリッシャー - Kubecost
- 名前 – `kubecost_kubecost`
- 名前空間 – `kubecost`
- サービスアカウント名 – サービスアカウントはこのアドオンでは使用されません。
- AWS マネージド IAM ポリシー – このアドオンでは管理ポリシーは使用されません。
- カスタム IAM アクセス許可 – このアドオンではカスタムアクセス許可は使用されません。
- 設定と使用手順 – Kubecost ドキュメントの「AWS クラウド請求の統合」を参照してください。
- クラスターがバージョン `1.23` 以降の場合は、the section called “Amazon EBS CSI ドライバー” がクラスターにインストールされている必要があります。そうでない場合、エラーが発生します。

Kasten

- パブリッシャー - Kasten by Veeam
- 名前 – `kasten_k10`
- 名前空間 – `kasten-io`
- サービスアカウント名 – `k10-k10`
- AWS マネージド IAM ポリシー – AWSLicenseManagerConsumptionPolicy
- 必要な IAM ロールを作成するコマンド – 次のコマンドでは、クラスターの IAM OpenID Connect (OIDC) プロバイダーが必要です。既に存在しているかどうかを確認する、または作成するには「クラスターの IAM OIDC プロバイダーを作成する」を参照してください。*my-cluster* をクラ

スターの名前に、*my-kasten-role* をロールに使用する名前に置き換えます。このコマンドを使用するには、デバイスに eksctl がインストールされている必要があります。別のツールを使用してロールを作成し、Kubernetes サービスアカウントに注釈を付ける必要がある場合は、「IAM ロールを引き受けるように Kubernetes サービスアカウントを設定する」を参照してください。

```
eksctl create iamserviceaccount --name k10-k10 --namespace kasten-io --cluster my-cluster --role-name my-kasten-role \
    --role-only --attach-policy-arn arn:aws:iam::aws:policy/service-role/AWSLicenseManagerConsumptionPolicy --approve
```

- カスタム IAM アクセス許可 - このアドオンではカスタムアクセス許可は使用されません。
- セットアップと使用方法 — Kasten ドキュメントの「Amazon EKS アドオンを使って AWS に K10 をインストール」を参照してください。
- 追加情報 — Amazon EKS クラスターがバージョン Kubernetes 1.23 またはそれ以降の場合は、クラスターに Amazon EBS CSI ドライバーがデフォルト StorageClass でインストールされている必要があります。

Kong

- パブリッシャー – Kong
- 名前 – kong_konnect-ri
- 名前空間 – kong
- サービスアカウント名 – サービスアカウントはこのアドオンでは使用されません。
- AWS マネージド IAM ポリシー – このアドオンでは管理ポリシーは使用されません。
- カスタム IAM アクセス許可 – このアドオンではカスタムアクセス許可は使用されません。
- セットアップと使用方法 – Kong ドキュメンテーションの「Kong Gateway EKS アドオンのインストール」を参照してください。

LeakSignal

- パブリッシャー – LeakSignal
- 名前 – leaksignal_leakagent
- 名前空間 – leakagent
- サービスアカウント名 – サービスアカウントはこのアドオンでは使用されません。
- AWS マネージド IAM ポリシー – このアドオンでは管理ポリシーは使用されません。

- カスタム IAM アクセス許可 – このアドオンではカスタムアクセス許可は使用されません。
- セットアップと使用方法 — LeakSignal ドキュメントの「LeakAgent アドオンのインストール」を参照してください。

NetApp

- パブリッシャー – NetApp
- 名前 – `netapp_trident-operator`
- 名前空間 – `trident`
- サービスアカウント名 – サービスアカウントはこのアドオンでは使用されません。
- AWS マネージド IAM ポリシー – このアドオンでは管理ポリシーは使用されません。
- カスタム IAM アクセス許可 – このアドオンではカスタムアクセス許可は使用されません。
- セットアップと使用方法 — NetApp ドキュメントの「Astra Trident EKS アドオンの設定」を参照してください。

New Relic

- パブリッシャー – New Relic
- 名前 – `new-relic_kubernetes-operator`
- 名前空間 – `newrelic`
- サービスアカウント名 – サービスアカウントはこのアドオンでは使用されません。
- AWS マネージド IAM ポリシー – このアドオンでは管理ポリシーは使用されません。
- カスタム IAM アクセス許可 – このアドオンではカスタムアクセス許可は使用されません。
- セットアップと使用方法 – New Relic ドキュメントの「EKS 用 New Relic アドオンのインストール」を参照してください。

Rafay

- パブリッシャー – Rafay
- 名前 – `rafay-systems_rafay-operator`
- 名前空間 – `rafay-system`
- サービスアカウント名 – サービスアカウントはこのアドオンでは使用されません。
- AWS マネージド IAM ポリシー – このアドオンでは管理ポリシーは使用されません。

- カスタム IAM アクセス許可 – このアドオンではカスタムアクセス許可は使用されません。
- セットアップと使用方法 – Rafay ドキュメントの「Rafay Amazon EKS アドオンのインストール」を参照してください。

Solo.io

- パブリッシャー – Solo.io
- 名前 – `solo-io_istio-distro`
- 名前空間 – `istio-system`
- サービスアカウント名 – サービスアカウントはこのアドオンでは使用されません。
- AWS マネージド IAM ポリシー – このアドオンでは管理ポリシーは使用されません。
- カスタム IAM アクセス許可 – このアドオンではカスタムアクセス許可は使用されません。
- セットアップと使用方法 — Solo.io ドキュメントの「Installing Istio」を参照してください。

Stormforge

- パブリッシャー – Stormforge
- 名前 – `stormforge_optimize-Live`
- 名前空間 – `stormforge-system`
- サービスアカウント名 – サービスアカウントはこのアドオンでは使用されません。
- AWS マネージド IAM ポリシー – このアドオンでは管理ポリシーは使用されません。
- カスタム IAM アクセス許可 – このアドオンではカスタムアクセス許可は使用されません。
- セットアップと使用方法 – StormForge ドキュメントの「StormForge エージェントのインストール」を参照してください。

Splunk

- パブリッシャー – Splunk
- 名前 – `splunk_splunk-otel-collector-chart`
- 名前空間 – `splunk-monitoring`
- サービスアカウント名 – サービスアカウントはこのアドオンでは使用されません。
- AWS マネージド IAM ポリシー – このアドオンでは管理ポリシーは使用されません。

- カスタム IAM アクセス許可 – このアドオンではカスタムアクセス許可は使用されません。
- セットアップと使用方法 — Splunk ドキュメントの「Amazon EKS に Splunk アドオンをインストール」を参照してください。

Teleport

- パブリッシャー – Teleport
- 名前 – `teleport_teleport`
- 名前空間 – `teleport`
- サービスアカウント名 – サービスアカウントはこのアドオンでは使用されません。
- AWS マネージド IAM ポリシー – このアドオンでは管理ポリシーは使用されません。
- カスタム IAM アクセス許可 - このアドオンではカスタムアクセス許可は使用されません。
- セットアップと使用方法 -「Teleport ドキュメント」の「テレポートの仕組み」を参照してください。

Tetrate

- パブリッシャー - Tetrate Io
- 名前 – `tetrate-io_istio-distro`
- 名前空間 – `istio-system`
- サービスアカウント名 – サービスアカウントはこのアドオンでは使用されません。
- AWS マネージド IAM ポリシー – このアドオンでは管理ポリシーは使用されません。
- カスタム IAM アクセス許可 - このアドオンではカスタムアクセス許可は使用されません。
- セットアップと使用方法 -「Tetrate Istio Distro」のウェブサイトを参照してください。

Upbound Universal Crossplane

- パブリッシャー - Upbound
- 名前 – `upbound_universal-crossplane`
- 名前空間 – `upbound-system`
- サービスアカウント名 – サービスアカウントはこのアドオンでは使用されません。
- AWS マネージド IAM ポリシー – このアドオンでは管理ポリシーは使用されません。

- カスタム IAM アクセス許可 - このアドオンではカスタムアクセス許可は使用されません。
- セットアップと使用方法 -「Upbound ドキュメント」の「アップバウンドユニバーサルクロスプレーン (UXP)」を参照してください。

Upwind

- パブリッシャー – Upwind
- 名前 – upwind
- 名前空間 – upwind
- サービスアカウント名 – サービスアカウントはこのアドオンでは使用されません。
- AWS マネージド IAM ポリシー – このアドオンでは管理ポリシーは使用されません。
- カスタム IAM アクセス許可 – このアドオンではカスタムアクセス許可は使用されません。
- セットアップと使用方法 -「Upwind ドキュメント」の「インストール手順」を参照してください。

Amazon EKS アドオンの管理

Amazon EKS アドオンは、Amazon EKS クラスター用のアドオンソフトウェアを厳選したセットです。すべての Amazon EKS アドオン:

- 最新のセキュリティパッチやバグ修正が含まれています。
- AWS によって Amazon EKS で動作することが検証されています。
- アドオンソフトウェアの管理に必要な作業量を削減します。

Amazon EKS アドオンで新しいバージョンが利用可能になると、AWS Management Console が通知します。更新を開始すると、Amazon EKS がアドオンソフトウェアを更新します。

利用可能なアドオンのリストについては、「Amazon EKS で利用可能な Amazon EKS アドオン」を参照してください。Kubernetes フィールド管理の詳細については、「 Kubernetes フィールド管理」を参照してください。

前提条件

- 既存の Amazon EKS クラスター。デプロイするには、「Amazon EKS の使用開始」を参照してください。

アドオンの作成

Amazon EKS アドオンは eksctl、AWS Management Console、AWS CLI を使用して作成できます。アドオンに IAM ロールが必要な場合、ロールの作成の詳細については、Amazon EKS で利用可能な Amazon EKS アドオン の特定のアドオンの詳細を参照してください。

eksctl

前提条件

デバイスまたは AWS CloudShell にインストールされている eksctl コマンドラインツールのバージョン 0.183.0 以降。eksctl をインストールまたはアップグレードするには、eksctl ドキュメントの「インストール」を参照してください。

eksctl を使用して Amazon EKS アドオンを作成する方法

1. クラスターバージョンで利用可能なアドオンの名前を表示します。*1.30* をお持ちのクラスターのバージョンに置き換えます。

```
eksctl utils describe-addon-versions --kubernetes-version 1.30 | grep AddonName
```

出力例は次のとおりです。

```
"AddonName": "aws-ebs-csi-driver",
                        "AddonName": "coredns",
                        "AddonName": "kube-proxy",
                        "AddonName": "vpc-cni",
                        "AddonName": "adot",
                        "AddonName": "dynatrace_dynatrace-operator",
                        "AddonName": "upbound_universal-crossplane",
                        "AddonName": "teleport_teleport",
                        "AddonName": "factorhouse_kpow",
                        [...]
```

2. 作成するアドオンで使用できるバージョンを表示します。*1.30* をお持ちのクラスターのバージョンに置き換えます。*name-of-addon* を表示するバージョンのアドオン名に置き換えます。名前は、前のステップで返された名前のいずれかである必要があります。

```
eksctl utils describe-addon-versions --kubernetes-version 1.30 --name name-of-addon | grep AddonVersion
```

次の出力は、vpc-cni という名前のアドオンに対して返される内容の例です。このアドオンには複数のバージョンがあることがわかります。

```
"AddonVersions": [
    "AddonVersion": "v1.12.0-eksbuild.1",
    "AddonVersion": "v1.11.4-eksbuild.1",
    "AddonVersion": "v1.10.4-eksbuild.1",
    "AddonVersion": "v1.9.3-eksbuild.1",
```

3. 作成するアドオンが Amazon EKS か AWS Marketplace アドオンか判断します。AWS Marketplace にはサードパーティ製のアドオンがあり、アドオンを作成するために追加の手順を実行する必要があります。

```
eksctl utils describe-addon-versions --kubernetes-version 1.30 --name name-of-addon | grep ProductUrl
```

出力が返されない場合、アドオンは Amazon EKS です。出力が返された場合、アドオンは AWS Marketplace アドオンです。次の出力は、teleport_teleport という名前のアドオンのものです。

```
"ProductUrl": "https://aws.amazon.com/marketplace/pp?
sku=3bda70bb-566f-4976-806c-f96faef18b26"
```

返される URL を含んだ AWS Marketplace のアドオンの詳細について説明します。アドオンにサブスクリプションが必要な場合、AWS Marketplace を通じてアドオンにサブスクライブできます。AWS Marketplace からアドオンを作成する場合、アドオンの作成に使用する IAM プリンシパルには、AWSServiceRoleForAWSLicenseManagerRole のサービスにリンクされたロールを作成する許可が必要です。IAM エンティティへの許可の割り当てに関する詳細については、「IAM ユーザーガイド」の「IAM アイデンティティ許可の追加と削除」を参照してください。

4. Amazon EKS アドオンを作成します。デバイスに沿ったコマンドをコピーします。必要に応じてコマンドに次の変更を加え、変更したコマンドを実行します。

- *my-cluster* を自分のクラスター名に置き換えます。
- *name-of-addon* を作成するアドオンの名前に置き換えます。
- 最新バージョンよりも前のバージョンのアドオンが必要な場合、*latest* を使用する前のステップの出力で返されたバージョン番号に置き換えます。

- アドオンがサービスアカウントロールを使用している場合、*111122223333* をアカウント ID に置き換え、*role-name* をロールの名前に置き換えます。サービスアカウントのロールを作成する手順については、作成するアドオンのドキュメントを参照してください。サービスアカウントロールを指定するには、クラスター用に IAM OpenID Connect (OIDC) プロバイダーが必要です。クラスター用に持っているかどうかを確認、あるいは作成するには、「クラスターの IAM OIDC プロバイダーを作成する」を参照してください。

  アドオンがサービスアカウントロールを使用していない場合、**--service-account-role-arn arn:aws:iam::*111122223333*:role/*role-name*** を削除してください。
- このコマンド例は、アドオンの既存のセルフマネージドバージョンがある場合、その設定を上書きします。既存のセルフマネージド型アドオンの設定を上書きしたくない場合、*--force* オプションを削除してください。オプションを削除し、Amazon EKS アドオンが既存のセルフマネージド型アドオンの設定を上書きする必要がある場合、Amazon EKS アドオンの作成が失敗します。その場合、競合を解決するためのエラーメッセージが表示されます。このオプションを指定する前に、自分が管理する必要がある設定を Amazon EKS アドオンが管理していないことを確認してください。これらの設定は、このオプションの指定により上書きされます。

```
eksctl create addon --cluster my-cluster --name name-of-addon --version latest
 \
    --service-account-role-arn arn:aws:iam::111122223333:role/role-name --
force
```

コマンドにすべての利用可能なオプションのリストを確認できます。

```
eksctl create addon --help
```

利用可能なオプションの詳細については、「eksctl ドキュメント」の「アドオン」を参照してください。

AWS Management Console

AWS Management Console を使用して Amazon EKS アドオンを作成する方法

1. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。
2. 左のナビゲーションペインで [クラスター] を選択し、次にアドオンを作成するクラスター名を選択します。

3. [アドオン] タブを選択します。
4. [その他のアドオンを入手] を選択します。
5. クラスターに追加するアドオンを選択します。必要な数だけ [Amazon EKS アドオン] や [AWS Marketplace アドオン] を追加できます。

   AWS Marketplace アドオンの場合、アドオンを作成するために使用する IAM プリンシパルには、AWS LicenseManager からアドオンの資格を読み取る許可が必要です。AWSLicenseManager は、ユーザーに代わってライセンスを管理する AWS リソースを許可する「AWSServiceRoleForAWSLicenseManagerRole」のサービスリンクロール (SLR) が必要になります。SLR はアカウントごとに 1 回限りの要件であり、アドオンやクラスターごとに別々の SLR を作成する必要はありません。IAM プリンシパルへの許可の割り当ての詳細については、「IAM ユーザーガイド」の「IAM ID のアクセス許可の追加および削除」を参照してください。

   インストールする AWS Marketplace アドオンがリストされていない場合は、検索ボックスにテキストを入力して、利用可能なアドオンを検索できます。[フィルタリング] オプションで、[カテゴリ]、[ベンダー]、または [料金モデル] によってフィルタリングして、検索結果からアドオンを選択することもできます。インストールするアドオンを選択したら、[次へ] を選択します。
6. [選択したアドオンセッティングの設定] ページで次のことを行います。

   - [サブスクリプションオプションを表示] を選択し、[サブスクリプションのオプション] フォームを開きます。[料金の詳細] と [法務] のセクションを確認し、[登録] ボタンを選択して続行します。
   - [バージョン] で、インストールするバージョンを選択します。作成する個々のアドオンが別のバージョンを推奨している場合を除き、[最新] と表示されたバージョンをお勧めします。アドオンに推奨バージョンがあるかどうかを確認するには、作成しているアドオンのドキュメントを参照してください。
   - 選択したすべてのアドオンが [ステータス] の下で [サブスクリプションが必要] と表示されている場合、[次へ] を選択します。クラスターが作成された後にサブスクライブするまで、それ以上これらのアドオンを設定することはできません。[ステータス] の下で [サブスクリプションが必要] と表示されていないアドオンの場合、次のことを行います。
     - [IAM ロールの選択] では、アドオンに IAM アクセス許可が必要でない限り、デフォルトのオプションをそのまま使用します。アドオンに AWS アクセス許可が必要な場合、ノードの IAM ロール ([未設定])、またはアドオンで使用するために作成した既存のロールを使用できます。選択するロールがない場合、既存のロールがありません。選択した

オプションを問わず、作成するアドオンのドキュメントを参照し、IAM ポリシーを作成してロールにアタッチしてください。IAM ロールを選択するには、クラスター用に IAM OpenID Connect (OIDC) プロバイダーが必要です。クラスター用に持っているかどうかを確認、あるいは作成するには、「クラスターの IAM OIDC プロバイダーを作成する」を参照してください。

- [オプション設定のセッティング] を選択します。
  - アドオンに設定が必要な場合、[設定値] ボックスに入力します。アドオンに設定情報が必要かどうかを判断するには、作成するアドオンのドキュメントを参照してください。
  - [競合解決方法] で利用可能なオプションをいずれか選択します。
- [次へ] をクリックします。

7. [確認と追加] ページで、[作成] を選択します。アドオンのインストールが完了した後、インストールしたアドオンが表示されます。
8. インストールしたアドオンにサブスクリプションが必要な場合、次の手順を実行してください。

   1. アドオンの右下隅にある [サブスクライブ] ボタンを選択します。AWS Marketplace のアドオンのページが表示されます。[製品概要] や [価格情報] など、アドオンに関する情報をお読みください。
   2. アドオンページの右上にある [サブスクライブを続ける] ボタンを選択します。
   3. [利用規約] をよくお読みください。利用規約に同意する場合、[利用規約に同意] を選択します。サブスクリプションの処理に数分かかることがあります。サブスクリプションの処理中、[Amazon EKS コンソールに戻る] ボタンがグレー表示されます。
   4. サブスクリプションの処理が完了すると、[Amazon EKS コンソールに戻る] ボタンがグレー表示されなくなります。ボタンを選択し、クラスターの Amazon EKS コンソールの [アドオン] タブに戻ります。
   5. サブスクライブしているアドオンについては、[削除して再インストール] を選択し、次に [アドオンの再インストール] を選択します。アドオンのインストールに数分かかることがあります。インストールが完了すると、アドオンを設定できます。

AWS CLI

前提条件

ご使用のデバイスまたは AWS CloudShell で、バージョン 2.12.3 以降、または AWS Command Line Interface (AWS CLI) のバージョン 1.27.160 以降がインストールおよび設定されていること。現在のバージョンを確認するには、「**aws --version | cut -d / -f2 | cut -d ' ' -f1**」を参照してください。macOS の yum、apt-get、または Homebrew などのパッケージマネージャは、AWS CLI の最新バージョンより数バージョン遅れることがあります。最新バージョンをインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI のインストール、更新、およびアンインストール」と「aws configure でのクイック設定」を参照してください。AWS CloudShell にインストールされている AWS CLI バージョンは、最新バージョンより数バージョン遅れている可能性もあります。更新するには、「AWS CloudShellユーザーガイド」の「ホームディレクトリへの AWS CLI のインストール」を参照してください。

AWS CLI を使用して Amazon EKS アドオンを作成する方法

1. 利用可能なアドオンを確認します。利用可能なすべてのアドオン、種類、発行元を確認できます。AWS Marketplace を介して利用可能なアドオンの URL も確認できます。*1.30* をお持ちのクラスターのバージョンに置き換えます。

```
aws eks describe-addon-versions --kubernetes-version 1.30 \
    --query 'addons[].{MarketplaceProductUrl: marketplaceInformation.productUrl,
 Name: addonName, Owner: owner Publisher: publisher, Type: type}' --output table
```

出力例は次のとおりです。

```
-------------------------------------------------------------------------------------------
|
 DescribeAddonVersions
       |
+-----------------------------------------------------------
+------------------------------+-----------------+-------------
+---------------------+
|                     MarketplaceProductUrl                    |
 Name              |      Owner       |  Publisher   |        Type         |
+-----------------------------------------------------------
+------------------------------+-----------------+-------------
+---------------------+
|  None                                                        |  aws-ebs-csi-
driver           |  aws             |  eks         |  storage            |
|  None                                                        |  coredns
                |  aws             |  eks         |  networking         |
```

```
|  None                                                                  |  kube-proxy
                |  aws             |  eks         |  networking          |
|  None                                                                  |  vpc-cni
                |  aws             |  eks         |  networking          |
|  None                                                                  |  adot
                |  aws             |  eks         |  observability       |
|  https://aws.amazon.com/marketplace/pp/prodview-brb73nceicv7u |
 dynatrace_dynatrace-operator |  aws-marketplace |  dynatrace   |  monitoring
      |
|  https://aws.amazon.com/marketplace/pp/prodview-uhc2iwi5xysoc |
 upbound_universal-crossplane |  aws-marketplace |  upbound     |  infra-
management   |
|  https://aws.amazon.com/marketplace/pp/prodview-hd2ydsrgqy4li |
 teleport_teleport            |  aws-marketplace |  teleport    |  policy-
management  |
|  https://aws.amazon.com/marketplace/pp/prodview-vgghgqdsplhvc |
 factorhouse_kpow             |  aws-marketplace |  factorhouse |  monitoring
      |
|  [...]                                                                 |  [...]
                |  [...]           |  [...]       |  [...]               |
+------------------------------------------------------------------
+-------------------------------+------------------+--------------
+---------------------+
```

出力が異なる場合があります。この出力例では、タイプ networking で利用可能なアドオンが 3 種類あり、タイプ eks の発行元でアドオンが 5 種類あります。Owner 列に aws-marketplace があるアドオンは、インストールする前にサブスクリプションが必要な場合があります。URL にアクセスしてアドオンの詳細を確認したり、アドオンをサブスクライブしたりできます。

2. アドオンごとに利用可能なバージョンを確認できます。*1.30* をクラスターのバージョンに置き換え、*vpc-cni* を前のステップで返されたアドオンの名前に置き換えます。

```
aws eks describe-addon-versions --kubernetes-version 1.30 --addon-name vpc-cni \
    --query 'addons[].addonVersions[].{Version: addonVersion, Defaultversion:
 compatibilities[0].defaultVersion}' --output table
```

出力例は次のとおりです。

```
-----------------------------------------
|          DescribeAddonVersions         |
+-----------------+----------------------+
```

```
| Defaultversion  |       Version        |
+-----------------+----------------------+
|  False          |  v1.12.0-eksbuild.1  |
|  True           |  v1.11.4-eksbuild.1  |
|  False          |  v1.10.4-eksbuild.1  |
|  False          |  v1.9.3-eksbuild.1   |
+-----------------+----------------------+
```

Defaultversion 列に True と表示されているバージョンは、デフォルトでアドオンが作成されたバージョンです。

3. (オプション) 次のコマンドを実行して、選択したアドオンの設定オプションを調べます。

```
aws eks describe-addon-configuration --addon-name vpc-cni --addon-
version v1.12.0-eksbuild.1
```

```
{
    "addonName": "vpc-cni",
    "addonVersion": "v1.12.0-eksbuild.1",
    "configurationSchema": "{\"$ref\":\"#/definitions/VpcCni\",\"$schema
\":\"http://json-schema.org/draft-06/schema#\",\"definitions\":{\"Cri\":
{\"additionalProperties\":false,\"properties\":{\"hostPath\":{\"$ref\":
\"#/definitions/HostPath\"}},\"title\":\"Cri\",\"type\":\"object\"},\"Env
\":{\"additionalProperties\":false,\"properties\":{\"ADDITIONAL_ENI_TAGS
\":{\"type\":\"string\"},\"AWS_VPC_CNI_NODE_PORT_SUPPORT\":{\"format\":
\"boolean\",\"type\":\"string\"},\"AWS_VPC_ENI_MTU\":{\"format\":\"integer
\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_CONFIGURE_RPFILTER\":{\"format
\":\"boolean\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG\":
{\"format\":\"boolean\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_EXTERNALSNAT
\":{\"format\":\"boolean\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_LOGLEVEL
\":{\"type\":\"string\"},\"AWS_VPC_K8S_CNI_LOG_FILE\":{\"type
\":\"string\"},\"AWS_VPC_K8S_CNI_RANDOMIZESNAT\":{\"type\":
\"string\"},\"AWS_VPC_K8S_CNI_VETHPREFIX\":{\"type\":\"string
\"},\"AWS_VPC_K8S_PLUGIN_LOG_FILE\":{\"type\":\"string\"},
\"AWS_VPC_K8S_PLUGIN_LOG_LEVEL\":{\"type\":\"string\"},\"DISABLE_INTROSPECTION
\":{\"format\":\"boolean\",\"type\":\"string\"},\"DISABLE_METRICS\":{\"format
\":\"boolean\",\"type\":\"string\"},\"DISABLE_NETWORK_RESOURCE_PROVISIONING
\":{\"format\":\"boolean\",\"type\":\"string\"},\"ENABLE_POD_ENI\":{\"format
\":\"boolean\",\"type\":\"string\"},\"ENABLE_PREFIX_DELEGATION\":{\"format
\":\"boolean\",\"type\":\"string\"},\"WARM_ENI_TARGET\":{\"format\":\"integer
\",\"type\":\"string\"},\"WARM_PREFIX_TARGET\":{\"format\":\"integer\",
\"type\":\"string\"}},\"title\":\"Env\",\"type\":\"object\"},\"HostPath\":
{\"additionalProperties\":false,\"properties\":{\"path\":{\"type\":\"string\"}},
```

```
\"title\":\"HostPath\",\"type\":\"object\"},\"Limits\":{\"additionalProperties
\":false,\"properties\":{\"cpu\":{\"type\":\"string\"},\"memory\":{\"type
\":\"string\"}},\"title\":\"Limits\",\"type\":\"object\"},\"Resources\":
{\"additionalProperties\":false,\"properties\":{\"limits\":{\"$ref\":\"#/
definitions/Limits\"},\"requests\":{\"$ref\":\"#/definitions/Limits\"}},
\"title\":\"Resources\",\"type\":\"object\"},\"VpcCni\":{\"additionalProperties
\":false,\"properties\":{\"cri\":{\"$ref\":\"#/definitions/Cri\"},\"env\":
{\"$ref\":\"#/definitions/Env\"},\"resources\":{\"$ref\":\"#/definitions/
Resources\"}},\"title\":\"VpcCni\",\"type\":\"object\"}}}"
}
```

出力は標準の JSON スキーマです。

上記のスキーマで使用できる有効な設定値の例 (JSON フォーマット) を次に示します。

```
{
  "resources": {
    "limits": {
      "cpu": "100m"
    }
  }
}
```

上記のスキーマで使用できる有効な設定値の例 (YAML フォーマット) を次に示します。

```
  resources:
    limits:
      cpu: 100m
```

4. アドオンに IAM アクセス許可が必要かどうかを確認します。その場合は、(1) EKS Pod Identity またはサービスアカウントの IAM ロール (IRSA) を使用するかどうかを決定し、(2) アドオンで使用する IAM ロールの ARN を決定し、(3) アドオンで使用される Kubernetes サービスアカウントの名前を決定する必要があります。この情報は、ドキュメントまたは AWS API を使用して確認できます。「Retrieve IAM info about an Add-on」を参照してください。

   - Amazon EKS では、アドオンが EKS Pod Identity をサポートしている場合は、EKS Pod Identity の使用を推奨しています。そのためには、Pod Identity エージェントがクラスターにインストールされている必要があります。アドオンで Pod Identity を使用する場合の詳細については、「Pod Identity を使用して Amazon EKS アドオンに IAM ロールをアタッチする」を参照してください。

- アドオンまたはクラスターが EKS Pod Identity 用にセットアップされていない場合は、IRSA を使用します。クラスターで IRSA が設定されていることを確認します。
- Amazon EKS アドオンのドキュメントを参照して、アドオンに IAM アクセス許可と、関連する Kubernetes サービスアカウントの名前が必要かどうかを確認します。

5. Amazon EKS アドオンを作成します。デバイスに沿ったコマンドをコピーします。必要に応じてコマンドに次の変更を加え、変更したコマンドを実行します。

- *my-cluster* を自分のクラスター名に置き換えます。
- *vpc-cni* を前のステップの出力で返された作成するアドオン名に置き換えます。
- *version-number* を使用する前のステップの出力で返されたバージョンに置き換えます。
- アドオンに IAM アクセス許可が必要ない場合は、*<service-account-configuration>* を削除します。
- アドオン (1) に IAM アクセス許可が必要で、(2) クラスターが EKS Pod Identity を使用している場合は、*<service-account-configuration>* を次の Pod Identity の関連付けに置き換えます。*<service-account-name>* をアドオンで使用されるサービスアカウント名に置き換えます。*<role-arn>* を IAM ロールの ARN に置き換えます。ロールには、EKS Pod Identity に必要な信頼ポリシーが必要です。
  - ```
    --pod-identity-associations 'serviceAccount=<service-account-name>,roleArn=<role-arn>'
    ```
- アドオン (1) に IAM アクセス許可が必要で、(2) クラスターが IRSA を使用している場合は、*<service-account-configuration>* を次の IRSA 設定に置き換えます。*111122223333* を、アカウント ID に置き換えます。また、*role-name* を、作成した既存の IAM ロールの名前に置き換えます。ロールの作成手順については、作成するアドオンのドキュメントを参照してください。サービスアカウントロールを指定するには、クラスター用に IAM OpenID Connect (OIDC) プロバイダーが必要です。クラスター用に持っているかどうかを確認、あるいは作成するには、「クラスターの IAM OIDC プロバイダーを作成する」を参照してください。
  - ```
    --service-account-role-arn arn:aws:iam::111122223333:role/role-name
    ```
- このコマンド例は、アドオンの既存のセルフマネージドバージョンの `--configuration-values` オプションがある場合、それを上書きします。これを、文字列やファイル入力などの目的の設定値に置き換えます。設定値を指定したくない場合、`--configuration-values` オプションを削除します。AWS CLI に既存のセルフマネージ

ド型アドオンの設定を上書きされたくない場合、*--resolve-conflicts OVERWRITE* オプションを削除してください。オプションを削除し、Amazon EKS アドオンが既存のセルフマネージド型アドオンの設定を上書きする必要がある場合、Amazon EKS アドオンの作成が失敗します。その場合、競合を解決するためのエラーメッセージが表示されます。このオプションを指定する前に、自分が管理する必要がある設定を Amazon EKS アドオンが管理していないことを確認してください。これらの設定は、このオプションの指定により上書きされます。

```
aws eks create-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version version-number \
    <service-account-configuration> --configuration-values '{"resources":
{"limits":{"cpu":"100m"}}}' --resolve-conflicts OVERWRITE
```

```
aws eks create-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version version-number \
   <service-account-configuration> --configuration-values 'file://example.yaml'
 --resolve-conflicts OVERWRITE
```

使用可能なすべてのオプションのリストについては、「Amazon EKS コマンドラインリファレンス」の「create-addon」を参照してください。作成したアドオンが前のステップの Owner 列に aws-marketplace が一覧表示されている場合、作成が失敗して次のエラーと同様なエラーメッセージが表示されることがあります。

```
{
    "addon": {
        "addonName": "addon-name",
        "clusterName": "my-cluster",
        "status": "CREATE_FAILED",
        "addonVersion": "version",
        "health": {
            "issues": [
                {
                    "code": "AddonSubscriptionNeeded",
                    "message": "You are currently not subscribed to this add-
on. To subscribe, visit the AWS Marketplace console, agree to the seller EULA,
 select the pricing type if required, then re-install the add-on"
[...]
```

前の出力のエラーと同様なものが表示された場合、前のステップに出力された URL にアクセスしてアドオンをサブスクライブしてください。サブスクライブしたら `create-addon` コマンドを再実行します。

アドオンの更新

Amazon EKS は、新しいバージョンがリリース、あるいはクラスターを新しい Kubernetes マイナーバージョンに更新しても、アドオンを自動的に更新しません。既存のクラスターのアドオンを更新するには、更新を開始する必要があります。更新を開始した後、Amazon EKS がアドオンを更新します。アドオンを更新する前に、アドオンの現行のドキュメントを確認してください。利用可能なアドオンのリストについては、「Amazon EKS で利用可能な Amazon EKS アドオン」を参照してください。アドオンに IAM ロールが必要な場合、ロールの作成の詳細については、Amazon EKS で利用可能な Amazon EKS アドオン の特定のアドオンの詳細を参照してください。

`eksctl`、AWS Management Console、AWS CLI のいずれかを使用して Amazon EKS アドオンを更新できます。

eksctl

前提条件

デバイスまたは AWS CloudShell にインストールされている `eksctl` コマンドラインツールのバージョン `0.183.0` 以降。`eksctl` をインストールまたはアップグレードするには、`eksctl` ドキュメントの「インストール」を参照してください。

`eksctl` を使用して Amazon EKS アドオンを更新する方法

1. クラスターにインストールされている現在のアドオンおよびアドオンのバージョンを確認します。*`my-cluster`* を自分のクラスター名に置き換えます。

```
eksctl get addon --cluster my-cluster
```

出力例は次のとおりです。

```
NAME        VERSION              STATUS  ISSUES  IAMROLE  UPDATE AVAILABLE
coredns     v1.8.7-eksbuild.2    ACTIVE  0
kube-proxy  v1.23.7-eksbuild.1   ACTIVE  0                v1.23.8-eksbuild.2
```

```
vpc-cni     v1.10.4-eksbuild.1   ACTIVE  0                 v1.12.0-
eksbuild.1,v1.11.4-eksbuild.1,v1.11.3-eksbuild.1,v1.11.2-eksbuild.1,v1.11.0-
eksbuild.1
```

クラスターにあるアドオンおよびバージョンによって、出力が異なる場合があります。前述の出力例では、クラスターの既存のアドオン 2 つは UPDATE AVAILABLE 列に新しいバージョンがあることがわかります。

2. アドオンを更新します。

   1. デバイスに沿ったコマンドをコピーします。必要に応じてコマンドに次の変更を加えます。

      - *my-cluster* を自分のクラスター名に置き換えます。
      - *region-code* をクラスターのある AWS リージョン に置き換えます。
      - *vpc-cni* を更新する前のステップの出力で返されたアドオンの名前に置き換えます。
      - 利用可能な最新バージョンよりも前のバージョンに更新する場合、*latest* を使用する前のステップの出力で返されたバージョン番号に置き換えます。一部のアドオンには推奨バージョンがあります。詳細については、更新するアドオンのドキュメントを参照してください。
      - アドオンが Kubernetes サービスアカウントおよび IAM ロールを使用する場合、*111122223333* をアカウント ID に置き換え、*role-name* を作成した既存の IAM ロールの名前に置き換えます。ロールの作成手順については、作成するアドオンのドキュメントを参照してください。サービスアカウントロールを指定するには、クラスター用に IAM OpenID Connect (OIDC) プロバイダーが必要です。クラスター用に持っているかどうかを確認、あるいは作成するには、「クラスターの IAM OIDC プロバイダーを作成する」を参照してください。

        アドオンが Kubernetes サービスアカウントおよび IAM ロールを使用しない場合、**serviceAccountRoleARN: arn:aws:iam::*111122223333*:role/*role-name*** 行を削除してください。
      - *preserve* オプションはアドオンの既存値を保存します。アドオン設定にカスタム値を設定していて、このオプションを使用しない場合、Amazon EKS は値をデフォルト値で上書きします。このオプションを使用する場合、実稼働クラスターのアドオンを更新する前に、非稼動クラスターのフィールドおよび値変更をテストすることをお勧めします。この値を overwrite に変更する場合、すべての設定が Amazon EKS のデフォルト値に変更されます。いずれかの設定にカスタム値を設定した場合、Amazon EKS のデフォルト値で上書きされる可能性があります。この値を none に変更した場

合、Amazon EKS は設定の値を一切変更しませんが、更新が失敗する可能性があります。更新に失敗した場合、競合の解決に役立つエラーメッセージが返されます。

```
cat >update-addon.yaml <<EOF
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: my-cluster
  region: region-code

addons:
- name: vpc-cni
  version: latest
  serviceAccountRoleARN: arn:aws:iam::111122223333:role/role-name
  resolveConflicts: preserve
EOF
```

2. 変更コマンドを実行して update-addon.yaml ファイルを作成します。
3. クラスターに設定ファイルを適用します。

```
eksctl update addon -f update-addon.yaml
```

アドオン更新の詳細については、「eksctl ドキュメント」の「アドオン」を参照してください。

AWS Management Console

AWS Management Console を使用して Amazon EKS アドオンを更新する方法

1. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。
2. 左のナビゲーションペインで、[クラスター] を選択し、次にアドオンを設定するクラスター名を選択します。
3. [アドオン] タブを選択します。
4. アドオンボックスの右上にあるボックスを選択し、次に [編集] を選択します。
5. [#######の設定] ページで次のことを行います。

- 使用する [バージョン] を選択します。アドオンには推奨バージョンがある場合があります。詳細については、更新するアドオンのドキュメントを参照してください。
- [IAM ロールの選択] では、ノードの IAM ロール ([未設定])、またはアドオンで使用するために作成した既存のロールを使用できます。選択するロールがない場合、既存のロールがありません。選択したオプションを問わず、作成するアドオンのドキュメントを参照し、IAM ポリシーを作成してロールにアタッチしてください。IAM ロールを選択するには、クラスター用に IAM OpenID Connect (OIDC) プロバイダーが必要です。クラスター用に持っているかどうかを確認、あるいは作成するには、「クラスターの IAM OIDC プロバイダーを作成する」を参照してください。
- `Code editor` には、アドオン固有の設定情報を入力します。詳細については、更新するアドオンのドキュメントを参照してください。
- [コンフリクト解決方法] で、いずれかのオプションを選択します。アドオン設定にカスタム値を設定している場合、[保存] オプションをお勧めします。このオプションを選択しない場合、Amazon EKS は値をデフォルト値で上書きします。このオプションを使用する場合、実稼働クラスターのアドオンを更新する前に、非稼動クラスターのフィールドおよび値変更をテストすることをお勧めします。

6. [Update] (更新) を選択します。

AWS CLI

前提条件

ご使用のデバイスまたは AWS CloudShell で、バージョン `2.12.3` 以降、または AWS Command Line Interface (AWS CLI) のバージョン `1.27.160` 以降がインストールおよび設定されていること。現在のバージョンを確認するには、「**`aws --version | cut -d / -f2 | cut -d ' ' -f1`**」を参照してください。macOS の `yum`、`apt-get`、または Homebrew などのパッケージマネージャは、AWS CLI の最新バージョンより数バージョン遅れることがあります。最新バージョンをインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI のインストール、更新、およびアンインストール」と「aws configure でのクイック設定」を参照してください。AWS CloudShell にインストールされている AWS CLI バージョンは、最新バージョンより数バージョン遅れている可能性もあります。更新するには、「AWS CloudShellユーザーガイド」の「ホームディレクトリへの AWS CLI のインストール」を参照してください。

AWS CLI を使用して Amazon EKS アドオンを更新する方法

1. インストールされているアドオンのリストを参照してください。*my-cluster* を自分のクラスター名に置き換えます。

```
aws eks list-addons --cluster-name my-cluster
```

出力例は次のとおりです。

```
{
    "addons": [
        "coredns",
        "kube-proxy",
        "vpc-cni"
    ]
}
```

2. 更新するアドオンの現在のバージョンを表示します。*my-cluster* をクラスターの名前に置き換えて、*vpc-cni* を更新するアドオンの名前に置き換えます。

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 "addon.addonVersion" --output text
```

出力例は次のとおりです。

```
v1.10.4-eksbuild.1
```

3. クラスターのバージョンに使用できるアドオンのバージョンを確認できます。*1.30* をクラスターのバージョンに置き換えて、*vpc-cni* を更新するアドオンの名前に置き換えます。

```
aws eks describe-addon-versions --kubernetes-version 1.30 --addon-name vpc-cni \
    --query 'addons[].addonVersions[].{Version: addonVersion, Defaultversion:
 compatibilities[0].defaultVersion}' --output table
```

出力例は次のとおりです。

```
-----------------------------------------
|          DescribeAddonVersions         |
+-----------------+-----------------------+
| Defaultversion  |       Version         |
```

```
+-----------------+----------------------+
|  False          |  v1.12.0-eksbuild.1  |
|  True           |  v1.11.4-eksbuild.1  |
|  False          |  v1.10.4-eksbuild.1  |
|  False          |  v1.9.3-eksbuild.1   |
+-----------------+----------------------+
```

Defaultversion 列に True と表示されているバージョンは、デフォルトでアドオンが作成されたバージョンです。

4. アドオンを更新してください。デバイスに沿ったコマンドをコピーします。必要に応じてコマンドに次の変更を加え、変更したコマンドを実行します。

   - *my-cluster* を自分のクラスター名に置き換えます。
   - *vpc-cni* を前のステップの出力で返された更新するアドオンの名前に置き換えます。
   - *version-number* を更新する前のステップの出力で返されたバージョンに置き換えます。一部のアドオンには推奨バージョンがあります。詳細については、更新するアドオンのドキュメントを参照してください。
   - アドオンが Kubernetes サービスアカウントおよび IAM ロールを使用する場合、*111122223333* をアカウント ID に置き換え、*role-name* を作成した既存の IAM ロールの名前に置き換えます。ロールの作成手順については、作成するアドオンのドキュメントを参照してください。サービスアカウントロールを指定するには、クラスター用に IAM OpenID Connect (OIDC) プロバイダーが必要です。クラスター用に持っているかどうかを確認、あるいは作成するには、「クラスターの IAM OIDC プロバイダーを作成する」を参照してください。

     アドオンが Kubernetes サービスアカウントおよび IAM ロールを使用しない場合、**serviceAccountRoleARN: arn:aws:iam::*111122223333*:role/*role-name*** 行を削除してください。
   - **--resolve-conflicts** *##*オプションはアドオンの既存値を保存します。アドオン設定にカスタム値を設定していて、このオプションを使用しない場合、Amazon EKS は値をデフォルト値で上書きします。このオプションを使用する場合、実稼働クラスターのアドオンを更新する前に、非稼動クラスターのフィールドおよび値変更をテストすることをお勧めします。この値を overwrite に変更する場合、すべての設定が Amazon EKS のデフォルト値に変更されます。いずれかの設定にカスタム値を設定した場合、Amazon EKS のデフォルト値で上書きされる可能性があります。この値を none に変更した場合、Amazon EKS は設定の値を一切変更しませんが、更新が失敗する可能性があります。更新に失敗した場合、競合の解決に役立つエラーメッセージが返されます。

- すべてのカスタム設定を削除する場合は、*--configuration-values '{}'* オプションを使用して更新を実行します。これにより、すべてのカスタム設定がデフォルト値に設定されます。カスタム設定を変更しない場合、*--configuration-values* フラグを指定しないでください。カスタム設定を調整する場合、*{}* を新しいパラメータに置き換えます。パラメータのリストを確認するには、「アドオンの作成」セクションの設定スキーマの表示ステップを参照してください。

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version version-number \
    --service-account-role-arn arn:aws:iam::111122223333:role/role-name --
configuration-values '{}' --resolve-conflicts PRESERVE
```

5. 更新のステータスを確認します。*my-cluster* をクラスターの名前に置き換えて、*vpc-cni* を更新するアドオンの名前に置き換えます。

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni
```

出力例は次のとおりです。

```
{
    "addon": {
        "addonName": "vpc-cni",
        "clusterName": "my-cluster",
        "status": "UPDATING",
[...]
```

ステータスが ACTIVE に変わると更新が完了します。

アドオンの削除

Amazon EKS アドオンを削除するときに次のことを行います。

- アドオンが提供する機能にはダウンタイムはありません。
- サービスアカウントの IAM ロール (IRSA) を使用していて、アドオンに IAM ロールが関連付けられている場合、IAM ロールは削除されません。

- Pod Identity を使用している場合、アドオンが所有する Pod Identity の関連付けはすべて削除されます。AWS CLI に `--preserve` オプションを指定すると、関連付けは保持されます。
- Amazon EKS はアドオンの設定の管理を停止します。
- 新しいバージョンが利用可能になると、コンソールが通知を停止します。
- AWS ツールや API を使用してアドオンを更新することはできません。
- 自己管理できるようにアドオンソフトウェアをクラスターに保持するか、クラスターからアドオンソフトウェアを削除するかを選択できます。そのアドオンが提供している機能に依存するクラスターにリソースがない場合のみ、クラスターからアドオンソフトウェアを削除してください。

`eksctl`、AWS Management Console、AWS CLI を使用して Amazon EKS アドオンをクラスターから削除できます。

eksctl

前提条件

デバイスまたは AWS CloudShell にインストールされている `eksctl` コマンドラインツールのバージョン `0.183.0` 以降。`eksctl` をインストールまたはアップグレードするには、`eksctl` ドキュメントの「インストール」を参照してください。

`eksctl` を使用して Amazon EKS アドオンを削除する方法

1. クラスターに現在インストールされているアドオンを確認します。*`my-cluster`* を自分のクラスター名に置き換えます。

```
eksctl get addon --cluster my-cluster
```

出力例は次のとおりです。

```
NAME        VERSION              STATUS  ISSUES  IAMROLE  UPDATE AVAILABLE
coredns     v1.8.7-eksbuild.2    ACTIVE  0
kube-proxy  v1.23.7-eksbuild.1   ACTIVE  0
vpc-cni     v1.10.4-eksbuild.1   ACTIVE  0
[...]
```

クラスターにあるアドオンおよびバージョンによって、出力が異なる場合があります。

2. アドオンを削除します。*`my-cluster`* をクラスターの名前に置き換えて、*`name-of-add-on`* を削除する前のステップの出力で返されたアドオン名に置き換えます。***`--preserve`*** オプショ

ンを削除する場合、Amazon EKS がアドオンを管理しなくなるだけではなく、アドオンソフトウェアがクラスターから削除されます。

```
eksctl delete addon --cluster my-cluster --name name-of-addon --preserve
```

AWS Management Console

AWS Management Console を使用して Amazon EKS アドオンを削除する方法

1. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。
2. 左のナビゲーションペインで、[クラスター] を選択し、次に Amazon EKS アドオンを削除するクラスターの名前を選択します。
3. [アドオン] タブを選択します。
4. アドオンボックス右上にあるチェックボックスを選択し、次に [削除] を選択します。Amazon EKS がアドオンの設定の管理を停止しても、クラスターのアドオンソフトウェアを保持し、アドオンのすべての設定を自己管理できるようにする場合、[クラスターに保存] を選択します。アドオン名を入力して [削除] を選択します。

AWS CLI

前提条件

デバイスまたは AWS CloudShell にインストールされている `eksctl` コマンドラインツールのバージョン `0.183.0` 以降。`eksctl` をインストールまたはアップグレードするには、`eksctl` ドキュメントの「インストール」を参照してください。

AWS CLI を使用して Amazon EKS アドオンを削除する方法

1. インストールされているアドオンのリストを参照してください。*my-cluster* を自分のクラスター名に置き換えます。

```
aws eks list-addons --cluster-name my-cluster
```

出力例は次のとおりです。

```
{
    "addons": [
```

```
        "coredns",
        "kube-proxy",
        "vpc-cni",
        "name-of-addon"
    ]
}
```

2. インストールされているアドオンを削除します。*my-cluster* をクラスターの名前に置き換えて、*name-of-add-on* を削除するアドオンの名前に置き換えます。***--preserve*** を削除することにより、アドオンがクラスターから削除されます。

```
aws eks delete-addon --cluster-name my-cluster --addon-name name-of-addon --
preserve
```

簡略化した出力例を次に示します。

```
{
    "addon": {
        "addonName": "name-of-add-on",
        "clusterName": "my-cluster",
        "status": "DELETING",
[...]
```

3. 削除の状態を確認します。*my-cluster* をクラスターの名前に置き換えて、*name-of-addon* を削除するアドオンの名前に置き換えます。

```
aws eks describe-addon --cluster-name my-cluster --addon-name name-of-addon
```

アドオンを削除した後の出力例は次のとおりです。

```
An error occurred (ResourceNotFoundException) when calling the DescribeAddon
 operation: No addon: name-of-addon found in cluster: my-cluster
```

アドオンバージョンの互換性を取得する

describe-addon-verisions API を使用して、EKS アドオンの利用可能なバージョンと、各アドオンバージョンがサポートしている Kubernetes バージョンを一覧表示します。

アドオンバージョンの互換性を取得する (AWS CLI)

1. AWS CLI がインストールされ、`aws sts get-caller-identity` と連携していることを確認します。このコマンドが機能しない場合は、「Get started with the AWS CLI」を参照してください。
2. バージョンの互換性情報を取得するアドオンの名前 (`amazon-cloudwatch-observability` など) を特定します。
3. クラスターの Kubernetes バージョンを特定します (`1.28` など)。
4. AWS CLI を使用して、クラスターの Kubernetes バージョンと互換性のあるアドオンのバージョンを取得します。

```
aws eks describe-addon-versions --addon-name amazon-cloudwatch-observability --
kubernetes-version 1.29
```

出力例は次のとおりです。

```
{
    "addons": [
        {
            "addonName": "amazon-cloudwatch-observability",
            "type": "observability",
            "addonVersions": [
                {
                    "addonVersion": "v1.5.0-eksbuild.1",
                    "architecture": [
                        "amd64",
                        "arm64"
                    ],
                    "compatibilities": [
                        {
                            "clusterVersion": "1.28",
                            "platformVersions": [
                                "*"
                            ],
                            "defaultVersion": true
                        }
                    ],
[...]
```

この出力は、アドオンのバージョン `v1.5.0-eksbuild.1` が、Kubernetes クラスターのバージョン `1.28` と互換性があることを示しています。

Kubernetes フィールド管理

Amazon EKS アドオンは、標準的なベストプラクティスによる設定を使用して、クラスターにインストールされます。Amazon EKS アドオンをクラスターに追加する方法については、「Amazon EKS アドオン」を参照してください。

Amazon EKS アドオンの設定をカスタマイズすると、高度な機能を有効にすることが可能です。Amazon EKS は Kubernetes サーバー側の適用機能を使用して、Amazon EKS で管理されていない設定の内容を上書きすることなく、Amazon EKS によるアドオンの管理を可能にします。詳細については、「Kubernetes ドキュメント」の「Server-side Apply」(サーバー側の適用) を参照してください。これを実現するために、Amazon EKS は、インストールするアドオンごとに最低限のフィールドセットを管理します。Amazon EKS、または別の Kubernetes コントロールプレーンプロセス (`kube-controller-manager` など) によって管理されていないすべてのフィールドは、ユーザーが問題なく管理できます。

> **Important**
>
> Amazon EKS で管理しているフィールドを変更すると、Amazon EKS がアドオンを管理できなくなるため、アドオンの更新時にユーザーによる変更が上書きされる可能性があります。

フィールド管理ステータスを表示する

`kubectl` を使用すると、任意の Amazon EKS アドオンについて、どのフィールドが Amazon EKS によって管理されているかを知ることができます。

フィールドの管理ステータスを表示する

1. 確認するアドオンを決定します。クラスターにデプロイされた、すべての `deployments` と `DaemonSets` を表示するには、「Kubernetes リソースを表示する」を参照してください。
2. アドオンのマネージド型フィールドを表示するには、次のコマンドを実行します。

```
kubectl get type/add-on-name -n add-on-namespace -o yaml
```

例えば、次のコマンドを使用すると、CoreDNS アドオンの用マネージド型フィールドを表示できます。

```
kubectl get deployment/coredns -n kube-system -o yaml
```

フィールドの管理状態は、返される出力の中の、次のセクションに記載されています。

```
[...]
managedFields:
  - apiVersion: apps/v1
    fieldsType: FieldsV1
    fieldsV1:
[...]
```

> **Note**
>
> 出力に、managedFields が表示されない場合、**--show-managed-fields** をコマンドに追加し、もう一度実行します。使用している kubectl のバージョンによって、管理フィールドがデフォルトで返されるかどうかが決まります。

Kubernetes API でのフィールド管理構文を理解する

Kubernetes オブジェクトの詳細を表示すると、マネージド型フィールドとアンマネージド型フィールドの両方が出力に返されます。管理対象フィールドは、次のいずれかになります。

- 完全マネージド型 - フィールドのすべてのキーは Amazon EKS によって管理されます。このフィールドの値を変更すると、競合の原因となります。
- 部分的マネージド型 - フィールドの一部のキーは Amazon EKS によって管理されます。Amazon EKS によって明示的に管理されているキーに対する変更のみが競合の原因となります。

どちらのタイプのフィールドも `manager: eks` でタグづけされています。

各キーは、フィールド自体を表す `.` (これは常に空のセットがマッピングされます) であるか、またはサブフィールドまたは項目を表す文字列です。フィールドの管理状況の出力は、以下のタイプの宣言で構成されます。

- `f:`*`name`* (*name* は、リスト内のフィールドの名前)。

- k:*keys* (*keys* は、リスト項目のフィールドのマップ)。
- v:*value* (*value* は、リスト項目を正確な JSON 形式で記述した値)。
- i:*index* (*index* は、リスト内の項目の位置)。

出力内で、以下の CoreDNS アドオンに関する部分には、前の宣言が表示されます。

- フルマネージド型フィールド - マネージド型フィールドに f: (フィールド) が指定されていますが、k: (キー) はありません。この場合は、フィールド全体が管理されています。このフィールドの値を変更すると、競合の原因となります。

  次の出力では、coredns という名前のコンテナが、eks によって管理されていることがわかります。args、image、および imagePullPolicy サブフィールドも eks によって管理されています。このフィールドの値を変更すると、競合の原因となります。

  ```
  [...]
  f:containers:
    k:{"name":"coredns"}:
    .: {}
    f:args: {}
    f:image: {}
    f:imagePullPolicy: {}
  [...]
  manager: eks
  [...]
  ```

- 部分的マネージド型キー - マネージド型キーに値が指定されている場合、宣言されたキーはそのフィールドで管理されます。指定されたキーを変更すると、競合の原因となります。

  次の出力では、name キーを使用する eks が、config-volume および tmp ボリュームセットを管理していることがわかります。

  ```
  [...]
  f:volumes:
    k:{"name":"config-volume"}:
      .: {}
      f:configMap:
        f:items: {}
        f:name: {}
      f:name: {}
    k:{"name":"tmp"}:
  ```

```
    .: {}
    f:name: {}
[...]
manager: eks
[...]
```

- 部分的マネージド型フィールドへのキーの追加 - 特定のキーバリューのみが管理されている場合であれば、競合を引き起こすことなく、引数などの追加のキーをフィールドに追加できます。キーを追加する場合は、まず、そのフィールドがマネージド型でないことを確認してください。マネージド型の値を追加または変更すると、競合の原因となります。

  次の出力では、name キーおよび name フィールドの両方が、マネージド型であることがわかります。コンテナ名を追加または変更すると、このマネージド型のキーとの競合が発生します。

```
[...]
f:containers:
  k:{"name":"coredns"}:
[...]
    f:name: {}
[...]
manager: eks
[...]
```

Pod Identity を使用して Amazon EKS アドオンに IAM ロールをアタッチする

特定の Amazon EKS アドオンでは、AWS API を呼び出す IAM ロールアクセス許可が必要です。例えば、Amazon VPC CNI アドオンは特定の AWS API を呼び出して、アカウント内のネットワークリソースを設定します。これらのアドオンには、AWS IAM を使用してアクセス許可を付与する必要があります。具体的には、アドオンを実行しているポッドのサービスアカウントを、十分な IAM ポリシーを持つ IAM ロールに関連付ける必要があります。

クラスターワークロードに AWS アクセス許可を付与する際の推奨方法は、Amazon EKS 機能 Pod Identity を使用することです。Pod Identity の関連付けを使用して、アドオンのサービスアカウントを IAM ロールにマッピングできます。ポッドが関連付けられたサービスアカウントを使用する場合、Amazon EKS はポッドのコンテナに環境変数を設定します。環境変数は、AWS CLI を含む AWS SDK が EKS Pod Identity の認証情報を使用するように設定します。EKS Pod Identity の詳細をご覧ください。

Amazon EKS アドオンは、アドオンに対応する Pod Identity の関連付けのライフサイクルを管理するのに役立ちます。例えば、Amazon EKS アドオンと必要な Pod Identity の関連付けを 1 回の API コールで作成または更新できます。Amazon EKS には、推奨される IAM ポリシーを取得するための API も用意されています。

推奨される使用法:

1. Amazon EKS Pod Identity エージェントがクラスターで設定されていることを確認します。
2. インストールするアドオンに、`describe-addon-versions` AWS CLI オペレーションを使用して IAM アクセス許可が必要かどうかを確認します。`requiresIamPermissions` フラグが `true` の場合、`describe-addon-configurations` オペレーションを使用してアドオンに必要なアクセス許可を決定する必要があります。レスポンスには、推奨されるマネージド IAM ポリシーのリストが含まれます。
3. `describe-addon-configuration` CLI オペレーションを使用して、Kubernetes サービスアカウントの名前と推奨される IAM ポリシーを取得します。提案されたポリシーの範囲をセキュリティ要件に照らして評価します。
4. 提案されたアクセス許可ポリシーと Pod Identity に必要な信頼ポリシーを使用して IAM ロールを作成します。詳細については、「EKS Pod Identity の関連付けの作成」を参照してください。
5. CLI を使用して Amazon EKS アドオンを作成または更新します。少なくとも 1 つの Pod Identity の関連付けを指定します。Pod Identity の関連付けは、(1) Kubernetes サービスアカウントの名前、および (2) IAM ロールの ARN です。

考慮事項:

- アドオン API を使用して作成された Pod Identity の関連付けは、それぞれのアドオンによって所有されます。アドオンを削除すると、Pod Identity の関連付けも削除されます。AWS CLI または API を使用してアドオンを削除するときに、`preserve` オプションを使用してこのカスケード削除を防ぐことができます。必要に応じて、Pod Identity の関連付けを直接更新または削除することもできます。アドオンは、既存の Pod Identity の関連付けの所有権を引き受けることはできません。既存の関連付けを削除し、アドオンの作成または更新オペレーションを使用して再作成する必要があります。
- Amazon EKS では、Pod Identity の関連付けを使用してアドオンの IAM アクセス許可を管理することをお勧めします。前の方法であるサービスアカウント (IRSA) の IAM ロールは引き続きサポートされています。アドオンの IRSA `serviceAccountRoleArn` と Pod Identity の関連付けの両方を指定できます。EKS Pod Identity エージェントがクラスターにインストールされている場

合、serviceAccountRoleArn は無視され、EKS は指定された Pod Identity の関連付けを使用します。Pod Identity が有効になっていない場合、serviceAccountRoleArn が使用されます。

- 既存のアドオンの Pod Identity の関連付けを更新すると、Amazon EKS はアドオンポッドのローリング再起動を開始します。

アドオンに関する IAM 情報を取得する

AWS CLI を使用して、(1) アドオンに IAM アクセス許可が必要かどうか、および (2) そのアドオンに推奨される IAM ポリシーを決定できます。

Amazon EKS アドオン (AWS CLI) に関する IAM 情報を取得する

1. インストールするアドオンの名前とクラスターの Kubernetes バージョンを決定します。Amazon EKS アドオンの詳細をご覧ください。
2. AWS CLI を使用してアドオンに IAM アクセス許可が必要かどうかを確認します。

```
aws eks describe-addon-versions \
--addon-name <addon-name> \
--kubernetes-version <kubernetes-version>
```

例:

```
aws eks describe-addon-versions \
--addon-name aws-ebs-csi-driver \
--kubernetes-version 1.30
```

以下の出力例を確認します。requiresIamPermissions は true であり、デフォルトのアドオンバージョンであることに注意してください。推奨される IAM ポリシーを取得するときは、アドオンバージョンを指定する必要があります。

```
{
    "addons": [
        {
            "addonName": "aws-ebs-csi-driver",
            "type": "storage",
            "addonVersions": [
                {
                    "addonVersion": "v1.31.0-eksbuild.1",
```

```
                    "architecture": [
                        "amd64",
                        "arm64"
                    ],
                    "compatibilities": [
                        {
                            "clusterVersion": "1.30",
                            "platformVersions": [
                                "*"
                            ],
                            "defaultVersion": true
                        }
                    ],
                    "requiresConfiguration": false,
                    "requiresIamPermissions": true
                },
[...]
```

3. アドオンに IAM アクセス許可が必要な場合は、AWS CLI を使用して推奨される IAM ポリシーを取得します。

```
aws eks describe-addon-configuration \
--query podIdentityConfiguration \
--addon-name <addon-name> \
--addon-version <addon-version>
```

例:

```
aws eks describe-addon-configuration \
--query podIdentityConfiguration \
--addon-name aws-ebs-csi-driver \
--addon-version v1.31.0-eksbuild.1
```

以下の出力を確認します。recommendedManagedPolicies を書き留めます。

```
[
    {
        "serviceAccount": "ebs-csi-controller-sa",
        "recommendedManagedPolicies": [
            "arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy"
        ]
    }
```

```
]
```

4. IAM ロールを作成し、管理ポリシーをアタッチします。または、管理ポリシーを確認し、必要に応じてアクセス許可の範囲を絞り込みます。EKS Pod Identity で使用する IAM ロールを作成する手順を確認します。

IAM ロールでアドオンを更新する

Pod Identity Association (AWS CLI) を使用するために Amazon EKS アドオンを更新する

1. 決定事項:
   - `cluster-name` – アドオンをインストールする EKS クラスターの名前。
   - `addon-name` – インストールする Amazon EKS アドオンの名前。
   - `service-account-name` – アドオンで使用される Kubernetes サービスアカウントの名前。
   - `iam-role-arn` – アドオンに十分なアクセス許可を持つ IAM ロールの ARN。IAM ロールには、EKS Pod Identity に必要な信頼ポリシーが必要です。
2. AWS CLI を使用してアドオンを更新します。同じ `--pod-identity-assocations` 構文を使用して、アドオンの作成時に Pod Identity の関連付けを指定することもできます。アドオンの更新中に Pod Identity の関連付けを指定すると、以前のすべての Pod Identity の関連付けが上書きされることに注意してください。

   ```
   aws eks update-addon --cluster-name <cluster-name> \
   --addon-name <addon-name> \
   --pod-identity-associations 'serviceAccount=<service-account-name>,roleArn=<role-arn>'
   ```

   例:

   ```
   aws eks update-addon --cluster-name mycluster \
   --addon-name aws-ebs-csi-driver \
   --pod-identity-associations 'serviceAccount=ebs-csi-controller-sa,roleArn=arn:aws:iam::123456789012:role/StorageDriver'
   ```

3. Pod Identity の関連付けが作成されたことを確認します。

   ```
   aws eks list-pod-identity-associations --cluster-name <cluster-name>
   ```

成功すると、次のような出力が表示されます。EKS アドオンの OwnerARN を書き留めます。

```
{
    "associations": [
        {
            "clusterName": "mycluster",
            "namespace": "kube-system",
            "serviceAccount": "ebs-csi-controller-sa",
            "associationArn": "arn:aws:eks:us-
west-2:123456789012:podidentityassociation/mycluster/a-4wvljrezsukshq1bv",
            "associationId": "a-4wvljrezsukshq1bv",
            "ownerArn": "arn:aws:eks:us-west-2:123456789012:addon/mycluster/aws-
ebs-csi-driver/9cc7ce8c-2e15-b0a7-f311-426691cd8546"
        }
    ]
}
```

アドオンから関連付けを削除する

Amazon EKS アドオン (AWS CLI) からすべての Pod Identity の関連付けを削除する

1. 決定事項:
   - `cluster-name` – アドオンをインストールする EKS クラスターの名前。
   - `addon-name` – インストールする Amazon EKS アドオンの名前。
2. アドオンを更新して、Pod Identity の関連付けの空の配列を指定します。

```
aws eks update-addon --cluster-name <cluster-name> \
--addon-name <addon-name> \
--pod-identity-associations "[]"
```

EKS アドオンの Pod Identity のトラブルシューティング

AWS API、SDK、または CLI オペレーションの試行中にアドオンでエラーが発生した場合は、以下を確認します。

- Pod Identity エージェントがクラスター にインストールされている。
  - Pod Identity エージェントのセットアップ方法を確認する。

- アドオンに有効な Pod Identity の関連付けがある。
  - AWS CLI を使用して、アドオンで使用されるサービスアカウント名の関連付けを取得します。

```
aws eks list-pod-identity-associations --cluster-name <cluster-name>
```

- 目的の IAM ロールに、EKS Pod Identity に必要な信頼ポリシーがある。
  - AWS CLI を使用して、アドオンの信頼ポリシーを取得します。

```
aws iam get-role --role-name <role-name> --query Role.AssumeRolePolicyDocument
```

- 目的の IAM ロールに、アドオンに必要なアクセス許可がある。
  - AWS CloudTrail を使用して `AccessDenied` または `UnauthorizedOperation` イベントを確認します。
- Pod Identity の関連付けのサービスアカウント名は、アドオンで使用されるサービスアカウント名と一致している。
  - アドオンのドキュメントを確認して、サービスアカウント名を確認します。

デプロイ中のコンテナイメージの検証

AWS Signer を使用し、デプロイ時に署名済みのコンテナイメージを検証したい場合は、次のいずれかのソリューションが使用可能です。

- Gatekeeper と Ratify — Gatekeeper をアドミッションコントローラーとして使用し、AWS Signer プラグインで構成された Ratify を署名を検証するためのウェブフックとして使用します。
- Kyverno — 署名を検証するための AWS Signer プラグインで構成された Kubernetes ポリシーエンジン。

> **Note**
>
> コンテナイメージの署名を検証する前に、選択したアドミッションコントローラーの要求に応じて、Notation トラストストアとトラストポリシーを構成します。

Elastic Fabric Adapter を使用した機械学習トレーニング

このトピックでは、Elastic Fabric Adapter (EFA) を Amazon EKS クラスターにデプロイした Pods と統合する方法について説明します。Elastic Fabric Adapter (EFA) は、Amazon EC2 インスタンス向けのネットワークインターフェイスです。これにより、AWS で高レベルのノード間通信を必要とするアプリケーションを実行できます。カスタムビルドされたオペレーティングシステムのバイパスハードウェアインターフェイスにより、インスタンス間通信のパフォーマンスが向上します。これは、これらのアプリケーションのスケーリングに不可欠です。EFA を使用すると、Message Passing Interface (MPI) を使用するハイパフォーマンスコンピューティング (HPC) アプリケーションと NVIDIA Collective Communications Library (NCCL) を使用する Machine Learning (ML) アプリケーションを、数千の CPU または GPU にスケールできます。その結果、AWS クラウドのオンデマンドの伸縮自在性と柔軟性を備えたオンプレミスの HPC クラスターのアプリケーションパフォーマンスを実現できます。EFA と Amazon EKS クラスターで実行中のアプリケーションを統合すると、大規模な分散型トレーニングワークロードを完了する時間を短縮できます。クラスターにインスタンスを追加する必要はありません。EFA の詳細については、「Elastic Fabric Adapter」を参照してください。

このトピックで説明する EFA プラグインは、クラウドでの分散型機械学習における最先端の技術を象徴する Amazon EC2 の P4d インスタンスを完全にサポートしています。各 `p4d.24xlarge` インスタンスには、NVIDIA A100 GPU が 8個、400 Gbps の GPUDirectRDMA over EFA が搭載されています。GPUDirectRDMA を使用すると、CPU バイパスを使用することでノード間での GPU の直接通信が可能になります。これにより総合的な通信帯域幅が増加し、レイテンシーが短縮されます。Amazon EKS および EFA が P4d インスタンスと統合することで、分散型機械学習トレーニング向けの最適なパフォーマンスを提供する Amazon EC2 コンピューティングインスタンスをシームレスに活用できます。

前提条件

- 既存の Amazon EKS クラスター。既存のクラスターがない場合は、Amazon EKS の使用開始 ガイドのいずれかを活用して作成してください。クラスターは、ノードをデプロイするのに十分な IP アドレスを持つ、少なくとも 1 つのプライベートサブネットがある VPC にデプロイする必要があります。プライベートサブネットには、NAT ゲートウェイなどの外部のデバイスから提供されるアウトバウンドのインターネットアクセスが必要です。

  `eksctl` を使用してノードグループを作成する予定がある場合、`eksctl` でクラスターを作成することもできます。
- ご使用のデバイスまたは AWS CloudShell で、バージョン `2.12.3` 以降、または AWS Command Line Interface (AWS CLI) のバージョン `1.27.160` 以降がインストールおよび設定されているこ

と。現在のバージョンを確認するには、「**aws --version | cut -d / -f2 | cut -d ' ' -f1**」を参照してください。macOS の yum、apt-get、または Homebrew などのパッケージマネージャは、AWS CLI の最新バージョンより数バージョン遅れることがあります。最新バージョンをインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI のインストール、更新、およびアンインストール」と「aws configure でのクイック設定」を参照してください。AWS CloudShell にインストールされている AWS CLI バージョンは、最新バージョンより数バージョン遅れている可能性もあります。更新するには、「AWS CloudShell ユーザーガイド」の「ホームディレクトリへの AWS CLI のインストール」を参照してください。

- デバイスまたは AWS CloudShell に、kubectl コマンドラインツールがインストールされていること。バージョンは、ご使用のクラスターの Kubernetes バージョンと同じか、1 つ前のマイナーバージョン以前、あるいはそれより新しいバージョンが使用できます。例えば、クラスターのバージョンが 1.29 である場合、kubectl のバージョン 1.28、1.29、または 1.30 が使用できます。kubectl をインストールまたはアップグレードする方法については、「kubectl のインストールまたは更新」を参照してください。
- p4d.24xlarge などの複数の Elastic Fabric Adapters をサポートするワーカーノードを起動する前に、Amazon VPC CNI plugin for Kubernetes バージョン 1.7.10 以降をインストールしておく必要があります。Amazon VPC CNI plugin for Kubernetes バージョンの更新する方法の詳細については、「Amazon VPC CNI plugin for Kubernetes Amazon EKS アドオンの使用」を参照してください。

ノードグループの作成

次の手順は、EFA インターフェイスと GPUDirect RDMA を使用して、p4d.24xlarge がバックアップされたノードグループでノードグループを作成し、EFA を使用したマルチノード NCCL パフォーマンス向けに NVIDIA Collective Communications Library (NCCL) のサンプルテストを実行するのに役立ちます。この例では、EFA を使用した Amazon EKS の分散型深層学習トレーニングのテンプレートを使用できます。

1. ノードをデプロイする対象の AWS リージョンで利用可能な、EFA をサポートする Amazon EC2 インスタンスを特定します。*region-code* を、ノードグループをデプロイする対象の AWS リージョンに置き換えます。

```
aws ec2 describe-instance-types --region region-code --filters Name=network-
info.efa-supported,Values=true \
    --query "InstanceTypes[*].[InstanceType]" --output text
```

ノードをデプロイする場合、デプロイするインスタンスタイプは、ユーザーのクラスターがある AWS リージョン で利用可能である必要があります。

2. デプロイするインスタンスタイプが利用可能なアベイラビリティーゾーンを特定します。このチュートリアルでは、p4d.24xlarge インスタンスタイプが使用されているため、前のステップで指定した AWS リージョン の出力が返されているはずです。本番クラスターにノードをデプロイするときは、*p4d.24xlarge* を前のステップで返されたいずれかのインスタンスタイプに置き換えます。

```
aws ec2 describe-instance-type-offerings --region region-code --location-type
 availability-zone --filters Name=instance-type,Values=p4d.24xlarge \
    --query 'InstanceTypeOfferings[*].Location' --output text
```

出力例は次のとおりです。

```
us-west-2a    us-west-2c    us-west-2b
```

後のステップで使用するために、返されたアベイラビリティーゾーンをメモします。ノードをクラスターにデプロイする場合、VPC には、出力で返されるアベイラビリティーゾーンの 1 つに使用可能な IP アドレスを持つサブネットが必要です。

3. eksctl、または AWS CLI と AWS CloudFormation を使用してノードグループを使用します。

eksctl

前提条件

デバイスまたは AWS CloudShell にインストールされている eksctl コマンドラインツールのバージョン 0.183.0 以降。eksctl をインストールまたはアップグレードするには、eksctl ドキュメントの「インストール」を参照してください。

1. 次の内容を *efa-cluster.yaml* という名前のファイルにコピーします。*example values* を自分の値に置き換えます。*p4d.24xlarge* を異なるインスタンスに置き換えることができますが、その場合、availabilityZones の値がステップ 1 でインスタンスタイプに対して返されたアベイラビリティーゾーンであることを確認してください。

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
```

```
  name: my-efa-cluster
  region: region-code
  version: "1.XX"

iam:
  withOIDC: true

availabilityZones: ["us-west-2a", "us-west-2c"]

managedNodeGroups:
  - name: my-efa-ng
    instanceType: p4d.24xlarge
    minSize: 1
    desiredCapacity: 2
    maxSize: 3
    availabilityZones: ["us-west-2a"]
    volumeSize: 300
    privateNetworking: true
    efaEnabled: true
```

2. 既存のクラスターにマネージド型ノードグループを作成します。

```
eksctl create nodegroup -f efa-cluster.yaml
```

既存のクラスターがない場合は、次のコマンドを実行してクラスターとノードグループを作成できます。

```
eksctl create cluster -f efa-cluster.yaml
```

> **Note**
>
> この例で使用されているインスタンスタイプには GPU があるため、eksctl は NVIDIA Kubernetes デバイスプラグインを各インスタンスに自動的にインストールします。

AWS CLI and AWS CloudFormation

EFA ネットワークには、EFA 固有のセキュリティグループの作成、Amazon EC2 プレイスメントグループの作成、1 つまたは複数の EFA インターフェイスを指定する起動テンプレー

トの作成など、いくつかの要件があります。また、Amazon EC2 ユーザーデータの一部としての EFA ドライバーのインストールも含みます。EFA の要件の詳細については、「Amazon EC2 ユーザーガイド」の「EFA と MPI の開始方法」を参照してください。次の手順では、これらのすべてを作成します。すべての *###* を独自の値で置き換えます。

1. 後の手順で使用するいくつかの変数を設定します。すべての *example values* を独自の値で置き換えます。*my-cluster* を既存のクラスターの名前に置き換えます。node_group_resources_name の値は、後で AWS CloudFormation スタックを作成するために使用されます。node_group_name の値は、後でクラスターにノードグループを作成するために使用されます。

```
cluster_name="my-cluster"
cluster_region="region-code"
node_group_resources_name="my-efa-nodegroup-resources"
node_group_name="my-efa-nodegroup"
```

2. デプロイするインスタンスタイプが利用可能であるように、同じアベイラビリティーゾーンにある VPC 内のプライベートサブネットを決定します。

   a. クラスターのバージョンを取得し、後のステップで使用するために変数に格納します。

```
cluster_version=$(aws eks describe-cluster \
    --name $cluster_name \
    --query "cluster.version" \
    --output text)
```

   b. クラスターがある VPC の ID を取得し、後のステップで使用するために変数に格納します。

```
vpc_id=$(aws eks describe-cluster \
    --name $cluster_name \
    --query "cluster.resourcesVpcConfig.vpcId" \
    --output text)
```

   c. クラスターのコントロールプレーンセキュリティグループの ID を取得し、後のステップで使用するために変数に格納します。

```
control_plane_security_group=$(aws eks describe-cluster \
    --name $cluster_name \
    --query "cluster.resourcesVpcConfig.clusterSecurityGroupId" \
    --output text)
```

d. ステップ 1 で返されたアベイラビリティーゾーンにある VPC 内のサブネット ID のリストを取得します。

```
aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=$vpc_id" "Name=availability-
zone,Values=us-west-2a" \
    --query 'Subnets[*].SubnetId' \
    --output text
```

出力が返されない場合は、ステップ 1 で返された別のアベイラビリティーゾーンを試します。ステップ 1 で返されたアベイラビリティーゾーンにサブネットがない場合は、そのアベイラビリティーゾーンにサブネットを作成する必要があります。VPC 内に別のサブネットを作成するスペースがない場合は、VPC に CIDR ブロックを追加して、新しい CIDR ブロック内にサブネットを作成するか、新しい VPC 内に新しいクラスターを作成することができます。

e. サブネットのルートテーブルを確認して、サブネットがプライベートサブネットかどうかを判定してください。

```
aws ec2 describe-route-tables \
    --filter Name=association.subnet-id,Values=subnet-0d403852a65210a29 \
    --query "RouteTables[].Routes[].GatewayId" \
    --output text
```

出力例は次のとおりです。

```
local
```

出力が `local igw-02adc64c1b72722e2` の場合、サブネットはパブリックサブネットになります。ステップ 1 で返されたアベイラビリティーゾーンで、プライベートサブネットを選択する必要があります。プライベートサブネットを決定したら、後のステップで使用するために ID を書き留めておきます。

f. 後の手順で使用するために、前のステップのプライベートサブネット ID を持つ変数を設定します。

```
subnet_id=your-subnet-id
```

3. AWS CloudFormation テンプレートをダウンロードします。

```
curl -O https://raw.githubusercontent.com/aws-samples/aws-efa-eks/main/
cloudformation/efa-p4d-managed-nodegroup.yaml
```

4. 次のテキストをコンピュータにコピーします。*p4d.24xlarge* を、ステップ 1 のインスタンスタイプに置き換えます。*subnet-0d403852a65210a29* を、ステップ 2。b.v で特定したプライベートサブネットの ID で置き換えます。*path-to-downloaded-cfn-template* を、前のステップでダウンロードした efa-p4d-managed-nodegroup.yaml へのパスで置き換えます。*your-public-key-name* を、パブリックキーの名前に置き換えます。置き換えが完了したら、変更したコマンドを実行します。

```
aws cloudformation create-stack \
    --stack-name ${node_group_resources_name} \
    --capabilities CAPABILITY_IAM \
    --template-body file://path-to-downloaded-cfn-template \
    --parameters \
       ParameterKey=ClusterName,ParameterValue=${cluster_name} \
       ParameterKey=ClusterControlPlaneSecurityGroup,ParameterValue=
${control_plane_security_group} \
       ParameterKey=VpcId,ParameterValue=${vpc_id} \
       ParameterKey=SubnetId,ParameterValue=${subnet_id} \
       ParameterKey=NodeGroupName,ParameterValue=${node_group_name} \
       ParameterKey=NodeImageIdSSMParam,ParameterValue=/aws/service/eks/
optimized-ami/${cluster_version}/amazon-linux-2-gpu/recommended/image_id \
       ParameterKey=KeyName,ParameterValue=your-public-key-name \
       ParameterKey=NodeInstanceType,ParameterValue=p4d.24xlarge
```

5. 前のステップでデプロイしたスタックをいつデプロイするかを決定します。

```
aws cloudformation wait stack-create-complete --stack-name
 $node_group_resources_name
```

前のコマンドからの出力はありませんが、スタックが作成されるまでシェルプロンプトは返されません。

6. 前のステップで AWS CloudFormation スタックによって作成されたリソースを使用して、ノードグループを作成します。

   a. デプロイされた AWS CloudFormation スタックから情報を取得し、変数に格納します。

```
node_instance_role=$(aws cloudformation describe-stacks \
    --stack-name $node_group_resources_name \
    --query='Stacks[].Outputs[?OutputKey==`NodeInstanceRole`].OutputValue'
 \
    --output text)
launch_template=$(aws cloudformation describe-stacks \
    --stack-name $node_group_resources_name \
    --query='Stacks[].Outputs[?OutputKey==`LaunchTemplateID`].OutputValue'
 \
    --output text)
```

b. 前のステップで作成した起動テンプレートとノード IAM ロールを使用するマネージド型ノードグループを作成します。

```
aws eks create-nodegroup \
    --cluster-name $cluster_name \
    --nodegroup-name $node_group_name \
    --node-role $node_instance_role \
    --subnets $subnet_id \
    --launch-template id=$launch_template,version=1
```

c. ノードが作成されたことを確認します。

```
aws eks describe-nodegroup \
   --cluster-name ${cluster_name} \
   --nodegroup-name ${node_group_name} | jq -r .nodegroup.status
```

前のコマンドから返されたステータスが [ACTIVE] になるまで続行しないでください。ノードの準備が完了するまで数分かかることがあります。

7. GPU インスタンスタイプを選択した場合は、Kubernetes 用の NVIDIA デバイスプラグインをデプロイする必要があります。次のコマンドを実行する前に、*vX.X.X* を必要となる NVIDIA/k8s-device-plugin バージョンに置き換えます。

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-
plugin/vX.X.X/nvidia-device-plugin.yml
```

4. EFA Kubernetes デバイスプラグインをデプロイします。

EFA Kubernetes デバイスプラグインでは、EFA インターフェイスを Kubernetes に割り当て可能なリソースとして検出およびアドバタイズできます。アプリケーションでは、拡張リソース

タイプ vpc.amazonaws.com/efa を CPU やメモリのように Pod リクエスト仕様で使用できます。詳細については、「Kubernetes ドキュメント」の「拡張したリソースの消費」を参照してください。リクエストされると、プラグインは EFA インターフェイスを自動的に Pod に割り当ておよびマウントします。デバイスのプラグインを使用すると、EFA の設定が簡単になります。Pod を特権モードで実行する必要はありません。

```
helm repo add eks https://aws.github.io/eks-chart
helm install aws-efa-k8s-device-plugin --namespace kube-system eks/aws-efa-k8s-device-plugin
```

(オプション) サンプルの EFA 互換アプリケーションをデプロイする

Kubeflow MPI Operator のデプロイ

NCCL テストでは、Kubeflow MPI Operator を適用できます。MPI Operator を使用すると、Kubernetes で AllReduce スタイルの分散型トレーニングを簡単に実行できます。詳細については、「GitHub」の「MPI Operator」を参照してください。

```
kubectl apply -f https://raw.githubusercontent.com/kubeflow/mpi-operator/master/deploy/v2beta1/mpi-operator.yaml
```

マルチノードの NCCL パフォーマンステストを実行して、GPUDirectRDMA/EFA を検証する

GPUDirectRDMA over EFA を使用した NCCL パフォーマンスを検証するには、標準の NCCL パフォーマンステストを実施します。詳細については、「GitHub」で公式の「NCCL-Tests」(NCCL テスト) を参照してください。サンプルの Dockerfile を使用できます。サンプルには、このテストを NVIDIA CUDA 11.2 および EFA の最新バージョン向けに構築したものが付属しています。

または、Amazon ECR リポジトリから取得可能な AWS Docker イメージをダウンロードできます。

> **Important**
>
> Kubernetes で EFA を導入するために必要となる重要な考慮事項は、クラスター内のリソースとして Huge Page を設定および管理することです。詳細については、「Kubernetes ドキュメント」の「Huge Page の管理」を参照してください。EFA ドライバーがインストールされた Amazon EC2 インスタンスでは、5128 2M Huge Page が事前に割り当てられます。このページは、ジョブの仕様で使用するリソースとしてリクエストできます。

2 ノード NCCLL パフォーマンステストを実行するには、次の手順を実行します。サンプルの NCCL テストジョブでは、各ワーカーが 8 つの GPU、5210Mi の hugepages-2Mi、4 つの EFA、8000Mi のメモリをリクエストします。これは事実上、各ワーカーが `p4d.24xlarge` インスタンスのすべてのリソースを使用することを意味します。

1. NCCL テストのジョブを作成します。

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/aws-efa-eks/main/examples/simple/nccl-efa-tests.yaml
```

出力例は次のとおりです。

mpijob.kubeflow.org/nccl-tests-efa が作成されました

2. 実行中の Pods を表示する。

```
kubectl get pods
```

出力例は次のとおりです。

```
NAME                              READY   STATUS     RESTARTS   AGE
nccl-tests-efa-launcher-nbql9  0/1     Init:0/1   0          2m49s
nccl-tests-efa-worker-0           1/1     Running    0          2m49s
nccl-tests-efa-worker-1           1/1     Running    0          2m49s
```

MPI Operator は、ランチャー Pod と 2 つのワーカー Pods (各ノードに 1 つ) を作成します。

3. `efa-launcher` Pod のログを表示します。*`wzr8j`* を出力の値で置き換えます。

```
kubectl logs -f nccl-tests-efa-launcher-nbql9
```

その他の例については、「GitHub」の「Amazon EKS EFA samples repository」(Amazon EKS EFA サンプルリポジトリ) を参照してください。

AWS Inferentia を使用した機械学習推論

このトピックでは、Amazon EC2 Inf1 インスタンスを実行中のノードで Amazon EKS クラスターを作成する方法、(オプションで) サンプルアプリケーションをデプロイする方法について説明します。Amazon EC2 Inf1 インスタンスは AWS Inferentia チップを利用しています。Inf1 チップは、ク

ラウド内で高パフォーマンスと低コストの推論を提供するために、AWS によってカスタムビルドされたものです。機械学習モデルは、AWS Neuron を使用してコンテナにデプロイされます。Neuron は、コンパイラー、ランタイム、および Inferentia チップの機械学習推論パフォーマンスを最適化するプロファイリングツールで構成されるソフトウェア開発キット (SDK) です。AWSNeuron は、TensorFlow、PyTorch、MXNet などの一般的な機械学習フレームワークをサポートしています。

> **Note**
>
> Neuron デバイスの論理 ID は連続している必要があります。複数の Neuron デバイスをリクエストする Pod が `inf1.6xlarge` または `inf1.24xlarge` インスタンス タイプ (複数の Neuron デバイスを持つ) でスケジュールされている場合、Kubernetes スケジューラが連続していないデバイス ID を選択すると、その Pod は起動に失敗します。詳細については、「GitHub」の「Device logical IDs must be contiguous」(デバイスロジカル ID は連続している必要があります) を参照してください。

前提条件

- コンピュータに `eksctl` がインストールされている。インストールされていない場合は、`eksctl` ドキュメントの「インストール」を参照してください。
- コンピュータに `kubectl` がインストールされている。詳細については、「kubectl のインストールまたは更新」を参照してください。
- (オプション) コンピュータに `python3` がインストールされている。インストールされていない場合は、「Python downloads」でインストール手順を参照してください。

クラスターの作成

Inf1 Amazon EC2 インスタンスを実行するノードを持つクラスターを作成するには

1. Inf1 Amazon EC2 インスタンスを実行するノードを持つクラスターを作成します。*`inf1.2xlarge`* を任意の inf1 インスタンスタイプ に置き換えることができます。`eksctl` ユーティリティは、`Inf1` インスタンスタイプでノードグループを起動していることを検出し、アクセラレーションされた Amazon EKS 最適化 Amazon Linux AMI のいずれかを使用してノードを起動します。

> **Note**
>
> TensorFlow Serving でサービスアカウント用の IAM ロールを使用することはできません。

```
eksctl create cluster \
    --name inferentia \
    --region region-code \
    --nodegroup-name ng-inf1 \
    --node-type inf1.2xlarge \
    --nodes 2 \
    --nodes-min 1 \
    --nodes-max 4 \
    --ssh-access \
    --ssh-public-key your-key \
    --with-oidc
```

> **Note**
>
> 出力の次の行の値をメモします。これは、後の (オプションの) ステップで使用されます。
>
> ```
> [9]  adding identity "arn:aws:iam::111122223333:role/
> eksctl-inferentia-nodegroup-ng-in-NodeInstanceRole-FI7HIYS3BS09" to auth
>  ConfigMap
> ```

Inf1 インスタンスでノードグループを起動すると、eksctl によって AWS Neuron Kubernetes デバイスプラグインが自動的にインストールされます。このプラグインは、Neuron デバイスをシステムリソースとして Kubernetes スケジューラにアドバタイズします。このスケジューラはコンテナによってリクエストできます。デフォルトの Amazon EKS ノードの IAM ポリシーに加えて、Amazon S3 読み取り専用のアクセスポリシーが追加されます。これにより、後のステップで説明するサンプルアプリケーションで Amazon S3 からトレーニングされたモデルをロードできるようになります。

2. すべての Pods が正しく起動していることを確認します。

```
kubectl get pods -n kube-system
```

簡略化された出力:

```
NAME                                   READY   STATUS    RESTARTS   AGE
[...]
neuron-device-plugin-daemonset-6djhp   1/1     Running   0          5m
neuron-device-plugin-daemonset-hwjsj   1/1     Running   0          5m
```

(オプション) TensorFlow Serving アプリケーションイメージのデプロイ

トレーニング済みモデルは、Inferentia インスタンスにデプロイする前に、Inferentia ターゲットにコンパイルする必要があります。続行するには、Amazon S3 に保存されている Neuron optimized TensorFlow モデルが必要になります。SavedModel がまだない場合は、Neuron 互換の ResNet50 モデルの作成についてのチュートリアルに従って、結果の SavedModel を S3 にアップロードします。ResNet-50 は、画像認識のタスクに使用される一般的な機械学習モデルです。Neuron モデルのコンパイルについての詳細は、「AWS デベロッパーガイド」の「AWS Deep Learning AMI Inferentia チップと DLAMI」を参照してください。

サンプルのデプロイメントマニフェストでは、AWS Deep Learning Containers によって提供される TensorFlow の事前構築済みの推論サービングコンテナが管理されます。コンテナ内には、AWS Neuron ランタイムと TensorFlow Serving アプリケーションがあります。Neuron 用に最適化された事前構築済みの深層学習コンテナの完全なリストは、GitHub で [Available Images] (利用可能なイメージ) に保持されます。起動時に、DLC は Amazon S3 からモデルを取得し、保存したモデルで Neuron TensorFlow Serving を起動して、予測リクエストのために待機します。

サービングアプリケーションに割り当てられる Neuron デバイスの数は、デプロイメント yaml で `aws.amazon.com/neuron` リソースを変更することで調整できます。TensorFlow Serving と Neuron ランタイム間の通信は GRPC 経由で行われることに注意してください。GRPC では、`IPC_LOCK` 機能をコンテナに追加する必要があります。

1. クラスターの作成 のステップ 1 で作成したノードのインスタンスロールに `AmazonS3ReadOnlyAccess` IAM ポリシーを追加します。これは、サンプルアプリケーションが Amazon S3 からトレーニングされたモデルをロードできるようにするために必要です。

```
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess \
```

```
    --role-name eksctl-inferentia-nodegroup-ng-in-NodeInstanceRole-FI7HIYS3BS09
```

2. 次の内容で、rn50_deployment.yaml という名前のファイルを作成します。希望の設定に合わせて、リージョンコードとモデルパスを更新します。モデル名は、クライアントが TensorFlow サーバーにリクエストを行う際の識別を目的としています。この例では、モデル名を使用してサンプルの ResNet50 のクライアントスクリプトと照合します。このスクリプトは、後のステップで予測リクエストを送信する際に使用されます。

```
aws ecr list-images --repository-name neuron-rtd --registry-id 790709498068 --
region us-west-2
```

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: eks-neuron-test
  labels:
    app: eks-neuron-test
    role: master
spec:
  replicas: 2
  selector:
    matchLabels:
      app: eks-neuron-test
      role: master
  template:
    metadata:
      labels:
        app: eks-neuron-test
        role: master
    spec:
      containers:
        - name: eks-neuron-test
          image: 763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference-
neuron:1.15.4-neuron-py37-ubuntu18.04
          command:
            - /usr/local/bin/entrypoint.sh
          args:
            - --port=8500
            - --rest_api_port=9000
            - --model_name=resnet50_neuron
            - --model_base_path=s3://your-bucket-of-models/resnet50_neuron/
          ports:
```

```
            - containerPort: 8500
            - containerPort: 9000
          imagePullPolicy: IfNotPresent
          env:
            - name: AWS_REGION
              value: "us-east-1"
            - name: S3_USE_HTTPS
              value: "1"
            - name: S3_VERIFY_SSL
              value: "0"
            - name: S3_ENDPOINT
              value: s3.us-east-1.amazonaws.com
            - name: AWS_LOG_LEVEL
              value: "3"
          resources:
            limits:
              cpu: 4
              memory: 4Gi
              aws.amazon.com/neuron: 1
            requests:
              cpu: "1"
              memory: 1Gi
          securityContext:
            capabilities:
              add:
                - IPC_LOCK
```

3. モデルをデプロイします。

```
kubectl apply -f rn50_deployment.yaml
```

4. 次の内容で、rn50_service.yaml というファイルを作成します。予測リクエストを受け入れるために HTTP ポートと gRPC ポートが開かれます。

```
kind: Service
apiVersion: v1
metadata:
  name: eks-neuron-test
  labels:
    app: eks-neuron-test
spec:
  type: ClusterIP
  ports:
```

```
    - name: http-tf-serving
      port: 8500
      targetPort: 8500
    - name: grpc-tf-serving
      port: 9000
      targetPort: 9000
  selector:
    app: eks-neuron-test
    role: master
```

5. TensorFlow モデルサービスアプリケーション用の Kubernetes サービスを作成します。

```
kubectl apply -f rn50_service.yaml
```

(オプション) TensorFlow Serving サービスに対する予測を行う

1. ローカルでテストするには、gRPC ポートを eks-neuron-test サービスに転送します。

```
kubectl port-forward service/eks-neuron-test 8500:8500 &
```

2. 次の内容で tensorflow-model-server-infer.py という Python スクリプトを作成します。このスクリプトは、サービスフレームワークである gRPC を介して推論を実行します。

```
import numpy as np
   import grpc
   import tensorflow as tf
   from tensorflow.keras.preprocessing import image
   from tensorflow.keras.applications.resnet50 import preprocess_input
   from tensorflow_serving.apis import predict_pb2
   from tensorflow_serving.apis import prediction_service_pb2_grpc
   from tensorflow.keras.applications.resnet50 import decode_predictions

   if __name__ == '__main__':
       channel = grpc.insecure_channel('localhost:8500')
       stub = prediction_service_pb2_grpc.PredictionServiceStub(channel)
       img_file = tf.keras.utils.get_file(
           "./kitten_small.jpg",
           "https://raw.githubusercontent.com/awslabs/mxnet-model-server/master/
docs/images/kitten_small.jpg")
       img = image.load_img(img_file, target_size=(224, 224))
       img_array = preprocess_input(image.img_to_array(img)[None, ...])
```

```
        request = predict_pb2.PredictRequest()
        request.model_spec.name = 'resnet50_inf1'
        request.inputs['input'].CopyFrom(
            tf.make_tensor_proto(img_array, shape=img_array.shape))
        result = stub.Predict(request)
        prediction = tf.make_ndarray(result.outputs['output'])
        print(decode_predictions(prediction))
```

3. スクリプトを実行して、予測をサービスに送信します。

```
python3 tensorflow-model-server-infer.py
```

出力例は次のとおりです。

```
[[(u'n02123045', u'tabby', 0.68817204), (u'n02127052', u'lynx', 0.12701613),
 (u'n02123159', u'tiger_cat', 0.08736559), (u'n02124075', u'Egyptian_cat',
 0.063844085), (u'n02128757', u'snow_leopard', 0.009240591)]]
```

クラスターの管理

この章では、クラスターの管理に役立つ以下のトピックについて説明します。AWS Management Console を使用して Kubernetes リソース に関する情報を表示することもできます。

- Kubernetes ダッシュボードは、Kubernetes クラスター用の汎用で Web ベースの UI です。これにより、ユーザーはクラスター自体を管理するだけでなく、クラスターで実行されているアプリケーションの管理およびトラブルシューティングを行うことができます。詳細については、「Kubernetes ダッシュボード」GitHub リポジトリを参照してください。
- Kubernetes メトリクスサーバーのインストール - Kubernetes メトリクスサーバーは、クラスター内のリソース使用状況データを集約します。これはデフォルトではクラスターにデプロイされませんが、Kubernetes ダッシュボードや Horizontal Pod Autoscaler などの Kubernetes アドオンによって使用されます。このトピックでは、メトリクスサーバーのインストール方法を説明します。
- Amazon EKS での Helm の使用 - Kubernetes 用の Helm パッケージマネージャーは、Kubernetes クラスターでアプリケーションをインストールおよび管理するために役立ちます。このトピックは、Helm バイナリをインストールして実行する際に役立ちます。これにより、ローカルコンピューターで Helm CLI を使用してチャートをインストールおよび管理することができます。
- Amazon EKS リソースのタグ付け – Amazon EKS リソースを管理しやすくするために、タグ形式で各リソースに独自のメタデータを割り当てることができます。ここでは、タグとその作成方法について説明します。
- Amazon EKS Service Quotas – AWS アカウントには、AWS のサービスごとにデフォルトのクォータ (以前は制限と呼ばれていました) があります。Amazon EKS のクォータと拡大方法について説明します。

コストモニタリング

コストのモニタリングは、Amazon EKS で Kubernetes クラスターを管理する上で不可欠な要素です。クラスターのコストを可視化することで、リソースの使用率を最適化し、予算を設定し、デプロイに関するデータ主導の意思決定を行うことができます。Amazon EKS には 2 つのコストモニタリングソリューションがあり、それぞれに独自の利点があり、コストを効果的に追跡して割り当てるのに役立ちます。

AWS 請求を活用した Amazon EKS のコスト配分データの分割 — このネイティブ機能は、AWS 請求コンソールにシームレスに統合されるため、他の AWS のサービスと同様の使い慣れたインターフェイスとワークフローを使って、コストの分析および割り当てを行うことができます。コスト配分

の分割を使用すると、Kubernetes コストに関するインサイトを他の AWS 支出と並行して直接得ることができるため、AWS 環境全体でコストを簡単に、総合的に最適化できるようになります。コストカテゴリーやコスト異常検出など、既存の AWS 請求の機能を活用して、コスト管理機能をさらに強化することもできます。詳細については、「AWS ユーザーガイド」の「Understanding split cost allocation data」を参照してください。

Kubecost — Amazon EKS は、Kubernetes のコストモニタリングツールである Kubecost をサポートしています。Kubecost には、リソースごとの詳細なコストの内訳、コスト最適化のために推奨される方法、すぐに使用できるダッシュボードやレポートなど、Kubernetes ネイティブのコストモニタリングに特化した機能が多数用意されています。また、Kubecost を使用して AWS コストと使用状況レポートと統合すると、正確な価格データを取得でき、Amazon EKSのコストを正確に把握することができます。インストール方法については「Kubecost をインストールするには」を参照してください。

AWS 請求 — コスト配分の分割

Amazon EKS の AWS コスト配分データの分割を使用したコストモニタリング

Amazon EKS の AWS コスト配分データの分割を使用すると、Amazon EKS クラスターのコストをきめ細かく把握することができます。これにより、Kubernetes アプリケーションのコストと使用状況の分析、最適化、チャージバックが可能になります。アプリケーションのコストは、Kubernetes アプリケーションが使用した Amazon EC2 CPU とメモリリソースに基づいて個々の部門およびチームに割り当てられます。Amazon EKS のコスト配分データの分割を使用することで、ポッドあたりのコストを可視化し、名前空間、クラスター、その他 Kubernetes プリミティブを使用しているポッドごとのコストデータを集計することができます。以下は、Amazon EKS コスト配分データの分析に使用できる Kubernetes プリミティブの一例です。

- クラスター名
- デプロイ
- 名前空間
- ノード
- ワークロード名
- ワークロードタイプ

コスト配分データの分割の詳細については、「AWS Billing ユーザーガイド」の「Understanding split cost allocation data」を参照してください。

コストと使用状況レポートの設定

コスト管理のコンソール、AWS Command Line Interface、AWS SDK のいずれかで、[EKS のコスト配分データの分割] をオンにします。

コスト配分データの分割には以下を使用します。

1. コスト配分データの分割にオプトインします。詳細については、「AWS Cost and Usage Report ユーザーガイド」の「分割コスト配分データの有効化」を参照してください。
2. 新しいレポートまたは既存のレポートにデータを含めます。
3. レポートを表示します。請求とコスト管理コンソールを使用するか、Amazon Simple Storage Service でレポートファイルを表示できます。

Kubecost

Amazon EKS がサポートする Kubecost を使用して、Pods、ノード、名前空間、ラベルなどの Kubernetes リソースごとにコストを分類して監視することができます。Kubernetes のプラットフォーム管理者および財務リーダーとして、Kubecost を使用して Amazon EKS の請求の内訳を可視化できます。また、コストを配分し、アプリケーションチームなどの組織単位にチャージバックすることも可能です。社内チームや事業部門に、実際の AWS の請求に基づく透明で正確なコストデータを提供できます。さらに、インフラストラクチャ環境とクラスター内の使用パターンに基づいて、コストを最適化するためのカスタマイズされた推奨事項を取得することもできます。Kubecost の詳細については、Kubecost のドキュメントを参照してください。

Amazon EKS は、クラスターコストの可視化のため、AWS に最適化された Kubecost のバンドルを提供します。既存の AWS サポート契約を使用してサポートを受けることができます。

前提条件

- 既存の Amazon EKS クラスター。デプロイするには、「Amazon EKS の使用開始」を参照してください。Fargate ノードでは Kubecost を実行できないため、クラスターには Amazon EC2 ノードが必要です。
- デバイスまたは AWS CloudShell に、`kubectl` コマンドラインツールがインストールされていること。バージョンは、ご使用のクラスターの Kubernetes バージョンと同じか、1 つ前のマイナーバージョン以前、あるいはそれより新しいバージョンが使用できます。例えば、クラスターのバージョンが `1.29` である場合、`kubectl` のバージョン `1.28`、`1.29`、または `1.30` が使用できます。`kubectl` をインストールまたはアップグレードする方法については、「`kubectl` のインストールまたは更新」を参照してください。

- Helm バージョン 3.9.0 以降がデバイスまたは AWS CloudShell に設定されていること。Helm をインストールまたは更新するには、「the section called “Helm の使用”」を参照してください。
- クラスターがバージョン 1.23 以降の場合は、the section called “Amazon EBS CSI ドライバー” がクラスターにインストールされている必要があります。

Kubecost をインストールするには

1. インストールする Kubecost のバージョンを確認します。利用可能なバージョンは、Amazon ECR Public Gallery の kubecost/cost-analyzer で確認できます。Kubecost バージョンと Amazon EKS の互換性の詳細については、Kubecost ドキュメントの「環境要件」を参照してください。
2. 以下のコマンドを使用して、Kubecost をインストールします。*kubecost-version* は、*1.108.1* などの ECR から取得された値に置き換えてください。

```
helm upgrade -i kubecost oci://public.ecr.aws/kubecost/cost-analyzer --
version kubecost-version \
    --namespace kubecost --create-namespace \
    -f https://raw.githubusercontent.com/kubecost/cost-analyzer-helm-chart/develop/
cost-analyzer/values-eks-cost-monitoring.yaml
```

   Kubecost は新しいバージョンを定期的にリリースします。`helm upgrade` を使用して、バージョンを更新できます。デフォルトでは、インストールにはローカルの Prometheus サーバーと `kube-state-metrics` が含まれています。「Integrating with Amazon EKS cost monitoring」(Amazon EKS コスト監視との統合) のドキュメントに従って、Amazon Managed Service for Prometheus を使用するようにデプロイをカスタマイズできます。使用できる他のすべての設定のリストについては、「GitHub」の「サンプル設定ファイル」を参照してください。
3. 必要な Pods が動作していることを確認します。

```
kubectl get pods -n kubecost
```

   出力例は次のとおりです。

```
NAME                                          READY   STATUS    RESTARTS   AGE
kubecost-cost-analyzer-b9788c99f-5vj5b        2/2     Running   0          3h27m
kubecost-kube-state-metrics-99bb8c55b-bn2br   1/1     Running   0          3h27m
kubecost-prometheus-server-7d9967bfc8-9c8p7   2/2     Running   0          3h27m
```

4. デバイス上でポート転送を有効にして、Kubecost ダッシュボードを公開します。

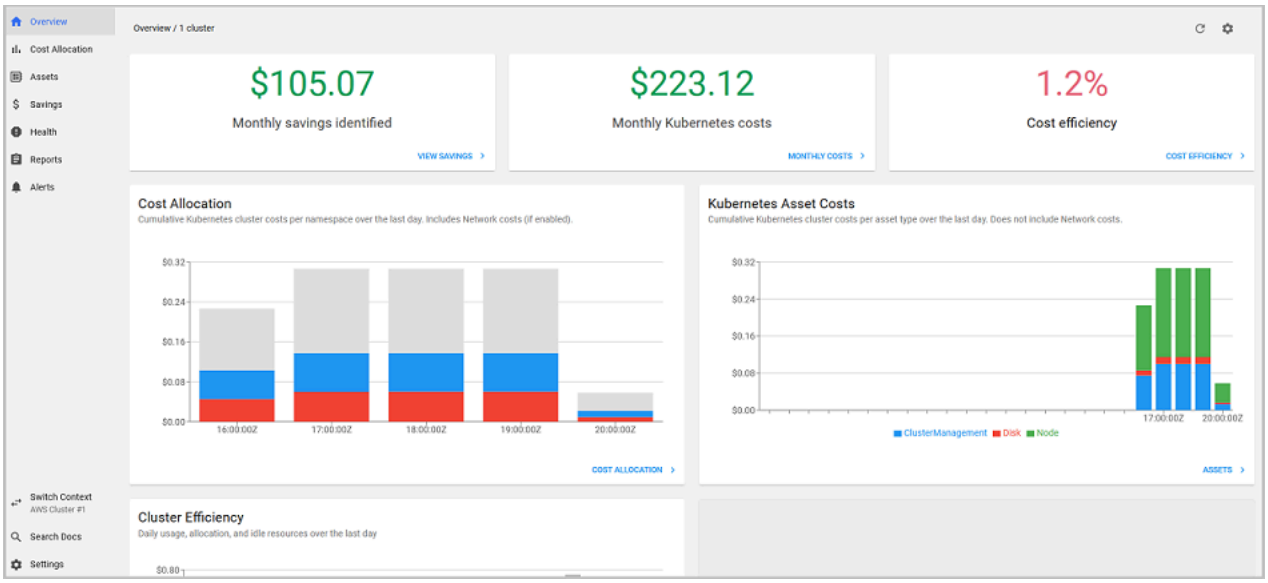
```
kubectl port-forward --namespace kubecost deployment/kubecost-cost-analyzer 9090
```

または、AWS Load Balancer Controller を使用して Kubecost を公開し、Amazon Cognito を使用して認証、認可、およびユーザー管理を行うこともできます。詳細については、「Application Load Balancer および Amazon Cognitoを使用して Kubernetes Web アプリのユーザーを認証する方法」を参照してください。

5. 前の手順を実行したのと同じデバイスで、Web ブラウザを開き、次のアドレスを入力します。

```
http://localhost:9090
```

ブラウザに Kubecost の概要ページが表示されます。Kubecost がメトリクスを収集するには 5 ~ 10 分かかる場合があります。クラスターにかかる累積コスト、関連する Kubernetes 資産コスト、および毎月の総支出を含む Amazon EKS の支出額を確認できます。



6. クラスターレベルでコストを追跡するには、請求対象の Amazon EKS リソースにタグを付けます。詳細については、「請求用のリソースにタグを付ける」を参照してください。

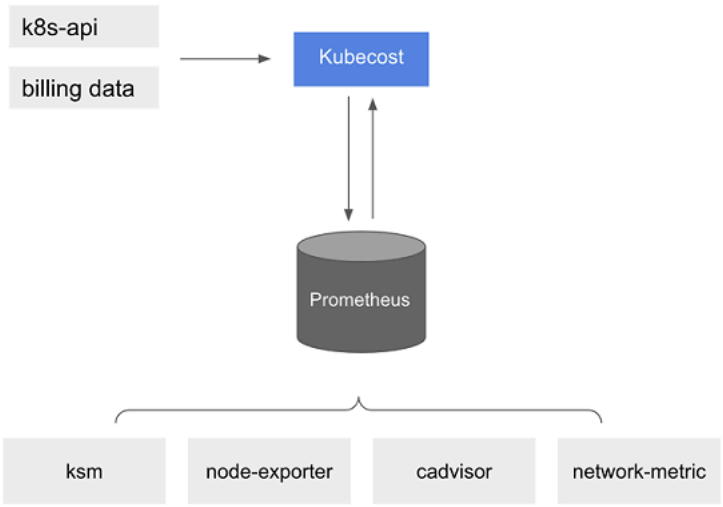
ダッシュボードの左ペインで選択すると、次の情報も表示できます。

- コスト配分 — Amazon EKS の月間費用と、各名前空間およびその他のディメンションの過去 7 日間の累積費用を表示します。これは、アプリケーションのどの部分が Amazon EKS の支出に寄与しているかを理解するのに役立ちます。

- アセット — Amazon EKS リソースに関連する AWS インフラストラクチャアセットの費用を表示します。

その他の機能

- コストメトリクスのエクスポート — Amazon EKS に最適化されたコストモニタリングは、オープンソースのモニタリングシステムおよび時系列データベースである Kubecost および Prometheus と一緒にデプロイされます。Kubecost は Prometheus からメトリックを読み取り、コスト配分計算を実行して、メトリックを Prometheus に書き込みます。Kubecost フロントエンドが Prometheus からメトリクスを読み込み、それらを Kubecost ユーザーインターフェイスに表示します。このアーキテクチャを以下の図に示します。



  Prometheus がプリインストールされていると、クエリを記述して Kubecost データをインジェストし、現在のビジネスインテリジェンスシステムに取り込んでさらに分析できます。また、現在の Grafana ダッシュボードのデータソースとして使用して、社内チームが精通している Amazon EKS クラスターの費用を表示することもできます。Prometheus クエリの記述方法の詳細については、GitHub の「Prometheus 設定」 `readme` ファイルを参照するか、Kubecost Github リポジトリの「Grafana JSON モデル」サンプルを参考として使用してください。

- AWS Cost and Usage Report 統合 — Amazon EKS クラスターのコスト配分計算を実行するため、Kubecost は AWS Price List API から AWS のサービス および AWS リソースの公開価格情報を取得します。Kubecost と AWS Cost and Usage Report を統合して、AWS アカウントに固有の価格情報の正確性を高めることもできます。この情報には、エンタープライズディスカウントプログラム、リザーブドインスタンスの使用量、Savings Plans、スポット使用量が含まれます。AWS

Cost and Usage Report との統合の動作方法の詳細については、Kubecost ドキュメントの「AWS クラウド請求の統合」を参照してください。

Kubecost を削除します。

次のコマンドを使用して、クラスターから Kubecost を削除できます。

```
helm uninstall kubecost --namespace kubecost
kubectl delete ns kubecost
```

よくある質問

Amazon EKS での Kubecost の使用に関する次の一般的な質問と回答を参照してください。

Kubecost のカスタムバンドルと Kubecost の無料バージョン (OpenCost としても知られている) の違いは何ですか?

AWS と Kubecost が協力して Kubecost のカスタマイズバージョンを提供しました。このバージョンには、追加料金なしで商用機能のサブセットが含まれています。Kubecost のカスタムバンドルに含まれる機能については、次の表を参照してください。

機能	Kubecost 無料利用枠	Amazon EKS 最適化済み Kubecost カスタムバンドル	Kubecost Enterprise
デプロイメント	ユーザーホスト	ユーザーホスト	ユーザーホストまたは Kubecost ホスト (SaaS)
[サポートされるクラスター数]	無制限	無制限	無制限
[サポートされるデータベース]	ローカル Prometheus	ローカル Prometheus または Amazon Managed Service for Prometheus	Prometheus、Amazon Managed Service for Prometheus、Cortex、または Thanos

機能	Kubecost 無料利用枠	Amazon EKS 最適化済み Kubecost カスタムバンドル	Kubecost Enterprise
[データベース保持サポート]	15 日間	無制限の履歴データ	無制限の履歴データ
Kubecost API 保持 (ETL)	15 日間	15 日間	無制限の履歴データ
[クラスターコストの可視化]	単一クラスター	統合マルチクラスター	統合マルチクラスター
[ハイブリッドクラウドの可視性]	-	Amazon EKS と Amazon EKS Anywhere クラスター	マルチクラウドとハイブリッドクラウドのサポート
[アラートと定期レポート]	-	効率性アラート、予算アラート、支出変更アラートなどをサポート	効率性アラート、予算アラート、支出変更アラートなどをサポート
[保存されたレポート]	-	15 日間のデータを使用したレポート	無制限の履歴データを使用したレポート
[クラウド請求の統合]	個々のクラスターに必須	AWS のカスタム価格サポート (複数のクラスターと複数のアカウントを含む)	AWS のカスタム価格サポート (複数のクラスターと複数のアカウントを含む)
[割引に関する推奨事項]	単一クラスターの洞察	単一クラスターの洞察	マルチクラスターの洞察
[ガバナンス: 監査]	-	-	過去のコストイベントの監査
[シングルサインオン (SSO) のサポート]	-	Amazon Cognito のサポート	Okta、Auth 0、PingID、KeyCloak

機能	Kubecost 無料利用枠	Amazon EKS 最適化済み Kubecost カスタムバンドル	Kubecost Enterprise
[SAML **2.0** を使用したロールベースのアクセスコントロール (RBAC)]	-	-	Okta, Auth0, PingID, Keycloak
[エンタープライズトレーニングとオンボーディング]	-	-	フルサービストレーニングと FinOps オンボーディング

Kubecost API 保持 (ETL) 機能とは?

Kubecost ETL 機能では、メトリクスを集約して整理し、さまざまな詳細レベルでコストを可視化します (`namespace-level`、`pod-level`、`deployment-level` など)。カスタム Kubecost バンドルでは、過去 15 日間のメトリクスからデータと洞察を取得することができます。

アラート機能と定期レポート機能とは何ですか? どのようなアラートとレポートが含まれますか?

Kubecost アラートでは、チームはリアルタイムで Kubernetes 支出とクラウド支出に関する最新情報を受け取ることができます。定期レポートでは、チームは履歴 Kubernetes とクラウド支出のカスタマイズされたビューを取得できます。どちらも、Kubecost UI または Helm 値で構成できます。メール、Slack、Microsoft Teams をサポートしています。

保存レポートには何が含まれますか?

Kubecost 保存レポートは、コストと効率のメトリクスを示す事前定義済みのビューです。クラスター、名前空間、ラベル、その他を基準とするコストが含まれます。

クラウド請求の統合とは何ですか?

AWS 請求 API との統合で、Kubecost にクラスター以外のコスト (Amazon S3 など) が表示されます。さらに、Kubecost は、スポット使用状況、Savings Plans、企業割引を考慮して、Kubecost のクラスター内予測を実際の請求データと調整できるようになります。

割引に関する推奨事項にはどのようなものがありますか?

Kubecost は、ユーザーが Kubernetes インフラと支出を最適化するのに役立つ洞察と自動化を提供します。

この機能には料金がかかりますか?

いいえ。このバージョンの Kubecost は追加料金なしで使用できます。このバンドルに含まれない追加の Kubecost 機能が必要な場合は、AWS Marketplace を通じて Kubecost のエンタープライズライセンスを購入するか、Kubecost から直接購入できます。

サポートは受けられますか?

はい。AWS へのお問い合わせから AWS Support チームに対してサポートケースを開くことができます。

Amazon EKS 統合によって提供される Kubecost 機能を使用するには、ライセンスが必要ですか?

いいえ。

より正確なレポートを作成するために、Kubecost を AWS Cost and Usage Report と統合できますか?

はい。AWS Cost and Usage Report からデータを取り込むように Kubecost を構成することで、割引、スポット料金、予約インスタンス料金などを含む正確なコストを可視化することができます。詳細については、Kubecost ドキュメントの「AWS クラウド請求の統合」を参照してください。

このバージョンは、Amazon EC2 の自己管理型 Kubernetes クラスターのコスト管理をサポートしていますか?

いいえ。このバージョンは Amazon EKS クラスターとのみ互換性があります。

Kubecost は AWS Fargate で Amazon EKS のコストを追跡できますか?

Kubecost では、Fargate での Amazon EKS のクラスターコストが最大限に可視化されますが、Amazon EC2 の Amazon EKS に比べて精度が低くなります。これは主に、使用量に対する課金方法の違いによるものです。Fargate の Amazon EKS では、消費したリソースに対して課金されます。Amazon EC2 ノードの Amazon EKS では、プロビジョニングされたリソースに対して課金されます。Kubecost は、CPU、RAM、一時ストレージを含むノード仕様に基づいて Amazon EC2 ノードのコストを計算します。Fargate のコストは、Fargate Pods のリクエストされたリソースに基づいて計算されます。

Kubecost の更新と新バージョンはどのように入手できますか?

標準の Helm アップグレード手順を使用して、Kubecost バージョンをアップグレードできます。最新バージョンは、Amazon ECR Public Gallery にあります。

`kubectl-cost` CLI はサポートされていますか? どのようにインストールすればよいですか?

はい。`Kubectl-cost` は、Kubecost (Apache 2.0 ライセンス) によるオープンソースツールで、Kubernetes コスト配分の指標に CLI でアクセスできるようにするものです。`kubectl-cost` をインストールするには、「GitHub」の「インストール」を参照してください。

Kubecost ユーザーインタフェイスはサポートされていますか? どのようにアクセスすればよいですか?

Kubecost は、`kubectl` ポートフォワーディング、イングレス、またはロードバランサーを介してアクセスできる Web ダッシュボードを提供します。または、AWS Load Balancer Controller を使用して Kubecost を公開し、Amazon Cognito を使用して認証、承認、およびユーザー管理を行うこともできます。詳細については、AWS ブログの「Application Load Balancer および Amazon Cognito を使用して Kubernetes Web アプリのユーザーを認証する方法」を参照してください。

Amazon EKS Anywhere はサポートされていますか?

いいえ。

Kubernetes メトリクスサーバーのインストール

Kubernetes メトリクスサーバーは、クラスター内のリソース使用状況データを集約し、デフォルトでは Amazon EKS クラスターにデプロイされません。詳細については、「GitHub」の「Kubernetes メトリクスサーバー」を参照してください。メトリクスサーバーは一般的に、他の Kubernetes アドオン Horizontal Pod Autoscaler や Kubernetes ダッシュボード などによって使用されます。詳細については、「Kubernetes ドキュメント」の「リソースメトリクスパイプライン」を参照してください。このトピックでは、Kubernetes メトリクスサーバーを Amazon EKS クラスターにデプロイする方法について説明します。

> **Important**
>
> メトリクスはポイントインタイム分析を目的としており、履歴分析の正確なソースではありません。監視ソリューションとして、またはその他の自動スケーリング以外の目的には使用できません。監視ツールの詳細については、「Amazon EKS でのオブザーバビリティ」を参照してください。

メトリクスサーバーをデプロイする

1. 次のコマンドを使用してメトリクスサーバーをデプロイします。

```
kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml
```

Fargate を使用している場合は、このファイルを変更する必要があります。デフォルト設定では、メトリクスサーバーはポート 10250 を使用します。このポートは Fargate で予約されています。components.yaml のポート 10250 への参照を 10251 などの別のポートに置き換えます。

2. 次のコマンドを使用して、`metrics-server` デプロイで必要な数の Pods が実行されていることを確認します。

```
kubectl get deployment metrics-server -n kube-system
```

出力例は次のとおりです。

```
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
metrics-server   1/1     1            1           6m
```

Amazon EKS での Helm の使用

Kubernetes の Helm パッケージマネージャーは、Kubernetes クラスターにアプリケーションをインストールして管理するのに役立ちます。詳細については、Helm のドキュメントを参照してください。このトピックは、Helm バイナリをインストールして実行する際に役立ちます。これにより、ローカルシステムで Helm CLI を使用してチャートをインストールおよび管理することができます。

Important

Amazon EKS クラスターに Helm チャートをインストールするには、あらかじめ Amazon EKS で動作するように `kubectl` を設定しておく必要があります。この設定をまだ行っていない場合は、続行する前に「Amazon EKS クラスターの kubeconfig ファイルを作成または更新する」を参照してください。次のコマンドがクラスターに対して成功した場合は、正しく設定されています。

```
kubectl get svc
```

ローカルシステムに Helm バイナリをインストールするには

1. クライアントオペレーティングシステムに適したコマンドを実行します。

   - macOS で Homebrew を使用している場合は、次のコマンドを使用してバイナリをインストールします。

     ```
     brew install helm
     ```

   - Chocolatey で Windows を使用している場合は、次のコマンドを使用してバイナリをインストールします。

     ```
     choco install kubernetes-helm
     ```

   - Linux を使用している場合は、次のコマンドを使用してバイナリをインストールします。

     ```
     curl https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 >
      get_helm.sh
     chmod 700 get_helm.sh
     ./get_helm.sh
     ```

     > **Note**
     >
     > 最初に openssl をインストールする必要があるというメッセージを受け取った場合、次のコマンドを使用してインストールできます。
     >
     > ```
     > sudo yum install openssl
     > ```

2. PATH で新しいバイナリを取得するため、現在のターミナルウィンドウを閉じて新しいターミナルウィンドウを開きます。
3. インストールした Helm のバージョンを確認してください。

   ```
   helm version | cut -d + -f 1
   ```

   出力例は次のとおりです。

   ```
   v3.9.0
   ```

4. この時点で、任意の Helm コマンド (`helm install` *`chart-name`* など) を実行して、クラスター内の Helm チャートをインストール、変更、削除、またはクエリすることができます。Helm を初めて使用する場合で、インストールするチャートがないときは、次の操作を実行できます。

   - サンプルチャートをインストールして試します。Helm の「クイックスタートガイド」の「サンプルチャートをインストールする」を参照してください。
   - サンプルチャートを作成し、Amazon ECR にプッシュします。詳細については、「Amazon Elastic Container Registry ユーザーガイド」の「Pushing a Helm chart (Helm チャートをプッシュする)」を参照してください。
   - [eks-charts] GitHub レプロ、および [ArtifactHub] から Amazon EKS チャートをインストールします。

Amazon EKS リソースのタグ付け

タグ を使用すると、Amazon EKS リソースの管理に役立ちます。このトピックでは、タグ機能の概要とタグの作成方法について説明します。

トピック

> **Note**
>
> タグは、Kubernetes ラベルや注釈とは別のメタデータの一種です。これらの他のメタデータタイプの詳細については、Kubernetes ドキュメントの次のセクションを参照してください。
>
> - ラベル (Labels) とセレクター (Selectors)
> - 注釈

タグの基本

タグとは、AWS リソースに割り当てるラベルです。各タグはキーとオプションの値で構成されます。

タグを使用して AWS リソースを分類できます。例えば、目的、所有者、環境などに基づいてリソースを分類できます。同じ型のリソースが多い場合に、特定のリソースに割り当てたタグを使用して、そのリソースをすばやく識別できます。例えば Amazon EKS クラスターに一連のタグを定義して、各クラスターの所有者とスタックレベルを追跡できます。リソースタイプごとに一貫した一連のタグキーを考案することをお勧めします。追加したタグに基づいてリソースを検索およびフィルタリングできます。

タグを追加したら、いつでもタグキーと値は編集でき、タグはリソースからいつでも削除できます。リソースを削除すると、リソースのタグも削除されます。

タグには Amazon EKS に関する意味論的な意味はなく、完全に文字列として解釈されます。タグの値は空の文字列に設定できます。ただし、タグの値を null に設定することはできません。そのリソースの既存のタグと同じキーを持つタグを追加した場合、古い値は新しい値によって上書きされます。

AWS Identity and Access Management (IAM) を使用すると、AWS アカウント内のどのユーザーがタグを管理するアクセス許可を持っているかを制御できます。

リソースのタグ付け

Amazon EKS の以下のリソースがタグをサポートしています。

- クラスター
- マネージド型ノードグループ
- Fargate プロファイル

以下を使用して、これらのリソースにタグ付けできます。

- Amazon EKS コンソールを使用している場合、新規または既存のリソースにいつでもタグを適用できます。これを行うには、関連リソースページの [Tags] (タグ) タブを使用します。詳細については、「コンソールでのタグの処理」を参照してください。
- `eksctl` を使用している場合は、`--tags` オプションを使ってリソースを作成する際に、リソースにタグを適用できます。

- AWS CLI、Amazon EKS API、または AWS SDK を使用している場合は、関連する API アクションの `tags` パラメータを使用して、新しいリソースにタグを適用できます。`TagResource` API アクションを使用して既存のリソースにタグを適用することもできます。詳細については、「TagResource」を参照してください。

リソース作成アクションによっては、リソースの作成時にリソースのタグを指定することもできます。リソースの作成中にタグを適用できない場合、リソースの作成は失敗します。これにより、タグ付けするリソースが、指定したタグで作成されているか、まったく作成されていないかが確認できます。リソースの作成時にタグ付けを行う場合、リソースの作成後にカスタムタグ付けスクリプトを実行する必要はありません。

タグは、作成したリソースに関連付けられている他のリソースには伝達されません。例えば、Fargate プロファイルタグは、Fargate プロファイルに関連付けられている Pods など、Fargate プロファイルに関連付けられている他のリソースには伝播されません。

タグの制限

タグには以下の制限があります。

- 1 つのリソースに対して最大 50 個のタグを関連付けることができます。
- 1 つのリソースに対してタグキーを繰り返すことはできません。各タグキーは一意である必要があり、それぞれに使用できる値は 1 つのみです。
- キーの値には最大 128 UTF-8 文字を使用できます。
- 値の最大長は 256 UTF-8 文字です。
- 複数の AWS のサービスおよびリソースがタグ付けスキーマを使用する場合、使用する文字の種類を制限します。一部のサービスでは、使用できる文字に制限がある場合があります。通常、使用できる文字は、英字、数字、スペース、および特殊文字 **+**、**-**、**=**、**.**、**_**、**:**、**/**、**@** です。
- タグのキーと値では、大文字と小文字が区別されます。
- `aws:`、`AWS:`、またはその大文字または小文字の組み合わせを、キーまたは値のプレフィックスとして使用しないでください。これらは AWS でのみ使用するように予約されています。このプレフィックスを持つタグのキーや値を編集または削除することはできません。このプレフィックスを持つタグは、リソースあたりのタグ数の制限にはカウントされません。

請求用のリソースにタグを付ける

Amazon EKS クラスターにタグを適用すると、そのタグを [Cost & Usage Reports] (コストと使用状況レポート) 内のコスト配分に使用できます。[Cost & Usage Reports] (コストと使用状況レポート) の計測データには、すべての Amazon EKS タスクの使用状況が示されます。詳細については、「AWS Billing ユーザーガイド」の「AWS コストと使用状況レポート」を参照してください。

AWS 生成されたコスト配分タグ、特に `aws:eks:cluster-name` では、[Cost Explorer] (コストエクスプローラー) で Amazon EC2 インスタンスのコストを個々の Amazon EKS クラスターごとに分類できます。ただし、このタグにはコントロールプレーンの費用を捕捉するものではありません。このタグは、Amazon EKS クラスターに参加している Amazon EC2 インスタンスに自動的に追加されます。この動作は、インスタンスが Amazon EKS マネージドノードグループを使用してプロビジョニングされているかどうかに関係なく発生します。Karpenter、または Amazon EC2 で直接発生します。この特定のタグは 50 個のタグの上限にはカウントされません。タグを使用するには、アカウント所有者が AWS Billing コンソールまたは API を使用してタグをアクティブ化する必要があります。AWS Organizations 管理アカウントの所有者がこのタグをアクティブ化すると、すべての組織のメンバーアカウントにも有効になります。

同じタグキー値を持つリソースに基づいて、請求情報を整理することもできます。例えば、複数のリソースに特定のアプリケーション名のタグを付け、請求情報を整理することができます。これにより、複数のサービスを利用しているアプリケーションの合計コストを確認することができます。タグによるコスト配分レポートの設定の詳細については、AWS Billing ユーザーガイドのコスト配分月次レポートを参照してください。

> **Note**
>
> レポートを有効にすると、約 24 時間後に、今月のデータを表示できるようになります。

[Cost Explorer] (コストエクスプローラー) は、AWS 無料利用枠の一部として利用できるレポートツールです。Cost Explorer を使用すると、過去 13 か月間の Amazon EKS リソースのグラフを表示できます。また、次の 3 か月間のリソース使用量も予測できます。時間の経過と共に AWS リソースに費やす金額のパターンを確認できます。例えば、Cost Explorer を使用して、さらに調べる必要がある分野を特定し、コストを把握するために使用できる傾向を確認できます。データの時間範囲を指定したり、時間データを日または月ごとに表示することもできます。

コンソールでのタグの処理

Amazon EKS コンソールを使用して、新規または既存のクラスターおよびマネージド型ノードグループに関連付けられたタグを管理できます。

Amazon EKS コンソールでリソース固有のページを選択すると、そのページにリソースのリストが表示されます。例えば、左のナビゲーションペインで [Clusters] (クラスター) を選択すると、Amazon EKS クラスターのリストがコンソールに表示されます。これらのリストの1つ(例えば、特定のクラスター)からタグをサポートするリソースを選択すると、Tagsタブでタグを表示および管理できます。

また、AWS Management Console の [Tag Editor] (タグエディタ) を使用することもできます。これにより、タグを一貫性のある方法で管理できます。詳細については、「AWS タグエディタユーザーガイド」の「タグエディタを使用して AWS リソースにタグを付ける」を参照してください。

リソース作成時のタグの追加

Amazon EKS クラスターおよびマネージド型ノードグループ、および Fargate プロファイルの作成時に、タグを追加できます。詳細については、「Amazon EKS クラスターの作成」を参照してください。

リソースでのタグの追加と削除

クラスターに関連付けられたタグをリソースのページから直接追加または削除できます。

個々のリソースのタグを追加または削除するには

1. https://console.aws.amazon.com/eks/home#/clusters で Amazon EKS コンソールを開きます。
2. ナビゲーションバーから、使用する AWS リージョンを選択します。
3. 左のナビゲーションペインで [クラスター] を選択します。
4. 指定するクラスターを選択します。
5. [タグ] タブを選択し、[タグ管理] を選択します。
6. [Manage tags] (タグの管理) ページで、必要に応じてタグを追加または削除します。
   - タグを追加するには、タグの追加 を選択します。その後、タグごとにキーと値を指定します。
   - タグを削除するには、[Remove tag] (タグの削除) を選択します。
7. 追加または削除するタグごとにこのプロセスを繰り返します。

8. [更新] を選択して終了します。

CLI、API または **eksctl** でのタグの操作

リソースのタグの追加、更新、リスト表示、および削除には、次の AWS CLI コマンドまたは Amazon EKS API オペレーションを使用します。1 つのコマンドで新しいリソースを同時に作成しながらタグを追加するのに使用できるのは `eksctl` だけです。

Amazon EKS リソースのタグ付けのサポート

タスク	AWS CLI	AWS Tools for Windows PowerShell	API アクション
1 つ以上のタグを追加、または上書きします。	tag-resource	Add-EKSResourceTag	TagResource
1 つ以上のタグを削除します。	untag-resource	Remove-EKSResourceTag	UntagResource

以下の例では、AWS CLI を使用して、リソースに対してタグ付けまたはタグ削除する方法を示しています。

例 1: 既存のクラスターへのタグ付け

次のコマンドは既存のクラスターにタグ付けします。

```
aws eks tag-resource --resource-arn resource_ARN --tags team=devs
```

例 2: 既存のクラスターでのタグ削除

次のコマンドは既存のクラスターからタグを削除します。

```
aws eks untag-resource --resource-arn resource_ARN --tag-keys tag_key
```

例 3: リソースのタグのリスト取得

次のコマンドは、既存のリソースに関連付けられているタグのリストを取得します。

```
aws eks list-tags-for-resource --resource-arn resource_ARN
```

一部のリソース作成アクションでは、リソースの作成と同時にタグを指定できます。次のアクションでは、リソース作成時のタグの指定がサポートされます。

タスク	AWS CLI	AWS Tools for Windows PowerShell	API アクション	**eksctl**
クラスターの作成	create-cluster	New-EKSCluster	CreateCluster	create cluster
マネージド型ノードグループの作成*	create-nodegroup	New-EKSNodegroup	CreateNodegroup	create nodegroup
Fargate プロファイルの作成	create-fargate-profile	New-EKSFargateProfile	CreateFargateProfile.html	create fargateprofile

* マネージド型ノードグループの作成時に Amazon EC2 インスタンスにもタグ付けを行う場合は、起動テンプレートを使用してマネージド型ノードグループを作成します。詳細については、「Amazon EC2 インスタンスへのタグ付け」を参照してください。インスタンスがすでに存在する場合は、インスタンスに手動でタグ付けを行うことができます。詳細については、「Amazon EC2 ユーザーガイド」の「リソースのタグ付け」を参照してください。

Amazon EKS Service Quotas

Amazon EKS は、Service Quotas と統合されています。Service Quotas は、クォータを一元的な場所から表示および管理するために使用できる AWS サービスです。詳細については、「Service Quotas ユーザーガイド」の「Service Quotas とは」を参照してください。Service Quotas との統合により、AWS Management Console および AWS CLI を使用して、Amazon EKS および AWS Fargate Service Quotas の値を簡単に検索できます。

AWS Management Console

AWS Management Console を使用して Amazon EKS サービスクォータおよび Fargate サービスクォータを表示するには

1. Service Quotas のコンソールを開きます。https://console.aws.amazon.com/servicequotas/
2. 左側のナビゲーションペインで、[AWS のサービス] を選択します。
3. [AWS のサービス] リストから、[Amazon Elastic Kubernetes Service (Amazon EKS)] または [AWS Fargate] を選択します。

   [Service Quotas] の一覧には、Service Quotas 名、適用された値 (使用可能な場合)、AWS のデフォルトのクォータ、クォータ値が調整可能かどうかが表示されます。
4. 説明など、Service Quotas に関する追加情報を表示するには、クォータ名を選択します。
5. (オプション) クォータの引き上げをリクエストするには、[Request quota increase (クォータ引き上げリクエスト)] を選択、または必要な情報を入力または選択して、[Request (リクエスト)] を選択します。

AWS Management Console を使用してさらに Service Quotas の操作を行うには、「Service Quotas ユーザーガイド」を参照してください。クォータの引き上げをリクエストするには、Service Quotas ユーザーガイド の「クォータ引き上げリクエスト」を参照してください。

AWS CLI

AWS CLI を使用して Amazon EKS サービスクォータおよび Fargate サービスクォータを表示するには

次のコマンドを実行して、Amazon EKS クォータを表示します。

```
aws service-quotas list-aws-default-service-quotas \
    --query 'Quotas[*].
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
    --service-code eks \
    --output table
```

次のコマンドを実行して、Fargate クォータを表示します。

```
aws service-quotas list-aws-default-service-quotas \
    --query 'Quotas[*].
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
```

```
    --service-code fargate \
    --output table
```

> **Note**
>
> 返されるクォータは、現在の AWS リージョンのこのアカウントの Fargate で同時に実行可能な Amazon ECS タスクまたは Amazon EKS Pods の数です。

AWS CLI を使用して、さらにサービスクォータの操作を行うには、「AWS CLI コマンドリファレンス」で service-quotas を参照してください。クォータの引き上げをリクエストするには、「AWS CLI コマンド リファレンス」で request-service-quota-increase コマンドを参照してください。

サービスクォータ

名前	デフォルト	引き上げ可能	説明
クラスターあたりのアクセスエントリー	サポートされている各リージョン: 3,000	[いいえ]	クラスターあたりのデータアクセスエントリーの最大数。
クラスター	サポートされている各リージョン: 100	はい	現在のリージョンのこのアカウントの EKS クラスターの最大数。
クラスターあたりのコントロールプレーンセキュリティグループ	サポートされている各リージョン: 4	[いいえ]	クラスターあたりのコントロールプレーンセキュリティグループの最大数 (クラスターの作成時に指定されます)。

名前	デフォルト	引き上げ可能	説明
EKS Anywhere エンタープライズサブスクリプション	サポートされている各リージョン: 10	はい	現在のリージョンにおけるこのアカウントのイベントサブスクリプションの最大数。
クラスターあたりの Fargate プロファイル	サポートされている各リージョン: 10	はい	クラスターあたりの Fargate プロファイルの最大数。
Fargate プロファイルのセレクタあたりのラベルペア	サポートされている各リージョン : 5	はい	Fargate プロファイルのセレクタあたりのラベルペアの最大数。
クラスターあたりのマネージドノードグループ	サポートされている各リージョン: 30	はい	クラスターあたりのマネージド型ノードグループの最大数。
マネージド型ノードグループあたりのノード数	サポートされている各リージョン: 450	はい	マネージド型ノードグループあたりのノードの最大数。
クラスターあたりのパブリックエンドポイントアクセスの CIDR 範囲	サポートされている各リージョン: 40	[いいえ]	クラスターあたりのパブリックエンドポイントアクセスの CIDR 範囲の最大数 (クラスターの作成または更新時に指定されます)。
登録済みクラスター	サポートされている各リージョン: 10	はい	現在のリージョンでこのアカウント内に登録されたクラスターの最大数。

名前	デフォルト	引き上げ可能	説明
Fargate プロファイルあたりのセレクター	サポートされている各リージョン：5	はい	Fargate プロファイルあたりのセレクタの最大数。

> **Note**
>
> デフォルト値は、AWS によって設定された初期クォータです。これらのデフォルト値は、実際に適用されたクォータ値および適用可能な最大 Service Quotas とは異なります。詳細については、「Service Quotas ユーザーガイド」の「Service Quotas の用語」を参照してください。

これらの Service Quotas は、Service Quotas コンソールの [Amazon Elastic Kubernetes Service (Amazon EKS)] に表示されます。調整可能として表示される値のクォータの引き上げをリクエストすることもできます。「Service Quotas ユーザーガイド」の「クォータ引き上げのリクエスト」を参照してください。

AWS Fargate Service Quotas

Service Quotas コンソールの AWS Fargate サービスは、いくつかのサービスクォータを一覧表示します。次の表では、Amazon EKS に適用されるクォータについてのみ説明しています。使用量がサービスクォータに近づいたときに警告するアラームを設定することもできます。詳細については、「Fargate のリソース使用状況メトリクスをモニタリングするための CloudWatch アラームの作成」を参照してください。

新規の AWS アカウントの低い初期クォータは、経時的に引き上げられる可能性があります。Fargate は、各 AWS リージョン内のアカウント使用率を常に監視し、使用率に基づいてクォータを自動的に引き上げます。調整可能として表示される値のクォータの引き上げをリクエストすることもできます。詳細については、「Service Quotas ユーザーガイド」の「クォータ引き上げのリクエスト」を参照してください。

名前	デフォルト	引き上げ可能	説明
Fargate オンデマンド vCPU リソース数	6	はい	現在のリージョンのこのアカウントで Fargate On-Demand として同時に実行される Fargate vCPU の数。

> **Note**
>
> デフォルト値は、AWS によって設定された初期クォータです。これらのデフォルト値は、実際に適用されたクォータ値および適用可能な最大 Service Quotas とは異なります。詳細については、「Service Quotas ユーザーガイド」の「Service Quotas の用語」を参照してください。

> **Note**
>
> Fargate はさらに、Amazon ECS タスクと Amazon EKS Pods の起動レートクォータを適用します。詳細については、「Amazon ECS ガイド」の「AWS Fargate スロットリングのクォータ」を参照してください。

Amazon EKS のセキュリティ

AWS ではクラウドセキュリティが最優先事項です。セキュリティを最も重視する組織の要件を満たすために構築された AWS のデータセンターとネットワークアーキテクチャは、お客様に大きく貢献します。

セキュリティは、AWS と顧客の間の責任共有です。責任共有モデルでは、この責任がクラウドのセキュリティおよびクラウド内のセキュリティとして説明されています。

- クラウドのセキュリティ – AWS は、AWS クラウドで AWS のサービスを実行するインフラストラクチャを保護する責任を担います。Amazon EKS の場合、AWS はコントロールプレーンノードと `etcd` データベースを含む Kubernetes コントロールプレーンを担当します。AWS コンプライアンスプログラムの一環として、サードパーティーの監査が定期的にセキュリティの有効性をテストおよび検証しています。Amazon EKS に適用されるコンプライアンスプログラムについては、「コンプライアンスプログラムによる対象範囲内の AWS サービス」を参照してください。
- クラウド内のセキュリティ – お客様の責任事項には、次の領域が含まれます。
  - データプレーンのセキュリティ設定。これには、Amazon EKS コントロールプレーンからお客様の VPC 内へのトラフィックの通過を許可する、セキュリティグループの設定を含みます。
  - ノードおよびコンテナに対する設定。
  - ノードのオペレーティングシステム (更新やセキュリティパッチを含みます)
  - その他の関連アプリケーションソフトウェア :
    - ファイアウォールのルールなど、ネットワークコントロールの設定および管理。
    - IAM を使用する、またはそれに追加して行う、Identity and Access Managementのプラットフォームレベルの管理。
  - お客様のデータの機密性、企業の要件、該当する法律および規制

このドキュメントは、Amazon EKS を使用する際の責任共有モデルの適用について理解するのに役立ちます。以下のトピックで、セキュリティおよびコンプライアンスの目的を満たすように、Amazon EKS を設定する方法について説明します。Amazon EKS リソースのモニタリングやセキュリティ確保に役立つ他の AWS サービスの使用方法についても説明します。

> **Note**
>
> Linux コンテナはコントロールグループ (cgroup) と名前空間で設定されています。これらにより、コンテナのアクセスが可能な対象を制限しながら、すべてのコンテナに、ホストの

Amazon EC2 インスタンスと同じ Linux カーネルを共有させることができます。コンテナをルートユーザー (UID 0) として実行したり、ホストリソースや (ホストネットワークやホスト PID の) 名前空間へのアクセスをコンテナに許可したりすることは一切お勧めしません。これは、コンテナが提供する分離の有効性を低下させます。

トピック

証明書への署名

Kubernetes 認証情報 API により、X.509 認証情報のプロビジョニングを自動化します。この API は、Kubernetes API クライアントが「X.509 証明書」を認証機関 (CA) にリクエストし、取得するためのコマンドラインインターフェイスを備えています。`CertificateSigningRequest` (CSR) リソースを使用して、示された署名者に証明書への署名を要求できます。リクエストは署名される前に承認または拒否されます。Kubernetes は、明確に定義された動作を持つ組み込み署名者とカスタム署名者の両方をサポートします。この構成により、クライアントは自分の CSR に何が起こるかを予測できます。証明書の署名の詳細については、「signing requests」(リクエストへの署名) を参照してください。

組み込みの署名者の中には、`kubernetes.io/legacy-unknown` も含まれます。CSR リソースの `v1beta1` API では、この未知のレガシー署名者を拒否していません。ただし、CSR の安定版 `v1` APIでは、`signerName` を `kubernetes.io/legacy-unknown` に設定することはできません。

Amazon EKS バージョン 1.21 以前のバージョンでは、v1beta1 CSR API の signerName として legacy-unknown の値を使用することができました。この API により、Amazon EKS 認証局 (CA) は証明書を生成できます。ただし Kubernetes バージョン 1.22 では、v1beta1 CSR API は v1 CSR API に置き換えられています。この API では、signerName を「legacy-unknown」とすることはできません。クラスターでの証明書の生成に Amazon EKS CA を使用する場合は、カスタム署名者を使用する必要があります。この署名者は、Amazon EKS バージョン 1.22 で導入されたものです。CSR v1 API バージョンを使用して新しい証明書を生成するには、既存のマニフェストと API クライアントを移行する必要があります。古い v1beta1 API で作成された既存の証明書の有効性は保たれ、その有効期限が切れるまで機能します。これには以下が含まれます。

- 信頼のディストリビューション: なし Kubernetes クラスターには、この署名者のための標準的な信頼またはディストリビューションはありません。
- 許可された対象: 任意
- 許可された x509 拡張機能: subjectAltName とキーの使用の拡張機能を受け入れ、他の拡張機能を破棄します
- 許可された鍵の使用法: ["key encipherment", "digital signature", "server auth"] (鍵暗号化、デジタル署名、サーバー認証) 以外の使用法を含めないでください。

> **Note**
>
> クライアント証明書署名はサポートされていません。

- 有効期限/証明書のライフタイム: 1 年 (デフォルトかつ最大の値)
- CA ビットの許可/不許可: 不許可

signerName を使用した CSR 生成の例

次の手順では、signerName: beta.eks.amazonaws.com/app-serving を使用した DNS 名 myserver.default.svc の配信用証明書の生成方法を示します。これは、実際の環境を構築するためのガイドとして使用できます。

1. openssl genrsa -out myserver.key 2048 コマンドを実行して、RSA プライベートキーを生成します。

```
openssl genrsa -out myserver.key 2048
```

2. 次のコマンドを実行して、証明書リクエストを生成します。

```
openssl req -new -key myserver.key -out myserver.csr -subj "/
CN=myserver.default.svc"
```

3. CSR リクエストの base64 値を生成し、後の手順で使用するために変数に格納します。

```
base_64=$(cat myserver.csr | base64 -w 0 | tr -d "\n")
```

4. 次のコマンドを実行して、mycsr.yaml という名前のファイルを作成します。次の例では、beta.eks.amazonaws.com/app-serving が signerName に使用されています。

```
cat >mycsr.yaml <<EOF
apiVersion: certificates.k8s.io/v1
kind: CertificateSigningRequest
metadata:
  name: myserver
spec:
  request: $base_64
  signerName: beta.eks.amazonaws.com/app-serving
  usages:
    - digital signature
    - key encipherment
    - server auth
EOF
```

5. CSR を送信します。

```
kubectl apply -f mycsr.yaml
```

6. 配信用証明書を承認します。

```
kubectl certificate approve myserver
```

7. 証明書が発行されたことを確認します。

```
kubectl get csr myserver
```

出力例は次のとおりです。

```
NAME       AGE     SIGNERNAME                            REQUESTOR
 CONDITION
```

```
myserver   3m20s   beta.eks.amazonaws.com/app-serving   kubernetes-admin
 Approved,Issued
```

8. 発行された証明書をエクスポートします。

```
kubectl get csr myserver -o jsonpath='{.status.certificate}'| base64 -d
 > myserver.crt
```

クラスターをKubernetes 1.24 にアップグレードする前の証明書署名に関する考慮事項

Kubernetes 1.23 以前のバージョンでは、検証不能な IP と DNS サブジェクト代替名 (SAN) を含む kubelet サービング証明書は、検証不能な SAN とともに自動的に発行されます。プロビジョニングされた証明書からは、SAN は除外されます。1.24 以降のクラスターでは、SAN を検証できない場合、kubelet サービング証明書は発行されません。これにより、kubectl exec および kubectl logs コマンドが機能しなくなります。

クラスターを 1.24 にアップグレードする前に、以下の手順を実行して、クラスターにまだ承認されていない証明書署名要求 (CSR) があるかどうかを確認します。

1. 以下のコマンドを実行します。

```
kubectl get csr -A
```

出力例は次のとおりです。

```
NAME        AGE   SIGNERNAME                        REQUESTOR
                        REQUESTEDDURATION   CONDITION
csr-7znmf   90m   kubernetes.io/kubelet-serving
 system:node:ip-192-168-42-149.region.compute.internal      <none>
 Approved
csr-9xx5q   90m   kubernetes.io/kubelet-serving
 system:node:ip-192-168-65-38.region.compute.internal      <none>
 Approved, Issued
```

返された出力に、ノードに対して Approved ではあるが Issued ではない「kubernetes.io/kubelet-serving」署名者付きの CSR が表示されている場合は、要求を承認する必要があります。

2. CSR を手動で承認します。`csr-7znmf` を独自の値に置き換えます。

```
kubectl certificate approve csr-7znmf
```

将来、CSR を自動承認するには、Amazon EKS で検証できない IP または DNS SAN を含む CSR を自動的に検証および承認できる承認コントローラーを作成することをお勧めします。

Amazon EKS の Identity and Access Management

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御するために役立つ AWS のサービスです。IAM 管理者は、誰の (サインイン) を認証し、誰による Amazon EKS リソースの使用を承認する (アクセス許可を付与する) かを制御します。IAM は、追加費用なしで使用できる AWS のサービスです。

対象者

AWS Identity and Access Management (IAM) の用途は、Amazon EKS で行う作業によって異なります。

サービスユーザー – ジョブを実行するために Amazon EKS サービスを使用する場合は、管理者から必要なアクセス許可と認証情報が与えられます。さらに多くの Amazon EKS 機能を使用して作業を行う場合は、追加のアクセス許可が必要になることがあります。アクセスの管理方法を理解すると、管理者から適切なアクセス許可をリクエストするのに役に立ちます。Amazon EKS の機能にアクセスできない場合は、「IAM のトラブルシューティング」を参照してください。

サービス管理者 – 社内の Amazon EKS リソースを管理しているユーザーには、通常、Amazon EKS への完全なアクセス権限があります。サービスのユーザーがどの Amazon EKS 機能やリソースにアクセスするかを決めるのは管理者の仕事です。その後、IAM 管理者にリクエストを送信して、サービスユーザーの権限を変更する必要があります。このページの情報を確認して、IAM の基本概念を理解してください。企業が Amazon EKS で IAM を利用する方法については、「Amazon EKS で IAM を使用する方法」を参照してください。

IAM 管理者 – IAM 管理者には、Amazon EKS へのアクセスを管理するポリシーの作成方法を、詳細に把握する必要が生じる場合があります。IAM で使用可能な、Amazon EKS アイデンティティベースのポリシーの例を確認するには、「Amazon EKS でのアイデンティティベースのポリシーの例」を参照してください。

アイデンティティを使用した認証

認証とは、アイデンティティ認証情報を使用して AWSにサインインする方法です。ユーザーは、AWS アカウントのルートユーザー として、IAM ユーザーとして、または IAM ロールを引き受けることによって、認証済み (AWS にサインイン済み) である必要があります。

ID ソースから提供された認証情報を使用すると、フェデレーティッドアイデンティティとして AWS にサインインできます。AWS IAM Identity Centerフェデレーティッドアイデンティティの例としては、(IAM アイデンティティセンター) ユーザー、貴社のシングルサインオン認証、Google または Facebook の認証情報などがあります。フェデレーティッドアイデンティティとしてサインインする場合、IAM ロールを使用して、前もって管理者により ID フェデレーションが設定されています。フェデレーションを使用して AWSにアクセスする場合、間接的にロールを引き受けることになります。

ユーザーのタイプに応じて、AWS Management Console または AWSアクセスポータルにサインインできます。AWS へのサインインの詳細については、『AWS サインイン ユーザーガイド』の「AWS アカウント にサインインする方法」を参照してください。

プログラムで AWS にアクセスする場合、AWS は Software Development Kit (SDK) とコマンドラインインターフェイス (CLI) を提供し、認証情報でリクエストに暗号で署名します。AWS ツールを使用しない場合は、リクエストに自分で署名する必要があります。リクエストに署名する推奨方法の使用については、『IAM ユーザーガイド』の「AWS API リクエストの署名」を参照してください。

使用する認証方法を問わず、追加セキュリティ情報の提供をリクエストされる場合もあります。例えば、AWS では、アカウントのセキュリティ強化のために多要素認証 (MFA) の使用をお勧めしています。詳細については、『AWS IAM Identity Center ユーザーガイド』の「Multi-factor authentication」(多要素認証) および『IAM ユーザーガイド』の「AWS における多要素認証 (MFA) の使用」を参照してください。

AWS アカウントのルートユーザー

AWS アカウントを作成する場合は、このアカウントのすべての AWS のサービスとリソースに対して完全なアクセス権を持つ 1 つのサインインアイデンティティから始めます。この ID は AWS アカウント ルートユーザーと呼ばれ、アカウントの作成に使用したメールアドレスとパスワードでサインインすることによってアクセスできます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザーの認証情報は保護し、ルートユーザーでしか実行できないタスクを実行するときに使用します。ルートユーザーとしてサインインする必要があるタスクの完全なリストについては、「IAM ユーザーガイド」の「ルートユーザー認証情報が必要なタスク」を参照してください。

IAM ユーザーとグループ

IAM ユーザーは、1 人のユーザーまたは 1 つのアプリケーションに対して特定の権限を持つ AWS アカウント 内のアイデンティティです。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を保有する IAM ユーザーを作成する代わりに、一時認証情報を使用することをお勧めします。ただし、IAM ユーザーでの長期的な認証情報が必要な特定のユースケースがある場合は、アクセスキーをローテーションすることをお勧めします。詳細については、『IAM ユーザーガイド』の「長期的な認証情報を必要とするユースケースのためにアクセスキーを定期的にローテーションする」を参照してください。

IAM グループは、IAM ユーザーの集団を指定するアイデンティティです。グループとしてサインインすることはできません。グループを使用して、複数のユーザーに対して一度に権限を指定できます。多数のユーザーグループがある場合、グループを使用することで権限の管理が容易になります。例えば、IAMAdmins という名前のグループを設定して、そのグループに IAM リソースを管理する権限を与えることができます。

ユーザーは、ロールとは異なります。ユーザーは 1 人の人または 1 つのアプリケーションに一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユーザーには永続的な長期の認証情報がありますが、ロールでは一時的な認証情報が提供されます。詳細については、『IAM ユーザーガイド』の「IAM ユーザー (ロールではなく) の作成が適している場合」を参照してください。

IAM ロール

IAM ロールは、特定の権限を持つ、AWS アカウント 内のアイデンティティです。これは IAM ユーザーに似ていますが、特定のユーザーには関連付けられていません。ロールの切り替えによって、AWS Management Console で IAM ロールを一時的に引き受けることができます。ロールを引き受けるには、AWS CLI または AWSAPI オペレーションを呼び出すか、カスタム URL を使用します。ロールを使用する方法の詳細については、『IAM ユーザーガイド』の「IAM ロールの使用」を参照してください。

IAM ロールと一時的な認証情報は、次の状況で役立ちます:

- フェデレーションユーザーアクセス – フェデレーティッドアイデンティティに権限を割り当てるには、ロールを作成してそのロールの権限を定義します。フェデレーティッドアイデンティティが認証されると、そのアイデンティティはロールに関連付けられ、ロールで定義されている権限が付与されます。フェデレーションの詳細については、『IAM ユーザーガイド』の「サードパーティーアイデンティティプロバイダー向けロールの作成」 を参照してください。IAM アイデンティティセンターを使用する場合、権限セットを設定します。アイデンティティが認証後にアク

セスできるものを制御するため、IAM Identity Center は、権限セットを IAM のロールに関連付けます。権限セットの詳細については、『AWS IAM Identity Center ユーザーガイド』の「権限セット」を参照してください。

- 一時的な IAM ユーザー権限 - IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる権限を一時的に IAM ロールで引き受けることができます。
- クロスアカウントアクセス - IAM ロールを使用して、自分のアカウントのリソースにアクセスすることを、別のアカウントの人物 (信頼済みプリンシパル) に許可できます。クロスアカウントアクセス権を付与する主な方法は、ロールを使用することです。ただし、一部の AWS のサービスでは、(ロールをプロキシとして使用する代わりに) リソースにポリシーを直接アタッチできます。クロスアカウントアクセスにおけるロールとリソースベースのポリシーの違いについては、『IAM ユーザーガイド』の「IAM ロールとリソースベースのポリシーとの相違点」を参照してください。
- クロスサービスアクセス権 - 一部の AWS のサービスでは、他の AWS のサービスの機能を使用します。例えば、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。サービスでは、呼び出し元プリンシパルの権限、サービスロール、またはサービスにリンクされたロールを使用してこれを行う場合があります。
  - 転送アクセスセッション (FAS) – IAM ユーザーまたはロールを使用して AWS でアクションを実行するユーザーは、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、AWS のサービスを呼び出すプリンシパルの権限を、AWS のサービスのリクエストと合わせて使用し、ダウンストリームのサービスに対してリクエストを行います。FAS リクエストは、サービスが、完了するために他の AWS のサービス またはリソースとのやりとりを必要とするリクエストを受け取ったときにのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。
  - サービスロール - サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、『IAM ユーザーガイド』の「AWS のサービス に権限を委任するロールの作成」を参照してください。
  - サービスリンクロール - サービスリンクロールは、AWS のサービス にリンクされたサービスロールの一種です。サービスがロールを引き受け、ユーザーに代わってアクションを実行できるようになります。サービスリンクロールは、AWS アカウント に表示され、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールの権限を表示できますが、編集することはできません。

- Amazon EC2 で実行されるアプリケーション - EC2 インスタンスで実行され、AWS CLI または AWS API 要求を行っているアプリケーションの一時的な認証情報を管理するために、IAM ロールを使用できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。AWS ロールを EC2 インスタンスに割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスに添付されたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、『IAM ユーザーガイド』の「Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用して権限を付与する」を参照してください。

IAM ロールと IAM ユーザーのどちらを使用するかについては、『IAM ユーザーガイド』の「(IAM ユーザーではなく) IAM ロールをいつ作成したら良いのか?」を参照してください。

ポリシーを使用したアクセス権の管理

AWS でアクセス権を管理するには、ポリシーを作成して AWS アイデンティティまたはリソースにアタッチします。ポリシーは AWS のオブジェクトであり、アイデンティティやリソースに関連付けて、これらの権限を定義します。AWS は、プリンシパル (ユーザー、ルートユーザー、またはロールセッション) がリクエストを行うと、これらのポリシーを評価します。ポリシーでの権限により、リクエストが許可されるか拒否されるかが決まります。大半のポリシーは JSON ドキュメントとして AWSに保存されます。JSON ポリシードキュメントの構造と内容の詳細については、『IAM ユーザーガイド』の「JSON ポリシー概要」を参照してください。

管理者は AWSJSON ポリシーを使用して、だれが何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

デフォルトでは、ユーザーやロールに権限はありません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き継ぐことができます。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションの権限を定義します。例えば、`iam:GetRole` アクションを許可するポリシーがあるとします。このポリシーがあるユーザーは、AWS Management Console、AWS CLI、または AWS API からロール情報を取得できます。

アイデンティティベースポリシー

アイデンティティベースポリシーは、IAM ユーザー、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 権限ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。アイデンティティ

ベースのポリシーを作成する方法については、『IAM ユーザーガイド』の「IAM ポリシーの作成」を参照してください。

アイデンティティベースポリシーは、さらにインラインポリシーまたはマネージドポリシーに分類できます。インラインポリシーは、単一のユーザー、グループ、またはロールに直接埋め込まれます。管理ポリシーは、AWS アカウント 内の複数のユーザー、グループ、およびロールにアタッチできるスタンドアロンポリシーです。マネージドポリシーには、AWS マネージドポリシーおよびカスタマーマネージドポリシーがあります。マネージドポリシーまたはインラインポリシーのいずれかを選択する方法については、『IAM ユーザーガイド』の「マネージドポリシーとインラインポリシーの比較」を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーションユーザー、または AWS のサービス を含めることができます。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーでは IAM の AWSマネージドポリシーは使用できません。

アクセスコントロールリスト (ACL)

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための権限を持つかをコントロールします。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Amazon S3、AWS WAF、および Amazon VPC は、ACL をサポートするサービスの例です。ACL の詳細については、『Amazon Simple Storage Service デベロッパーガイド』の「アクセスコントロールリスト (ACL) の概要」を参照してください。

その他のポリシータイプ

AWS では、他の一般的ではないポリシータイプをサポートしています。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の権限を設定できます。

- アクセス許可の境界 - アクセス許可の境界は、アイデンティティベースのポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる権限の上限を設定する高度な機能です。エンティティにアクセス許可の境界を設定できます。結果として得られる権限は、エンティティのアイデンティティベースポリシーとそのアクセス許可の境界の共通部分になります。`Principal` フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、アクセス許可の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。アクセス許可の境界界の詳細については、『IAM ユーザーガイド』の「IAM エンティティのアクセス許可の境界」を参照してください。
- サービスコントロールポリシー (SCP) - SCP は、AWS Organizations で組織や組織単位 (OU) の最大権限を指定する JSON ポリシーです。AWS Organizations は、顧客のビジネスが所有する複数の AWS アカウント をグループ化し、一元的に管理するサービスです。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP はメンバーアカウントのエンティティに対する権限を制限します (各 AWS アカウントのルートユーザー など)。Organizations と SCP の詳細については、『AWS Organizations ユーザーガイド』の「SCP の仕組み」を参照してください。
- セッションポリシー - セッションポリシーは、ロールまたはフェデレーションユーザーの一時的なセッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。結果としてセッションの権限は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポリシーの共通部分になります。また、リソースベースのポリシーから権限が派生する場合もあります。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。詳細については、『IAM ユーザーガイド』の「セッションポリシー」を参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成される権限を理解するのがさらに難しくなります。複数のポリシータイプが関連するとき、リクエストを許可するかどうかを AWS が決定する方法の詳細については、IAM ユーザーガイドのポリシーの評価ロジックを参照してください。

Amazon EKS で IAM を使用する方法

IAM を使用して Amazon EKS へのアクセスを管理する前に、Amazon EKS で使用できる IAM 機能について理解しておく必要があります。Amazon EKS およびその他の AWS のサービスが IAM と連携する方法の概要を把握するには、IAM ユーザーガイド の「IAM と連携する AWS のサービス」を参照してください。

トピック

- Amazon EKS のアイデンティティベースのポリシー
- Amazon EKS でのリソースベースのポリシー
- Amazon EKS タグに基づいた認証
- Amazon EKS でのIAM ロール

Amazon EKS のアイデンティティベースのポリシー

IAM アイデンティティベースのポリシーでは、許可または拒否するアクションとリソース、またアクションを許可または拒否する条件を指定できます。Amazon EKS は、特定のアクション、リソース、および条件キーをサポートしています。JSON ポリシーで使用するすべての要素については、「IAM ユーザーガイド」の「IAM JSON ポリシーエレメントのリファレンス」を参照してください。

アクション

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

JSON ポリシーの`Action`要素には、ポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。ポリシーアクションの名前は通常、関連する AWSAPI オペレーションと同じです。一致する API オペレーションのない権限のみのアクションなど、いくつかの例外があります。また、ポリシーに複数アクションが必要なオペレーションもあります。これらの追加アクションは、 依存アクション と呼ばれます。

このアクションは、関連付けられたオペレーションを実行するためのアクセス許可を付与するポリシーで使用されます。

Amazon EKS のポリシーアクションは、アクションの前にプレフィックス `eks:` を付加します。例えば、Amazon EKS クラスターについて説明する情報を入手するアクセス許可を他のユーザーに付与するには、ポリシーに `DescribeCluster` アクションを含めます。ポリシーステートメントには、`Action` または `NotAction` 要素を含める必要があります。

単一のステートメントに複数のアクションを指定するには、次のようにコンマで区切ります。

```
"Action": ["eks:action1", "eks:action2"]
```

ワイルドカード (*) を使用して複数アクションを指定できます。例えば、`Describe` という単語で始まるすべてのアクションを指定するには、次のアクションを含めます。

```
"Action": "eks:Describe*"
```

Amazon EKS アクションの一覧については、「サービス認証リファレンス」の「Amazon Elastic Kubernetes Service で定義されるアクション」を参照してください。

リソース

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

Resource JSON ポリシー要素は、アクションが適用されるオブジェクトを指定します。ステートメントには、Resource または NotResource要素を含める必要があります。ベストプラクティスとして、Amazon リソースネーム (ARN) を使用してリソースを指定します。これは、リソースレベルの権限と呼ばれる特定のリソースタイプをサポートするアクションに対して実行できます。

オペレーションのリスト化など、リソースレベルの権限をサポートしないアクションの場合は、ステートメントがすべてのリソースに適用されることを示すために、ワイルドカード (*) を使用します。

```
"Resource": "*"
```

Amazon EKS クラスターリソースの ARN は次のようになります。

```
arn:aws:eks:region-code:account-id:cluster/cluster-name
```

ARN の形式の詳細については、「Amazon リソースネーム (ARN) と AWS のサービスの名前空間」を参照してください。

例えば、ステートメントで *my-cluster* という名前のクラスターを指定するには、次の ARN を使用します。

```
"Resource": "arn:aws:eks:region-code:111122223333:cluster/my-cluster"
```

特定のアカウントと AWS リージョン に属するすべてのクラスターを指定するには、ワイルドカード (*) を使用します。

```
"Resource": "arn:aws:eks:region-code:111122223333:cluster/*"
```

リソースの作成を含む、一部の Amazon EKS アクションは、特定のリソースで実行できません。このような場合は、ワイルドカード *を使用する必要があります。

```
"Resource": "*"
```

Amazon EKS リソースタイプとその ARN の一覧については、「サービス認証リファレンス」の「Amazon Elastic Kubernetes Service で定義されるリソースタイプ」を参照してください。各リソースの ARN を、どのアクションで指定できるかについては、「Amazon Elastic Kubernetes Service で定義されるアクション」を参照してください。

条件キー

Amazon EKS では独自の条件キーが定義されており、また一部のグローバル条件キーの使用がサポートされています。すべての AWS グローバル条件キーを確認するには、IAM ユーザーガイドの「AWS グローバル条件コンテキストキー」を参照してください。

OpenID Connect プロバイダーをクラスターに関連付ける場合に条件キーを設定できます。詳細については、「IAM ポリシーの例」を参照してください。

すべての Amazon EC2 アクションは、`aws:RequestedRegion` および `ec2:Region` 条件キーをサポートします。詳細については、「例: 特定の AWS リージョン へのアクセスの制限」を参照してください。

Amazon EKS 条件キーのリストについては、「サービス認証リファレンス」の「Amazon Elastic Kubernetes Service によって定義された条件」を参照してください。条件キーを使用できるアクションとリソースについては、「Amazon Elastic Kubernetes Service で定義されるアクション」を参照してください。

例

Amazon EKS でのアイデンティティベースのポリシーの例は、「Amazon EKS でのアイデンティティベースのポリシーの例」でご確認ください。

Amazon EKS クラスターを作成すると、このクラスターを作成する IAM プリンシパルには、Amazon EKS コントロールプレーンのクラスターロールベースアクセスコントロール (RBAC) 設定で、`system:masters` 許可が自動的に付与されます。このプリンシパルは表示可能な設定の中には表示されません。したがって、どのプリンシパルが最初にクラスターを作成したのかを追跡する必要があります。追加の IAM プリンシパルがクラスターとインタラクションできるようにするには、Kubernetes 内の `aws-auth ConfigMap` を編集し、`aws-auth ConfigMap` で指定した `group` の名前を使用して、Kubernetes `rolebinding` または `clusterrolebinding` を作成します。

ConfigMap の使用に関する詳細については、「Kubernetes API へのアクセスを許可する」を参照してください。

Amazon EKS でのリソースベースのポリシー

Amazon EKS では、 リソースベースのポリシーはサポートされていません。

Amazon EKS タグに基づいた認証

タグは、Amazon EKS リソースにアタッチすることも、Amazon EKS へのリクエストの際に渡すこともできます。タグに基づいてアクセスを制御するには、`aws:ResourceTag/key-name`、`aws:RequestTag/key-name`、または `aws:TagKeys` の条件キーを使用して、ポリシーの条件要素でタグ情報を提供します。Amazon EKS リソースのタグ付けの詳細については、「Amazon EKS リソースのタグ付け」を参照してください。条件キーでタグを使用できるアクションの詳細については、「Service Authorization Reference」(サービス認可のリファレンス) の「Amazon EKS で定義されたアクション」を参照してください。

Amazon EKS でのIAM ロール

IAM ロール は、特定のアクセス許可を持つ、AWS アカウント内のエンティティです。

Amazon EKS での一時的な認証情報の使用

一時的な認証情報を使用して、フェデレーションでサインイン、IAM ロールを引き受ける、またはクロスアカウントロールを引き受けることができます。一時的なセキュリティ認証情報を取得するには、AssumeRole または GetFederationToken などの AWS STS API オペレーションを呼び出します。

Amazon EKS では、一時認証情報が使用できます。

サービスにリンクされたロール

サービスリンクロールは、AWS サービスが他のサービスのリソースにアクセスしてお客様の代わりにアクションを完了することを許可します。サービスリンクロールは IAM アカウント内に表示され、サービスによって所有されます。管理者は、サービスにリンクされたロールの許可を表示できますが、編集することはできません。

Amazon EKS は、サービスにリンクされたロールをサポートしています。Amazon EKS でのサービスにリンクされたロールの作成または管理の詳細については、「Amazon EKS でのサービスにリンクされたロールの使用」を参照してください。

サービスロール

この機能により、ユーザーに代わってサービスがサービスロールを引き受けることが許可されます。このロールにより、サービスがお客様に代わって他のサービスのリソースにアクセスし、アクションを完了することが許可されます。サービスロールは、IAM アカウントに表示され、アカウントによって所有されます。つまり、IAM 管理者は、このロールの権限を変更できます。ただし、これを行うことにより、サービスの機能が損なわれる場合があります。

Amazon EKS ではサービスロールがサポートされています。詳細については、「Amazon EKS クラスター の IAM ロール」および「Amazon EKS ノードの IAM ロール」を参照してください。

Amazon EKS での IAM ロールの選択

Amazon EKS でクラスターリソースを作成する場合は、他のいくつかの AWS リソースに Amazon EKS が自動的にアクセスすることを許可するためのロールを、ユーザーが選択する必要があります。以前に作成したサービスロールがある場合、Amazon EKS により、選択できるロールのリストが提示されます。Amazon EKS 管理のポリシーがアタッチされたロールを選択することが重要です。詳細については、既存のクラスターロールの確認および既存のノードロールの確認を参照してください。

Amazon EKS でのアイデンティティベースのポリシーの例

デフォルトでは、IAM ユーザーおよびロールにはAmazon EKS リソースを作成または変更するアクセス許可はありません。また、AWS Management Console や AWS CLI、AWS API を使用してタスクを実行することもできません。IAM 管理者は、ユーザーとロールに必要な、指定されたリソースで特定の API オペレーションを実行する権限をユーザーとロールに付与する IAM ポリシーを作成する必要があります。続いて、管理者はそれらの権限が必要な IAM ユーザーまたはグループにそのポリシーをアタッチする必要があります。

これらの JSON ポリシードキュメント例を使用して IAM のアイデンティティベースのポリシーを作成する方法については、『IAM ユーザーガイド』の「JSON タブでのポリシーの作成」を参照してください。

Amazon EKS クラスターを作成すると、このクラスターを作成する IAM プリンシパルには、Amazon EKS コントロールプレーンのクラスターロールベースアクセスコントロール (RBAC) 設定で、`system:masters` 許可が自動的に付与されます。このプリンシパルは表示可能な設定の中には表示されません。したがって、どのプリンシパルが最初にクラスターを作成したのかを追跡する必要があります。追加の IAM プリンシパルがクラスターとインタラクションできるようにするには、Kubernetes 内の `aws-auth ConfigMap` を編集し、`aws-auth ConfigMap` で指定した `group` の名前を使用して、Kubernetes `rolebinding` または `clusterrolebinding` を作成します。

ConfigMap の使用に関する詳細については、「Kubernetes API へのアクセスを許可する」を参照してください。

トピック

- ポリシーのベストプラクティス
- Amazon EKS コンソールの使用
- 自分のアクセス許可の表示を IAM ユーザーに許可する
- AWS クラウド に Kubernetes クラスターを作成します
- Outpost にローカル Kubernetes クラスターを作成します
- Kubernetes クラスターを更新する
- すべてのクラスターの一覧表示または説明

ポリシーのベストプラクティス

ID ベースのポリシーは、ユーザーのアカウント内で誰かが Amazon EKS リソースを作成、アクセス、または削除できるどうかを決定します。これらのアクションを実行すると、AWS アカウント に料金が発生する可能性があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

- AWS マネージドポリシーを使用して開始し、最小特権の権限に移行する – ユーザーとワークロードへの権限の付与を開始するには、多くの一般的なユースケースのために権限を付与する AWS マネージドポリシーを使用します。これらは AWS アカウントで使用できます。ユースケースに応じた AWS カスタマーマネージドポリシーを定義することで、権限をさらに減らすことをお勧めします。詳細については、『IAM ユーザーガイド』の「AWS マネージドポリシー」または「AWS ジョブ機能の管理ポリシー」を参照してください。
- 最小特権を適用する – IAM ポリシーで権限を設定するときは、タスクの実行に必要な権限のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権権限とも呼ばれています。IAM を使用して権限を適用する方法の詳細については、『IAM ユーザーガイド』の「IAM でのポリシーと権限」を参照してください。
- IAM ポリシーで条件を使用してアクセスをさらに制限する - ポリシーに条件を追加して、アクションやリソースへのアクセスを制限できます。例えば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。また、AWS CloudFormation などの特定の AWS のサービスを介して使用する場合、条件を使ってサービスアクションへのアクセス権を付与することもできます。詳細については、『IAM ユーザーガイド』の [IAM JSON policy elements: Condition] (IAM JSON ポリシー要素：条件) を参照してください。

- IAM Access Analyzer を使用して IAM ポリシーを検証し、安全で機能的な権限を確保する - IAM Access Analyzer は、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語 (JSON) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、『IAM ユーザーガイド』の「IAM Access Analyzer ポリシーの検証」を参照してください。
- 多要素認証 (MFA) を要求する – AWS アカウント で IAM ユーザーまたはルートユーザーを要求するシナリオがある場合は、セキュリティを強化するために MFA をオンにします。API オペレーションが呼び出されるときに MFA を必須にするには、ポリシーに MFA 条件を追加します。詳細については、『IAM ユーザーガイド』の「MFA 保護 API アクセスの設定」を参照してください。

IAM でのベストプラクティスの詳細については、『IAM ユーザーガイド』の「IAM でのセキュリティのベストプラクティス」を参照してください。

Amazon EKS コンソールの使用

Amazon EKS コンソールにアクセスするには、IAM プリンシパルに最小限の許可のセットが必要です。これらの許可により、プリンシパルは AWS アカウントの Amazon EKS リソースの詳細をリストおよび表示できます。最小限必要な許可よりも制限されたアイデンティティベースのポリシーを作成すると、そのポリシーをアタッチしたプリンシパルに対してはコンソールが意図したとおりに機能しません。

これらの IAM プリンシパルがまだ Amazon EKS コンソールを使用できるようにするには、`AmazonEKSAdminPolicy` など、独自の名前を付けてポリシーを作成します。ポリシーをプリンシパルにアタッチします。詳細については、「 IAM ユーザーガイド」の「IAM ID アクセス許可の追加および削除」を参照してください。

> **Important**
>
> 次のポリシー例では、プリンシパルはコンソールの [設定] タブで情報を表示できます。AWS Management Console で [概要] タブと [リソース] タブの情報を表示するには、プリンシパルに Kubernetes の許可も必要です。詳細については、「必要なアクセス許可」を参照してください。

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "eks.amazonaws.com"
                }
            }
        }
    ]
}
```

AWS CLI または AWS API のみを呼び出すプリンシパルには、最小限のコンソール許可を付与する必要はありません。代わりに、実行しようとしている API オペレーションに一致するアクションのみへのアクセスが許可されます。

自分のアクセス許可の表示を IAM ユーザーに許可する

この例では、ユーザーアイデンティティにアタッチされたインラインおよびマネージドポリシーの表示を IAM ユーザーに許可するポリシーの作成方法を示します。このポリシーには、コンソールで、または AWS CLI か AWS API を使用してプログラム的に、このアクションを完了する権限が含まれています。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
```

```
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS クラウド に Kubernetes クラスターを作成します

このポリシー例には、*us-west-2* AWS リージョン に *my-cluster* という名前の Amazon EKS クラスターを作成するために必要な最小限の許可が含まれています。AWS リージョン を、クラスターを作成する AWS リージョン に置き換えることができます。AWS Management Console で [ポリシーのアクションはリソースレベルの許可をサポートしていないため、**All resources** を選択する必要があります] という警告が表示された場合、無視しても問題ありません。アカウントに *AWSServiceRoleForAmazonEKS* ロールが既にある場合は、ポリシーから `iam:CreateServiceLinkedRole` アクションを削除できます。アカウントで Amazon EKS クラスターを作成したことがある場合、削除していない限り、このロールは既に存在します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "eks:CreateCluster",
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-cluster"
        },
```

```
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::111122223333:role/aws-service-role/
eks.amazonaws.com/AWSServiceRoleForAmazonEKS",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iam:AWSServiceName": "eks"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::111122223333:role/cluster-role-name"
        }
    ]
}
```

Outpost にローカル Kubernetes クラスターを作成します

このポリシー例には、*us-west-2* AWS リージョン に Outpost で *my-cluster* という名前の Amazon EKS ローカルクラスターを作成するために必要な最小限の許可が含まれています。AWS リージョン を、クラスターを作成する AWS リージョン に置き換えることができます。AWS Management Console で [ポリシーのアクションはリソースレベルの許可をサポートしていないため、**All resources** を選択する必要があります] という警告が表示された場合、無視しても問題ありません。アカウントに AWSServiceRoleForAmazonEKSLocalOutpost ロールがすでにある場合、ポリシーから iam:CreateServiceLinkedRole アクションを削除できます。アカウントで Outpost 上に Amazon EKS ローカルクラスターを作成したことがある場合、削除していない限り、このロールはすでに存在します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "eks:CreateCluster",
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-cluster"
        },
        {
            "Action": [
                "ec2:DescribeSubnets",
```

```
                "ec2:DescribeVpcs",
                "iam:GetRole"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::111122223333:role/aws-service-role/outposts.eks-
local.amazonaws.com/AWSServiceRoleForAmazonEKSLocalOutpost"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole",
                "iam:ListAttachedRolePolicies"
            ]
            "Resource": "arn:aws:iam::111122223333:role/cluster-role-name"
        },
        {
            "Action": [
                "iam:CreateInstanceProfile",
                "iam:TagInstanceProfile",
                "iam:AddRoleToInstanceProfile",
                "iam:GetInstanceProfile",
                "iam:DeleteInstanceProfile",
                "iam:RemoveRoleFromInstanceProfile"
            ],
            "Resource": "arn:aws:iam::*:instance-profile/eks-local-*",
            "Effect": "Allow"
        },
    ]
}
```

Kubernetes クラスターを更新する

このポリシー例には、us-west-2 AWS リージョン に *my-cluster* という名前のクラスターを更新するために必要な最小限の許可が含まれています。

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": "eks:UpdateClusterVersion",
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-cluster"
        }
    ]
}
```

すべてのクラスターの一覧表示または説明

このポリシーの例には、アカウント内のすべてのクラスターを一覧表示して記述するために必要な最小限の許可が含まれています。update-kubeconfig AWS CLI コマンドを使用するには、IAM プリンシパルがクラスターを一覧表示および記述できる必要があります。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster",
                "eks:ListClusters"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon EKS でのサービスにリンクされたロールの使用

Amazon Elastic Kubernetes Service は、AWS Identity and Access Management (IAM) のサービスにリンクされたロールを使用します。サービスにリンクされたロールは、Amazon EKS に直接リンクされた一意のタイプの IAM ロールです。サービスにリンクされたロールは、Amazon EKS で事前定義されています。このロールには、サービスがユーザーに代わって他の AWS のサービスを呼び出すために必要な、すべてのアクセス許可が付与されています。

トピック

- Amazon EKS クラスターのロールの使用
- Amazon EKS ノードグループでのロールの使用
- Amazon EKS Fargate プロファイルのロールの使用

- ロールを使用して Kubernetes クラスターを Amazon EKS に接続する
- Outpost で Amazon EKS ローカルクラスターのロールの使用

Amazon EKS クラスターのロールの使用

Amazon Elastic Kubernetes Service は、AWS Identity and Access Management (IAM) のサービスにリンクされたロールを使用します。サービスにリンクされたロールは、Amazon EKS に直接リンクされた一意のタイプの IAM ロールです。サービスにリンクされたロールは、Amazon EKS で事前定義されています。このロールには、サービスがユーザーに代わって他の AWS のサービスを呼び出すために必要な、すべてのアクセス許可が付与されています。

サービスにリンクされたロールを使用することで、必要なアクセス許可を手動で追加する必要がなくなるため、Amazon EKS の設定が簡単になります。サービスにリンクされたロールのアクセス許可は、Amazon EKS により定義されます。特に指定されている場合を除き、Amazon EKS のみがそのロールを引き受けることができます。定義される許可は、信頼ポリシーと許可ポリシーに含まれており、その許可ポリシーを他の IAM エンティティにアタッチすることはできません。

サービスリンクロールを削除するには、まずその関連リソースを削除します。これにより、リソースへのアクセス許可を不用意に削除することが防止され、Amazon EKS リソースが保護されます。

サービスにリンクされたロールをサポートするその他のサービスについては、「IAM と連携する AWS サービス」を参照し、「サービスにリンクされたロール」列に「はい」があるサービスを探してください。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[はい] リンクを選択します。

Amazon EKS でのサービスにリンクされたロールのアクセス許可

Amazon EKS では、`AWSServiceRoleForAmazonEKS` という名前のサービスにリンクされたロールを使用します。 – このロールにより、Amazon EKS はアカウント内のクラスターを管理できるようになります。アタッチされたポリシーにより、ロールはネットワークインターフェイス、セキュリティグループ、ログ、VPC のリソースを管理できるようになります。

> **Note**
>
> `AWSServiceRoleForAmazonEKS` サービスにリンクされたロールは、クラスターの作成に必要なロールとは異なります。詳細については、「Amazon EKS クラスター の IAM ロール」を参照してください。

AWSServiceRoleForAmazonEKS サービスリンクロールは、ロールの引き受けについて以下のサービスを信頼します。

- eks.amazonaws.com

ロールのアクセス許可ポリシーは、指定したリソースに対して以下のアクションを実行することを Amazon EKS に許可します。

- AmazonEKSServiceRolePolicy

サービスにリンクされたロールの作成、編集、削除を IAM エンティティ (ユーザー、グループ、ロールなど) に許可するには、アクセス許可を設定する必要があります。詳細については、IAM ユーザーガイド の「サービスにリンクされたロールのアクセス許可」を参照してください。

Amazon EKS でのサービスにリンクされたロールの作成

サービスにリンクされたロールを手動で作成する必要はありません。AWS Management Console、AWS CLI、または AWS API でクラスターを作成すると、Amazon EKS によってサービスリンクロールが作成されます。

このサービスにリンクされたロールを削除した後で再度作成する必要が生じた場合は、同じ方法でアカウントにロールを再作成できます。サービスにリンクされたロールは、クラスターの作成時に Amazon EKS で自動的に再作成されます。

Amazon EKS でのサービスにリンクされたロールの編集

Amazon EKS では、AWSServiceRoleForAmazonEKS のサービスにリンクされたロールを編集することはできません。サービスにリンクされたロールを作成すると、多くのエンティティによってロールが参照される可能性があるため、ロール名を変更することはできません。ただし、IAM を使用したロールの説明の編集はできます。詳細については、『IAM ユーザーガイド』の「サービスにリンクされたロールの編集」を参照してください。

Amazon EKS でのサービスにリンクされたロールの削除

サービスにリンクされたロールが必要な機能またはサービスが不要になった場合には、そのロールを削除することをお勧めします。そうすることで、使用していないエンティティがアクティブにモニタリングされたり、メンテナンスされたりすることがなくなります。ただし、手動で削除する前に、サービスにリンクされたロールをクリーンアップする必要があります。

サービスリンクロールのクリーンアップ

IAM を使用してサービスにリンクされたロールを削除するには、最初に、そのロールで使用されているリソースをすべて削除する必要があります。

> **Note**
>
> リソースの削除を試みた際に、対応するロールが Amazon EKS サービスで使用されている場合、削除が失敗することがあります。失敗した場合は、数分待ってから再度オペレーションを実行してください。

`AWSServiceRoleForAmazonEKS` ロールで使用されている Amazon EKS リソースを削除するには

1. Amazon EKS コンソール https://console.aws.amazon.com/eks/home#/clusters を開きます。
2. 左のナビゲーションペインで [Clusters (クラスター)] を選択します。
3. クラスターにノードグループまたは Fargate プロファイルがある場合は、クラスターを削除する前にそれらを削除する必要があります。詳細については、マネージド型ノードグループの削除およびFargate プロファイルの削除を参照してください。
4. [Clusters (クラスター)] ページで、削除するクラスターを選択し、[Delete (削除)] を選択します。
5. 削除確認ウィンドウにクラスター名を入力し、[Delete (削除)] を選択します。
6. この手順をアカウント内の他のすべてのクラスターに対して繰り返します。すべての削除操作が完了するまで待ちます。

サービスにリンクされたロールを手動で削除する

IAM コンソール、AWS CLI、または AWS API を使用して、`AWSServiceRoleForAmazonEKS` サービスにリンクされたロールを削除します。詳細については、IAM ユーザーガイド の「サービスにリンクされたロールの削除」を参照してください。

Amazon EKS のサービスにリンクされたロールがサポートされるリージョン

Amazon EKS では、このサービスを利用できるすべてのリージョンで、サービスにリンクされたロールの使用がサポートされます。詳細については、「Amazon EKS endpoints and quotas」(Amazon EKS エンドポイントとクォータ) を参照してください。

Amazon EKS ノードグループでのロールの使用

Amazon EKS は、AWS Identity and Access Management (IAM) のサービスにリンクされたロールを使用しています。サービスにリンクされたロールは、Amazon EKS に直接リンクされた一意のタイプの IAM ロールです。サービスにリンクされたロールは、Amazon EKS で事前定義されています。このロールには、サービスがユーザーに代わって他の AWS のサービスを呼び出すために必要な、すべてのアクセス許可が付与されています。

サービスにリンクされたロールを使用することで、必要なアクセス許可を手動で追加する必要がなくなるため、Amazon EKS の設定が簡単になります。サービスにリンクされたロールのアクセス許可は、Amazon EKS により定義されます。特に指定されている場合を除き、Amazon EKS のみがそのロールを引き受けることができます。定義される許可は、信頼ポリシーと許可ポリシーに含まれており、その許可ポリシーを他の IAM エンティティにアタッチすることはできません。

サービスリンクロールを削除するには、まずその関連リソースを削除します。これにより、リソースへのアクセス許可を不用意に削除することが防止され、Amazon EKS リソースが保護されます。

サービスにリンクされたロールをサポートするその他のサービスについては、「IAM と連携する AWS サービス」を参照し、「サービスにリンクされたロール」列に「はい」があるサービスを探してください。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[はい] リンクを選択します。

Amazon EKS でのサービスにリンクされたロールのアクセス許可

Amazon EKS では、AWSServiceRoleForAmazonEKSNodegroup という名前のサービスにリンクされたロールを使用します。 – このロールにより、Amazon EKS はアカウント内のノードグループを管理できるようになります。アタッチされたポリシーにより、ロールは Auto Scaling グループ、セキュリティグループ、起動テンプレート、および IAM インスタンスプロファイルのリソースを管理できるようになります。

AWSServiceRoleForAmazonEKSNodegroup サービスリンクロールは、ロールの引き受けについて以下のサービスを信頼します。

- eks-nodegroup.amazonaws.com

ロールのアクセス許可ポリシーは、指定したリソースに対して以下のアクションを実行することを Amazon EKS に許可します。

- AWSServiceRoleForAmazonEKSNodegroup

サービスにリンクされたロールの作成、編集、削除を IAM エンティティ (ユーザー、グループ、ロールなど) に許可するには、アクセス許可を設定する必要があります。詳細については、IAM ユーザーガイド の「サービスにリンクされたロールのアクセス許可」を参照してください。

Amazon EKS でのサービスにリンクされたロールの作成

サービスにリンクされたロールを手動で作成する必要はありません。AWS Management Console、AWS CLI、または AWS API でノードグループを作成すると、Amazon EKS によってサービスリンクロールが作成されます。

> **Important**
>
> このサービスにリンクされたロールがアカウントに表示されるのは、このロールでサポートされている機能を使用する別のサービスでアクションが完了した場合です。2017 年 1 月 1 日より前に Amazon EKS サービスを使用している場合、サービスにリンクされたロールのサポートが開始された時点で、AWSServiceRoleForAmazonEKSNodegroup ロールは Amazon EKS によりアカウントに作成されています。詳細については、 IAM アカウントに新しいロールが表示されるを参照してください。

Amazon EKS でのサービスにリンクされたロールの作成 (AWS API)

サービスにリンクされたロールを手動で作成する必要はありません。AWS Management Console、AWS CLI、または AWS API でマネージド型ノードグループを作成すると、Amazon EKS によってサービスリンクロールが作成されます。

このサービスにリンクされたロールを削除した後で再度作成する必要が生じた場合は、同じ方法でアカウントにロールを再作成できます。別のマネージド型ノードグループを作成すると、Amazon EKS によってサービスリンクロールが再度作成されます。

Amazon EKS でのサービスにリンクされたロールの編集

Amazon EKS では、`AWSServiceRoleForAmazonEKSNodegroup` のサービスにリンクされたロールを編集することはできません。サービスにリンクされたロールを作成すると、多くのエンティティによってロールが参照される可能性があるため、ロール名を変更することはできません。ただし、IAM を使用したロールの説明の編集はできます。詳細については、『IAM ユーザーガイド』の「サービスにリンクされたロールの編集」を参照してください。

Amazon EKS でのサービスにリンクされたロールの削除

サービスにリンクされたロールが必要な機能またはサービスが不要になった場合には、そのロールを削除することをお勧めします。そうすることで、使用していないエンティティがアクティブにモニタリングされたり、メンテナンスされたりすることがなくなります。ただし、手動で削除する前に、サービスにリンクされたロールをクリーンアップする必要があります。

サービスリンクロールのクリーンアップ

IAM を使用してサービスにリンクされたロールを削除するには、最初に、そのロールで使用されているリソースをすべて削除する必要があります。

> **Note**
>
> リソースの削除を試みた際に、対応するロールが Amazon EKS サービスで使用されている場合、削除が失敗することがあります。失敗した場合は、数分待ってから再度オペレーションを実行してください。

`AWSServiceRoleForAmazonEKSNodegroup` ロールで使用されている Amazon EKS リソースを削除するには

1. Amazon EKS コンソール https://console.aws.amazon.com/eks/home#/clusters を開きます。
2. 左のナビゲーションペインで [クラスター] を選択します。
3. [Compute] (コンピューティング) タブを選択します。
4. [Node groups] (ノードグループ) セクションで、削除するノードグループを選択します。
5. 削除確認ウィンドウにノードグループの名前を入力し、[Delete (削除)] を選択します。
6. この手順をクラスター内の他のすべてのノードグループに対して繰り返します。すべての削除操作が完了するまで待ちます。

サービスにリンクされたロールを手動で削除する

IAM コンソール、AWS CLI、または AWS API を使用して、`AWSServiceRoleForAmazonEKSNodegroup` サービスにリンクされたロールを削除します。詳細については、IAM ユーザーガイド の「サービスにリンクされたロールの削除」を参照してください。

Amazon EKS のサービスにリンクされたロールがサポートされるリージョン

Amazon EKS では、このサービスを利用できるすべてのリージョンで、サービスにリンクされたロールの使用がサポートされます。詳細については、「Amazon EKS endpoints and quotas」(Amazon EKS エンドポイントとクォータ) を参照してください。

Amazon EKS Fargate プロファイルのロールの使用

Amazon EKS は、AWS Identity and Access Management (IAM) のサービスにリンクされたロールを使用しています。サービスにリンクされたロールは、Amazon EKS に直接リンクされた一意のタイプの IAM ロールです。サービスにリンクされたロールは、Amazon EKS で事前定義されています。このロールには、サービスがユーザーに代わって他の AWS のサービスを呼び出すために必要な、すべてのアクセス許可が付与されています。

サービスにリンクされたロールを使用することで、必要なアクセス許可を手動で追加する必要がなくなるため、Amazon EKS の設定が簡単になります。サービスにリンクされたロールのアクセス許可は、Amazon EKS により定義されます。特に指定されている場合を除き、Amazon EKS のみがそのロールを引き受けることができます。定義される許可は、信頼ポリシーと許可ポリシーに含まれており、その許可ポリシーを他の IAM エンティティにアタッチすることはできません。

サービスリンクロールを削除するには、まずその関連リソースを削除します。これにより、リソースへのアクセス許可を不用意に削除することが防止され、Amazon EKS リソースが保護されます。

サービスにリンクされたロールをサポートするその他のサービスについては、「IAM と連携する AWS サービス」を参照し、「サービスにリンクされたロール」列に「はい」があるサービスを探してください。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[はい] リンクを選択します。

Amazon EKS でのサービスにリンクされたロールのアクセス許可

Amazon EKS では、`AWSServiceRoleForAmazonEKSForFargate` という名前のサービスにリンクされたロールを使用します。このロールにより、Amazon EKS Fargate は Fargate Pods に必要な VPC ネットワーキングを設定できるようになります。アタッチされたポリシーにより、ロールは Elastic Network Interface の作成と削除、Elastic Network Interface とリソースの記述ができるようになります。

`AWSServiceRoleForAmazonEKSForFargate` サービスリンクロールは、ロールの引き受けについて以下のサービスを信頼します。

- `eks-fargate.amazonaws.com`

ロールのアクセス許可ポリシーは、指定したリソースに対して以下のアクションを実行することを Amazon EKS に許可します。

- AmazonEKSForFargateServiceRolePolicy

サービスにリンクされたロールの作成、編集、削除を IAM エンティティ (ユーザー、グループ、ロールなど) に許可するには、アクセス許可を設定する必要があります。詳細については、IAM ユーザーガイド の「サービスにリンクされたロールのアクセス許可」を参照してください。

Amazon EKS でのサービスにリンクされたロールの作成

サービスにリンクされたロールを手動で作成する必要はありません。AWS Management Console、AWS CLI、または AWS API で Fargate プロファイルを作成すると、Amazon EKS がサービスにリンクされたロールを作成します。

> **Important**
>
> このサービスにリンクされたロールがアカウントに表示されるのは、このロールでサポートされている機能を使用する別のサービスでアクションが完了した場合です。2019 年 12 月 13 日より前に Amazon EKS サービスを使用している場合、サービスにリンクされたロールのサポートが開始された時点で、AWSServiceRoleForAmazonEKSForFargate ロールは Amazon EKS によりアカウントに作成されています。詳細については、「IAM アカウントに表示される新しいロール」を参照してください。

Amazon EKS でのサービスにリンクされたロールの作成 (AWS API)

サービスにリンクされたロールを手動で作成する必要はありません。AWS Management Console、AWS CLI、または AWS API で Fargate プロファイルを作成すると、Amazon EKS がサービスにリンクされたロールを作成します。

このサービスにリンクされたロールを削除した後で再度作成する必要が生じた場合は、同じ方法でアカウントにロールを再作成できます。別のマネージド型ノードグループを作成すると、Amazon EKS によってサービスリンクロールが再度作成されます。

Amazon EKS でのサービスにリンクされたロールの編集

Amazon EKS では、AWSServiceRoleForAmazonEKSForFargate のサービスにリンクされたロールを編集することはできません。サービスにリンクされたロールを作成すると、多くのエンティ

ティによってロールが参照される可能性があるため、ロール名を変更することはできません。ただし、IAM を使用したロールの説明の編集はできます。詳細については、『IAM ユーザーガイド』の「サービスにリンクされたロールの編集」を参照してください。

Amazon EKS でのサービスにリンクされたロールの削除

サービスにリンクされたロールが必要な機能またはサービスが不要になった場合には、そのロールを削除することをお勧めします。そうすることで、使用していないエンティティがアクティブにモニタリングされたり、メンテナンスされたりすることがなくなります。ただし、手動で削除する前に、サービスにリンクされたロールをクリーンアップする必要があります。

サービスリンクロールのクリーンアップ

IAM を使用してサービスにリンクされたロールを削除するには、最初に、そのロールで使用されているリソースをすべて削除する必要があります。

> **Note**
>
> リソースの削除を試みた際に、対応するロールが Amazon EKS サービスで使用されている場合、削除が失敗することがあります。失敗した場合は、数分待ってから再度オペレーションを実行してください。

`AWSServiceRoleForAmazonEKSForFargate` ロールで使用されている Amazon EKS リソースを削除するには

1. Amazon EKS コンソール https://console.aws.amazon.com/eks/home#/clusters を開きます。
2. 左のナビゲーションペインで [Clusters] (クラスター) を選択します。
3. [Clusters] (クラスター) ページで、クラスターを選択します。
4. [Compute] (コンピューティング) タブを選択します。
5. [Fargate profiles] (Fargateプロファイル) セクションに Fargate プロファイルがある場合、それぞれを個別に選択し、[Delete] (削除)を選択します。
6. 削除確認ウィンドウにプロファイルの名前を入力し、[Delete (削除)] を選択します。
7. クラスター内のその他の Fargate プロファイルと、アカウント内のその他のクラスターについて、この手順を繰り返します。

サービスにリンクされたロールを手動で削除する

IAM コンソール、AWS CLI、または AWS API を使用して、AWSServiceRoleForAmazonEKSForFargate のサービスにリンクされたロールを削除します。詳細については、IAM ユーザーガイド の「サービスにリンクされたロールの削除」を参照してください。

Amazon EKS のサービスにリンクされたロールがサポートされるリージョン

Amazon EKS では、このサービスを利用できるすべてのリージョンで、サービスにリンクされたロールの使用がサポートされます。詳細については、「Amazon EKS endpoints and quotas」(Amazon EKS エンドポイントとクォータ) を参照してください。

ロールを使用して Kubernetes クラスターを Amazon EKS に接続する

Amazon EKS は、AWS Identity and Access Management (IAM) のサービスにリンクされたロールを使用しています。サービスにリンクされたロールは、Amazon EKS に直接リンクされた一意のタイプの IAM ロールです。サービスにリンクされたロールは、Amazon EKS で事前定義されています。このロールには、サービスがユーザーに代わって他の AWS のサービスを呼び出すために必要な、すべてのアクセス許可が付与されています。

サービスにリンクされたロールを使用することで、必要なアクセス許可を手動で追加する必要がなくなるため、Amazon EKS の設定が簡単になります。サービスにリンクされたロールのアクセス許可は、Amazon EKS により定義されます。特に指定されている場合を除き、Amazon EKS のみがそのロールを引き受けることができます。定義される許可は、信頼ポリシーと許可ポリシーに含まれており、その許可ポリシーを他の IAM エンティティにアタッチすることはできません。

サービスリンクロールを削除するには、まずその関連リソースを削除します。これにより、リソースへのアクセス許可を不用意に削除することが防止され、Amazon EKS リソースが保護されます。

サービスにリンクされたロールをサポートするその他のサービスについては、「IAM と連携する AWS サービス」を参照し、「サービスにリンクされたロール」列に「はい」があるサービスを探してください。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[はい] リンクを選択します。

Amazon EKS でのサービスにリンクされたロールのアクセス許可

Amazon EKS では、`AWSServiceRoleForAmazonEKSConnector` という名前のサービスにリンクされたロールを使用します。- このロールにより、Amazon EKS は Kubernetes クラスターに接続で

きます。添付されたポリシーにより、ロールは、登録された Kubernetes クラスターに接続するために必要なリソースを管理できます。

AWSServiceRoleForAmazonEKSConnector サービスリンクロールは、ロールの引き受けについて以下のサービスを信頼します。

- eks-connector.amazonaws.com

ロールのアクセス許可ポリシーは、指定したリソースに対して以下のアクションを実行することを Amazon EKS に許可します。

- AmazonEKSConnectorServiceRolePolicy

サービスにリンクされたロールの作成、編集、削除を IAM エンティティ (ユーザー、グループ、ロールなど) に許可するには、アクセス許可を設定する必要があります。詳細については、IAM ユーザーガイド の「サービスにリンクされたロールのアクセス許可」を参照してください。

Amazon EKS でのサービスにリンクされたロールの作成

サービスにリンクされたロールを手動で作成してクラスターを接続する必要はありません。AWS Management Console、AWS CLI、eksctl または AWS API でクラスターを接続すると、Amazon EKS によってサービスにリンクされたロールが作成されます。

このサービスリンクロールを削除した後で再度作成する必要が生じた場合は、同じ方法でアカウントにロールを再作成できます。サービスにリンクされたロールは、クラスターの接続時に Amazon EKS で自動的に再作成されます。

Amazon EKS でのサービスにリンクされたロールの編集

Amazon EKS では、AWSServiceRoleForAmazonEKSConnector のサービスにリンクされたロールを編集することはできません。サービスにリンクされたロールを作成すると、多くのエンティティによってロールが参照される可能性があるため、ロール名を変更することはできません。ただし、IAM を使用したロールの説明の編集はできます。詳細については、『IAM ユーザーガイド』の「サービスにリンクされたロールの編集」を参照してください。

Amazon EKS でのサービスにリンクされたロールの削除

サービスにリンクされたロールが必要な機能またはサービスが不要になった場合には、そのロールを削除することをお勧めします。そうすることで、使用していないエンティティがアクティブにモニ

タリングされたり、メンテナンスされたりすることがなくなります。ただし、手動で削除する前に、サービスにリンクされたロールをクリーンアップする必要があります。

サービスリンクロールのクリーンアップ

IAM を使用してサービスにリンクされたロールを削除するには、最初に、そのロールで使用されているリソースをすべて削除する必要があります。

> **Note**
>
> リソースの削除を試みた際に、対応するロールが Amazon EKS サービスで使用されている場合、削除が失敗することがあります。失敗した場合は、数分待ってから再度オペレーションを実行してください。

`AWSServiceRoleForAmazonEKSConnector` ロールで使用されている Amazon EKS リソースを削除するには

1. Amazon EKS コンソール https://console.aws.amazon.com/eks/home#/clusters を開きます。
2. 左のナビゲーションペインで [Clusters] (クラスター) を選択します。
3. [Clusters] (クラスター) ページで、クラスターを選択します。
4. 登録解除タブをクリックし、Ok タブを選択します。

サービスリンクロールの手動による削除

IAM コンソール、AWS CLI、または AWS API を使用して、AWSServiceRoleForAmazonEKSConnector のサービスにリンクされたロールを削除します。詳細については、 IAM ユーザーガイド の「サービスにリンクされたロールの削除」を参照してください。

Outpost で Amazon EKS ローカルクラスターのロールの使用

Amazon Elastic Kubernetes Service は、AWS Identity and Access Management (IAM) のサービスにリンクされたロールを使用します。サービスにリンクされたロールは、Amazon EKS に直接リンクされた一意のタイプの IAM ロールです。サービスにリンクされたロールは、Amazon EKS で事前定義されています。このロールには、サービスがユーザーに代わって他の AWS のサービスを呼び出すために必要な、すべてのアクセス許可が付与されています。

サービスにリンクされたロールを使用することで、必要なアクセス許可を手動で追加する必要がなくなるため、Amazon EKS の設定が簡単になります。サービスにリンクされたロールのアクセス許可は、Amazon EKS により定義されます。特に指定されている場合を除き、Amazon EKS のみがそのロールを引き受けることができます。定義される許可は、信頼ポリシーと許可ポリシーに含まれており、その許可ポリシーを他の IAM エンティティにアタッチすることはできません。

サービスリンクロールを削除するには、まずその関連リソースを削除します。これにより、リソースへのアクセス許可を不用意に削除することが防止され、Amazon EKS リソースが保護されます。

サービスにリンクされたロールをサポートするその他のサービスについては、「IAM と連携する AWS サービス」を参照し、「サービスにリンクされたロール」列に「はい」があるサービスを探してください。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[はい] リンクを選択します。

Amazon EKS でのサービスにリンクされたロールのアクセス許可

Amazon EKS では、`AWSServiceRoleForAmazonEKSLocalOutpost` という名前のサービスにリンクされたロールを使用します。このロールにより、Amazon EKS はアカウント内のローカルクラスターを管理できるようになります。アタッチされたポリシーにより、ロールはネットワークインターフェイス、セキュリティグループ、ログ、Amazon EC2 インスタンスのリソースを管理できるようになります。

> **Note**
>
> `AWSServiceRoleForAmazonEKSLocalOutpost` サービスにリンクされたロールは、クラスターの作成に必要なロールとは異なります。詳細については、「Amazon EKS クラスターの IAM ロール」を参照してください。

`AWSServiceRoleForAmazonEKSLocalOutpost` サービスリンクロールは、ロールの引き受けについて以下のサービスを信頼します。

- `outposts.eks-local.amazonaws.com`

ロールのアクセス許可ポリシーは、指定したリソースに対して以下のアクションを実行することを Amazon EKS に許可します。

- `AmazonEKSServiceRolePolicy`

サービスにリンクされたロールの作成、編集、削除を IAM エンティティ (ユーザー、グループ、ロールなど) に許可するには、アクセス許可を設定する必要があります。詳細については、IAM ユーザーガイド の「サービスにリンクされたロールのアクセス許可」を参照してください。

Amazon EKS でのサービスにリンクされたロールの作成

サービスにリンクされたロールを手動で作成する必要はありません。AWS Management Console、AWS CLI、または AWS API でクラスターを作成すると、Amazon EKS によってサービスリンクロールが作成されます。

このサービスにリンクされたロールを削除した後で再度作成する必要が生じた場合は、同じ方法でアカウントにロールを再作成できます。サービスにリンクされたロールは、クラスターの作成時に Amazon EKS で自動的に再作成されます。

Amazon EKS でのサービスにリンクされたロールの編集

Amazon EKS では、`AWSServiceRoleForAmazonEKSLocalOutpost` のサービスにリンクされたロールを編集することはできません。サービスリンクロールを作成すると、多くのエンティティによってロールが参照される可能性があるため、ロール名を変更することはできません。ただし、IAM を使用したロール記述の編集はできます。詳細については、『IAM ユーザーガイド』の「サービスにリンクされたロールの編集」を参照してください。

Amazon EKS でのサービスにリンクされたロールの削除

サービスにリンクされたロールが必要な機能またはサービスが不要になった場合には、そのロールを削除することをお勧めします。そうすることで、使用していないエンティティがアクティブにモニタリングされたり、メンテナンスされたりすることがなくなります。ただし、手動で削除する前に、サービスにリンクされたロールをクリーンアップする必要があります。

サービスリンクロールのクリーンアップ

IAM を使用してサービスにリンクされたロールを削除するには、最初に、そのロールで使用されているリソースをすべて削除する必要があります。

> **Note**
>
> リソースの削除を試みた際に、対応するロールが Amazon EKS サービスで使用されている場合、削除が失敗することがあります。失敗した場合は、数分待ってから再度オペレーションを実行してください。

`AWSServiceRoleForAmazonEKSLocalOutpost` ロールで使用されている Amazon EKS リソースを削除するには

1. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。
2. 左のナビゲーションペインで Amazon EKS [Clusters] (クラスター) を選択します。
3. クラスターにノードグループまたは Fargate プロファイルがある場合は、クラスターを削除する前にそれらを削除する必要があります。詳細については、マネージド型ノードグループの削除およびFargate プロファイルの削除を参照してください。
4. [Clusters (クラスター)] ページで、削除するクラスターを選択し、[Delete (削除)] を選択します。
5. 削除確認ウィンドウにクラスター名を入力し、[Delete (削除)] を選択します。
6. この手順をアカウント内の他のすべてのクラスターに対して繰り返します。すべての削除操作が完了するまで待ちます。

サービスにリンクされたロールを手動で削除する

IAM コンソール、AWS CLI、または AWS API を使用して、`AWSServiceRoleForAmazonEKSLocalOutpost` サービスにリンクされたロールを削除します。詳細については、IAM ユーザーガイド の「サービスにリンクされたロールの削除」を参照してください。

Amazon EKS のサービスにリンクされたロールがサポートされるリージョン

Amazon EKS では、このサービスを利用できるすべてのリージョンで、サービスにリンクされたロールの使用がサポートされます。詳細については、「Amazon EKS endpoints and quotas」(Amazon EKS エンドポイントとクォータ) を参照してください。

Amazon EKS クラスター の IAM ロール

Amazon EKS クラスターの IAM ロールは各クラスターに必要です。Amazon EKS によって管理される Kubernetes クラスターはこのロールを使用してノードを管理し、レガシークラウドプロバイダーはこのロールを使用して Elastic Load Balancing によるロードバランサーをサービス用に作成します。

Amazon EKS クラスターを使用するには、事前に次の IAM ポリシーのいずれかを持つ IAM ロールを作成する必要があります。

- `AmazonEKSClusterPolicy`

- カスタム IAM ポリシー。以下の最小限の許可では、Kubernetes クラスターがノードを管理することはできますが、レガシークラウドプロバイダーが Elastic Load Balancing によるロードバランサーを作成することはできません。カスタム IAM ポリシーには、少なくとも以下の許可が必要です。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:CreateTags"
      ],
      "Resource": "arn:aws:ec2:*:*:instance/*",
      "Condition": {
        "ForAnyValue:StringLike": {
          "aws:TagKeys": "kubernetes.io/cluster/*"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeVpcs",
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeAvailabilityZones",
        "kms:DescribeKey"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Note**
>
> 2023 年 10 月 3 日より前は、各クラスターの IAM ロールには AmazonEKSClusterPolicy が必要でした。
> 2020 年 4 月 16 日までは、AmazonEKSServicePolicy および AmazonEKSClusterPolicy が必須であり、ロールに推奨される名前は eksServiceRole でし

た。AWSServiceRoleForAmazonEKS のサービスにリンクされたロールにより、2020 年 4 月 16 日以降に作成されたクラスターに対しては、AmazonEKSServicePolicy ポリシーは必要なくなりました。

既存のクラスターロールの確認

次の手順を使用して、アカウントに Amazon EKS クラスターロールが既に存在しているかどうかを確認できます。

IAM コンソールで **eksClusterRole** を確認するには

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. 左のナビゲーションペインで、[ロール] を選択します。
3. ロールのリストで eksClusterRole を検索します。eksClusterRole が含まれているロールが存在しない場合は、Amazon EKS でのクラスターロールの作成を参照してロールを作成します。eksClusterRole が含まれているロールが存在する場合は、このロールを選択してアタッチされているポリシーを表示します。
4. 「アクセス許可」 を選択します。
5. AmazonEKSClusterPolicy 管理ポリシーがロールにアタッチされていることを確認します。ポリシーがアタッチされている場合、Amazon EKS クラスターロールは適切に設定されています。
6. [Trust relationships] (信頼関係) を選択し、[Edit trust policy] (信頼ポリシーの編集) を選択します。
7. 信頼関係に以下のポリシーが含まれていることを確認します。信頼関係が以下のポリシーと一致する場合、[キャンセル] を選択します。信頼関係が一致しない場合、ポリシーを [信頼ポリシーの編集] ウィンドウにコピーし、[ポリシーの更新] を選択します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
```

```
}
```

Amazon EKS でのクラスターロールの作成

クラスターロールを作成するには、AWS Management Console または AWS CLI を使用できます。

AWS Management Console

IAM コンソールで Amazon EKS クラスターロールを作成するには

1. https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。
2. [ロール] を選択してから [ロールの作成] を選びます。
3. [Trusted entity type] (信頼されたエンティティタイプ) の下で、[AWS service] (のサービス) を選択します。
4. [Use cases for other AWS のサービス] (他の ユースケース) ドロップダウンリストから、[EKS] を選択します。
5. ユースケースに [EKS - Cluster] (EKS - クラスター) を選択し、[Next] (次へ) を選択します。
6. [Add permissions] (アクセス許可を追加する) タブで、[Next] (次へ) を選択します。
7. [ロール名] に、**eksClusterRole** などのロールの一意の名前を入力します。
8. [Description] (説明) には、**Amazon EKS - Cluster role** のような説明文を入力します。
9. [ロールの作成] を選択します。

AWS CLI

1. 次の内容を *cluster-trust-policy.json* という名前のファイルにコピーします。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
```

```
}
```

2. ロールを作成します。*eksClusterRole* は、任意の名前で置き換えることができます。

```
aws iam create-role \
  --role-name eksClusterRole \
  --assume-role-policy-document file://"cluster-trust-policy.json"
```

3. 必要な IAM ポリシーをロールにアタッチします。

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy \
  --role-name eksClusterRole
```

Amazon EKS ノードの IAM ロール

Amazon EKS ノード kubelet デーモンが、ユーザーに代わって AWS API への呼び出しを実行します。ノードは、IAM インスタンスプロファイルおよび関連ポリシーを通じて、これらの API コールのアクセス許可を受け取ります。ノードを起動してクラスターに登録する前に、起動するときに使用するノード用の IAM ロールを作成する必要があります。この要件は、Amazon が提供する Amazon EKS 最適化 AMI で起動されたノード、または使用する予定の他のノード AMI で起動されたノードに適用されます。さらに、この要件はマネージド型ノードグループとセルフマネージド型ノードの両方に適用されます。

> **Note**
>
> クラスターの作成に使用したロールは使用できません。

ノードを使用するには、次の アクセス許可を持つ IAM ロールを作成する必要があります。

- kubelet が VPC 内の Amazon EC2 リソースを記述するためのアクセス許可 (AmazonEKSWorkerNodePolicy ポリシーで規定されているなど)。このポリシーは、Amazon EKS Pod Identity エージェントのアクセス許可も提供します。
- kubelet が Amazon Elastic Container Registry (Amazon ECR) のコンテナイメージを使用するためのアクセス許可 (AmazonEC2ContainerRegistryReadOnlyポリシーで規定されているなど)。ネットワーキング用の組み込みアドオンは Amazon ECR のコンテナイメージを使用するポッ

ドを実行するため、Amazon Elastic Container Registry (Amazon ECR) のコンテナイメージを使用する許可が必要です。

- (オプション) Amazon EKS Pod Identity エージェントが `eks-auth:AssumeRoleForPodIdentity` アクションを使用してポッドの認証情報を取得するためのアクセス許可。AmazonEKSWorkerNodePolicy を使用しない場合、EKS Pod Identity を使用するには EC2 アクセス許可に加えてこのアクセス権許可も提供する必要があります。
- (オプション) IRSA または EKS Pod Identity を使用して VPC CNI ポッドにアクセス許可を与えない場合は、インスタンスロールで VPC CNI へのアクセス許可を与える必要があります。AmazonEKS_CNI_Policy マネージドポリシー (IPv4 ファミリーを使用してクラスターを作成した場合) または作成した IPv6 ポリシー (IPv6 ファミリーを使用してクラスターを作成した場合) のいずれかを使用できます。ただし、このロールにポリシーをアタッチするのではなく、Amazon VPC CNI アドオン専用の別のロールにポリシーをアタッチすることをお勧めします。Amazon VPC CNI アドオン用に別のロールを作成する方法の詳細については、「サービスアカウントの IAM ロールを使用する Amazon VPC CNI plugin for Kubernetes の設定」を参照してください。

> **Note**
>
> 2023 年 10 月 3 日より前は、各マネージドノードグループの IAM ロールに AmazonEKSWorkerNodePolicy と AmazonEC2ContainerRegistryReadOnly が必要でした。
> Amazon EC2 ノードグループには、Fargate プロファイルとは異なる IAM ロールが必要です。詳細については、「Amazon EKS Pod 実行 IAM ロール」を参照してください。

既存のノードロールの確認

以下の手順を使用して、アカウントに既に Amazon EKS ノードロールがあるかどうかを確認できます。

IAM コンソールで **eksNodeRole** を確認するには

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. 左のナビゲーションペインで、[ロール] を選択します。
3. ロールのリストで `eksNodeRole`、`AmazonEKSNodeRole`、または `NodeInstanceRole` を検索します。これらの名前のいずれかを持つロールが存在しない場合、

「Amazon EKS ノードでの IAM ロールの作成」を参照してロールを作成してください。eksNodeRole、AmazonEKSNodeRole、あるいは NodeInstanceRole を含むロールが存在する場合、そのロールを選択して、アタッチされているポリシーを表示します。

4. [許可] を選択します。
5. AmazonEKSWorkerNodePolicy および AmazonEC2ContainerRegistryReadOnly のマネージドポリシーがロールにアタッチされていること、またはカスタムポリシーが最小限の許可でアタッチされていることを確認します。

> **Note**
>
> AmazonEKS_CNI_Policy ポリシーがロールに添付されている場合は、それを削除して、代わりに aws-node Kubernetes サービスアカウントにマップされている IAM ロールに添付することをお勧めします。詳細については、「サービスアカウントの IAM ロールを使用する Amazon VPC CNI plugin for Kubernetes の設定」を参照してください。

6. [信頼関係] を選択し、[信頼ポリシーの編集] を選択します。
7. 信頼関係に以下のポリシーが含まれていることを確認します。信頼関係が以下のポリシーと一致する場合、[キャンセル] を選択します。信頼関係が一致しない場合、ポリシーを [信頼ポリシーの編集] ウィンドウにコピーし、[ポリシーの更新] を選択します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Amazon EKS ノードでの IAM ロールの作成

AWS Management Console または AWS CLI を使用して、ノードの IAM ロールを作成できます。

AWS Management Console

IAM コンソールで Amazon EKS ノードロールを作成するには

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. 左のナビゲーションペインで、[ロール] を選択します。
3. [ロール] ページで、[ロールの作成] を選択します。
4. [信頼されたエンティティを選択] ページで、以下の操作を実行します。
   a. [信頼するエンティティのタイプ] で [AWS サービス] を選択します。
   b. [ユースケース] で、[EC2] を選択します。
   c. [Next] を選択します。
5. [アクセス許可を追加] ページで、カスタムポリシーをアタッチするか、以下の操作を行います。
   a. [フィルタポリシー] ボックスに **AmazonEKSWorkerNodePolicy** と入力します。
   b. 検索結果の [AmazonEKSWorkerNodePolicy] の左にあるチェックボックスを選択します。
   c. [フィルターのクリア] を選択します。
   d. [フィルタポリシー] ボックスに **AmazonEC2ContainerRegistryReadOnly** と入力します。
   e. 検索結果の [AmazonEC2ContainerRegistryReadOnly] の左にあるチェックボックスを選択します。

      このロールか、aws-node Kubernetes サービスアカウントにマッピングされた別のロールのいずれかに、AmazonEKS_CNI_Policy マネージドポリシーまたは作成した IPv6 ポリシー を添付する必要もあります。このロールに対してではなく、Kubernetes サービスアカウントに関連付けられたロールに、ポリシーを割り当てることをお勧めします。詳細については、「サービスアカウントの IAM ロールを使用する Amazon VPC CNI plugin for Kubernetes の設定」を参照してください。
   f. [Next] を選択します。
6. [名前を付けて、レビューし、作成する] ページで、以下の操作を実行します。
   a. [ロール名] に、**AmazonEKSNodeRole** などのロールの一意の名前を入力します。
   b. [説明] では、現在のテキストを「**Amazon EKS - Node role**」などの説明文に置き換えます。

c. [タグの追加 (オプション)] で、タグをキーバリューのペアとして添付して、メタデータをロールに追加します。IAM でのタグの使用に関する詳細については、『IAM ユーザーガイド』の「IAM リソースにタグを付ける」を参照してください。

d. [ロールの作成] を選択します。

AWS CLI

1. 次のコマンドを実行して、node-role-trust-relationship.json ファイルを作成します。

```
cat >node-role-trust-relationship.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

2. IAM ロールを作成します。

```
aws iam create-role \
  --role-name AmazonEKSNodeRole \
  --assume-role-policy-document file://"node-role-trust-relationship.json"
```

3. IAM ロールに、2 つの必須なマネージド IAM ポリシーをアタッチします。

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name AmazonEKSNodeRole
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name AmazonEKSNodeRole
```

4. クラスターを作成した IP ファミリーに応じて、次の IAM ポリシーの 1 つを IAM ロールにアタッチします。ポリシーは、このロールまたは Amazon VPC CNI plugin for Kubernetes に使用される Kubernetes aws-node サービスアカウントに関連付けられたロールに添付する必要があります。Kubernetes サービスアカウントに関連付けられているロールにポリシーを割り当てることをお勧めします。Kubernetes サービスアカウントに関連付けられたロールにポリシーを割り当てるには、「サービスアカウントの IAM ロールを使用する Amazon VPC CNI plugin for Kubernetes の設定」を参照しください。

   - IPv4

     ```
     aws iam attach-role-policy \
       --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
       --role-name AmazonEKSNodeRole
     ```

   - IPv6

     1. 次のテキストをコピーし、*vpc-cni-ipv6-policy*.json という名前のファイルに保存します。

        ```
        {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Action": [
                        "ec2:AssignIpv6Addresses",
                        "ec2:DescribeInstances",
                        "ec2:DescribeTags",
                        "ec2:DescribeNetworkInterfaces",
                        "ec2:DescribeInstanceTypes"
                    ],
                    "Resource": "*"
                },
                {
                    "Effect": "Allow",
                    "Action": [
                        "ec2:CreateTags"
                    ],
                    "Resource": [
                        "arn:aws:ec2:*:*:network-interface/*"
                    ]
                }
        ```

```
    ]
}
```

2. IAM ポリシーを作成する

```
aws iam create-policy --policy-name AmazonEKS_CNI_IPv6_Policy --policy-
document file://vpc-cni-ipv6-policy.json
```

3. IAM ロールに IAM ポリシーをアタッチします。*111122223333* をアカウントID に置き換えます。

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy \
  --role-name AmazonEKSNodeRole
```

Amazon EKS Pod 実行 IAM ロール

Pods を AWS Fargate インフラストラクチャで実行するために、Amazon EKS Pod 実行ロールが必要になります。

クラスターが AWS Fargate インフラストラクチャ上で Pods を作成する場合、Fargate インフラストラクチャ上で実行されているコンポーネントは、ユーザーに代わって AWS API にコールを実行する必要があります。これは、Amazon ECR からコンテナイメージをプルしたり、ログを他の AWS サービスにルーティングしたりするなどのアクションを実行できるようにするためです。Amazon EKS Pod 実行ロールにより、これらを行うための IAM アクセス許可が付与されます。

Fargate プロファイルを作成する際には、プロファイルを使用して Fargate インフラストラクチャで実行される Amazon EKS コンポーネントのために、Pod 実行ロールを指定する必要があります。このロールは、認証のためにクラスターの Kubernetes [ロールベースのアクセスコントロール] (RBAC) に追加されます。これにより、Fargate インフラストラクチャで実行されている `kubelet` が Amazon EKS クラスターに登録され、クラスター内でノードとして表示されるようになります。

> **Note**
>
> Fargate プロファイルには、Amazon EC2 ノードグループとは異なる IAM ロールが必要です。

> **Important**
>
> Fargate Pod で実行されているコンテナは、Pod 実行ロールに関連付けられた IAM アクセス許可を引き受けることはできません。Fargate Pod 内のコンテナに他の AWS のサービスへのアクセス許可を付与するには、サービスアカウントの IAM ロール を使用する必要があります。

Fargate プロファイルを作成する前に、`AmazonEKSFargatePodExecutionRolePolicy` で IAM ロールを作成する必要があります。

正しく設定された既存の Pod 実行ロールをチェックする

次の手順を使用して、正しく設定された Amazon EKS の Pod 実行ロールがアカウントにすでに存在しているかをチェックします。「混乱した代理」のセキュリティ上の問題を回避するには、ロールが `SourceArn` に基づいてアクセスを制限することが重要です。必要に応じて実行ロールを変更し、他のクラスター上で Fargate プロファイルのサポートを含めることができます。

IAM コンソールで Amazon EKS Pod 実行ロールをチェックするには

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. 左のナビゲーションペインで、[ロール] を選択します。
3. [ロール] ページで、ロールの一覧から AmazonEKSFargatePodExecutionRole を検索します。ロールが存在しない場合は、「Amazon EKS の Pod 実行ロールの作成」を参照してロールを作成します。ロールが存在する場合は、そのロールを選択します。
4. [AmazonEKSFargatePodExecutionRole] ページで、次の操作を実行します。
   a. [許可] を選択します。
   b. AmazonEKSFargatePodExecutionRolePolicy Amazon マネージドポリシーがロールにアタッチされていることを確認します。
   c. [信頼関係] を選択します。
   d. [信頼ポリシーを編集] を選択します。
5. [信頼ポリシーを編集] ページで、信頼関係に次のポリシーが含まれており、クラスター上で Fargate プロファイルの行が存在していることを確認します。そうである場合は、[キャンセル] を選択します。

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Condition": {
         "ArnLike": {
            "aws:SourceArn": "arn:aws:eks:region-
code:111122223333:fargateprofile/my-cluster/*"
         }
      },
      "Principal": {
        "Service": "eks-fargate-pods.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

ポリシーは一致するが、クラスター上で Fargate プロファイルを指定する行が存在しない場合は、ArnLike オブジェクトの上部に次の行を追加します。*region-code* をクラスターが存在する AWS リージョン で、*111122223333* をアカウント ID で、*my-cluster* を自分のクラスター名に置き換えます。

```
"aws:SourceArn": "arn:aws:eks:region-code:111122223333:fargateprofile/my-cluster/
*",
```

ポリシーが一致しない場合は、前のポリシー全体をフォームにコピーして、[ポリシーの更新] を選択します。*region-code* をクラスターのある AWS リージョン に置き換えます。アカウントのすべての AWS リージョン で同じロールを使用する場合は、*region-code* を * で置き換えます。*111122223333* をアカウント ID に置き換え、*my-cluster* を自分のクラスター名に置き換えます。アカウント内のすべてのクラスターに同じロールを使用する場合は、*my-cluster* を * に置き換えます。

Amazon EKS の Pod 実行ロールの作成

クラスター用の Amazon EKS Pod 実行ロールをまだお持ちでない場合は、AWS Management Console または AWS CLI を使用して作成できます。

AWS Management Console

AWS Management Console を使用して AWS FargatePod ポッド実行ロールを作成するには

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. 左のナビゲーションペインで、[ロール] を選択します。
3. [ロール] ページで、[ロールの作成] を選択します。
4. [信頼されたエンティティを選択] ページで、以下の操作を実行します。
   a. [信頼するエンティティのタイプ] で [AWS サービス] を選択します。
   b. [他の AWS のサービス のユースケース] ドロップダウンリストから、[EKS] を選択します。
   c. [EKS - Fargate Pod] を選択します。
   d. [Next] を選択します。
5. [アクセス許可を追加] ページで [次へ] を選択します。
6. [名前を付けて、レビューし、作成する] ページで、以下の操作を実行します。
   a. [ロール名] に、**AmazonEKSFargatePodExecutionRole** などのロールの一意の名前を入力します。
   b. [タグの追加 (オプション)] で、タグをキーバリューのペアとして添付して、メタデータをロールに追加します。IAM でのタグの使用に関する詳細については、『IAM ユーザーガイド』の「IAM リソースにタグを付ける」を参照してください。
   c. [ロールの作成] を選択します。
7. [ロール] ページで、ロールの一覧から AmazonEKSFargatePodExecutionRole を検索します。ロールを選択します。
8. [AmazonEKSFargatePodExecutionRole] ページで、次の操作を実行します。
   a. [信頼関係] を選択します。
   b. [信頼ポリシーを編集] を選択します。
9. [信頼ポリシーを編集] ページで、次の操作を実行します。
   a. 次の内容をコピーして、[信頼ポリシーを編集] フォームに貼り付けます。*region-code* を、クラスターが存在する AWS リージョン で置き換えます。アカウントのすべての AWS リージョン で同じロールを使用する場合は、*region-code* を * で置き換えます。*111122223333* をアカウント ID に置き換え、*my-cluster* を自分のクラスター

名に置き換えます。アカウント内のすべてのクラスターに同じロールを使用する場合は、*my-cluster* を * に置き換えます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Condition": {
         "ArnLike": {
            "aws:SourceArn": "arn:aws:eks:region-
code:111122223333:fargateprofile/my-cluster/*"
         }
      },
      "Principal": {
        "Service": "eks-fargate-pods.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

b. [ポリシーの更新] を選択します。

AWS CLI

AWS CLI を使用して AWS FargatePod ポッド実行ロールを作成するには

1. 次の内容をコピーして、*pod-execution-role-trust-policy*.json という名前のファイルに貼り付けます。*region-code* を、クラスターが存在する AWS リージョン で置き換えます。アカウントのすべての AWS リージョン で同じロールを使用する場合は、*region-code* を * で置き換えます。*111122223333* をアカウント ID に置き換え、*my-cluster* を自分のクラスター名に置き換えます。アカウント内のすべてのクラスターに同じロールを使用する場合は、*my-cluster* を * に置き換えます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Condition": {
```

```
          "ArnLike": {
             "aws:SourceArn": "arn:aws:eks:region-
code:111122223333:fargateprofile/my-cluster/*"
          }
       },
       "Principal": {
         "Service": "eks-fargate-pods.amazonaws.com"
       },
       "Action": "sts:AssumeRole"
     }
  ]
}
```

2. Pod 実行 IAM ロールを作成します。

```
aws iam create-role \
  --role-name AmazonEKSFargatePodExecutionRole \
  --assume-role-policy-document file://"pod-execution-role-trust-policy.json"
```

3. このロールに、必要な Amazon EKS 管理の IAM ポリシーをアタッチします。

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSFargatePodExecutionRolePolicy \
  --role-name AmazonEKSFargatePodExecutionRole
```

Amazon EKS コネクタの IAM ロール

Kubernetes クラスターを接続して AWS Management Console で表示することができます。Kubernetes クラスターに接続するには、IAM ロールを作成します。

既存の EKS Connector ロールの確認

以下の手順を使用して、アカウントに既に Amazon EKS コネクタロールがあるかどうかを確認できます。

IAM コンソールで **AmazonEKSConnectorAgentRole** を確認するには

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. 左のナビゲーションペインで、[ロール] を選択します。

3. ロールのリストで AmazonEKSConnectorAgentRole を検索します。AmazonEKSConnectorAgentRole が含まれているロールが存在しない場合は、Amazon EKS コネクタエージェントロールの作成を参照してロールを作成します。AmazonEKSConnectorAgentRole が含まれているロールが存在する場合は、このロールを選択してアタッチされているポリシーを表示します。
4. 「アクセス許可」 を選択します。
5. AmazonEKSConnectorAgentPolicy 管理ポリシーがロールにアタッチされていることを確認します。ポリシーがアタッチされている場合、Amazon EKS Connector ロールは適切に設定されています。
6. [信頼関係] を選択し、[信頼ポリシーの編集] を選択します。
7. 信頼関係に以下のポリシーが含まれていることを確認します。信頼関係が以下のポリシーと一致する場合、[キャンセル] を選択します。信頼関係が一致しない場合、ポリシーを [信頼ポリシーの編集] ウィンドウにコピーし、[ポリシーの更新] を選択します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ssm.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Amazon EKS コネクタエージェントロールの作成

コネクタエージェントロールを作成するには、AWS Management Console または AWS CloudFormation を使用できます。

AWS CLI

1. IAM ロールに使用する次の JSON が含まれる eks-connector-agent-trust-policy.json という名前のファイルを作成します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ssm.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

2. IAM ロールに使用する次の JSON が含まれる eks-connector-agent-policy.json という名前のファイルを作成します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SsmControlChannel",
            "Effect": "Allow",
            "Action": [
                "ssmmessages:CreateControlChannel"
            ],
            "Resource": "arn:aws:eks:*:*:cluster/*"
        },
        {
            "Sid": "ssmDataplaneOperations",
            "Effect": "Allow",
            "Action": [
                "ssmmessages:CreateDataChannel",
                "ssmmessages:OpenDataChannel",
                "ssmmessages:OpenControlChannel"
            ],
            "Resource": "*"
        }
    ]
}
```

3. 前のリストアイテムで作成した信頼ポリシーとポリシーを使用して、Amazon EKS Connector エージェントロールを作成します。

```
aws iam create-role \
     --role-name AmazonEKSConnectorAgentRole \
     --assume-role-policy-document file://eks-connector-agent-trust-policy.json
```

4. Amazon EKS Connector エージェントのロールにポリシーを添付します。

```
aws iam put-role-policy \
     --role-name AmazonEKSConnectorAgentRole \
     --policy-name AmazonEKSConnectorAgentPolicy \
     --policy-document file://eks-connector-agent-policy.json
```

AWS CloudFormation

Amazon EKS Connector エージェントロールを AWS CloudFormation で作成するには

1. 以下の AWS CloudFormation テンプレートをローカルシステムのテキストファイルに保存します。

> **Note**
>
> このテンプレートでは、ここで作成されなければ `registerCluster` API が呼び出されたときに作成される、サービスにリンクされたロールも作成されます。詳細については、「ロールを使用して Kubernetes クラスターを Amazon EKS に接続する」を参照してください。

```
---
AWSTemplateFormatVersion: '2010-09-09'
Description: 'Provisions necessary resources needed to register clusters in EKS'
Parameters: {}
Resources:
  EKSConnectorSLR:
    Type: AWS::IAM::ServiceLinkedRole
    Properties:
      AWSServiceName: eks-connector.amazonaws.com

  EKSConnectorAgentRole:
```

```
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Action: [ 'sts:AssumeRole' ]
            Principal:
              Service: 'ssm.amazonaws.com'

  EKSConnectorAgentPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyName: EKSConnectorAgentPolicy
      Roles:
        - {Ref: 'EKSConnectorAgentRole'}
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: 'Allow'
            Action: [ 'ssmmessages:CreateControlChannel' ]
            Resource:
            - Fn::Sub: 'arn:${AWS::Partition}:eks:*:*:cluster/*'
          - Effect: 'Allow'
            Action: [ 'ssmmessages:CreateDataChannel',
 'ssmmessages:OpenDataChannel', 'ssmmessages:OpenControlChannel' ]
            Resource: "*"
Outputs:
  EKSConnectorAgentRoleArn:
    Description: The agent role that EKS connector uses to communicate with AWS #
####.
    Value: !GetAtt EKSConnectorAgentRole.Arn
```

2. https://console.aws.amazon.com/cloudformation で AWS CloudFormation コンソール を開きます。
3. [Create stack] (スタックの作成) (新しいリソースまたは既存のリソースを使用) を選択します。
4. [テンプレートの指定] で、[テンプレートファイルのアップロード] を選択し、[ファイルの選択] を選択します。
5. 作成したファイルを選択し、[Next (次へ)] を選択します。

6. [スタックの名前] に `eksConnectorAgentRole` などのロール名を入力し、[次へ] を選択します。
7. [スタックオプションの設定] ページで、[Next (次へ)] を選択します。
8. [Review (レビュー)] ページの情報から、スタックにより IAM リソースが作成されることを確認し、[Create stack (スタックの作成)] を選択します。

Amazon Elastic Kubernetes Service に関する AWS 管理ポリシー

AWS マネージドポリシーは、AWS が作成および管理するスタンドアロンポリシーです。AWS マネージドポリシーは、多くの一般的なユースケースで権限を提供できるように設計されているため、ユーザー、グループ、ロールへの権限の割り当てを開始できます。

AWS マネージドポリシーは、ご利用の特定のユースケースに対して最小特権の権限を付与しない場合があることにご注意ください。AWS のすべてのお客様が使用できるようになるのを避けるためです。ユースケース別にカスタマーマネージドポリシーを定義して、マネージドポリシーを絞り込むことをお勧めします。

AWS マネージドポリシーで定義したアクセス権限は変更できません。AWS が AWS マネージドポリシーに定義されている権限を更新すると、更新はポリシーがアタッチされているすべてのプリンシパルアイデンティティ (ユーザー、グループ、ロール) に影響します。新しい AWS のサービス を起動するか、既存のサービスで新しい API オペレーションが使用可能になると、AWS が AWS マネージドポリシーを更新する可能性が最も高くなります。

詳細については、「IAM ユーザーガイド」の「AWS マネージドポリシー」を参照してください。

AWS 管理ポリシー: AmazonEKS_CNI_Policy

`AmazonEKS_CNI_Policy` ポリシーを IAM エンティティにアタッチできます。Amazon EC2 ノードグループを作成する前に、このポリシーをノード IAM ロール、または Amazon VPC CNI plugin for Kubernetes で特に使用する IAM ロールにアタッチする必要があります。これは、ユーザーに代わってアクションを実行できるようにするためです。プラグインでのみ使用されるロールにポリシーをアタッチすることをお勧めします。詳細については、Amazon VPC CNI plugin for Kubernetes Amazon

EKS アドオンの使用およびサービスアカウントの IAM ロールを使用する Amazon VPC CNI plugin for Kubernetes の設定を参照してください。

許可の詳細

このポリシーには、Amazon EKS に以下のタスクを完了させるための以下の権限が含まれています。

- **`ec2:*NetworkInterface`** および **`ec2:*PrivateIpAddresses`** - Amazon VPC CNI プラグインが Pods の ElasticNetwork Interface や IP アドレスのプロビジョニングなどのアクションを実行し、Amazon EKS で実行されるアプリケーションにネットワークを提供できるようにします。
- **`ec2`** 読み取りアクション - Amazon VPC CNI プラグインがインスタンスやサブネットを記述して、Amazon VPC サブネット内の空き IP アドレスの量を確認するなどのアクションを実行できるようにします。VPC CNI は各サブネットの空き IP アドレスを使用して、Elastic Network Interface の作成時に使用する空き IP アドレスが最も多いサブネットを選択できます。

JSON ポリシードキュメントの最新バージョンを確認するには、『AWS管理ポリシーリファレンスガイド』の「AmazonEKS_CNI_Policy I_Policy」を参照してください。

AWS 管理ポリシー: AmazonEKSClusterPolicy

IAM エンティティに `AmazonEKSClusterPolicy` をアタッチできます。クラスターを作成する前に、このポリシーが添付されたクラスター IAM ロールを用意する必要があります。Amazon EKS によって管理される Kubernetes クラスターは、お客様に代わって他の AWS サービスへの呼び出しを行います。これにより、サービスで使用するリソースを管理します。

このポリシーには、Amazon EKS に以下のタスクを完了させるための以下の権限が含まれています。

- **`autoscaling`** – Auto Scaling グループの設定を読み取り、更新します。これらの権限は Amazon EKS では使用されませんが、下位互換性のためにポリシーに残ります。
- **`ec2`** – Amazon EC2 ノードに関連付けられているボリュームとネットワークリソースを操作します。これは、Kubernetes コントロールプレーンがインスタンスをクラスターに結合し、Kubernetes 永続ボリュームによってリクエストされる Amazon EBS ボリュームを動的にプロビジョニングおよび管理できるようにするために必要です。
- **`elasticloadbalancing`** – Elastic Load Balancer を操作し、ノードをターゲットとして追加します。これは、Kubernetes コントロールプレーンが Kubernetes サービスによってリクエストされる Elastic Load Balancer を動的にプロビジョニングできるようにするために必要です。

- **iam** - サービスにリンクされたロールを作成します。これは、Kubernetes コントロールプレーンが Kubernetes サービスによってリクエストされる Elastic Load Balancer を動的にプロビジョニングできるようにするために必要です。
- **kms** – AWS KMS からキーを読み込みます。これは、Kubernetes コントロールプレーンが `etcd` に保存される Kubernetes シークレットの シークレット暗号化 を管理するために必要です。

JSON ポリシードキュメントの最新バージョンを確認するには、『AWS管理ポリシーリファレンスガイド』の「AmazonEKSClusterPolicy Policy」を参照してください。

AWS 管理ポリシー: AmazonEKSFargatePodExecutionRolePolicy

IAM エンティティに `AmazonEKSFargatePodExecutionRolePolicy` をアタッチできます。Fargate プロファイルを作成する前に、Fargate Pod 実行ロールを作成し、このポリシーをアタッチする必要があります。詳細については、Fargate Pod 実行ロールを作成するおよびAWS Fargate プロファイルを参照してください。

このポリシーはロールに対して、Fargate で Amazon EKS Pods を実行するために必要な他の AWS サービスリソースにアクセスするための権限を付与します。

許可の詳細

このポリシーには、Amazon EKS に以下のタスクを完了させるための以下の権限が含まれています。

- **ecr** – Fargate で実行中のポッドが、Amazon ECR に格納されているコンテナイメージを取得できるようにします。

JSON ポリシードキュメントの最新バージョンを確認するには、『AWS管理ポリシーリファレンスガイド』の「AmazonEKSFargatePodExecutionRolePolicy Policy」を参照してください。

AWS 管理ポリシー: AmazonEKSForFargateServiceRolePolicy

IAM エンティティに `AmazonEKSForFargateServiceRolePolicy` をアタッチすることはできません。このポリシーは、ユーザーに代わって Amazon EKS がアクションを実行することを許可する、サービスにリンクされたロールにアタッチされます。詳細については、「AWSServiceRoleforAmazonEKSForFargate」を参照してください。

このポリシーは、Fargate タスクを実行するために必要なアクセス権限を Amazon EKS に付与します。このポリシーは、Fargate ノードがある場合にのみ使用されます。

許可の詳細

このポリシーには、Amazon EKS が以下のタスクを完了できるようにする以下の権限が含まれています。

- **ec2** – Elastic Network Interface を作成および削除し、Elastic Network Interface とリソースについて説明します。これは、Amazon EKS Fargate サービスが Fargate ポッドに必要な VPC ネットワーキングを設定できるようにするために必要です。

JSON ポリシードキュメントの最新バージョンを確認するには、『AWS管理ポリシーリファレンスガイド』の「AmazonEKSForFargateServiceRolePolicy Policy」を参照してください。

AWS 管理ポリシー: AmazonEKSServicePolicy

IAM エンティティに `AmazonEKSServicePolicy` をアタッチできます。2020 年 4 月 16 日より前に作成されたクラスターでは、IAM ロールを作成し、このポリシーをアタッチする必要がありました。2020 年 4 月 16 日以降に作成されたクラスターでは、ロールを作成する必要はなく、このポリシーの割り当ても必要ありません。IAM プリンシパルを使用してクラスターを作成する場合、`iam:CreateServiceLinkedRole` 権限がある場合、AWS ServiceRoleforAmazonEKS のサービスにリンクされたロールが自動的に作成されます。サービスにリンクされたロールには AWS 管理ポリシー: AmazonEKSServiceRolePolicy がアタッチされています。

このポリシーにより、Amazon EKS は Amazon EKS クラスターを操作するために必要なリソースを作成および管理できるようになります。

許可の詳細

このポリシーには、Amazon EKS が以下のタスクを完了できるようにする以下の権限が含まれています。

- **eks** - 更新を開始した後、クラスターの Kubernetes バージョンを更新します。この権限は Amazon EKS では使用されませんが、下位互換性のためにポリシーに残ります。
- **ec2** – Elastic Network Interface およびその他のネットワークリソースとタグを操作します。これは、ノードと Kubernetes コントロールプレーン間の通信を容易にするネットワーキングを設定するために、Amazon EKS で必要です。
- **route53** – VPC をホストゾーンに関連付けます。これは、Kubernetes クラスター API サーバーのプライベートエンドポイントネットワーキングを有効にするために、Amazon EKS で必要です。

- **logs** – ログイベント これは、Amazon EKS が Kubernetes コントロールプレーンログを CloudWatch に送信できるようにするために必要です。
- **iam** - サービスにリンクされたロールを作成します。これは、Amazon EKS がユーザーに代わって AWSServiceRoleForAmazonEKS のサービスにリンクされたロールを作成するために必要です。

JSON ポリシードキュメントの最新バージョンを確認するには、『AWS管理ポリシーリファレンスガイド』の「AmazonEKSServicePolicy olicy」を参照してください。

AWS 管理ポリシー: AmazonEKSServiceRolePolicy

IAM エンティティに AmazonEKSServiceRolePolicy をアタッチすることはできません。このポリシーは、ユーザーに代わって Amazon EKS がアクションを実行することを許可する、サービスにリンクされたロールにアタッチされます。詳細については、「Amazon EKS でのサービスにリンクされたロールのアクセス許可」を参照してください。iam:CreateServiceLinkedRole 権限のある IAM プリンシパルを使用してクラスターを作成する場合、AWSServiceRoleforAmazonEKS のサービスにリンクされたロールが自動的に作成され、このポリシーがアタッチされます。

このポリシーにより、サービスにリンクされたロールはユーザーに代わって AWS サービスを呼び出します。

許可の詳細

このポリシーには、Amazon EKS が以下のタスクを完了できるようにする以下の権限が含まれています。

- **ec2** – Elastic Network Interface と Amazon EC2 インスタンス、クラスターセキュリティグループ、およびクラスターの作成に必要な VPC を作成、記述します。
- **iam** – IAM ロールにアタッチされているすべての管理ポリシーを一覧表示します。これは、Amazon EKS がクラスターの作成に必要なすべての管理ポリシーと権限を一覧表示および検証できるようにするために必要です。
- VPC をホストゾーンに関連付ける - これは、Kubernetes クラスター API サーバーのプライベートエンドポイントネットワーキングを有効にするために、Amazon EKS で必要です。
- ログイベント - これは、Amazon EKS が Kubernetes コントロールプレーンログを CloudWatch に送信できるようにするために必要です。

JSON ポリシードキュメントの最新バージョンを確認するには、『AWS管理ポリシーリファレンスガイド』の「AmazonEKSServiceRolePolicy」を参照してください。

AWS 管理ポリシー: AmazonEKSVPCResourceController

AmazonEKSVPCResourceController ポリシーを IAM アイデンティティにアタッチできます。Pods のセキュリティグループ を使用している場合は、ユーザーに代わってアクションを実行させるために、このポリシーを Amazon EKS クラスター の IAM ロール にアタッチします。

このポリシーは、ノードの Elastic Network Interface と IP アドレスを管理するアクセス権限をクラスターロールに付与します。

許可の詳細

このポリシーには、Amazon EKS に以下のタスクを完了させるための以下の権限が含まれています。

- **ec2** - Elastic Network Interface と IP アドレスを管理し、Pod ポッドセキュリティグループと Windows ノードをサポートします。

JSON ポリシードキュメントの最新バージョンを確認するには、『AWS管理ポリシーリファレンスガイド』の「AmazonEKSVPCResourceController」を参照してください。

AWS 管理ポリシー: AmazonEKSWorkerNodePolicy

AmazonEKSWorkerNodePolicy ポリシーを IAM エンティティにアタッチできます。このポリシーを、Amazon EKS がユーザーに代わってアクションを実行することを許可する Amazon EC2 ノードを作成するときに指定するノード IAM ロールにアタッチしなければなりません。eksctl を使用してノードグループを作成すると、ノード IAM ロールが作成され、このポリシーをロールに自動的にアタッチします。

このポリシーは、Amazon EKS Amazon EC2 ノードに Amazon EKS クラスターに接続するためのアクセス権限を付与します。

許可の詳細

このポリシーには、Amazon EKS に以下のタスクを完了させるための以下の権限が含まれています。

- **ec2** – インスタンスボリュームとネットワーク情報を読み取ります。これは、Kubernetes ノードが Amazon EKS クラスターに参加するのに必要な Amazon EC2 リソースに関する情報を記述できるようにするために必要です。
- **eks** – オプションで、ノードのブートストラップの一部としてクラスターを記述します。

- **eks-auth:AssumeRoleForPodIdentity** – ノード上の EKS ワークロードの認証情報を取得できるようにします。これは、EKS Pod Identity が正しく機能するために必要です。

JSON ポリシードキュメントの最新バージョンを確認するには、『AWS管理ポリシーリファレンスガイド』の「AmazonEKSWorkerNodePolicy Policy」を参照してください。

AWS 管理ポリシー: AWSServiceRoleForAmazonEKSNodegroup

IAM エンティティに AWSServiceRoleForAmazonEKSNodegroup をアタッチすることはできません。このポリシーは、ユーザーに代わって Amazon EKS がアクションを実行することを許可する、サービスにリンクされたロールにアタッチされます。詳細については、「Amazon EKS でのサービスにリンクされたロールのアクセス許可」を参照してください。

このポリシーは、AWSServiceRoleForAmazonEKSNodegroup ロール権限を付与し、アカウント内の Amazon EC2 ノードグループを作成および管理できるようにします。

許可の詳細

このポリシーには、Amazon EKS に以下のタスクを完了させるための以下の権限が含まれています。

- **ec2** – セキュリティグループ、タグ、および起動テンプレートを操作します。これは、Amazon EKS マネージド型ノードグループがリモートアクセス設定を有効にするために必要です。さらに、Amazon EKS マネージド型ノードグループは、ユーザーに代わって起動テンプレートを作成します。これは、各マネージド型ノードグループをバックアップする Amazon EC2 Auto Scaling グループを設定するためです。
- **iam** – サービスにリンクされたロールを作成し、ロールを渡します。これは、Amazon EKS マネージド型ノードグループが、マネージドノードグループの作成時に渡されるロールのインスタンスプロファイルを管理するために必要です。このインスタンスプロファイルは、マネージド型ノードグループの一部として起動される Amazon EC2 インスタンスによって使用されます。Amazon EKS は、Amazon EC2 Auto Scaling グループなどの他のサービスに対してサービスリンクされたロールを作成する必要があります。これらの権限は、マネージド型ノードグループの作成に使用されます。
- **autoscaling** – セキュリティの Auto Scaling グループを使用します。これは、Amazon EKS マネージド型ノードグループが、各マネージドノードグループをバックアップする Amazon EC2 Auto Scaling グループを管理するために必要です。また、ノードグループの更新中にノードが終了またはリサイクルされたときに Pods を強制撤去させるなどの機能をサポートするためにも使用されます。

JSON ポリシードキュメントの最新バージョンを確認するには、「AWS マネージドポリシーリファレンスガイド」の「AWSServiceRoleForAmazonEKSNodegroup」を参照してください。

AWS 管理ポリシー: AmazonEBSCSIDriverPolicy

`AmazonEBSCSIDriverPolicy` ポリシーは、Amazon EBS Container Storage Interface (CSI) ドライバーが、ユーザーに代わってボリュームを作成、変更、アタッチ、デタッチ、削除できるようにします。また、EBS CSI ドライバーに、スナップショットの作成と削除、インスタンス、ボリューム、およびスナップショットの一覧表示を行う権限も付与します。

JSON ポリシードキュメントの最新バージョンを確認するには、『AWS管理ポリシーリファレンスガイド』の「AmazonEBSCSIDriverServiceRolePolicy」を参照してください。

AWS マネージドポリシー: AmazonEFSCSIDriverPolicy

`AmazonEFSCSIDriverPolicy` ポリシーでは、Amazon EFS コンテナストレージインターフェイス (CSI) がユーザーに代わってアクセスポイントを作成および削除できるようにすることができます。また、Amazon EFS CSI ドライバーに、アクセスポイント、ファイルシステム、マウントターゲット、Amazon EC2 アベイラビリティゾーンを一覧表示するアクセス許可を付与することもできます。

JSON ポリシードキュメントの最新バージョンを確認するには、AWS マネージドポリシーリファレンスガイドの「AmazonEFSCSIDriverServiceRolePolicy」を参照してください。

AWS マネージドポリシー: AmazonEKSLocalOutpostClusterPolicy

このポリシーを IAM エンティティにアタッチできます。ローカルクラスターを作成する前に、このポリシーをクラスターロールにアタッチする必要があります。Amazon EKS によって管理される Kubernetes クラスターは、お客様に代わって他の AWS サービスへの呼び出しを行います。これにより、サービスで使用するリソースを管理します。

`AmazonEKSLocalOutpostClusterPolicy` には以下の権限が含まれています。

- **`ec2`** - Amazon EC2 インスタンスがコントロールプレーンインスタンスとして正常にクラスターに参加するために必要なアクセス許可です。
- **`ssm`** - Amazon EC2 Systems Manager がコントロールプレーンインスタンスに接続できるようにします。これは、アカウントのローカルクラスターと通信して管理するために、Amazon EKS によって使用されます。
- **`logs`** - インスタンスが Amazon CloudWatch にログをプッシュできるようにします。

- **secretsmanager** - インスタンスがコントロールプレーンインスタンスのブートストラップデータを AWS Secrets Manager から安全に取得および削除できるようにします。
- **ecr** - コントロールプレーンインスタンス上で動作している Pods とコンテナが、Amazon Elastic Container Registry に格納されているコンテナイメージをプルできるようにします。

JSON ポリシードキュメントの最新バージョンを確認するには、『AWS管理ポリシーリファレンスガイド』の「AmazonEKSLocalOutPostClusterPolicy」を参照してください。

AWS マネージドポリシー: AmazonEKSLocalOutpostServiceRolePolicy

このポリシーを IAM エンティティにアタッチすることはできません。`iam:CreateServiceLinkedRole` 権限のある IAM プリンシパルを使用してクラスターを作成する場合、Amazon EKS は `AWSServiceRoleforAmazonEKSLocalOutpost` のサービスにリンクされたロールが自動的に作成され、このポリシーがアタッチされます。このポリシーは、サービスにリンクされたロールはローカルクラスターの代わりに AWS サービスを呼び出すことを許可します。

`AmazonEKSLocalOutpostServiceRolePolicy` には以下の権限が含まれています。

- **ec2** - Amazon EKS がセキュリティ、ネットワーク、その他のリソースと連携して、アカウント内のコントロールプレーンインスタンスを正常に起動および管理できるようにします。
- **ssm** - Amazon EC2 Systems Manager がコントロールプレーンインスタンスに接続できるようにします。これは、アカウントのローカルクラスターと通信して管理するために、Amazon EKS によって使用されます。
- **iam** - Amazon EKS がコントロールプレーンインスタンスに関連付けられたインスタンスプロファイルを管理できるようにします。
- **secretsmanager** - Amazon EKS がコントロールプレーンインスタンスのブートストラップデータを AWS Secrets Manager に格納できるようにします。これにより、インスタンスのブートストラップ中に安全に参照できるようになります。
- **outposts** - Amazon EKS がお客様のアカウントから Outpost 情報を取得して、Outpost でローカルクラスターを正常に起動できるようにします。

JSON ポリシードキュメントの最新バージョンを確認するには、『AWS管理ポリシーリファレンスガイド』の「AmazonEKSLocalOutPostServiceRolePolicy」を参照してください。

Amazon EKS による AWS 管理ポリシーの更新

Amazon EKS の AWS 管理ポリシーの更新に関する詳細を、このサービスがこれらの変更の追跡を開始した以降の分について表示します。このページの変更に関する自動通知を入手するには、Amazon EKS ドキュメントの履歴ページから、RSS フィードにサブスクライブしてください。

変更	説明	日付
AmazonEKS _CNI_Policy – 既存のポリシーへの更新	Amazon EKS は、Amazon VPC CNI plugin for Kubernetes が Amazon VPC サブネット内の空き IP アドレスの量を確認できるようにする新しい `ec2:DescribeSubnets` 許可を追加しました。 VPC CNI は各サブネットの空き IP アドレスを使用して、Elastic Network Interface の作成時に使用する空き IP アドレスが最も多いサブネットを選択できます。	2024 年 3 月 4 日
AmazonEKSWorkerNodePolicy – 既存のポリシーへの更新	Amazon EKS は EKS Pod Identity を許可する新しいアクセス権限を追加しました。 Amazon EKS Pod Identity エージェントはノードロールを使用します。	2023 年 11 月 26 日
AmazonEFSCSIDriverPolicy を導入しました。	AWS は `AmazonEFSCSIDriverPolicy` を導入しました。	2023 年 7 月 26 日
AmazonEKSClusterPolicy に権限を追加しました。	ロードバランサーを作成しながら、サブネットの自動検出中に Amazon EKS が AZ の詳細を取得できるようにする `ec2:DescribeAvailabilityZones` 許可が追加されました。	2023 年 2 月 7 日
AmazonEBSCSIDriverPolicy のポリシー条件が更新されました。	`StringLike` キーフィールドにワイルドカード文字を含む無効なポリシー条件を削除しました。また `ec2:Delet`	2022 年 11 月 17 日

変更	説明	日付
	eVolume に新しい条件 ec2:ResourceTag/kubernetes.io/created-for/pvc/name: "*" も追加され、ツリー内プラグインで作成されたボリュームを EBS CSI ドライバーが削除できるようになりました。	
AmazonEKSLocalOutpostServiceRolePolicy にアクセス許可を追加しました。	前提条件の検証とマネージドライフサイクルの管理が向上するように、ec2:DescribeVPCAttribute 、ec2:GetConsoleOutput 、ec2:DescribeSecret を追加しました。また、Outposts のコントロールプレーン Amazon EC2 インスタンスの配置制御をサポートするため、ec2:DescribePlacementGroups 、"arn:aws:ec2:*:*:placement-group/*"、ec2:RunInstances が追加しました。	2022 年 10 月 24 日
AmazonEKSLocalOutpostClusterPolicy の Amazon Elastic Container Registry のアクセス権限を更新します。	ecr:GetDownloadUrlForLayer アクションを、全リソースセクションからスコープされたセクションに移動しました。arn:aws:ecr:*:*:repository/eks/* リソースを追加しました。arn:aws:ecr:*:*:repository/eks/eks-certificates-controller-public リソースを削除しました。このリソースは、追加された arn:aws:ecr:*:*:repository/eks/* リソースでカバーします。	2022 年 10 月 20 日

変更	説明	日付
AmazonEKSLocalOutpostClusterPolicy にアクセス許可を追加しました。	クラスターコントロールプレーンインスタンスがいくつかの kubelet 引数を更新できるように、arn:aws:ecr:*:*:repository/kubelet-config-updater Amazon Elastic Container Registry リポジトリを追加しました。	2022 年 8 月 31 日
AmazonEKSLocalOutpostClusterPolicy を導入しました。	AWS は AmazonEKSLocalOutpostClusterPolicy を導入しました。	2022 年 8 月 24 日
AmazonEKSLocalOutpostServiceRolePolicy を導入しました。	AWS は AmazonEKSLocalOutpostServiceRolePolicy を導入しました。	2022 年 8 月 23 日
AmazonEBSCSIDriverPolicy を導入しました。	AWS は AmazonEBSCSIDriverPolicy を導入しました。	2022 年 4 月 4 日
AmazonEKSWorkerNodePolicy に権限を追加しました。	インスタンスレベルのプロパティを自動検出できる Amazon EKS に最適化された AMI を有効化する ec2:DescribeInstanceTypes を追加しました。	2022 年 3 月 21 日
AWSServiceRoleForAmazonEKSNodegroup に権限を追加しました。	Amazon EKS がメトリクスの収集を有効にすることを許可する autoscaling:EnableMetricsCollection アクセス許可を追加しました。	2021 年 12 月 13 日

変更	説明	日付
AmazonEKSClusterPolicy に権限を追加しました。	ec2:DescribeAccountAttributes 、ec2:DescribeAddresses 、ec2:DescribeInternetGateways の権限を追加し、Amazon EKS がNetwork Load Balancer のサービスにリンクされたロールを作成できるようになりました。	2021 年 6 月 17 日
Amazon EKS が変更の追跡を開始しました。	Amazon EKS が AWS 管理ポリシーの変更の追跡を開始しました。	2021 年 6 月 17 日

IAM のトラブルシューティング

このトピックでは、IAM での Amazon EKS の使用中に表示される一般的なエラーとその回避方法について説明します。

AccessDeniedException

AWS API オペレーションの呼び出し時に AccessDeniedException を受け取った場合、使用している IAM プリンシパルの認証情報には、その呼び出しに必要な許可がありません。

```
An error occurred (AccessDeniedException) when calling the DescribeCluster operation:
User: arn:aws:iam::111122223333:user/user_name is not authorized to perform:
eks:DescribeCluster on resource: arn:aws:eks:region:111122223333:cluster/my-cluster
```

前記のメッセージの例では、ユーザーには Amazon EKS DescribeCluster API オペレーションを呼び出す許可がありません。IAM プリンシパルに Amazon EKS 管理者の許可を付与するには、「Amazon EKS でのアイデンティティベースのポリシーの例」を参照してください。

IAM の一般的な情報については、IAM ユーザーガイドの「ポリシーを使用したアクセスの制御」を参照してください。

[Compute] (コンピューティング) タブに [Nodes] (ノード) が表示されず、また [Resources] (リソース) タブにも何も表示されず、AWS Management Console でエラーが表示される

`Your current user or role does not have access to Kubernetes objects on this EKS cluster`というコンソールのエラーメッセージが表示されることがあります。AWS Management Console を使用している IAM プリンシパルユーザーに必要な許可があることを確認してください。詳細については、「必要なアクセス許可」を参照してください。

aws-auth **`ConfigMap`** がクラスターへのアクセスを許可しない

AWS IAM Authenticator は、`ConfigMap` で使用されるロール ARN でパスを許可しません。したがって、`rolearn` を指定する前に、パスを削除してください。例えば、`arn:aws:iam::111122223333:role/team/developers/eks-admin` を `arn:aws:iam::111122223333:role/eks-admin` に変更します。

iam: PassRole を実行する権限がありません

`iam:PassRole` アクションを実行する権限がないというエラーが表示された場合は、ポリシーを更新して Amazon EKS にロールを渡せるようにする必要があります。

一部の AWS のサービスでは、新しいサービスロールまたはサービスリンクロールを作成せずに、既存のロールをサービスに渡すことが許可されています。そのためには、サービスにロールを渡す権限が必要です。

次の例のエラーは、`marymajor` という IAM ユーザーがコンソールを使用して Amazon EKS でアクションを実行しようとする場合に発生します。ただし、このアクションをサービスが実行するには、サービスロールから付与された権限が必要です。Mary には、ロールをサービスに渡す権限がありません。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

この場合、Mary のポリシーを更新してメアリーに `iam:PassRole` アクションの実行を許可する必要があります。

サポートが必要な場合は、AWS 管理者に問い合わせてください。管理者とは、サインイン認証情報を提供した担当者です。

AWS アカウント以外のユーザーに Amazon EKS リソースへのアクセスを許可したい

他のアカウントのユーザーや組織外の人が、リソースにアクセスするために使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまたはアクセス制御リスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください。

- Amazon EKS がこれらの機能をサポートしているかどうかについては、「Amazon EKS で IAM を使用する方法」を参照してください。
- 所有している AWS アカウント全体のリソースへのアクセス権を提供する方法については、「IAM ユーザーガイド」の「所有している別の AWS アカウントへのアクセス権を IAM ユーザーに提供」を参照してください。
- サードパーティーの AWS アカウント にリソースへのアクセス権を提供する方法については、『IAM ユーザーガイド』の「第三者が所有する AWS アカウント へのアクセス権を付与する」を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、『IAM ユーザーガイド』の「外部で認証されたユーザー (ID フェデレーション) へのアクセス権限」を参照してください。
- クロスアカウントアクセスでのロールとリソースベースのポリシーの使用の違いの詳細については、『IAM ユーザーガイド』の「IAM ロールとリソースベースのポリシーとの相違点」を参照してください。

ポッドコンテナは次のエラーを受け取ります: **An error occurred (SignatureDoesNotMatch) when calling the GetCallerIdentity operation: Credential should be scoped to a valid region**

アプリケーションが明示的に AWS STS グローバル エンドポイント (`https://sts.amazonaws`) にリクエストを送信し、Kubernetes サービス アカウントがリージョン エンドポイントを使用するように設定されている場合、コンテナはこのエラーを受け取ります。次のいずれかのオプションを使用して問題を解決できます。

- アプリケーションコードを更新して、AWS STS グローバルエンドポイントへの明示的な呼び出しを削除します。
- アプリケーションコードを更新して、`https://sts.us-west-2.amazonaws.com` などのリージョンエンドポイントへの明示的な呼び出しを行います。AWS リージョン 内のサービスに障害

が発生した場合に別の AWS リージョン を選択するよう、アプリケーションに冗長性が組み込まれている必要があります。詳細については、IAM ユーザーガイドの 「AWS リージョン での AWS STS の管理」を参照してください。

- グローバルエンドポイントを使用するようにサービスアカウントを設定します。1.22 より前のすべてのバージョンでは、デフォルトでグローバルエンドポイントを使用しましたが、バージョン 1.22 以降のクラスターでは、デフォルトでリージョンエンドポイントを使用します。詳細については、「サービスアカウントの AWS Security Token Service エンドポイントを設定する」を参照してください。

デフォルトの Amazon EKS は Kubernetes ロールとユーザーを作成しました。

Kubernetes クラスターを作成すると、Kubernetes が正常に機能するようにそのクラスターに複数のデフォルト Kubernetes ID が作成されます。Amazon EKS は、デフォルトコンポーネントごとに Kubernetes ID を作成します。ID は、クラスターコンポーネントの Kubernetes ロールベースの承認制御 (RBAC) を提供します。詳細については、「Kubernetes ドキュメント」の「RBAC 認証の使用」を参照してください。

オプションのアドオンをクラスターにインストールすると、クラスターに追加の Kubernetes ID が追加されることがあります。このトピックで扱われていない ID の詳細については、「アドオンのドキュメント」を参照してください。

AWS Management Console または `kubectl` コマンドラインツールを使用して、クラスターで Amazon EKS が作成した Kubernetes ID のリストを表示できます。すべてのユーザー ID はお客様に対して、Amazon CloudWatch を通じて利用可能な `kube` 監査ログに表示されます。

AWS Management Console

前提条件

使用する IAM プリンシパルには、必要なアクセス許可 に記載されている許可が必要です。

AWS Management Console を使用して Amazon EKS で作成された ID を表示するには

1. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。
2. [Clusters] (クラスター) のリストで、表示したい ID を含んでいるクラスターを選択します。
3. [リソース] タブを選択します。

4. [Resource types] (リソースタイプ) で [Authorization] (承認) を選択します。
5. [ClusterRoles]、[ClusterRoleBindings]、[Roles] (ロール)、または [RoleBindings] を選択します。[eks] というプレフィックスが付いたリソースはすべて、Amazon EKS によって作成されます。Amazon EKS で作成される追加の ID リソースは次のとおりです。

   - [aws-node] という名前の [ClusterRole] と [ClusterRoleBinding]。[aws-node] リソースは、Amazon EKS がすべてのクラスターにインストールする Amazon VPC CNI plugin for Kubernetes をサポートします。
   - [vpc-resource-controller-role] という名前の [ClusterRole] および [vpc-resource-controller-rolebinding] という名前の [ClusterRoleBinding]。これらのリソースは、Amazon EKS がすべてのクラスターにインストールする「Amazon VPC resource controller」(Amazon VPC リソースコントローラー) をサポートします。

   コンソールに表示されるリソースに加えて、次の特別なユーザー ID がクラスターに存在しますが、クラスターの設定には表示されません。

   - **`eks:cluster-bootstrap`** - クラスターブートストラップ中の `kubectl` 操作に使用されます。
   - **`eks:support-engineer`** - クラスター管理操作に使用されます。
6. 特定のリソースを選択すると、そのリソースの詳細が表示されます。デフォルトでは、情報は [Structured view] (構造化ビュー) で表示されます。詳細ページの右上隅で、[Raw View] (生のビュー) を選択すると、リソースの情報をすべて表示できます。

Kubectl

前提条件

クラスター上の Kubernetes リソースを一覧表示するために使用するエンティティ (AWS Identity and Access Management (IAM) または OpenID Connect (OIDC)) は、IAM または OIDC ID プロバイダーによって認証される必要があります。エンティティには、そのエンティティに使用させたいクラスター上の、`Role`、`ClusterRole`、`RoleBinding`、および `ClusterRoleBinding` リソースに対して Kubernetes `get` および `list` 動詞を使用するアクセス許可が付与されている必要があります。IAM エンティティにクラスターへのアクセスを付与する方法の詳細については、「the section called “Kubernetes API へのアクセスを許可する”」を参照してください。独自の OIDC プロバイダーによって認証されたエンティティにクラスターへのアクセスを付与する方法の詳細については、「OpenID Connect アイデンティティプロバイダーからクラスターのユーザーを認証する」を参照してください。

kubectl を使用して Amazon EKS で作成された ID を表示するには

表示するリソースのタイプのコマンドを実行します。[eks] というプレフィックスが付いたすべての返されるリソースは、Amazon EKS によって作成されます。コマンドからの出力で返されるリソースに加えて、次の特別なユーザー ID がクラスターに存在しますが、クラスターの設定には表示されません。

- **eks:cluster-bootstrap** - クラスターブートストラップ中の kubectl 操作に使用されます。
- **eks:support-engineer** - クラスター管理操作に使用されます。

ClusterRoles - ClusterRoles はクラスターにスコープが設定されているため、ロールに付与されたすべてのアクセス許可は、クラスター上の任意の Kubernetes 名前空間内のリソースに適用されます。

次のコマンドは、クラスターで作成されたすべての Amazon EKS Kubernetes ClusterRoles を返します。

```
kubectl get clusterroles | grep eks
```

出力で返されるのはプレフィックスが付けられた ClusterRoles の他に、次の ClusterRoles が存在します。

- **aws-node** - この ClusterRole は Amazon EKS がすべてのクラスターにインストールする Amazon VPC CNI plugin for Kubernetes をサポートします。
- **vpc-resource-controller-role** - この ClusterRole は、Amazon EKS がすべてのクラスターにインストールする「Amazon VPC resource controller」(Amazon VPC リソースコントローラー) をサポートします。

ClusterRole の仕様を確認するには、次のコマンドの *eks:k8s-metrics* を前のコマンドの出力で返された ClusterRole に置き換えます。次の例では、*eks:k8s-metrics* ClusterRole の仕様を返します。

```
kubectl describe clusterrole eks:k8s-metrics
```

出力例は次のとおりです。

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources         Non-Resource URLs  Resource Names  Verbs
  ---------         -----------------  --------------  -----
                    [/metrics]         []              [get]
  endpoints         []                 []              [list]
  nodes             []                 []              [list]
  pods              []                 []              [list]
  deployments.apps  []                 []              [list]
```

ClusterRoleBindings - `ClusterRoleBindings` はクラスターを対象としています。

次のコマンドは、クラスターで作成されたすべての Amazon EKS Kubernetes `ClusterRoleBindings` を返します。

```
kubectl get clusterrolebindings | grep eks
```

出力で返される `ClusterRoleBindings` 以外に、次の `ClusterRoleBindings` が存在します。

- **`aws-node`** - この `ClusterRoleBinding` は Amazon EKS がすべてのクラスターにインストールする Amazon VPC CNI plugin for Kubernetes をサポートします。
- **`vpc-resource-controller-rolebinding`** - この `ClusterRoleBinding` は、Amazon EKS がすべてのクラスターにインストールする「Amazon VPC resource controller」(Amazon VPC リソースコントローラー) をサポートします。

`ClusterRoleBinding` の仕様を確認するには、次のコマンドの *`eks:k8s-metrics`* を前のコマンドの出力で返された `ClusterRoleBinding` に置き換えます。次の例では、*`eks:k8s-metrics`* `ClusterRoleBinding` の仕様を返します。

```
kubectl describe clusterrolebinding eks:k8s-metrics
```

出力例は次のとおりです。

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
```

```
Role:
  Kind:  ClusterRole
  Name:  eks:k8s-metrics
Subjects:
  Kind  Name             Namespace
  ----  ----             ---------
  User  eks:k8s-metrics
```

ロール - Roles は Kubernetes 名前空間にスコープされます。Roles で作成されたすべての Amazon EKS は、kube-system 名前空間にスコープされます。

次のコマンドは、クラスターで作成されたすべての Amazon EKS Kubernetes Roles を返します。

```
kubectl get roles -n kube-system | grep eks
```

Role の仕様を確認するには、次のコマンドの *eks:k8s-metrics* を、前のコマンドの出力で返された Role の名前に置き換えます。次の例では、*eks:k8s-metrics* Role の仕様を返します。

```
kubectl describe role eks:k8s-metrics -n kube-system
```

出力例は次のとおりです。

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources         Non-Resource URLs  Resource Names              Verbs
  ---------         -----------------  --------------              -----
  daemonsets.apps   []                 [aws-node]                  [get]
  deployments.apps  []                 [vpc-resource-controller]  [get]
```

RoleBindings - RoleBindings は Kubernetes 名前空間にスコープされます。RoleBindings で作成されたすべての Amazon EKS は、kube-system 名前空間にスコープされます。

次のコマンドは、クラスターで作成されたすべての Amazon EKS Kubernetes RoleBindings を返します。

```
kubectl get rolebindings -n kube-system | grep eks
```

RoleBinding の仕様を確認するには、次のコマンドの *eks:k8s-metrics* を前のコマンドの出力で返された RoleBinding に置き換えます。次の例では、*eks:k8s-metrics* RoleBinding の仕様を返します。

```
kubectl describe rolebinding eks:k8s-metrics -n kube-system
```

出力例は次のとおりです。

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  Role
  Name:  eks:k8s-metrics
Subjects:
  Kind  Name             Namespace
  ----  ----             ---------
  User  eks:k8s-metrics
```

Amazon Elastic Kubernetes Service でのコンプライアンス検証

AWS のサービス が特定のコンプライアンスプログラムの範囲内にあるかどうかを確認するには、「コンプライアンスプログラム別の範囲」の「AWS のサービス」と「」の「AWS のサービス」を参照し、関心のあるコンプライアンスプログラムを選択してください。一般的な情報については、AWS コンプライアンスプログラム を参照してください。

AWS Artifact を使用して、サードパーティーの監査レポートをダウンロードできます。詳細については、「Downloading Reports in AWS Artifact」を参照してください。

AWS のサービスを使用する際のユーザーのコンプライアンス責任は、ユーザーのデータの機密性や貴社のコンプライアンス目的、適用される法律および規制によって決まります。AWS では、コンプライアンスに役立つ次のリソースを提供しています。

- セキュリティとコンプライアンスのクイックスタートガイド - これらのデプロイガイドでは、アーキテクチャ上の考慮事項について説明し、セキュリティとコンプライアンスに重点を置いたベースライン環境を AWS にデプロイするためのステップを示します。
- 「Amazon Web Services での HIPAA のセキュリティとコンプライアンスのためのアーキテクチャ」 - このホワイトペーパーは、企業が AWS を使用して HIPAA 対象アプリケーションを作成する方法を説明しています。

> **Note**
>
> すべての AWS のサービスが HIPAA 適格であるわけではありません。詳細については、「HIPAA 対応サービスのリファレンス」を参照してください。

- AWS コンプライアンスのリソース – このワークブックおよびガイドのコレクションは、顧客の業界と拠点に適用されるものである場合があります。
- AWS Customer Compliance Guide — コンプライアンスの観点から見た責任共有モデルを理解できます。このガイドは、AWS のサービスを保護するためのベストプラクティスを要約したものであり、複数のフレームワーク (米国標準技術研究所 (NIST)、ペイメントカード業界セキュリティ標準評議会 (PCI)、国際標準化機構 (ISO) など) にわたるセキュリティ統制へのガイダンスがまとめられています。
- 「AWS Config デベロッパーガイド」の「ルールでのリソースの評価」 - AWS Config サービスは、自社のプラクティス、業界ガイドライン、および規制に対するリソースの設定の準拠状態を評価します。
- AWS Security Hub - この AWS のサービスは、AWS 内のセキュリティ状態の包括的なビューを提供します。Security Hub では、セキュリティコントロールを使用して AWS リソースを評価し、セキュリティ業界標準とベストプラクティスに対するコンプライアンスをチェックします。サポートされているサービスとコントロールのリストについては、「Security Hub のコントロールリファレンス」を参照してください。
- Amazon GuardDuty – この AWS のサービスは、環境をモニタリングして、疑わしいアクティビティや悪意のあるアクティビティがないか調べることで、AWS アカウント、ワークロード、コンテナ、データに対する潜在的な脅威を検出します。GuardDuty を使用すると、特定のコンプライアンスフレームワークで義務付けられている侵入検知要件を満たすことで、PCI DSS などのさまざまなコンプライアンス要件に対応できます。
- AWS Audit Manager - この AWS のサービスは、AWS の使用状況を継続的に監査して、リスクの管理方法や、規制および業界標準へのコンプライアンスの管理方法を簡素化するために役立ちます。

Amazon EKS の耐障害性

AWS グローバルインフラストラクチャは AWS リージョン およびアベイラビリティーゾーンを中心に構築されています。AWS リージョン には、低レイテンシー、高いスループット、そして高度の冗長ネットワークで接続されている複数の物理的に独立・隔離されたアベイラビリティーゾーンがあります。アベイラビリティーゾーンでは、アベイラビリティーゾーン間で中断せずに、自動的にフェイ

ルオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャに比べて、可用性、耐障害性、および拡張性に優れています。

Amazon EKS は、複数の AWS アベイラビリティーゾーンで Kubernetes コントロールプレーンを実行およびスケーリングして、高可用性を実現します。Amazon EKS は、負荷に基づいてコントロールプレーンインスタンスを自動的にスケーリングします。また、異常なコントロールプレーンインスタンスを自動的に検出して置換し、そのコントロールプレーンに自動的にパッチを適用します。バージョン更新を開始すると、Amazon EKS によってコントロールプレーンが自動的に更新され、更新中のコントロールプレーンの高可用性が維持されます。

このコントロールプレーンは、少なくとも 2 つの API サーバーインスタンスと、(1 つの AWS リージョン で 3 つのアベイラビリティーゾーンにまたがっている) 3 つの `etcd` インスタンスを含みます。Amazon EKS:

- コントロールプレーンインスタンスの負荷をアクティブに監視し、高パフォーマンスを確保するために、それらのインスタンスを自動的にスケーリングします。
- 異常なコントロールプレーンインスタンスを自動的に検出して置換します。それらのコントロールプレーンインスタンスは、必要に応じて AWS リージョン 内のアベイラビリティーゾーンで再起動されます。
- 高可用性を維持するために、AWS リージョン のアーキテクチャを活用します。そのため、Amazon EKS では API サーバーエンドポイント向けの可用性の SLA をご利用いただけます。

AWS リージョン とアベイラビリティーゾーンの詳細については、「AWS グローバルインフラストラクチャ」を参照してください。

Amazon EKS でのインフラストラクチャセキュリティ

マネージドサービスの Amazon Elastic Kubernetes Service は、AWS グローバルネットワークセキュリティによって保護されています。AWS セキュリティサービスと AWS がインフラストラクチャを保護する方法については、「AWS クラウドセキュリティ」を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して AWS 環境を設計するには、「セキュリティの柱 - AWS Well-Architected フレームワーク」の「インフラストラクチャ保護」を参照してください。

AWS が公開している API コールを使用し、ネットワーク経由で Amazon EKS にアクセスします。クライアントは以下をサポートする必要があります:

- Transport Layer Security (TLS)。TLS 1.2、できれば TLS 1.3 が必要です。

- DHE (Ephemeral Diffie-Hellman) や ECDHE (Elliptic Curve Ephemeral Diffie-Hellman) などの Perfect Forward Secrecy (PFS) を使用した暗号スイート。これらのモードは、Java 7 以降など、ほとんどの最新システムでサポートされています。

また、リクエストには、アクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または、AWS Security Token Service (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

Amazon EKS のクラスターの作成時には、使用するクラスターの VPC サブネットを指定します。Amazon EKS には、2 つ以上のアベイラビリティーゾーンにサブネットが必要です。パブリックとプライベートのサブネットを持つ VPC をお勧めします。これにより、Kubernetes がパブリックサブネットにパブリックロードバランサーを作成し、プライベートサブネットにあるノードで実行されている Pods へのトラフィックを負荷分散します。

VPC に関する考慮事項の詳細については、「」を参照してくださいAmazon EKS VPC およびサブネットの要件と考慮事項

チュートリアル Amazon EKS の使用開始 で提供されている AWS CloudFormation テンプレートを使用して VPC およびノードグループを作成した場合、コントロールプレーンとノードのセキュリティグループは推奨設定で構成されます。

セキュリティグループの考慮事項の詳細については、「」を参照してくださいAmazon EKS セキュリティグループの要件および考慮事項

新しいクラスターを作成すると、Amazon EKS によってマネージド型の Kubernetes API サーバーのエンドポイントが作成されます。ユーザーはこのエンドポイントを、(`kubectl` などの Kubernetes 管理ツールを通じて) クラスターとの通信に使用します。デフォルトでは、この API サーバーエンドポイントはインターネットに公開され、API サーバーへのアクセスは、AWS Identity and Access Management (IAM) とネイティブの Kubernetes Role Based Access Control (RBAC) の組み合わせを使用して保護されます。

Kubernetes API サーバーへのプライベートアクセスを有効にすると、ノードと API サーバー間のすべての通信が VPC 内で行われるようにできます。インターネットから API サーバーにアクセスできる IP アドレスを制限したり、API サーバーへのインターネットアクセスを完全に無効にしたりできます。

クラスターエンドポイントアクセスの変更の詳細については、「」を参照してくださいクラスターエンドポイントのアクセスの変更

Amazon VPC CNI または Project Calico などのサードパーティー製ツールを使用して Kubernetes ネットワークポリシーを実装できます。Amazon VPC CNI をネットワークポリシーに使用する方法の詳細については、「Kubernetes ネットワークポリシーのクラスターを構成する」を参照してください。プロジェクト Calico はサードパーティのオープンソースプロジェクトです。詳細については、「プロジェクト Calico のドキュメント」を参照してください。

Amazon EKS での設定と脆弱性の分析

セキュリティは、Kubernetes クラスターとアプリケーションを設定および保守するための重要な考慮事項です。以下に、EKS クラスターのセキュリティ設定を分析するためのリソース、脆弱性をチェックするためのリソース、およびその分析を実行できる AWS サービスとの統合を示します。

Amazon EKS 用 Center for Internet Security (CIS) ベンチマーク

Center for Internet Security (CIS) Kubernetes ベンチマーク は、Amazon EKS のセキュリティ設定に関するガイダンスを提供します。ベンチマーク:

- Kubernetes コンポーネントのセキュリティ設定を担当する Amazon EC2 ノード (マネージドとセルフマネージドの両方) に適用できます。
- Amazon EKS を使用するときに、Kubernetes クラスターとノードを安全に設定したことを確認するための標準のコミュニティ承認済みの方法を提供します。
- これは、コントロールプレーンでのログ記録の設定、ノードのセキュリティ設定、ポリシー、マネージド型サービスの 4 つのセクションで構成されます。
- Amazon EKS で現在利用可能なすべての Kubernetes バージョンをサポートし、Kubernetes クラスターで CIS ベンチマークを使用して設定を確認するための標準のオープンソースツールである kube-bench を使用して実行できます。

詳細については、「Introducing The CIS Amazon EKS Benchmark (CIS Amazon EKSベンチマークの紹介)」を参照してください。

Amazon EKS のプラットフォームバージョン

Amazon EKS プラットフォームのバージョンは、クラスターコントロールプレーンの機能を表しています。これには、Kubernetes API サーバーでどのフラグが有効であるかや、現在の Kubernetes パッチバージョンなどの情報が含まれます。新しいクラスターは、最新のプラットフォームバージョンでデプロイされます。詳細については、「Amazon EKS のプラットフォームバージョン」を参照してください。

新しい Kubernetes バージョンに Amazon EKS クラスター を更新することができます。新しい Kubernetes バージョンが Amazon EKS で利用可能になったら、利用可能な最新のバージョンが使用できるよう、クラスターをタイムリーに更新することをお勧めします。EKS での Kubernetes のバージョンの詳細については、「Amazon EKS Kubernetes のバージョン」を参照してください。

オペレーティングシステムの脆弱性リスト

AL2023 脆弱性リスト

Amazon Linux セキュリティセンター で Amazon Linux 2023 のセキュリティまたはプライバシーイベントを追跡するか、関連する RSS フィードを購読します。セキュリティおよびプライバシーイベントには、影響を受ける問題の概要、パッケージ、および問題を修正するためにインスタンスを更新する手順などがあります。

Amazon Linux 2 脆弱性リスト

Amazon Linux セキュリティセンター で Amazon Linux 2 のセキュリティまたはプライバシーイベントを追跡するか、関連する RSS フィードをサブスクライブします。セキュリティおよびプライバシーイベントには、影響を受ける問題の概要、パッケージ、および問題を修正するためにインスタンスを更新する手順などがあります。

Amazon Inspector によるノード検出

Amazon Inspector を使用すると、ノードからネットワークに意図しない接続が可能か、また、それらの Amazon EC2 インスタンスに脆弱性があるかを確認できます。

Amazon GuardDuty によるクラスターとノードの検出

Amazon GuardDuty は、AWS 環境内のアカウント、コンテナ、ワークロード、データを保護する脅威検知サービスです。GuardDuty には、EKS クラスターに対する潜在的な脅威を検出する EKS Protection とランタイムモニタリングの 2 つの機能があります。

詳細については、「Amazon GuardDuty で脅威を検出する」を参照してください。

Amazon EKS のセキュリティベストプラクティス

Amazon EKS のセキュリティのベストプラクティスは Github: https://aws.github.io/aws-eks-best-practices/security/docs/ で公開されています。

ポッドのセキュリティポリシー

Kubernetes Pod のセキュリティポリシーのアドミッションコントローラーは、一連のルールに対して Pod の作成を検証し、リクエストを更新します。デフォルトでは、Amazon EKS クラスターには、制限のない、完全な許容度のセキュリティポリシーが設定されています。詳細については、「Kubernetes のドキュメント」の「ポッドセキュリティポリシー」を参照してください。

> **Note**
>
> PodSecurityPolicy (PSP) は Kubernetes バージョン 1.21 で非推奨となり、Kubernetes 1.25 で削除されました。PSPs は「ポッドセキュリティ標準 (PSS)」で概説されているセキュリティ制御を実装する組み込みアドミッションコントローラーである「ポッドセキュリティアドミッション (PSA)」に置き換えられています。PSA および PSS はどちらもベータ機能状態に達しており、Amazon EKS ではデフォルトで有効になっています。1.25 から PSP の削除に対処するには、Amazon EKS に PSS を実装することをお勧めします。詳細については、「AWS ブログ」の「Implementing Pod Security Standards in Amazon EKS」(Amazon EKS でのポッドセキュリティ標準の実装) を参照してください。

Amazon EKS での デフォルトの Pod セキュリティポリシー

Kubernetes バージョン 1.13 以降の Amazon EKS クラスターには、eks.privileged という名前のデフォルト Pod のセキュリティポリシーがあります。このポリシーには、システムに受け入れ可能な種類の Pod について制限がありません。これは、PodSecurityPolicy コントローラーを無効にして Kubernetes を実行するのと同じです。

> **Note**
>
> このポリシーは、PodSecurityPolicy コントローラーが有効になっていないクラスターとの下位互換性を維持するために作成されたものです。クラスターと個別名前空間およびサービスアカウントに対してより制限の高いポリシーを作成してから、デフォルトポリシーを削除し、より制限の高いポリシーを有効にすることができます。

次のコマンドを使用してデフォルトのポリシーを表示できます。

```
kubectl get psp eks.privileged
```

出力例は次のとおりです。

```
NAME              PRIV    CAPS    SELINUX    RUNASUSER    FSGROUP    SUPGROUP
 READONLYROOTFS   VOLUMES
eks.privileged    true    *       RunAsAny   RunAsAny     RunAsAny   RunAsAny    false
      *
```

詳細については、次のコマンドを使用してポリシーについて説明できます。

```
kubectl describe psp eks.privileged
```

出力例は次のとおりです。

```
Name:  eks.privileged

Settings:
  Allow Privileged:                       true
  Allow Privilege Escalation:             0xc0004ce5f8
  Default Add Capabilities:               <none>
  Required Drop Capabilities:             <none>
  Allowed Capabilities:                   *
  Allowed Volume Types:                   *
  Allow Host Network:                     true
  Allow Host Ports:                       0-65535
  Allow Host PID:                         true
  Allow Host IPC:                         true
  Read Only Root Filesystem:              false
  SELinux Context Strategy: RunAsAny
    User:                                 <none>
    Role:                                 <none>
    Type:                                 <none>
    Level:                                <none>
  Run As User Strategy: RunAsAny
    Ranges:                               <none>
  FSGroup Strategy: RunAsAny
    Ranges:                               <none>
  Supplemental Groups Strategy: RunAsAny
    Ranges:                               <none>
```

eks.privileged Pod セキュリティポリシー、そのクラスターロール、およびクラスターロールバインドに対する完全な YAML ファイルを デフォルトの Pod セキュリティポリシーをインストールまたは復元する で確認できます。

デフォルトの Amazon EKS Pod セキュリティポリシーを削除する

Pods に対してより制限の厳しいポリシーを作成する場合は、作成後にデフォルトの Amazon EKS eks.privileged Pod セキュリティポリシーを削除して、カスタムポリシーを有効にできます。

> **Important**
>
> CNI プラグインのバージョン 1.7.0 以降を使用していて、Daemonset によってデプロイされた aws-node Pods に使用した aws-node Kubernetes サービスアカウントにカスタム Pod セキュリティポリシーを割り当てる場合、ポリシーの allowedCapabilities セクションには NET_ADMIN が、またポリシーの spec には hostNetwork: true と privileged: true があることが必要です。

デフォルトの Pod セキュリティポリシーを削除するには

1. デフォルトの Pod セキュリティポリシーをインストールまたは復元する のサンプルファイルの内容を使用して、*privileged-podsecuritypolicy.yaml* という名前のファイルを作成します。
2. 次のコマンドを使用して YAML を削除します。これにより、デフォルトの Pod セキュリティポリシー、ClusterRole、およびそれに関連付けられた ClusterRoleBinding が削除されます。

```
kubectl delete -f privileged-podsecuritypolicy.yaml
```

デフォルトの Pod セキュリティポリシーをインストールまたは復元する

Kubernetes の以前のバージョンからアップグレードする場合、またはデフォルトの Amazon EKS eks.privileged Pod セキュリティポリシーを変更または削除した場合は、次のステップで復元できます。

デフォルトの Pod セキュリティポリシーをインストールまたは復元するには

1. *privileged-podsecuritypolicy.yaml* というファイルを次の内容で作成します。

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
```

```
  name: eks.privileged
  annotations:
    kubernetes.io/description: 'privileged allows full unrestricted access to
      Pod features, as if the PodSecurityPolicy controller was not enabled.'
    seccomp.security.alpha.kubernetes.io/allowedProfileNames: '*'
  labels:
    kubernetes.io/cluster-service: "true"
    eks.amazonaws.com/component: pod-security-policy
spec:
  privileged: true
  allowPrivilegeEscalation: true
  allowedCapabilities:
  - '*'
  volumes:
  - '*'
  hostNetwork: true
  hostPorts:
  - min: 0
    max: 65535
  hostIPC: true
  hostPID: true
  runAsUser:
    rule: 'RunAsAny'
  seLinux:
    rule: 'RunAsAny'
  supplementalGroups:
    rule: 'RunAsAny'
  fsGroup:
    rule: 'RunAsAny'
  readOnlyRootFilesystem: false

---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: eks:podsecuritypolicy:privileged
  labels:
    kubernetes.io/cluster-service: "true"
    eks.amazonaws.com/component: pod-security-policy
rules:
- apiGroups:
  - policy
  resourceNames:
  - eks.privileged
```

```
  resources:
  - podsecuritypolicies
  verbs:
  - use

---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks:podsecuritypolicy:authenticated
  annotations:
    kubernetes.io/description: 'Allow all authenticated users to create privileged
 Pods.'
  labels:
    kubernetes.io/cluster-service: "true"
    eks.amazonaws.com/component: pod-security-policy
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: eks:podsecuritypolicy:privileged
subjects:
  - kind: Group
    apiGroup: rbac.authorization.k8s.io
    name: system:authenticated
```

2. 次のコマンドを使用して YAML を適用します。

```
kubectl apply -f privileged-podsecuritypolicy.yaml
```

ポッドセキュリティポリシー (PSP) の削除に関するよくある質問

PodSecurityPolicy は Kubernetes1.21 で非推奨となり、Kubernetes1.25 で削除されました。クラスターで PodSecurityPolicy を使用している場合は、ワークロードの中断を避けるために、クラスターをバージョン **1.25** にアップグレードする前に、組み込みの Kubernetes ポッドセキュリティ標準 (PSS) または Policy as Code ソリューションに移行する必要があります。詳細については、よくある質問を選択してください。

PSP について

PodSecurityPolicy は、クラスター管理者が Pod 仕様のセキュリティセンシティブな側面を制御できるようにする組み込みのアドミッションコントローラーです。Pod がその PSP の要件を満たしてい

る場合、その Pod は通常どおりクラスターに受け入れられます。Pod が PSP の要件を満たしていない場合、Pod は拒否され、実行できません。

PSP の削除は Amazon EKS に固有のものですか? それともアップストリーム Kubernetes で削除されていますか?

これは Kubernetes プロジェクトにおけるアップストリームの変更であり、Amazon EKS で行われた変更ではありません。PSP は、Kubernetes `1.21` で非推奨となり、Kubernetes `1.25` で削除されました。Kubernetes コミュニティは、PSP の深刻なユーザビリティの問題を特定しました。これらには、意図したよりも広範な許可の誤った付与や、特定の状況で PSPs が適用する検査における困難が含まれます。これらの問題は、重大な変更を加えることなく対処することができませんでした。これを主な理由として、Kubernetes コミュニティは PSP を削除することを決定しました。

Amazon EKS クラスターで PSPs を使用しているかどうかを確認するにはどうすればよいですか?

クラスターで PSPs を使用しているかどうかを確認するために、次のコマンドを実行できます。

```
kubectl get psp
```

クラスター内の PSPs が影響を与えている Pods を確認するには、次のコマンドを実行します。このコマンドは Pod 名、名前空間、PSPs を出力します。

```
kubectl get pod -A -o jsonpath='{range.items[?(@.metadata.annotations.kubernetes
\.io/psp)]}{.metadata.name}{"\t"}{.metadata.namespace}{"\t"}
{.metadata.annotations.kubernetes\.io/psp}{"\n"}'
```

Amazon EKS クラスターで PSPs を使用している場合、どうすればよいですか?

クラスターを `1.25` にアップグレードする前に、PSPs を次のいずれかに移行する必要があります。

- Kubernetes PSS.
- Kubernetes 環境からの Policy as Code ソリューション。

PSP が非推奨となったことや、最初から Pod セキュリティを制御することに対する継続的なニーズに対応して、Kubernetes コミュニティは (PSS) と Pod Security Admission (PSA) を使用して組み込

みソリューションを作成しました。PSA ウェブフックは、PSS で定義されているコントロールを実装します。

組み込みの PSS に PSPs を移行するためのベストプラクティスは、「EKS ベストプラクティスガイド」で確認できます。また、「Amazon EKS でのポッドセキュリティ標準の実装」のブログを確認することをお勧めします。追加のリファレンスには、「PodSecurityPolicy から組み込み PodSecurity アドミッションコントローラーへの移行」、および「PodSecurityPolicies からポッドセキュリティ標準へのマッピング」が含まれます。

Policy as Code ソリューションは、クラスターユーザーをガイドするガードレールを提供し、規定の自動化されたコントロールを通じて望ましくない動作を防止します。Policy as Code ソリューションは通常、Kubernetes ダイナミックアドミッションコントローラーを使用して、ウェブフック呼び出しで Kubernetes API サーバーリクエストのフローをインターセプトします。Policy as Code ソリューションは、コードとして記述および保存されたポリシーに基づいて、リクエストペイロードを変更および検証します。

Kubernetes で利用できるオープンソースの Policy as Code ソリューションがいくつかあります。PSPs を Policy as Code ソリューションに移行するためのベストプラクティスを確認するには、GitHub のポッドセキュリティのページの「Policy as Code」セクションを参照してください。

クラスターで PSP が **eks.privileged** を呼び出しました。これは何ですか? そして、これについてどのように対応すればよいですか?

Kubernetes バージョン `1.13` 以降の Amazon EKS クラスターには、`eks.privileged` という名前のデフォルトの PSP があります。このポリシーは、`1.24` および以前のクラスターで作成されます。`1.25` 以降のクラスターでは使用されません。Amazon EKS は、この PSP を PSS ベースの適用に自動的に移行します。お客様側でのご対応は不要です。

クラスターをバージョン **1.25** に更新すると、Amazon EKS は既存のクラスターに存在する PSPs に対して変更を加えますか?

いいえ。Amazon EKS によって作成された PSP である `eks.privileged` に加えて、`1.25` にアップグレードしても、クラスター内の他の PSPs は変更されません。

PSP から移行していない場合、クラスターのバージョン **1.25** への更新は Amazon EKS によって妨げられますか?

いいえ。まだ PSP から移行していない場合でも、クラスターのバージョン `1.25` への更新が Amazon EKS によって妨げられることはありません。

クラスターをバージョン 1.25 に更新する前に、PSPs を PSS/PSA または Policy as Code ソリューションに移行するのを忘れた場合はどうなりますか? クラスターを更新した後に移行できますか?

PSP を含むクラスターが Kubernetes バージョン 1.25 にアップグレードされると、API サーバーは 1.25 の PSP リソースを認識しません。これにより、Pods が誤ったセキュリティスコープを取得する可能性があります。影響の詳細なリストについては、「PodSecurityPolicy から組み込みの PodSecurity アドミッションコントローラーに移行する」を参照してください。

この変更は、Windows ワークロードのポッドセキュリティにどのように影響しますか?

Windows ワークロードに対する特定の影響はないと想定されています。PodSecurityContext は、Windows Pods の PodSpec v1 API で windowsOptions というフィールドを持っています。これは Kubernetes 1.25 で PSS を使用します。Windows ワークロードの PSS の適用に関する詳細とベストプラクティスについては、「EKS ベストプラクティスガイド」と Kubernetes ドキュメントを参照してください。

Kubernetes で AWS Secrets Manager シークレットを使用する

Secrets Manager のシークレットと Parameter Store のパラメータを、Amazon EKS Pods にマウントされたファイルとして表示するには、Kubernetes Secrets Store CSI Driver 向けの AWS Secrets and Configuration Provider (ASCP) を使用します。

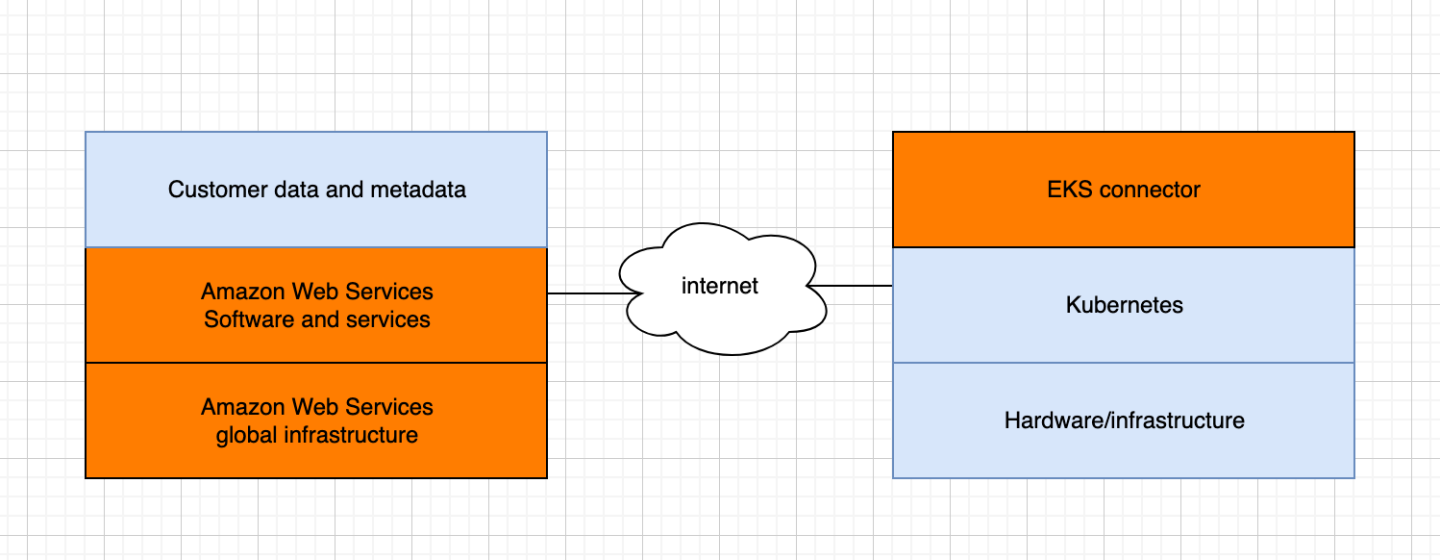
ASCP を使用すると、Secrets Manager でシークレットを保存および管理し、Amazon EKS で実行されているワークロードからシークレットを取得できます。IAM ロールとポリシーを使用して、シークレットへのアクセスをクラスター内の特定の Kubernetes Pods に制限できます。ASCP は Pod アイデンティティを取得し、そのアイデンティティを IAM ロールと交換します。ASCP が Pod の IAM ロールを引き受けると、そのロールに対して認証されている Secrets Manager からシークレットを取得できます。

自分のシークレットに対し、Secrets Manager の自動ローテーションを使用する場合は、Secrets Store CSI ドライバーのローテーション調停機能を使用することで、Secrets Manager から確実に最新のシークレットを取得できます。

詳細については、AWS Secrets Manager ユーザーガイドの「Using Secrets Manager secrets in Amazon EKS」を参照してください。

Amazon EKS Connector の考慮事項

Amazon EKS Connector は、Kubernetes クラスターで実行されるオープンソースコンポーネントです。このクラスターは、AWS 環境の外部に配置できます。これにより、セキュリティ上の責任に関する追加の考慮事項が作成されます。この設定については、次の図で示されています。オレンジ色は AWS の責任、青色はお客様の責任を表します。



このトピックでは、接続されているクラスターが AWS の外部にある場合の責任モデルの違いについて説明します。

AWS の責任

- お客様の Kubernetes クラスターで実行され、AWS と通信する オープンソースコンポーネント である Amazon EKS Connector の保守、構築および提供。
- 接続されている Kubernetes クラスターと AWS のサービス間のトランスポートおよびアプリケーションレイヤーの通信セキュリティの維持。

お客様の責任

- Kubernetes クラスター固有のセキュリティ、具体例には以下の通りです。
  - Kubernetes シークレットは適切に暗号化され、保護されている必要があります。
  - `eks-connector` 名前空間へのアクセスをロックダウンします。

- AWS からの IAM プリンシパルのアクセスを管理するロールベースのアクセスコントロール (RBAC) 許可の設定。手順については、「クラスター上の Kubernetes リソースを表示するためのアクセス権の IAM プリンシパルへの付与」を参照してください。
- Amazon EKS Connector のインストールおよびアップグレード。
- 接続された Kubernetes クラスターをサポートするハードウェア、ソフトウェア、およびインフラストラクチャを維持します。
- (例えばルートユーザーの認証情報を保護することを通じて) AWS アカウントを保護します。

Kubernetes リソースを表示する

クラスターにデプロイされた Kubernetes リソースを AWS Management Console で表示できます。Kubernetes リソースは、AWS CLI または `eksctl` で表示することはできません。コマンドラインツールを使用して Kubernetes リソースを表示するには、`kubectl` を使用します。

前提条件

AWS Management Console の [リソース] タブと [コンピューティング] タブの [ノード] セクションを表示するには、使用している IAM プリンシパルに特定の IAM と Kubernetes 許可が必要です。詳細については、「必要なアクセス許可」を参照してください。

Kubernetes リソースを AWS Management Console で表示する

1. Amazon EKS コンソールを https://console.aws.amazon.com/eks/home#/clusters で開きます。
2. [クラスター] のリストで、表示したい Kubernetes リソースを含んでいるクラスターを選択します。
3. [リソース] タブを選択します。
4. リソースを表示したい [リソースタイプ] グループを選択します。例えば、[ワークロード] などです。そのグループ内のリソースタイプのリストが表示されます。
5. リソースタイプを選択します。つまり、[ワークロード] グループ内の [デプロイ] などです。リソースタイプの説明、リソースタイプの詳細な説明の Kubernetes ドキュメントへのリンク、およびクラスターにデプロイされていてそのタイプのリソースのリストが表示されます。リストが空の場合、そのタイプのリソースはクラスターにデプロイされていません。
6. 詳細情報を表示するには、そのリソースを選択します。次の例を試してください。
   - [ワークロード] グループを選択し、[デプロイメント] のリソースタイプを選択し、次に [CoreDNS] リソースを選択します。リソースを選択すると、デフォルトで [構造化ビュー] が表示されます。リソースタイプによっては、[構造化ビュー] に [ポッド] セクションが表示されます。このセクションでは、ワークロードによって管理される Pods が一覧表示されます。一覧表示されている任意の Pod を選択し、Pod の情報を表示できます。すべてのリソースタイプについて、[構造化ビュー] で情報が表示されるわけではありません。リソース表示画面で右上隅にある [raw ビュー] を選択すると、そのリソースについて Kubernetes API からの完全な JSON レスポンスが表示されます。
   - [クラスター] グループを選択してから、[ノード] リソースタイプを選択します。クラスター内のすべてのノードのリストが表示されます。ノードは、任意の Amazon EKS ノードタイプと

なります。これはクラスターの [コンピューティング] () タブを選択したときに [ノード] セクションに表示されるのと同じリストです。リストからノードリソースを選択します。[構造化ビュー] では、[ポッド] セクションも表示されます。このセクションでは、ノード上で実行中のすべての Pods が表示されます。

必要なアクセス許可

AWS Management Console の [リソース] タブと [コンピューティング] タブの [ノード] セクションを表示するには、使用している IAM プリンシパルに特定の最小限の IAM と Kubernetes 許可が必要です。次のステップを実行して、IAM プリンシパルに必要な許可を割り当てます。

1. `eks:AccessKubernetesApi`、および Kubernetes リソースを表示するために必要なその他の IAM アクセス許可が、使用している IAM プリンシパルに割り当てられていることを確認してください。IAM プリンシパルの許可を編集する方法の詳細については、「IAM ユーザーガイド」の「プリンシパルへのアクセスの制御」を参照してください。ロールのアクセス許可を編集する方法の詳細については、IAM ユーザーガイドの「ロールのアクセス許可ポリシーの変更 (コンソール)」を参照してください。

   次のポリシーの例には、アカウント内のすべてのクラスター内にある Kubernetes リソースを表示するために必要な、プリンシパルへの許可が含まれています。*111122223333* を AWS アカウント ID に置き換えます。

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": [
                   "eks:ListFargateProfiles",
                   "eks:DescribeNodegroup",
                   "eks:ListNodegroups",
                   "eks:ListUpdates",
                   "eks:AccessKubernetesApi",
                   "eks:ListAddons",
                   "eks:DescribeCluster",
                   "eks:DescribeAddonVersions",
                   "eks:ListClusters",
                   "eks:ListIdentityProviderConfigs",
                   "iam:ListRoles"
               ],
   ```

```
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ssm:GetParameter",
            "Resource": "arn:aws:ssm:*:111122223333:parameter/*"
        }
    ]
}
```

接続されたクラスター内のノードを表示するには、Amazon EKS コネクタの IAM ロールがクラスター内のプリンシパルを偽装できる必要があります。これは、Amazon EKS Connector がプリンシパルを Kubernetes ユーザーにマッピングすることを許可します。

2. Kubernetes リソースを表示するのに必要なアクセス許可を持つ Kubernetes `role` または `clusterrole` にバインドされている Kubernetes `rolebinding` または `clusterrolebinding` を作成します。Kubernetes のロールとロールバインドの詳細については、「Kubernetes ドキュメント」の「RBAC 認可を使用する」を参照してください。次のマニフェストのいずれかをクラスターに適用して、必要な Kubernetes アクセス許可を持つ `role` および `rolebinding` または `clusterrole` および `clusterrolebinding` を作成できます。

   すべての名前空間の Kubernetes リソースを表示する

   ファイル内のグループ名は `eks-console-dashboard-full-access-group` です。次のコマンドを使用して、クラスターにマニフェストを適用します。

   ```
   kubectl apply -f https://s3.us-west-2.amazonaws.com/amazon-eks/docs/eks-console-full-access.yaml
   ```

   特定の名前空間の Kubernetes リソースを表示する

   このファイルの名前空間は `default` です。ファイル内のグループ名は `eks-console-dashboard-restricted-access-group` です。次のコマンドを使用して、クラスターにマニフェストを適用します。

   ```
   kubectl apply -f https://s3.us-west-2.amazonaws.com/amazon-eks/docs/eks-console-restricted-access.yaml
   ```

ファイル内の Kubernetes グループ名、名前空間、アクセス許可、またはその他の設定を変更する必要がある場合は、ファイルをダウンロードして編集してからクラスターに適用します。

1. 次のいずれかのコマンドを使って、ファイルをダウンロードします。

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/docs/eks-console-full-access.yaml
```

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/docs/eks-console-restricted-access.yaml
```

2. 必要に応じてファイルを編集します。
3. 次のコマンドのいずれかを使用して、クラスターにマニフェストを適用します。

```
kubectl apply -f eks-console-full-access.yaml
```

```
kubectl apply -f eks-console-restricted-access.yaml
```

3. aws-auth ConfigMap の Kubernetes ユーザーまたはグループに IAM プリンシパルをマッピングします。eksctl のようなツールにより ConfigMap を更新することができます。あるいは、手動で編集しての更新も可能です。

> **Important**
>
> ConfigMap の編集には、eksctl や、その他のツールを使用することをお勧めします。使用できる他のツールについては、Amazon EKS ベストプラクティスガイドの「ツールを使用して aws-authConfigMap を変更する」を参照してください。aws-auth ConfigMap の形式が不適切な場合、クラスターへのアクセスが失われる場合があります。

eksctl

前提条件

デバイスまたは AWS CloudShell にインストールされている eksctl コマンドラインツールのバージョン 0.183.0 以降。eksctl をインストールまたはアップグレードするには、eksctl ドキュメントの「インストール」を参照してください。

1. ConfigMap に現在のマッピングを表示します。*my-cluster* を自分のクラスター名に置き換えます。*region-code* をクラスターのある AWS リージョン に置き換えます。

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

出力例は次のとおりです。

```
ARN
                    USERNAME                                GROUPS
             ACCOUNT
arn:aws:iam::111122223333:role/eksctl-my-cluster-my-nodegroup-
NodeInstanceRole-1XLS7754U3ZPA    system:node:{{EC2PrivateDNSName}}
 system:bootstrappers,system:nodes
```

2. ロールのマッピングを追加します。この例では、最初のステップで *my-console-viewer-role* という名前のロールに IAM アクセス許可を添付したと仮定します。*111122223333* をアカウントID に置き換えます。

```
eksctl create iamidentitymapping \
    --cluster my-cluster \
    --region=region-code \
    --arn arn:aws:iam::111122223333:role/my-console-viewer-role \
    --group eks-console-dashboard-full-access-group \
    --no-duplicate-arns
```

> **Important**
>
> ロールの ARN には、role/my-team/developers/my-role などのパスを含めることはできません。ロールの ARN は、arn:aws:iam::*111122223333*:role/*my-role* のような形式にする必要があります。この例では、my-team/developers/ を削除する必要があります。

出力例は次のとおりです。

```
[...]
2022-05-09 14:51:20 [#]  adding identity "arn:aws:iam::111122223333:role/my-
console-viewer-role" to auth ConfigMap
```

3. ユーザーのマッピングを追加します。IAM のベストプラクティスでは、ユーザーではなくロールに許可を付与することが推奨されています。この例では、最初のステップで *my-user* という名前のユーザーに IAM アクセス許可を添付したと仮定します。*111122223333* をアカウント ID に置き換えます。

```
eksctl create iamidentitymapping \
    --cluster my-cluster \
    --region=region-code \
    --arn arn:aws:iam::111122223333:user/my-user \
    --group eks-console-dashboard-restricted-access-group \
    --no-duplicate-arns
```

出力例は次のとおりです。

```
[...]
2022-05-09 14:53:48 [#]  adding identity "arn:aws:iam::111122223333:user/my-
user" to auth ConfigMap
```

4. ConfigMap のマッピングを再度表示します。

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

出力例は次のとおりです。

```
ARN
                    USERNAME                                GROUPS
                      ACCOUNT
arn:aws:iam::111122223333:role/eksctl-my-cluster-my-nodegroup-
NodeInstanceRole-1XLS7754U3ZPA    system:node:{{EC2PrivateDNSName}}
 system:bootstrappers,system:nodes
arn:aws:iam::111122223333:role/my-console-viewer-role
                                                        eks-console-
dashboard-full-access-group
arn:aws:iam::111122223333:user/my-user
                                                        eks-console-
dashboard-restricted-access-group
```

Edit ConfigMap manually

aws-auth ConfigMap にユーザまたはロールを追加する方法の詳細については、「Amazon EKS クラスターに IAM プリンシパルを追加する」を参照してください。

1. 編集する aws-auth ConfigMap を開きます。

```
kubectl edit -n kube-system configmap/aws-auth
```

2. aws-auth ConfigMap にマッピングを追加しますが、既存のマッピングを置き換えないでください。次の例では、最初のステップで追加した許可を持つ IAM プリンシパルと、前のステップで作成した Kubernetes グループの間のマッピングを追加します。
   - *my-console-viewer-role* ロールと eks-console-dashboard-full-access-group。
   - *my-user* ユーザーと eks-console-dashboard-restricted-access-group。

   これらの例では、最初のステップで *my-console-viewer-role* という名前のロール、および *my-user* という名前のユーザーに、IAM アクセス許可を添付したと仮定します。*111122223333* を AWS アカウント ID に置き換えます。

```
apiVersion: v1
data:
mapRoles: |
  - groups:
    - eks-console-dashboard-full-access-group
    rolearn: arn:aws:iam::111122223333:role/my-console-viewer-role
    username: my-console-viewer-role
mapUsers: |
  - groups:
    - eks-console-dashboard-restricted-access-group
    userarn: arn:aws:iam::111122223333:user/my-user
    username: my-user
```

> **Important**
>
> ロールの ARN には、role/my-team/developers/my-console-viewer-role などのパスを含めることはできません。ロールの ARN は、arn:aws:iam::*111122223333*:role/*my-console-viewer-role* のよ

うな形式にする必要があります。この例では、my-team/developers/ を削除する必要があります。

3. ファイルを保存し、テキストエディタを終了します。

Amazon EKS でのオブザーバビリティ

Amazon EKS では、利用可能な多くのモニタリングツールまたはログ記録ツールを使用して、データを観測できます。Amazon EKS ログデータを AWS のサービス またはパートナーツールにストリーミングしてデータ分析することができます。Amazon EKS の問題をトラブルシューティングするためのデータを提供する多くのサービスが、AWS Management Console で利用可能です。また、AWS でサポートされているオープンソースソリューションを使用して、Amazon EKS インフラストラクチャをモニタリングすることもできます。

Amazon EKS コンソールの左側のナビゲーションペインで [クラスター] を選択後、お客様のクラスターの名前を選択すると、クラスターの健全性と詳細を表示できます。クラスタにデプロイされている既存の Kubernetes リソースの詳細を表示するには、「Kubernetes リソースを表示する」を参照してください。

モニタリングは、Amazon EKS と AWS ソリューションの信頼性、可用性、パフォーマンスを維持する上で重要なパートです。モニタリングのデータを AWS ソリューションのすべての部分から収集することをお勧めします。これにより、マルチポイント障害が発生した場合は、その障害をより簡単にデバッグできます。Amazon EKS のモニタリングを開始する前に、モニタリング計画が以下の質問に対処していることを確認してください。

- 目標は何ですか。クラスターが劇的に拡大する場合、リアルタイムの通知が必要ですか。
- どのリソースを観測する必要がありますか。
- これらのリソースをどのくらいの頻度で観測する必要がありますか。あなたの会社はリスクに迅速に対応したいですか。
- どのツールを使用する予定ですか。起動の一環として既に AWS Fargate を実行している場合は、組み込みログルーターを使用できます。
- 誰がモニタリングタスクを実行する予定ですか。
- 何か問題が発生したときに通知を誰に送信したいですか。

Amazon EKS でのログ記録とモニタリング

Amazon EKS には、ログ記録とモニタリング用の組み込みツールが用意されています。コントロールプレーンのログ記録は、クラスターへのすべての API 呼び出し、クラスターに対してどのユーザーがどのアクションを実行したかをキャプチャする監査情報、およびロールベースの情報を記録します。詳細については、AWS 規範的ガイダンスの「Amazon EKS でのロギングとモニタリング」を参照してください。

Amazon EKS コントロールプレーンのログ記録により、アカウント内で Amazon EKS コントロールプレーンから CloudWatch Logs に対し、監査および診断ログを直接送れるようになります。これらのログを使用すると、クラスターの保護と実行が容易になります。必要なログタイプを正確に選択することで、CloudWatch 内で各 Amazon EKS クラスターのためのグループに対し、ログストリームの形態でログを送信できます。詳細については、「Amazon EKS コントロールプレーンのログ記録」を参照してください。

> **Note**
>
> Amazon CloudWatch で Amazon EKS の認証ログをチェックすると、次のサンプルテキストのようなテキストを含むエントリが表示されます。
>
> ```
> level=info msg="mapping IAM role" groups="[]"
>  role="arn:aws:iam::111122223333:role/XXXXXXXXXXXXXXXXXX-
> NodeManagerRole-XXXXXXXX" username="eks:node-manager"
> ```
>
> このテキストを含むエントリが必要です。username は、マネージド型ノードグループと Fargate の特定のオペレーションを実行する、Amazon EKS の内部サービスロールです。低レベルでカスタマイズ可能なログ記録には、Kubernetes 口グ記録 が利用可能です。

Amazon EKS は、AWS CloudTrail と統合されています。このサービスにより、Amazon EKS のユーザー、ロールまたは AWS のサービスによって実行されたアクションを記録します。CloudTrail は、Amazon EKS へのすべての API コールをイベントとしてキャプチャします。キャプチャされた呼び出しには、Amazon EKS コンソールからの呼び出しと、Amazon EKS API オペレーションへのコード呼び出しが含まれます。詳細については、「AWS CloudTrail を使用した Amazon EKS API コールのログ記録」を参照してください。

Kubernetes API サーバーは、モニタリングと分析に役立つ多数のメトリクスを公開します。詳細については、「Prometheus のメトリクス」を参照してください。

Fluent Bit をカスタムの Amazon CloudWatch ログに設定するには、「Amazon CloudWatch ユーザーガイド」の「Fluent Bit のセットアップ」を参照してください。

Amazon EKS のログ記録とモニタリングのツール

Amazon Web Services では、Amazon EKS のモニタリングに使用できるさまざまなツールを提供しています。一部のツールは自動モニタリングを設定できますが、手動呼び出しが必要なツールもあり

ます。環境および既存のツールセットで許容される範囲で、できるだけモニタリングタスクを自動化することをお勧めします。

ログ記録ツール

エリア	ツール	説明	セットアップ
アプリケーション	Amazon CloudWatch Container Insights	コンテナ化されたアプリケーションとマイクロサービスから、メトリクスとログを収集、集計、要約します。	設定手順
コントロールプレーン	AWS CloudTrail	ユーザー、ロール、またはサービスによる API 呼び出しをログに記録します。	設定手順
AWS Fargate インスタンス用の複数のエリア	AWS Fargate ログルーター	AWS Fargate インスタンスの場合、ログを AWS サービスまたはパートナーツールにストリーミングします。AWS for Fluent Bit を使用します。ログは他の AWS のサービス またはパートナーツールにストリーミングできます。	設定手順

モニタリングツール

エリア	ツール	説明	セットアップ
アプリケーション	CloudWatch Container Insights	CloudWatch Container Insights は、コンテナ化されたアプリケーションとマイクロサービスのメトリクスとログを収集、集約、要約します。	設定手順
アプリケーション	AWS Distro for OpenTelemetry (ADOT)	相関メトリクス、トレースデータ、メタデータを収集し、AWS モニタリングサービスまたはパートナーに送信します。これは、CloudWatch Container Insights を介して設定できます。	設定手順
アプリケーション	Amazon DevOps Guru	ノードレベルの運用パフォーマンスと可用性を検出します。	設定手順
アプリケーション	AWS X-Ray	アプリケーションに関するトレースデータ	設定手順

エリア	ツール	説明	セットアップ
		を受信します。このトレースデータには、着信と送信のリクエスト、およびリクエストに関するメタデータが含まれます。Amazon EKS の場合、実装には OpenTelemetry アドオンが必要です。	
アプリケーション	Amazon CloudWatch オブザーバビリティオペレータ	Amazon CloudWatch オブザーバビリティオペレーターは、メトリクス、ログ、トレースデータを収集します。それらを Amazon CloudWatch と AWS X-Ray に送信します。	設定手順

エリア	ツール	説明	セットアップ
コントロールプレーン	Prometheus	CloudWatch Logs で設定された取り込み、アーカイブストレージ、およびデータスキャンレートが、有効化されたコントロールプレーンログに適用されます。	設定手順

Prometheus のメトリクス

Prometheus は、エンドポイントをスクレイピングするモニタリングおよび時系列データベースです。収集したデータをクエリ、集計、保存することができます。また、アラートやアラートの集計にも使用できます。このトピックでは、Prometheus を、マネージドオプションまたはオープンソースオプションとして設定する方法について説明します。Amazon EKS コントロールプレーンメトリクスのモニタリングは一般的なユースケースです。

Amazon Managed Service for Prometheus は Prometheus 互換のモニタリングおよびアラートサービスであり、コンテナ化されたアプリケーションやインフラストラクチャを大規模に監視することが容易になります。これは、メトリクスの取り込み、ストレージ、クエリ、アラートを自動的にスケーリングするフルマネージド型サービスです。また、AWS セキュリティサービスと統合して、データへの高速かつ安全なアクセスを可能にします。オープンソースの PromQL クエリ言語を使用して、メトリクスをクエリし、それらに関するアラートを作成できます。

Prometheus メトリクスをオンにした後の使用方法のさらなる詳細については、「Amazon Managed Service for Prometheus ユーザーガイド」を参照してください。

クラスターを作成するときは、Prometheus メトリクスを有効にしてください。

> **Important**
>
> Amazon Managed Service for Prometheus リソースはクラスターのライフサイクル外にあるため、クラスターとは別途維持する必要があります。クラスターを削除するときは、対象コストを抑えるために、該当するスクレイパーも削除してください。詳細については、Amazon Managed Service for Prometheus ユーザーガイドの「スクレイパーの検出と作成」を参照してください。

新しいクラスターを作成すると、メトリクスを Prometheus に送信するオプションをオンにできます。AWS Management Console では、このオプションはクラスターの新規作成の [オブザーバビリティの設定] ステップにあります。詳細については、「Amazon EKS クラスターの作成」を参照してください。

Prometheus は、スクレイピングと呼ばれるプルベースのモデルを通じて、クラスターからメトリクスを検出して収集します。スクレイパーは、クラスターインフラストラクチャーとコンテナー化されたアプリケーションからデータを収集するように設定されています。

Prometheus メトリクスを送信するオプションをオンにすると、Amazon Managed Service for Prometheus は完全マネージド型のエージェントレススクレイパーを提供します。以下の [詳細設定] オプションを使用して、必要に応じてデフォルトのスクレイパーをカスタマイズします。

スクレイパーエイリアスの名前

(オプション) スクレイパーの一意のエイリアスの入力。

デスティネーション

Amazon Managed Service for Prometheus ワークスペースを選択します。ワークスペースは、Prometheus メトリックの保存とクエリ専用の論理スペースです。このワークスペースでは、そのワークスペースにアクセスできるアカウントの Prometheus メトリクスを表示できます。[新しいワークスペースを作成する] オプションでは、指定した [ワークスペースエイリアス] を使用してユーザーに代わってワークスペースを作成するよう Amazon EKS に指示します。[既存のワークスペースを選択] オプションでは、ドロップダウンリストから既存のワークスペースを選択できます。ワークスペースの詳細については、「Amazon Managed Service for Prometheus ユーザーガイド」の「ワークスペースの管理」を参照してください。

サービスアクセス

このセクションでは、Prometheus メトリクスを送信する際に付与する権限の概要を説明します。

- Amazon Managed Service for Prometheus に、スクレイピングされた Amazon EKS クラスターを記述できるようにします
- Amazon マネージド Prometheus ワークスペースへのリモート書き込みを許可する

`AmazonManagedScraperRole` がすでに存在する場合、スクレイパーはそれを使用します。`AmazonManagedScraperRole` リンクを選択すると、[アクセス許可の詳細] が表示されます。`AmazonManagedScraperRole` がまだ存在しない場合は、[許可を表示] リンクを選択すると、Prometheus メトリクスを送信して付与している特定の権限を確認できます。

サブネット

スクレイパーが継承するサブネットを表示します。変更する必要がある場合は、クラスターの作成でネットワークを指定する手順に戻ってください。

セキュリティグループ

スクレイパーが継承するセキュリティグループを表示します。変更する必要がある場合は、クラスターの作成でネットワークを指定する手順に戻ってください。

スクレイパー設定

必要に応じて YAML 形式でスクレイパー設定を変更します。これを行うには、フォームを使用するか、代替の YAML ファイルをアップロードします。詳細については、「Amazon Managed Service for Prometheus ユーザーガイド」の「スクレイパー設定」を参照してください。

Amazon Managed Service for Prometheus は、AWS マネージドコレクターとしてクラスターと一緒に作成されるエージェントレススクレイパーを指します。AWS マネージドコレクターの詳細については、「Amazon Managed Service for Prometheus ユーザーガイド」の「AWSマネージドコレクター」を参照してください。

> **Important**
>
> スクレイパーにクラスター内のアクセス権限を付与するように `aws-auth ConfigMap` を設定する必要があります。詳細については、「Amazon Managed Service for Prometheus ユーザーガイド」の「Amazon EKS クラスターの設定」を参照してください。

Prometheus スクレイパーの詳細を表示する

Prometheus メトリクスオプションをオンにしてクラスターを作成すると、Prometheus スクレイパーの詳細を表示できます。AWS Management Console でクラスターを表示するときは、[オブザーバビリティ] タブを選択します。表には、スクレイパー ID、エイリアス、ステータス、作成日などの情報を含む、クラスターのスクレイパーのリストが表示されます。

スクレイパーの詳細を表示するには、スクレイパー ID リンクを選択します。例えば、スクレイパー設定、Amazon リソースネーム (ARN)、リモート書き込み URL、およびネットワーク情報を表示できます。スクレイパー ID は、`DescribeScraper` や `DeleteScraper` などの Amazon Managed Service for Prometheus API オペレーションへの入力として使用できます。API を使用して、さらにスクレイパーを作成することもできます。

Prometheus API の使用に関する詳細については、「Amazon Managed Service for Prometheus API リファレンス」を参照してください。

Helm を使用して Prometheus をデプロイする

または、Helm V3 を使用してクラスターに Prometheus をデプロイすることもできます。すでに Helm をインストールしている場合は、`helm version` コマンドでバージョンを確認できます。Helm は Kubernetes クラスター用のパッケージマネージャーです。Helm の詳細およびインストール方法については、「Amazon EKS での Helm の使用」を参照してください。

Amazon EKS クラスター用に Helm を設定した後、次の手順で Prometheus をデプロイするために使用できます。

Helm を使用して Prometheus をデプロイするには

1. Prometheus 名前空間を作成します。

```
kubectl create namespace prometheus
```

2. `prometheus-community` グラフリポジトリの追加

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
```

3. Prometheus をデプロイします。

```
helm upgrade -i prometheus prometheus-community/prometheus \
```

```
    --namespace prometheus \
    --set alertmanager.persistence.storageClass="gp2" \
    --set server.persistentVolume.storageClass="gp2"
```

> **Note**
>
> このコマンドの実行時に `Error: failed to download "stable/prometheus" (hint: running `helm repo update` may help)` エラーが発生した場合、`helm repo update prometheus-community` を実行し、ステップ 2 のコマンドをもう一度実行してみてください。
> `Error: rendered manifests contain a resource that already exists` エラーが発生した場合、`helm uninstall` *your-release-name* `-n` *namespace* を実行し、ステップ 3 のコマンドをもう一度実行してみてください。

4. `prometheus` 名前空間のすべての Pods が `READY` 状態になっていることを確認します。

```
kubectl get pods -n prometheus
```

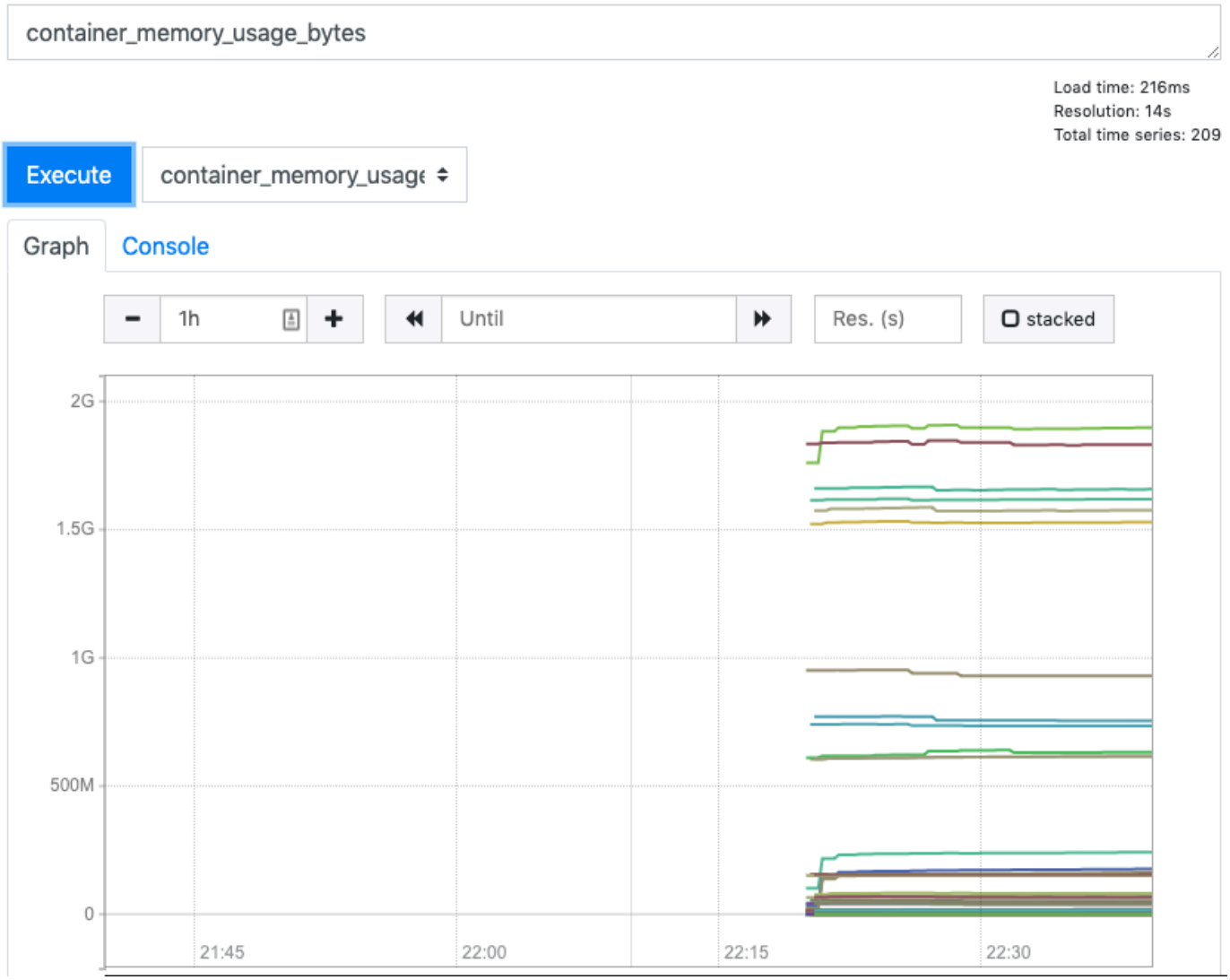
出力例は次のとおりです。

```
NAME                                             READY   STATUS    RESTARTS   AGE
prometheus-alertmanager-59b4c8c744-r7bgp         1/2     Running   0          48s
prometheus-kube-state-metrics-7cfd87cf99-jkz2f   1/1     Running   0          48s
prometheus-node-exporter-jcjqz                   1/1     Running   0          48s
prometheus-node-exporter-jxv2h                   1/1     Running   0          48s
prometheus-node-exporter-vbdks                   1/1     Running   0          48s
prometheus-pushgateway-76c444b68c-82tnw          1/1     Running   0          48s
prometheus-server-775957f748-mmht9               1/2     Running   0          48s
```

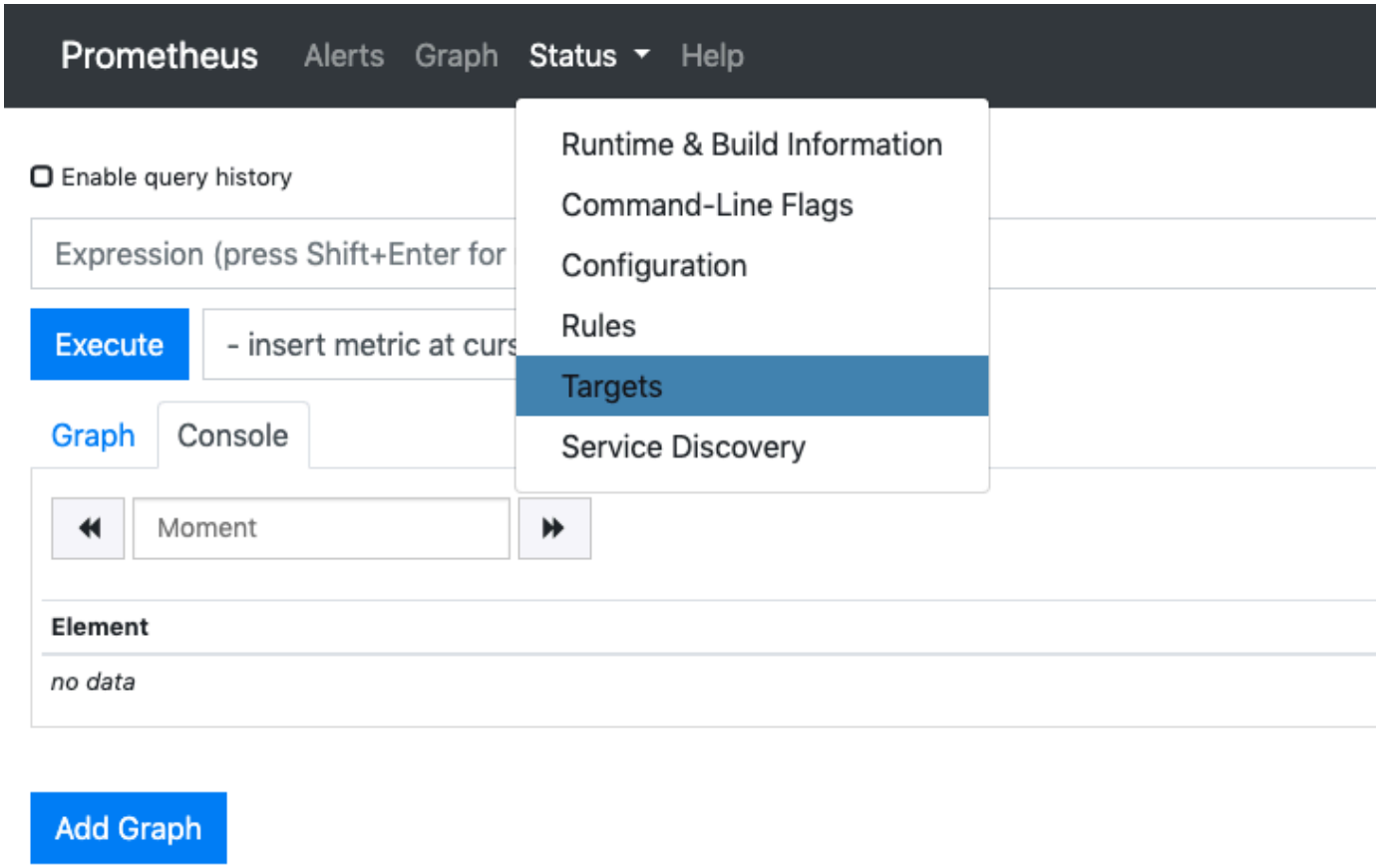
5. `kubectl` を使用して Prometheus コンソールをローカルマシンにポート転送します。

```
kubectl --namespace=prometheus port-forward deploy/prometheus-server 9090
```

6. ウェブブラウザを `http://localhost:9090` に指定して、Prometheus コンソールを表示します。
7. [- カーソルの位置のメトリクスを挿入] メニューからメトリクスを選択し、[実行] を選択します。[グラフ] タブを選択して、一定期間のメトリクスを表示します。次の図では、一定期間の `container_memory_usage_bytes` を示します。



8. 上部のナビゲーションバーから、[ステータス]、[ターゲット] の順に選択します。



サービス検出を使用して Prometheus に接続されているすべての Kubernetes エンドポイントが表示されます。

コントロールプレーンにおける未加工メトリクスの表示

Kubernetes API サーバーは、Prometheus をデプロイする代わりに、Prometheus の形式で表される多数のメトリクスを公開します。これらのメトリクスは、モニタリングおよび分析に役立ちます。これらのメトリクスは、`/metrics` HTTP API を参照するメトリクスエンドポイントを介して内部的に公開されます。他のエンドポイントと同様に、このエンドポイントは Amazon EKS コントロールプレーンに公開されます。このエンドポイントは、主に特定のメトリクスを調べるのに役立ちます。メトリクスを時系列的に分析するには、Prometheus をデプロイすることをおすすめします。

未加工メトリクスの出力を表示するには、`--raw` フラグを付けて `kubectl` を使用します。このコマンドを使用すると、任意の HTTP パスを渡して未加工のレスポンスを返すことができます。

```
kubectl get --raw /metrics
```

出力例は次のとおりです。

```
[...]
# HELP rest_client_requests_total Number of HTTP requests, partitioned by status code,
 method, and host.
# TYPE rest_client_requests_total counter
rest_client_requests_total{code="200",host="127.0.0.1:21362",method="POST"} 4994
rest_client_requests_total{code="200",host="127.0.0.1:443",method="DELETE"} 1
rest_client_requests_total{code="200",host="127.0.0.1:443",method="GET"} 1.326086e+06
rest_client_requests_total{code="200",host="127.0.0.1:443",method="PUT"} 862173
rest_client_requests_total{code="404",host="127.0.0.1:443",method="GET"} 2
rest_client_requests_total{code="409",host="127.0.0.1:443",method="POST"} 3
rest_client_requests_total{code="409",host="127.0.0.1:443",method="PUT"} 8
# HELP ssh_tunnel_open_count Counter of ssh tunnel total open attempts
# TYPE ssh_tunnel_open_count counter
ssh_tunnel_open_count 0
# HELP ssh_tunnel_open_fail_count Counter of ssh tunnel failed open attempts
# TYPE ssh_tunnel_open_fail_count counter
ssh_tunnel_open_fail_count 0
```

この未加工出力は、API サーバーが公開する内容をそのまま返します。さまざまなメトリクスが行ごとに一覧表示され、各行には指標名、タグ、値が含まれます。

```
metric_name{"tag"="value"[,...]}
            value
```

Amazon EKS アドオンのサポート (Amazon CloudWatch)

Amazon CloudWatch Observability は、リアルタイムのログ、メトリクス、トレースデータを収集します。それらを Amazon CloudWatch に送信し、AWS X-Ray. このアドオンをインストールすると、CloudWatch アプリケーションシグナルと Amazon EKS のオブザーバビリティが強化された CloudWatch Container Insights の両方を有効にできます。これにより、インフラストラクチャとコンテナ化されたアプリケーションのヘルスとパフォーマンスをモニタリングできます。この Amazon CloudWatch Observability Operator は、必要なコンポーネントをインストールして設定するように設計されています。

Amazon EKS は Amazon EKS アドオンとして、Amazon CloudWatch Observability Operator をサポートしています。このアドオンを使うと、クラスター内の Linux ワーカーノードと Windows ワーカーノードの両方で Container Insights を使用できます。Container Insights を Windows で

有効にするには、Amazon EKS アドオンのバージョンが 1.5.0 以上である必要があります。現在、CloudWatch Application Signals は Amazon EKS Windows ではサポートされていません。

以下のトピックでは、Amazon EKSクラスタで Amazon CloudWatch Observability Operator を使い始める方法について説明します。

- このアドオンのインストール手順については、Amazon CloudWatch ユーザーガイドの「CloudWatch オブザーバビリティ Amazon EKS アドオンを使用して Amazon CloudWatch エージェントをインストールする」を参照してください。
- CloudWatch アプリケーションシグナルの詳細については、「アプリケーションシグナル」を参照してください。
- Container Insights の詳細については、「Amazon CloudWatch ユーザーガイド」の「Using Container Insights」を参照してください。

Amazon EKS コントロールプレーンのログ記録

Amazon EKS コントロールプレーンのログ記録により、アカウント内で Amazon EKS コントロールプレーンから CloudWatch Logs に対し、監査および診断ログを直接送れるようになります。これらのログを使用すると、クラスターの保護と実行が容易になります。必要なログタイプを正確に選択することで、CloudWatch 内で各 Amazon EKS クラスターのためのグループに対し、ログストリームの形態でログを送信できます。詳細については、「Amazon CloudWatch logging」(Amazon CloudWatch ログ記録) を参照してください。

Amazon EKS コントロールプレーンのログ記録の使用を開始するには、新規または既存の Amazon EKS クラスターごとに有効にするログタイプを選択します。クラスターごとに各ログタイプを有効または無効にするには、AWS Management Console、AWS CLI (バージョン 1.16.139 以降)、または Amazon EKS API を使用します。有効化したログタイプのログが、Amazon EKS クラスターから同じアカウントの CloudWatch Logs に自動的に送信されるようになります。

Amazon EKS コントロールプレーンのログ記録を使用すると、実行しているクラスターごとに Amazon EKS の標準料金が発生します。クラスターから CloudWatch Logs に送信されるすべてのログに対して、CloudWatch Logs のデータ取り込みおよび保存に関する標準料金が発生します。また、Amazon EC2 インスタンスや Amazon EBS ボリュームなど、クラスターの一部としてプロビジョニングしている AWS リソースに対しても料金が発生します。

以下のクラスターコントロールプレーンのログタイプが使用可能です。各ログタイプは、Kubernetes コントロールプレーンのコンポーネントに対応しています。これらのコンポーネン

トの詳細については、Kubernetes ドキュメントの「Kubernetes コンポーネント」を参照してください。

API サーバー (**api**)

クラスターの API サーバーは、Kubernetes API を公開するコントロールプレーンコンポーネントです。クラスターを起動する際、またはその直後に API サーバーのログを有効にすると、ログには、API サーバーの起動に使用された API サーバーフラグが含まれます。詳細については、「Kubernetes ドキュメント」の「kube-apiserver」と「audit policy」(監査ポリシー) を参照してください。

監査 (**audit**)

Kubernetes 監査ログは、クラスターに影響を与えた個々のユーザー、管理者、またはシステムコンポーネントの記録を提供します。詳細については、Kubernetes ドキュメントの「Auditing」(監査) を参照してください。

Authenticator (**authenticator**)

Authenticator ログは、Amazon EKS に固有です。これらのログは、IAM 認証情報を使用した Kubernetes [ロールベースのアクセスコントロール] (RBAC) 認証のために Amazon EKS が使用する、コントロールプレーンコンポーネントを示します。詳細については、「クラスターの管理」を参照してください。

コントローラーマネージャー (**controllerManager**)

コントローラーマネージャーは、Kubernetes に付属するコアコントロールループを管理します。詳細については、Kubernetes ドキュメントの「kube-controller-manager」を参照してください。

スケジューラ (**scheduler**)

スケジューラコンポーネントは、クラスター内で Pods を実行するタイミングと場所を管理します。詳細については、Kubernetes ドキュメントの「kube-scheduler」を参照してください。

コントロールプレーンログの有効化と無効化

デフォルトでは、クラスターコントロールプレーンのログは CloudWatch Logs に送信されません。クラスターのログを送信するには、各ログタイプを個別に有効にする必要があります。CloudWatch Logs で設定された取り込み、アーカイブストレージ、およびデータスキャンレートが、有効化されたコントロールプレーンログに適用されます。詳細については、「CloudWatch 料金表」を参照してください。

コントロールプレーンのログ記録設定を更新するために、Amazon EKS は各サブネットで最大 5 つの使用可能な IP アドレスを必要とします。ログタイプを有効にすると、ログの詳細レベル 2 でログが送信されます。

AWS Management Console

AWS Management Console を使用してコントロールプレーンログを有効または無効にするには

1. https://console.aws.amazon.com/eks/home#/clusters で Amazon EKS コンソールを開きます。
2. クラスター名を選択すると、そのクラスターの情報を表示されます。
3. [オブザーバビリティ] タブを選択します。
4. [コントロールプレーンのロギング] セクションで、[ロギングの管理] を選択します。
5. ログタイプごとに、そのログタイプを [有効] にするか [無効] にするかを選択します。デフォルトでは、それぞれのログタイプは無効化されています。
6. [変更を保存] を選択して終了します。

AWS CLI

AWS CLI を使用してコントロールプレーンログを有効または無効にするには

1. 以下のコマンドを使用して、AWS CLI のバージョンを確認します。

```
aws --version
```

AWS CLI バージョンが `1.16.139` より前である場合は、まず最新バージョンに更新する必要があります。AWS CLI のインストールまたはアップグレードについては、AWS Command Line Interface ユーザーガイドの「AWS Command Line Interface のインストール」を参照してください。

2. 次の AWS CLI コマンドを使用して、クラスターのコントロールプレーンログのエクスポート設定を更新します。*`my-cluster`* をクラスター名に置き換え、目的のエンドポイントアクセス値を指定します。

> **Note**
>
> 次のコマンドは、使用可能なすべてのログタイプを CloudWatch Logs に送信します。

```
aws eks update-cluster-config \
    --region region-code \
    --name my-cluster \
    --logging '{"clusterLogging":[{"types":
["api","audit","authenticator","controllerManager","scheduler"],"enabled":true}]}'
```

出力例は次のとおりです。

```
{
    "update": {
        "id": "883405c8-65c6-4758-8cee-2a7c1340a6d9",
        "status": "InProgress",
        "type": "LoggingUpdate",
        "params": [
            {
                "type": "ClusterLogging",
                "value": "{\"clusterLogging\":[{\"types\":[\"api\",\"audit\",
\"authenticator\",\"controllerManager\",\"scheduler\"],\"enabled\":true}]}"
            }
        ],
        "createdAt": 1553271814.684,
        "errors": []
    }
}
```

3. 次のコマンドでログ設定更新のステータスをモニタリングします。その際、以前のコマンドで返されたクラスター名と更新 ID を使用します。ステータスが Successful と表示されたら、更新は完了です。

```
aws eks describe-update \
    --region region-code\
    --name my-cluster \
    --update-id 883405c8-65c6-4758-8cee-2a7c1340a6d9
```

出力例は次のとおりです。

```
{
    "update": {
        "id": "883405c8-65c6-4758-8cee-2a7c1340a6d9",
        "status": "Successful",
        "type": "LoggingUpdate",
        "params": [
            {
                "type": "ClusterLogging",
                "value": "{\"clusterLogging\":[{\"types\":[\"api\",\"audit\",
\"authenticator\",\"controllerManager\",\"scheduler\"],\"enabled\":true}]}"
            }
        ],
        "createdAt": 1553271814.684,
        "errors": []
    }
}
```

クラスターのコントロールプレーンログの表示

Amazon EKS クラスターのいずれかのコントロールプレーンログタイプを有効にすると、それらを CloudWatch コンソールで表示できるようになります。

CloudWatch でログを表示、分析、および管理する方法については、Amazon CloudWatch Logs ユーザーガイドを参照してください。

CloudWatch コンソールでクラスターのコントロールプレーンログを表示するには

1. CloudWatch コンソールを開きます。このリンクでは、現在使用可能なロググループが表示され、プレフィックス /aws/eks でフィルタリングされます。
2. ログを表示するクラスターを選択します。ロググループの名前の形式は /aws/eks/*my-cluster*/cluster です。
3. 表示するログストリームを選択します。次のリストで、各ログタイプのログストリーム名の形式について説明します。

> **Note**
>
> ログストリームデータが大きくなるにつれて、ログストリーム名のローテーションが行われます。特定のログタイプのログストリームが複数存在する場合、最新のログストリームを表示するには、[最終のイベント時刻] が最新のログストリーム名を見つけ出します。

- Kubernetes API サーバーコンポーネントログ (**api**) - kube-apiserver-*1234567890abcdef01234567890abcde*
- 監査 (**audit**) – kube-apiserver-audit-*1234567890abcdef01234567890abcde*
- 認証システム (**authenticator**) – authenticator-*1234567890abcdef01234567890abcde*
- コントローラーマネージャー (**controllerManager**) – kube-controller-manager-*1234567890abcdef01234567890abcde*
- スケジューラ (**scheduler**) – kube-scheduler-*1234567890abcdef01234567890abcde*

4. ログストリームのイベントを確認してください。

   たとえば、kube-apiserver-*1234567890abcdef01234567890abcde* のトップを表示すると、クラスターの初期 API サーバーが表示されます。

> **Note**
>
> ログストリームの先頭に API サーバーログが表示されない場合は、サーバーで API サーバーログ記録を有効にする前に API サーバーのログファイルがローテーションされた可能性があります。API サーバーのログ記録が有効になる前にローテーションされたログファイルは、CloudWatch にエクスポートできません。
> ただし、同じ Kubernetes バージョンで新しいクラスターを作成し、クラスターの作成時に API サーバーログを有効にすることができます。同じプラットフォームバージョンを持つクラスターでは、同じフラグが有効になるため、フラグは新しいクラスターのフラグと一致する必要があります。CloudWatch で新しいクラスターのフラグを確認し終えたら、新しいクラスターを削除します。

AWS CloudTrail を使用した Amazon EKS API コールのログ記録

Amazon EKS は AWS CloudTrail と統合します。CloudTrail は、Amazon EKS のユーザー、ロール、または AWS サービスによるアクションを記録するサービスです。CloudTrail は、Amazon EKS へのすべての API コールをイベントとしてキャプチャします。これには、Amazon EKS コンソールからの呼び出しと、Amazon EKS API オペレーションへのコード呼び出しが含まれます。

証跡を作成する場合は、 のイベントなど、Amazon S3 バケットへの CloudTrail イベントの継続的な配信を有効にすることができます これには、Amazon EKS のイベントが含まれます。追跡を設定しない場合でも、CloudTrail コンソールの [Event history] (イベント履歴) で最新のイベントを表示できます。CloudTrail で収集された情報に基づいて、リクエストに関する詳細を確認できます。例えば、Amazon EKS に対するリクエストの日時や、リクエスト元の IP アドレス、リクエストの実行者を確認できます。

CloudTrail の詳細については、「AWS CloudTrail ユーザーガイド」を参照してください。

トピック

- CloudTrail の Amazon EKS 情報
- Amazon EKS ログファイルエントリの理解
- Auto Scaling グループのメトリクス収集を有効にする

CloudTrail の Amazon EKS 情報

AWS アカウントを作成すると、AWS アカウントで CloudTrail も有効になります。Amazon EKS でイベントアクティビティが発生すると、そのアクティビティは [Event history] (イベント履歴) で他の AWS サービスのイベントとともに CloudTrail イベントに記録されます。AWS アカウントで最近のイベントを表示、検索、ダウンロードできます。詳細については、CloudTrail イベント履歴でのイベントの表示を参照してください。

Amazon EKS のイベントなど、AWS アカウントのイベントの継続的な記録については、証跡を作成します。追跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。デフォルトでは、コンソールで証跡を作成するときに、証跡がすべての AWS リージョン に適用されます。証跡では、AWS パーティション内のすべての AWS リージョン からのイベントがログに記録され、指定した Amazon S3 バケットにログファイルが配信されます。さらに、CloudTrail ログで収集したイベントデータをより詳細に分析し、それに基づく対応するようにその他の AWS サービスを設定できます。詳細については、以下のリソースを参照してください。

- 証跡作成の概要

- CloudTrail がサポートされているサービスと統合
- CloudTrail の Amazon SNS 通知の設定
- 複数のリージョンから CloudTrail ログファイルを受け取るおよび複数のアカウントから CloudTrail ログファイルを受け取る

すべての Amazon EKS アクションは、CloudTrail により記録され、Amazon EKS API リファレンスで文書化されます。例えば、`CreateCluster`、`ListClusters`、および `DeleteCluster` セクションを呼び出すと、CloudTrail ログファイルにエントリが生成されます。

すべてのイベントまたはログエントリには、リクエストを行った IAM アイデンティティのタイプとどの認証情報が使用されたかに関する詳細が含まれます。一時的認証情報が使用された場合、エレメントは、認証情報がどのように取得されたかを示します。

詳細については、「CloudTrail userIdentity エレメント」を参照してください。

Amazon EKS ログファイルエントリの理解

[トレイル] は、指定した Simple Storage Service (Amazon S3) バケットにイベントをログファイルとして配信するように構成できます。CloudTrail のログファイルには、単一か複数のログエントリがあります。イベントでは、あらゆるソースからの単一のリクエストが示され、リクエストされたアクションに関する情報が含まれています。これには、アクションの日時、使用されたリクエストパラメータなどの情報が含まれます。CloudTrail ログファイルは、パブリック API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

次の例は、`CreateCluster` アクションを示す CloudTrail ログエントリです。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AKIAIOSFODNN7EXAMPLE",
    "arn": "arn:aws:iam::111122223333:user/username",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "username"
  },
  "eventTime": "2018-05-28T19:16:43Z",
  "eventSource": "eks.amazonaws.com",
  "eventName": "CreateCluster",
  "awsRegion": "region-code",
```

```
  "sourceIPAddress": "205.251.233.178",
  "userAgent": "PostmanRuntime/6.4.0",
  "requestParameters": {
    "resourcesVpcConfig": {
      "subnetIds": [
        "subnet-a670c2df",
        "subnet-4f8c5004"
      ]
    },
    "roleArn": "arn:aws:iam::111122223333:role/AWSServiceRoleForAmazonEKS-
CAC1G1VH3ZKZ",
    "clusterName": "test"
  },
  "responseElements": {
    "cluster": {
      "clusterName": "test",
      "status": "CREATING",
      "createdAt": 1527535003.208,
      "certificateAuthority": {},
      "arn": "arn:aws:eks:region-code:111122223333:cluster/test",
      "roleArn": "arn:aws:iam::111122223333:role/AWSServiceRoleForAmazonEKS-
CAC1G1VH3ZKZ",
      "version": "1.10",
      "resourcesVpcConfig": {
        "securityGroupIds": [],
        "vpcId": "vpc-21277358",
        "subnetIds": [
          "subnet-a670c2df",
          "subnet-4f8c5004"
        ]
      }
    }
  },
  "requestID": "a7a0735d-62ab-11e8-9f79-81ce5b2b7d37",
  "eventID": "eab22523-174a-499c-9dd6-91e7be3ff8e3",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
```

Amazon EKS サービスにリンクされたロールのログエントリ

Amazon EKS サービスにリンクされたロールは、AWS リソースへの API コールを行います。Amazon EKS サービスにリンクされたロールによって行われた呼び出しには、CloudTrail ログエントリが username: AWSServiceRoleForAmazonEKS と username: AWSServiceRoleForAmazonEKSNodegroup で表示されます。Amazon EKS およびサービスにリンクされたロールの詳細については、「Amazon EKS でのサービスにリンクされたロールの使用」を参照してください。

次の例では、sessionContext に記録され、AWSServiceRoleForAmazonEKSNodegroup サービスにリンクされたロールによって行われた DeleteInstanceProfile アクションを示す CloudTrail ログエントリが示されています。


```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA3WHGPEZ7SJ2CW55C5:EKS",
        "arn": "arn:aws:sts::111122223333:assumed-role/
AWSServiceRoleForAmazonEKSNodegroup/EKS",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROA3WHGPEZ7SJ2CW55C5",
                "arn": "arn:aws:iam::111122223333:role/aws-service-role/eks-
nodegroup.amazonaws.com/AWSServiceRoleForAmazonEKSNodegroup",
                "accountId": "111122223333",
                "userName": "AWSServiceRoleForAmazonEKSNodegroup"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2020-02-26T00:56:33Z"
            }
        },
        "invokedBy": "eks-nodegroup.amazonaws.com"
    },
    "eventTime": "2020-02-26T00:56:34Z",
    "eventSource": "iam.amazonaws.com",
    "eventName": "DeleteInstanceProfile",
```

```
    "awsRegion": "region-code",
    "sourceIPAddress": "eks-nodegroup.amazonaws.com",
    "userAgent": "eks-nodegroup.amazonaws.com",
    "requestParameters": {
        "instanceProfileName": "eks-11111111-2222-3333-4444-abcdef123456"
    },
    "responseElements": null,
    "requestID": "11111111-2222-3333-4444-abcdef123456",
    "eventID": "11111111-2222-3333-4444-abcdef123456",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

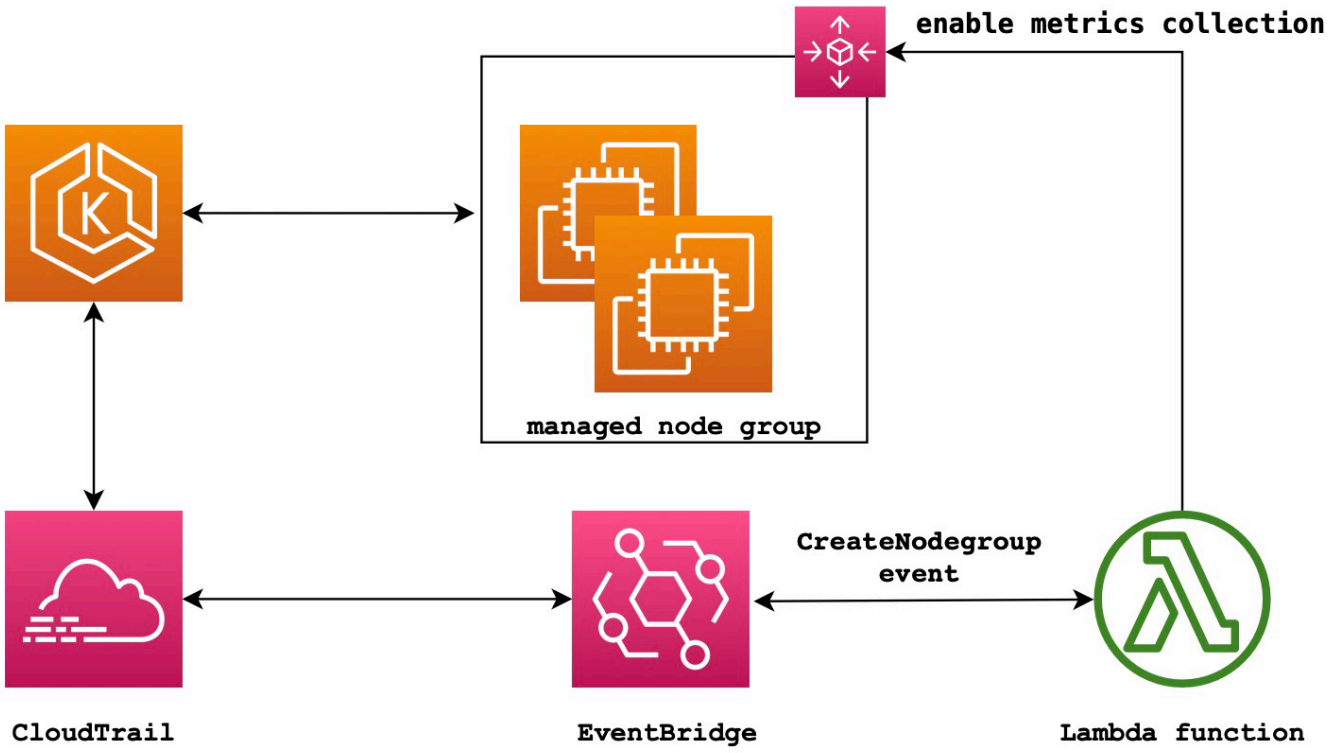
Auto Scaling グループのメトリクス収集を有効にする

このトピックでは、AWS Lambda および AWS CloudTrail を使用して Auto Scaling グループのメトリックス収集を有効にする方法について説明します。Amazon EKS は、マネージドノード用に作成された Auto Scaling グループのグループメトリックス収集を自動的に有効にしません。

Auto Scaling グループのメトリックスを使用して、Auto Scaling グループの変更を追跡し、しきい値にアラームを設定できます。Auto Scaling グループのメトリックスは、Auto Scaling コンソールまたは Amazon CloudWatch コンソールで使用できます。Auto Scaling グループが有効になると、サンプリングされたデータが毎分 Amazon CloudWatch に送信されます。これらのメトリクスの有効化には料金はかかりません。

Auto Scaling グループのメトリクス収集を有効にすることで、マネージド型ノードグループのスケーリングを監視することができるようになります。Auto Scaling グループのメトリクスは、Auto Scaling グループの最小サイズ、最大サイズ、および必要なサイズを報告します。ノードグループ内のノード数が最小サイズを下回った場合にアラームを作成できます。これは、ノードグループに異常があることを示しています。ノードグループサイズを追跡することは、データプレーンの容量が不足しないように最大数を調整する場合にも役立ちます。

マネージド型ノードグループを作成すると、AWS CloudTrail は `CreateNodegroup` イベントを Amazon EventBridge に送信します。`CreateNodegroup` イベントと一致する Amazon EventBridge ルールを作成することで、Lambda 関数をトリガーして、マネージド型ノードグループに関連付けられた Auto Scaling グループのグループメトリック収集を有効にします。



Auto Scaling グループのメトリクス収集を有効にするには

1. Lambda 用に IAM ロールを作成する。

```
LAMBDA_ROLE=$(aws iam create-role \
  --role-name lambda-asg-enable-metrics \
  --assume-role-policy-document '{"Version": "2012-10-17","Statement":
 [{ "Effect": "Allow", "Principal": {"Service": "lambda.amazonaws.com"}, "Action":
 "sts:AssumeRole"}]}' \
  --output text \
  --query 'Role.Arn')
echo $LAMBDA_ROLE
```

2. Amazon EKS ノードグループを記述し、Auto Scaling グループのメトリックス収集を有効にすることを許可するポリシーを作成します。

```
cat > /tmp/lambda-policy.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
```

```
      {
        "Effect": "Allow",
        "Action": [
            "eks:DescribeNodegroup",
            "autoscaling:EnableMetricsCollection"
            ],
        "Resource": [
            "*"
        ]
      }
    ]
}
EOF
LAMBDA_POLICY_ARN=$(aws iam create-policy \
  --policy-name lambda-asg-enable-metrics-policy \
  --policy-document file:///tmp/lambda-policy.json \
  --output text \
  --query 'Policy.Arn')
echo $LAMBDA_POLICY_ARN
```

3. Lambda の IAM ロールにポリシーをアタッチします。

```
aws iam attach-role-policy \
  --policy-arn $LAMBDA_POLICY_ARN \
  --role-name lambda-asg-enable-metrics
```

4. ログを CloudWatch Logs に書き込むために関数が必要とするアクセス許可を含む AWSLambdaBasicExecutionRole 管理ポリシーを追加します。

```
aws iam attach-role-policy \
  --role-name lambda-asg-enable-metrics \
  --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole
```

5. Lambda コードを作成します。

```
cat > /tmp/lambda-handler.py <<EOF
import json
import boto3
import time
import logging

eks = boto3.client('eks')
autoscaling = boto3.client('autoscaling')
```

```
logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):
    ASG_METRICS_COLLLECTION_TAG_NAME = "ASG_METRICS_COLLLECTION_ENABLED"
    initial_retry_delay = 10
    attempts = 0

    #print(event)

    if not event["detail"]["eventName"] == "CreateNodegroup":
        print("invalid event.")
        return -1

    clusterName = event["detail"]["requestParameters"]["name"]
    nodegroupName = event["detail"]["requestParameters"]["nodegroupName"]
    try:
        metricsCollectionEnabled = event["detail"]["requestParameters"]["tags"]
[ASG_METRICS_COLLLECTION_TAG_NAME]
    except KeyError:
        print(ASG_METRICS_COLLLECTION_TAG_NAME, "tag not found.")
        return

    # Check if metrics collection is enabled in tags
    if metricsCollectionEnabled.lower() != "true":
        print("Metrics collection is not enabled in nodegroup tags.")
        return

    # Get the name of the associated autoscaling group
    print("Getting the autoscaling group name for nodegroup=", nodegroupName, ",
 cluster=", clusterName )
    for i in range(0,10):
        try:
            autoScalingGroup =
 eks.describe_nodegroup(clusterName=clusterName,nodegroupName=nodegroupName)
["nodegroup"]["resources"]["autoScalingGroups"][0]["name"]
        except:
            attempts += 1
            print("Failed to obtain the associated autoscaling group for
 nodegroup", nodegroupName, "Retrying in", initial_retry_delay*attempts,
 "seconds.")
            time.sleep(initial_retry_delay*attempts)
        else:
```

```
            break

    print("Enabling metrics collection on autoscaling group ", autoScalingGroup)

    # Enable metrics collection in the autoscaling group
    try:
        enableMetricsCollection =
 autoscaling.enable_metrics_collection(AutoScalingGroupName=autoScalingGroup,Granularity="1
    except:
        print("Unable to enable metrics collection on nodegroup=",nodegroup)
    print("Enabled metrics collection on nodegroup", nodegroupName)
EOF
```

6. デプロイパッケージを作成します。

```
cd /tmp
zip function.zip lambda-handler.py
```

7. Lambda 関数を作成する。

```
LAMBDA_ARN=$(aws lambda create-function --function-name asg-enable-metrics-
collection \
  --zip-file fileb://function.zip --handler lambda-handler.lambda_handler \
  --runtime python3.9 \
  --timeout 600 \
  --role $LAMBDA_ROLE \
  --output text \
  --query 'FunctionArn')
echo $LAMBDA_ARN
```

8. EventBridge ルールを作成します。

```
RULE_ARN=$(aws events put-rule --name CreateNodegroupRuleToLambda \
  --event-pattern "{\"source\":[\"aws.eks\"],\"detail-type\":[\"AWS API Call via
 CloudTrail\"],\"detail\":{\"eventName\":[\"CreateNodegroup\"],\"eventSource\":
[\"eks.amazonaws.com\"]}}" \
  --output text \
  --query 'RuleArn')
echo $RULE_ARN
```

9. ターゲットとしての Lambda 関数を追加します。

```
aws events put-targets --rule CreateNodegroupRuleToLambda \
```

```
  --targets "Id"="1","Arn"="$LAMBDA_ARN"
```

10. EventBridge が Lambda 関数を呼び出すことを許可するポリシーを追加します。

```
aws lambda add-permission \
  --function-name asg-enable-metrics-collection \
  --statement-id CreateNodegroupRuleToLambda \
  --action 'lambda:InvokeFunction' \
  --principal events.amazonaws.com \
  --source-arn $RULE_ARN
```

Lambda 関数は、ASG_METRICS_COLLLECTION_ENABLED を TRUE に設定してタグ付けしたマネージド型ノードグループの Auto Scaling グループのメトリック収集を有効にします。[Auto Scaling group metrics collection] (Auto Scaling グループのメトリクス収集) が有効になっていることを確認するには、Amazon EC2 コンソールで、関連する Auto Scaling グループに移動します。[Monitoring] (モニタリング) タブで、[Enable] (有効化) チェックボックスが有効になっているはずです。

ADOT 演算子の Amazon EKS アドオンサポート

Amazon EKS は、AWS Management Console、AWS CLI、および Amazon EKS API を使用して、AWS Distro for OpenTelemetry (ADOT) Operator をインストールおよび管理することをサポートしています。これにより、シンプルな体験で Amazon EKS で実行されているアプリケーションをインストルメント化して、メトリクスおよびトレースデータを Amazon CloudWatch、Prometheus、X-Ray のような複数のモニタリングサービスオプションに送信できます。

追加情報 — 詳細については、 Distro for OpenTelemetry ドキュメントの「EKS アドオンを使用した Distro for OpenTelemetry の開始方法」を参照してください。

Amazon EKS と統合された AWS のサービス

他のセクションで説明されているサービスに加えて、Amazon EK はより多くの AWS サービスと連携して追加のソリューションを提供しています。このトピックで説明するサービスは、Amazon EKS を使用して機能を追加するか、タスクを実行するために Amazon EKS に使用されるかのいずれかです。

トピック

AWS CloudFormation を使用した Amazon EKS リソースの作成

Amazon EKS は AWS CloudFormation と統合されています。これは、リソースとインフラストラクチャの作成と管理の所要時間を短縮できるように AWS リソースをモデル化して設定するためのサービスです。すべての必要な AWS リソース (Amazon EKS クラスターなど) を記述するテンプレートを作成すれば、これらのリソースのプロビジョニングと設定は AWS CloudFormation で代行されます。

AWS CloudFormation を使用すると、テンプレートを再利用して Amazon EKS リソースをいつでも繰り返しセットアップできます。リソースを一度記述するだけで、同じリソースを複数の AWS リージョンで何度でもプロビジョニングできます。

Amazon EKS と AWS CloudFormation テンプレート

Amazon EKS および関連サービスのリソースをプロビジョニングして設定するには、AWS CloudFormation テンプレートについて理解しておく必要があります。テンプレートは、JSON または YAML でフォーマットされたテキストファイルです。これらのテンプレートには、AWS

CloudFormation スタックにプロビジョニングしたいリソースを記述します。JSONやYAMLに不慣れな方は、AWS CloudFormation Designerを使えば、 AWS CloudFormation テンプレートを使いこなすことができます。詳細については、AWS CloudFormation ユーザーガイドの「AWS CloudFormation Designer とは?」を参照してください。

Amazon EKS では、AWS CloudFormation でのクラスターとノードグループの作成がサポートされています。Amazon EKS リソースの JSON テンプレートと YAML テンプレートの例を含む詳細情報については、AWS CloudFormation ユーザーガイドの「Amazon EKS resource type reference (Amazon EKS リソースタイプのリファレンス)」を参照してください。

の詳細情報AWS CloudFormation

AWS CloudFormation の詳細については、以下のリソースを参照してください。

- AWS CloudFormation
- AWS CloudFormation ユーザーガイド
- AWS CloudFormation コマンドラインインターフェイスユーザーガイド

Amazon EKS と AWS Local Zones

AWS ローカルゾーンは AWS リージョン の拡張であり、ユーザーに近い場所に配置されます。Local Zones は、インターネットへの独自の接続を持ち、AWS Direct Connect をサポートします。Local Zone で作成したリソースは、低いレイテンシーの通信をローカルユーザーに提供できます。詳細については、「 Local Zones」を参照してください。

Amazon EKS では、Local Zones の特定のリソースをサポートしています。これには、セルフマネージド型の Amazon EC2 ノード、Amazon EBS ボリューム、および Application Load Balancer (ALB) が含まれます。Amazon EKS クラスターの一部として Local Zones を使用する場合は、次を考慮することをお勧めします。

ノード

Amazon EKS を使用して Local Zones に Fargate ノードまたはマネージド型ノードグループを作成することはできません。ただし、Amazon EC2 API、AWS CloudFormation、または `eksctl` を使用して、Local Zones でセルフマネージド型の Amazon EC2 ノードを作成できます。詳細については、「セルフマネージド型ノード」を参照してください。

ネットワークアーキテクチャ

- Amazon EKS が管理する Kubernetes コントロール プレーンは、常に AWS リージョン で実行されます。Amazon EKS が管理する Kubernetes コントロールプレーンは、ローカル ゾーンでは実行できません。Local Zones は VPC 内のサブネットとして表示されるため、Kubernetes は Local Zones リソースをそのサブネットの一部として認識します。
- Amazon EKS Kubernetes クラスターは、Amazon EKS が管理する エラスティックネットワークインターフェース を使用して AWS リージョン またはローカルゾーンで実行する Amazon EC2 インスタンスと通信します。Amazon EKS ネットワークアーキテクチャの詳細については、「Amazon EKS ネットワーク」を参照してください。
- リージョンサブネットとは異なり、Amazon EKS は Local Zone サブネットにネットワークインターフェイスを配置できません。つまり、クラスターの作成時に Local Zone サブネットを指定することはできません。

深層学習コンテナ

AWS Deep Learning Containers は、Amazon EKS および Amazon Elastic Container Service (Amazon ECS) の TensorFlow でモデルをトレーニングおよび提供するための Docker イメージのセットです。深層学習コンテナは、TensorFlow、NVIDIA CUDA (GPU インスタンス用)、Intel MKL (CPU インスタンス用) ライブラリを使用して最適化された環境を提供します。これは Amazon ECR で利用できます。

Amazon EKS で AWS 深層学習コンテナの使用を開始するには、「AWS 深層学習コンテナデベロッパーガイド」の「Amazon EKS の設定」を参照してください。

Amazon VPC Lattice

Amazon VPC Lattice は、AWS ネットワークインフラストラクチャに直接組み込まれた完全マネージド型のアプリケーションネットワーキングサービスで、複数のアカウントや仮想プライベートクラウド (VPC) を通してサービスの接続、保護、モニタリングに使用できます。Amazon EKS では、Kubernetes「ゲートウェイ API」の実装である AWS ゲートウェイ API コントローラーを使用して Amazon VPC Lattice を活用できます。Amazon VPC Lattice を使用すると、標準的な Kubernetes セマンティクスでクラスター間の接続を簡単かつ一貫した方法でセットアップできます。Amazon EKS で Amazon VPC Lattice の使用を開始するには、「AWS ゲートウェイ API コントローラのユーザーガイド」を参照してください。

AWS Resilience Hub

AWS Resilience Hub は、Amazon EKS クラスターのインフラストラクチャを分析して、その耐障害性を評価します。AWS Resilience Hub は、Kubernetes のロールベースのアクセスコントロール (RBAC) 設定を使用して、クラスターにデプロイされた Kubernetes のワークロードを評価します。詳細については、「AWS Resilience Hub ユーザーガイド」の「Amazon EKS クラスターへの AWS Resilience Hub のアクセスを有効にする」を参照してください。

Amazon GuardDuty で脅威を検出する

Amazon GuardDuty は、AWS 環境内のアカウント、コンテナ、ワークロード、データを保護する脅威検知サービスです。GuardDuty は、機械学習（ML）モデル、異常および脅威検出機能を使用して、さまざまなログソースとランタイムアクティビティを継続的に監視し、環境内の潜在的なセキュリティリスクと悪意のあるアクティビティを特定して優先順位を付けます。

GuardDuty には、EKS クラスターに対する潜在的な脅威を検出する EKS Protection とランタイムモニタリングの 2 つの機能があります。

EKS Protection

この機能は、関連する Kubernetes 監査ログをモニタリングすることで Amazon EKS クラスターを保護するのに役立つ脅威検出カバレッジを提供します。Kubernetes 監査ログは、ユーザー、Kubernetes API を使用するアプリケーション、コントロールプレーンからのアクティビティなど、クラスター内のシーケンシャルアクションをキャプチャします。例えば、GuardDuty は、Kubernetes クラスター内のリソースを改ざんするために呼び出された可能性のある API が、認証されていないユーザーによって呼び出された場合、そのことを識別できます。

EKS Protection を有効にすると、GuardDuty は継続的な脅威検出のためにのみ Amazon EKS 監査ログにアクセスできるようになります。GuardDuty によりクラスターに対する潜在的な脅威が識別されると、特定のタイプの関連する Kubernetes 監査ログの検出結果を生成します。Kubernetes 監査ログから得られる検出結果のタイプの詳細については、「Amazon GuardDuty ユーザーガイド」の「Kubernetes 監査ログの検出結果タイプ」を参照してください。

詳細については、「Amazon GuardDuty ユーザーガイド」の「EKS Protection」を参照してください。

Runtime Monitoring

この機能は、オペレーティングシステムレベル、ネットワーク、ファイルのイベントを監視して分析し、環境内の特定の AWS ワークロードにおける潜在的な脅威を検出するのに役立ちます。

ランタイムモニタリングを有効にして Amazon EKS クラスターに GuardDuty エージェントをインストールすると、GuardDuty はこのクラスターに関連付けられたランタイムイベントのモニタリングを開始します。GuardDuty がクラスターに対する潜在的な脅威を特定すると、関連するランタイムモニタリングの結果を生成します。例えば、脅威は、脆弱なウェブアプリケーションを実行している 1 つのコンテナを危険にさらすことから始まる可能性があります。このウェブアプリケーションには、基盤となるコンテナとワークロードへのアクセス権限が存在する可能性があります。この場合、認証情報が正しく設定されていないと、アカウントとその中に保存されているデータへのアクセスを制御できない可能性があります。

ランタイムモニタリングを設定するには Amazon EKS アドオンとしてクラスターに GuardDuty エージェントをインストールします。アドオンの詳細については、「Amazon EKS で利用可能な Amazon EKS アドオン」を参照してください。

詳細については、「Amazon GuardDuty ユーザーガイド」の「ランタイムモニタリング」を参照してください。

Amazon Security Lake を Amazon EKS で使用する

Amazon Security Lake は、Amazon EKS を含むさまざまなソースからのセキュリティデータを一元化するフルマネージドのセキュリティデータレイクサービスです。Amazon EKS を Security Lake と統合することで、Kubernetes リソースで実行されるアクティビティに関するより深いインサイトを入手し、Amazon EKS クラスターのセキュリティ体制を強化することができます。

> **Note**
>
> Amazon EKS で Security Lake を使用する方法、およびデータソースを設定する方法の詳細については、Amazon Security Lake のドキュメントを参照してください。

Security Lake を Amazon EKS で使用する利点

セキュリティデータを一元化 - Security Lake は、Amazon EKS クラスターのセキュリティデータを、他の AWS のサービス、SaaS プロバイダー、オンプレミスソース、サードパーティーソースのデータとともに、自動的に収集して一元化します。これにより、組織全体のセキュリティ体制を包括的に把握することができます。

標準化されたデータ形式 — Security Lake は、収集したデータを標準的なオープンソースのスキーマである Open Cybersecurity Schema Framework (OCSF) 形式に変換します。この標準化により、分析が容易になり、他のセキュリティツールやサービスとの統合が可能になります。

脅威検出の向上 — Amazon EKS コントロールプレーンのログを含む一元化されたセキュリティデータを分析することで、Amazon EKS クラスター内の疑わしいアクティビティをより効果的に検出することができます。これにより、セキュリティインシデントをすみやかに特定し、対応することができます。

データ管理の簡素化 — Security Lake は、カスタマイズ可能な保持とレプリケーションの設定によって、セキュリティデータのライフサイクルを管理します。これにより、データ管理のタスクが簡素化され、コンプライアンスや監査に必要なデータを確実に保持することができます。

Amazon EKS の Security Lake の有効化

Amazon EKS で Security Lake の使用を開始するには、以下のステップに従います。

1. EKS クラスターの Amazon EKS コントロールプレーンログ記録を有効にします。詳細な手順については、「コントロールプレーンログの有効化と無効化」を参照してください。
2. Amazon EKS 監査ログを Security Lake のソースとして追加します。次に、Security Lake が、EKS クラスターで実行中の Kubernetes リソースで行われたアクティビティの詳細情報の収集を開始します。
3. 要件に基づいて、Security Lake のセキュリティデータの保持とレプリケーションを設定します。
4. Security Lake に保存されている標準化された OCSF データは、インシデント対応、セキュリティ分析、その他の AWS のサービスやサードパーティーツールとの統合に使用できます。例えば、Amazon OpenSearch Ingestion を使用して Amazon Security Lake のデータからセキュリティインサイトを生成することができます。

Security Lake の EKS ログの分析

Security Lake は EKS ログイベントを OCSF 形式に標準化し、データの分析や、他のセキュリティイベントとの関連付けを簡単に行えるようにします。標準化されたデータは、Amazon Athena、Amazon QuickSight、サードパーティーのセキュリティ分析ツールなどさまざまなツールやサービスを使ってクエリしたり、可視化したりできます。

EKS ログイベントの OCSF マッピングに関する詳細は、OCSF GitHub リポジトリの「mapping reference」を参照してください。

Amazon Detective

Amazon Detective を使用すると、セキュリティに関する検出結果や疑わしいアクティビティの根本原因を分析、調査、および迅速に特定できます。Detective は、AWS リソースからログデータを自動的に収集します。その後、機械学習、統計分析、グラフ理論を使用して、セキュリティ調査をより迅速かつ効率的に行うのに役立つビジュアライゼーションを生成します。Detective の事前に作成されたデータの集計、要約、およびコンテキストは、考えられるセキュリティ問題の性質と範囲を迅速に分析および特定するのに役立ちます。詳細については、「Amazon Detective ユーザーガイド」を参照してください。

Detective は、Kubernetes と AWS のデータを次のような結果に整理します。

- Amazon EKS クラスターの詳細 (クラスターやクラスターのサービスロールを作成した IAM ID など)。Detective を使用して、これらの IAM ID の AWS および Kubernetes API アクティビティを調査できます。
- コンテナの詳細 (イメージやセキュリティコンテキストなど)。終了済みの Pods の詳細を確認することもできます。
- Kubernetes API アクティビティ (API アクティビティの全体的な傾向と特定の API 呼び出しの詳細の両方を含む)。例えば、選択した時間範囲中に発行されて、成功および失敗した Kubernetes API 呼び出しの数を表示できます。また、新たに確認された API 呼び出しに関するセクションは、疑わしいアクティビティを特定するのに役立つ場合があります。

Amazon EKS 監査ログは、Detective の動作グラフに追加できるオプションのデータソースパッケージです。利用可能なオプションのソースパッケージとそのステータスは、アカウント内で確認できます。詳細については、「Amazon Detective ユーザーガイド」の「Amazon EKS audit logs for Detective」を参照してください。

Amazon EKS との連携で Amazon Detective を使用する

Amazon EKS クラスターの結果を確認する方法

結果を確認できるようにするには、クラスターが配置されているのと同じ AWS リージョンで、Detective を少なくとも 48 時間有効にする必要があります。詳細については、「Amazon Detective ユーザーガイド」の「Amazon Detective の設定」を参照してください。

1. https://console.aws.amazon.com/detective/ で Detective コンソールを開きます。
2. 左のナビゲーションペインで [検索] を選択します。

3. [タイプを選択] を選択し、[EKS クラスター] を選択します。
4. クラスター名または ARN を入力し、[検索] を選択します。
5. 検索結果で、アクティビティを表示するクラスターの名前を選択します。表示できる内容の詳細については、「Amazon Detective ユーザーガイド」の「Overall Kubernetes API activity involving an Amazon EKS cluster」を参照してください。

Amazon EKS のトラブルシューティング

この章では、Amazon EKS の使用中に表示される一般的なエラーとその回避方法について説明します。特定の Amazon EKS 領域のトラブルシューティングが必要な場合、別の IAM のトラブルシューティング、Amazon EKS Connector での問題のトラブルシューティング、および「EKS アドオンを使用した ADOT のトラブルシューティング」を参照してください。

その他のトラブルシューティング情報については、AWS re:Postの 「Amazon Elastic Kubernetes Service に関するナレッジセンターのコンテンツ」を参照してください。

容量不足

Amazon EKS クラスターを作成しようとするときに次のエラーが表示された場合、指定したいずれかのアベイラビリティーゾーンに、クラスターをサポートするための十分な容量がありません。

```
Cannot create cluster 'example-cluster' because region-1d, the targeted Availability Zone, does not currently have sufficient capacity to support the cluster. Retry and choose from these Availability Zones: region-1a, region-1b, region-1c
```

このエラーメッセージによって返されるアベイラビリティーゾーンでホストされているクラスター VPC 内のサブネットを使用して、クラスターを再作成してください。

クラスターを配置できないアベイラビリティーゾーンがあります。サブネットが属するアベイラビリティーゾーンを、サブネットの要件と考慮事項 内のアベイラビリティーゾーンのリストと比較してください。

ノードをクラスターに結合できません

ノードをクラスターに結合できない一般的な理由はいくつかあります。

- ノードがマネージド型の場合、Amazon EKS はノードグループの作成時に、aws-auth ConfigMap にエントリを追加します。エントリが削除または変更された場合は、再度追加する必要があります。詳細については、ターミナルに **eksctl create iamidentitymapping --help** を入力してください。現在の aws-auth ConfigMap エントリを確認するには、次のコマンドの *my-cluster* をクラスターの名前に置き換えて、変更したコマンド **eksctl get iamidentitymapping --cluster *my-cluster*** を実行します。指定するロールの ARN には、/ 以外のパスを含めることはできません。例えば、ロールの名前が development/apps/

my-role の場合、ロールの ARN を指定するときに my-role に変更する必要があります。ノード IAM ロール ARN (インスタンスプロファイルの ARN ではない) を指定していることを確認してください。

ノードがセルフマネージド型で、ノードの IAM ロールの ARN のアクセスエントリを作成していない場合は、マネージド型ノード用にリストされているのと同じコマンドを実行します。ノード IAM ロールの ARN のアクセスエントリを作成している場合、そのエントリがアクセスエントリで正しく設定されていない可能性があります。aws-auth ConfigMap エントリまたはアクセスエントリのプリンシパル ARN として、ノード IAM ロール ARN (インスタンスプロファイルの ARN ではない) が指定されていることを確認してください。アクセスエントリの詳細については、「アクセスエントリを管理する」を参照してください。

- ノード AWS CloudFormation テンプレートの ClusterName が、ノードを結合するクラスターの名前と正確に一致しません。このフィールドに正しくない値を渡すと、ノードの /var/lib/kubelet/kubeconfig ファイルの設定が正しくないために、ノードはクラスターに結合されません。
- ノードは、クラスターによって所有されているためタグ付けされていません。ノードには、次のタグが適用されている必要があります。*my-cluster* はクラスターの名前に置き換えられます。

キー	値
kubernetes.io/cluster/*my-cluster*	owned

- ノードは、パブリック IP アドレスを使用してクラスターにアクセスできない場合があります。パブリックサブネットにデプロイされたノードにパブリック IP アドレスが割り当てられていることを確認してください。割り当てられていない場合は、起動後にノードに Elastic IP アドレスを関連付けることができます。詳細については、「Elastic IP アドレスをインスタンスまたはネットワークインターフェイスに関連付ける」を参照してください。デプロイされたインスタンスにパブリック IP アドレスを自動的に割り当てるようにパブリックサブネットが設定されていない場合は、その設定を有効にすることをお勧めします。詳細については、「サブネットのパブリック IPv4 アドレッシング属性の変更」を参照してください。ノードがプライベートサブネットにデプロイされている場合、サブネットには、パブリック IP アドレスが割り当てられた NAT ゲートウェイへのルートが必要です。
- ノードのデプロイ先の AWS リージョン の AWS STS エンドポイントは、アカウントに対して有効になっていません。リージョンを有効にするには、「AWS リージョン での AWS STS のアクティブ化と非アクティブ化」を参照してください。

- ノードにはプライベート DNS エントリがないため、node "" not found エラーを含む kubelet ログが発生する可能性があります。ノードが作成される VPC に、DHCP options set の Options として domain-name と domain-name-servers の値のセットがあることを確認してください。デフォルト値は domain-name:<region>.compute.internal および domain-name-servers:AmazonProvidedDNS です。詳細については、「Amazon VPC ユーザーガイド」の「DHCP オプションセット」を参照してください。
- マネージドノードグループ内のノードが 15 分以内にクラスターに接続しない場合、「NodeCreationFailure」という正常性の問題が出力され、コンソールのステータスが Create failed に設定されます。起動時間が遅い Windows AMI の場合、この問題は高速起動を使用して解決できます。

ワーカーノードがクラスターに参加できない一般的な原因を特定してトラブルシューティングするには、AWSSupport-TroubleshootEKSWorkerNode ランブックを使用できます。詳細については、「AWS Systems Manager Automation ランブックリファレンス」の「AWSSupport-TroubleshootEKSWorkerNode」を参照してください。

許可されていないか、アクセスが拒否されました (kubectl)

kubectl コマンドの実行中に次のいずれかのエラーが発生した場合は、kubectl が Amazon EKS に対して適切に設定されていないか、使用している IAM プリンシパルの認証情報 (ロールまたはユーザー) が、Amazon EKS クラスターで Kubernetes オブジェクトに対して十分なアクセス許可を持つ Kubernetes ユーザー名にマッピングされていません。

- could not get token: AccessDenied: Access denied
- error: You must be logged in to the server (Unauthorized)
- error: the server doesn't have a resource type "svc"

この原因としては、次のいずれかが考えられます。

- クラスターはある IAM プリンシパルの認証情報を使用して作成されており、kubectl は別の IAM プリンシパルの認証情報を使用するよう設定されています。この問題を解決するには、クラスターを作成した認証情報を使用するように kube config ファイルを更新してください。詳細については、「Amazon EKS クラスターの kubeconfig ファイルを作成または更新する」を参照してください。
- クラスターが アクセスエントリを管理する の前提条件セクションの最小プラットフォーム要件を満たしている場合、IAM プリンシパルにはアクセスエントリは存在しません。存在する場合、必

要な Kubernetes グループ名が定義されていないか、適切なアクセスポリシーが関連付けられていません。詳細については、「アクセスエントリを管理する」を参照してください。

- クラスターが アクセスエントリを管理する の最小プラットフォーム要件を満たしていない場合、IAM プリンシパルのエントリは aws-auth ConfigMap に存在しません。存在しても、Kubernetes Role または ClusterRole にバインドされている、または必要なアクセス許可を持つ Kubernetes グループ名にはマップされません。Kubernetes ロールベース認証 (RBAC) オブジェクトの詳細については、Kubernetes ドキュメントの「RBAC 承認の使用」を参照してください。現在の aws-auth ConfigMap エントリを確認するには、次のコマンドの *my-cluster* をクラスターの名前に置き換えて、変更したコマンド **eksctl get iamidentitymapping --cluster *my-cluster*** を実行します。IAM プリンシパルの ARN を含むエントリが ConfigMap にない場合は、ターミナルに **eksctl create iamidentitymapping --help** を入力して作成方法を確認してください。

AWS CLI をインストールして設定する場合は、使用する IAM 認証情報を設定できます。詳細については、「AWS CLI ユーザーガイド」の「AWS Command Line Interface の設定。」を参照してください。クラスター上の Kubernetes オブジェクトにアクセスする IAM ロールを引き受ける場合は、IAM ロールを使用するように kubectl を設定することもできます。詳細については、「Amazon EKS クラスターの kubeconfig ファイルを作成または更新する」を参照してください。

hostname doesn't match

システムの Python のバージョンは 2.7.9 以降であることが必要です。それ以外の場合は、AWS CLI で Amazon EKS を呼び出すと hostname doesn't match エラーが表示されます。詳細については、「Python Requests のよくある質問」の「What are "hostname doesn't match" errors?」を参照してください。

getsockopt: no route to host

Docker は、Amazon EKS クラスターの 172.17.0.0/16 CIDR 範囲で実行されます。クラスターの VPC サブネットがこの範囲と重ならないようにすることをお勧めします。重なっている場合は、以下のエラーが発生します。

```
Error: : error upgrading connection: error dialing backend: dial tcp
 172.17.<nn>.<nn>:10250: getsockopt: no route to host
```

Instances failed to join the Kubernetes cluster

AWS Management Console で Instances failed to join the Kubernetes cluster エラーが表示された場合、クラスターのプライベートエンドポイントアクセスが有効になっているか、パブリックエンドポイントアクセス用に CIDR ブロックが正しく設定されていることを確認します。詳細については、「Amazon EKS クラスターエンドポイントアクセスコントロール」を参照してください。

マネージド型ノードグループのエラー

マネージド型ノードグループでハードウェアのヘルスに問題が発生した場合は、Amazon EKS によって問題の診断に役立つエラーメッセージが返されます。これらのヘルスチェックは Amazon EC2 ヘルスチェックに基づいているため、ソフトウェアの問題は検出されません。次のリストは、エラーコードについての説明です。

AccessDenied

Amazon EKS または 1 つ以上の管理対象ノードが、Kubernetes クラスター API サーバーでの認証または承認に失敗しています。共通のレスポンスヘッダーの詳細については、「AccessDeniedマネージド型ノードグループエラーを修正する」を参照してください。プライベートWindows AMI では、Not authorized for imagesエラーメッセージとともにこのエラーコードが表示されることもあります。詳細については、「Not authorized for images」を参照してください。

AmiIdNotFound

起動テンプレートに関連付けられている AMI ID が見つかりませんでした。AMI が存在し、アカウントと共有されていることを確認してください。

AutoScalingGroupNotFound

マネージド型ノードグループに関連付けられている Auto Scaling グループが見つかりませんでした。同じ設定で Auto Scaling グループを再作成して復旧できる場合があります。

ClusterUnreachable

Amazon EKS または 1 つ以上の管理対象ノードが Kubernetes クラスター API サーバーと通信できません。このエラーはネットワークが中断したり、API サーバーがリクエストの処理をタイムアウトしている場合に発生する可能性があります。

Ec2SecurityGroupNotFound

クラスターのクラスターセキュリティグループが見つかりませんでした。クラスターを再作成する必要があります。

Ec2SecurityGroupDeletionFailure

マネージド型ノードグループのリモートアクセスセキュリティグループを削除できませんでした。セキュリティグループから依存関係を削除します。

Ec2LaunchTemplateNotFound

マネージド型ノードグループの Amazon EC2 起動テンプレートが見つかりませんでした。復旧には、ノードグループを再作成する必要があります。

Ec2LaunchTemplateVersionMismatch

マネージド型ノードグループの Amazon EC2 起動テンプレートのバージョンが、Amazon EKS が作成したバージョンと一致しません。Amazon EKS が作成したバージョンに戻すことで復旧できる場合があります。

IamInstanceProfileNotFound

マネージド型ノードグループの IAM インスタンスプロファイルが見つかりませんでした。同じ設定でインスタンスプロファイルを再作成して復旧できる場合があります。

IamNodeRoleNotFound

マネージド型ノードグループの IAM ロールが見つかりませんでした。同じ設定で IAM ロールを再作成して復旧できる場合があります。

AsgInstanceLaunchFailures

インスタンスの起動中に Auto Scaling グループに障害が発生しています。

NodeCreationFailure

起動したインスタンスを Amazon EKS クラスターに登録できません。この障害の一般的な原因は、ノード IAM ロールのアクセス許可が不十分か、ノードのアウトバウンドインターネットアクセスがないことです。ノードは以下の要件のいずれかを満たしている必要があります。

- パブリック IP アドレスを使用してインターネットにアクセス可能です。ノードが存在するサブネットに関連付けられているセキュリティグループが、通信を許可する必要があります。詳細については、サブネットの要件と考慮事項およびAmazon EKS セキュリティグループの要件および考慮事項を参照してください。

- ノードと VPC は、プライベートクラスターの要件 に記載される要件を満たしている必要があります。

InstanceLimitExceeded

AWS アカウントが、指定されたインスタンスタイプのインスタンスをこれ以上起動できません。Amazon EC2 のインスタンス制限の引き上げをリクエストして復旧できる場合があります。

InsufficientFreeAddresses

マネージド型ノードグループに関連付けられている 1 つ以上のサブネットに、新しいノードに使用できる十分な IP アドレスがありません。

InternalFailure

これらのエラーは通常、Amazon EKS サーバー側の問題が原因で発生します。

`AccessDenied`マネージド型ノードグループエラーを修正する

マネージド型ノードグループで操作を実行する際に `AccessDenied` エラーが発生する最も一般的な原因は、`eks:node-manager`、`ClusterRole` または `ClusterRoleBinding` がないことです。Amazon EKS は、マネージド型ノードグループでのオンボーディングの一環として、クラスター内にこれらのリソースを設定します。これらのリソースは、ノードグループの管理に必要です。

`ClusterRole` は時間の経過とともに変化する可能性がありますが、次の例のようになります。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: eks:node-manager
rules:
- apiGroups:
  - ''
  resources:
  - pods
  verbs:
  - get
  - list
  - watch
  - delete
- apiGroups:
  - ''
  resources:
```

```
  - nodes
  verbs:
  - get
  - list
  - watch
  - patch
- apiGroups:
  - ''
  resources:
  - pods/eviction
  verbs:
  - create
```

ClusterRoleBinding は時間の経過とともに変化する可能性がありますが、次の例のようになります。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks:node-manager
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: eks:node-manager
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: eks:node-manager
```

eks:node-manager ClusterRole が存在することを確認します。

```
kubectl describe clusterrole eks:node-manager
```

存在する場合は、出力を以前の ClusterRole の例と比較します。

eks:node-manager ClusterRoleBinding が存在することを確認します。

```
kubectl describe clusterrolebinding eks:node-manager
```

存在する場合は、出力を以前の ClusterRoleBinding の例と比較します。

マネージド型ノードグループの操作のリクエスト中に AcessDenied エラーの原因として ClusterRole や ClusterRoleBinding が紛失または破損していることを発見した場合は、それらを復元できます。次の内容を *eks-node-manager-role.yaml* という名前のファイルに保存します。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: eks:node-manager
rules:
- apiGroups:
  - ''
  resources:
  - pods
  verbs:
  - get
  - list
  - watch
  - delete
- apiGroups:
  - ''
  resources:
  - nodes
  verbs:
  - get
  - list
  - watch
  - patch
- apiGroups:
  - ''
  resources:
  - pods/eviction
  verbs:
  - create
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks:node-manager
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: eks:node-manager
```

```
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: eks:node-manager
```

ファイルを適用します。

```
kubectl apply -f eks-node-manager-role.yaml
```

ノードグループの操作を再試行して、問題が解決したかどうかを確認します。

Not authorized for images

Not authorized for images エラーメッセージの考えられる原因の 1 つは、プライベート Amazon EKS Windows AMI Windows を使用してマネージドノードグループを起動することです。新しい Windows AMI をリリースすると、AWS は 4 か月以上前の AMI を非公開にし、アクセス不可にします。マネージド型ノードグループがプライベート Windows AMI を使用している場合は、Windows マネージド型ノードグループを更新することを検討してください。非公開になった AMI へのアクセスを可能にできるかは保証できませんが、AWS サポートにチケットを提出することでアクセスをリクエストできます。詳細については、「Amazon EC2 ユーザーガイド」の「Patches, security updates, and AMI IDs」を参照してください。

ノードが NotReady 状態です

ノードが NotReady 状態になった場合、ノードに異常があり、新しい Pods をスケジュールできない可能性があります。これは、ノードに CPU、メモリ、利用可能なディスクスペースの十分なリソースがないなど、さまざまな理由で発生します。

Amazon EKS 最適化 Windows AMI の場合、kubelet 設定のデフォルトでは、コンピューティングリソースの予約は指定されていません。リソースの問題を防ぐために、kubelet に kube-reserved および/または system-reserved の設定値を指定することで、システムプロセスのコンピューティングリソースを予約できます。これを行うには、ブートストラップスクリプトの -KubeletExtraArgs コマンドラインパラメータを使用します。詳細については、「Kubernetes ドキュメント」の「Reserve Compute Resources for System Daemons」、およびこのユーザーガイドの「ブートストラップスクリプトの設定パラメータ」を参照してください。

CNI ログ収集ツール

Amazon VPC CNI plugin for Kubernetes には、/opt/cni/bin/aws-cni-support.sh のノードで使用できる独自のトラブルシューティング スクリプトがあります。このスクリプトを使用して、サポートケースおよび一般的なトラブルシューティングの診断ログを収集できます。

ノードでスクリプトを実行するには、次のコマンドを使用します。

```
sudo bash /opt/cni/bin/aws-cni-support.sh
```

> **Note**
>
> 指定された場所にスクリプトが存在しない場合は、CNI コンテナの実行に失敗します。手動でスクリプトをダウンロードして実行するには、次のコマンドを使用します。
>
> ```
> curl -O https://raw.githubusercontent.com/awslabs/amazon-eks-ami/master/log-collector-script/linux/eks-log-collector.sh
> sudo bash eks-log-collector.sh
> ```

このスクリプトは、次の診断情報を収集します。デプロイした CNI バージョンは、スクリプトバージョンより前のバージョンである可能性があります。

```
      This is version 0.6.1. New versions can be found at https://github.com/awslabs/amazon-eks-ami

Trying to collect common operating system logs...
Trying to collect kernel logs...
Trying to collect mount points and volume information...
Trying to collect SELinux status...
Trying to collect iptables information...
Trying to collect installed packages...
Trying to collect active system services...
Trying to collect Docker daemon information...
Trying to collect kubelet information...
Trying to collect L-IPAMD information...
Trying to collect sysctls information...
Trying to collect networking information...
Trying to collect CNI configuration information...
Trying to collect running Docker containers and gather container data...
```

```
Trying to collect Docker daemon logs...
Trying to archive gathered information...

 Done... your bundled logs are located in /var/
log/eks_i-0717c9d54b6cfaa19_2020-03-24_0103-UTC_0.6.1.tar.gz
```

診断情報が収集され、以下に保存されます。

```
/var/log/eks_i-0717c9d54b6cfaa19_2020-03-24_0103-UTC_0.6.1.tar.gz
```

コンテナランタイムネットワークの準備ができていません

以下のような Container runtime network not ready エラーと承認エラーが表示される場合があります。

```
4191 kubelet.go:2130] Container runtime network not ready: NetworkReady=false
 reason:NetworkPluginNotReady message:docker: network plugin is not ready: cni config
 uninitialized
4191 reflector.go:205] k8s.io/kubernetes/pkg/kubelet/kubelet.go:452: Failed to list
 *v1.Service: Unauthorized
4191 kubelet_node_status.go:106] Unable to register node
 "ip-10-40-175-122.ec2.internal" with API server: Unauthorized
4191 reflector.go:205] k8s.io/kubernetes/pkg/kubelet/kubelet.go:452: Failed to list
 *v1.Service: Unauthorized
```

これは、次のいずれかの理由で発生します。

1. クラスターに aws-auth ConfigMap がないか、ノードに設定した IAM ロールのエントリが含まれていないかのどちらかです。

   この ConfigMap エントリは、ノードが次の条件のうち 1 つを満たす場合に必要です。

   - 任意の Kubernetes またはプラットフォームバージョンのクラスター内のマネージド型ノード。
   - アクセスエントリを管理する トピックの前提条件セクションに記載されているいずれかのプラットフォームバージョンよりも前のクラスター内のセルフマネージド型ノード。

   この問題を解決するには、次のコマンドの *my-cluster* をクラスターの名前に置き換えて、変更したコマンド **eksctl get iamidentitymapping --cluster *my-cluster*** を実行することで、ConfigMap 内の現在のエントリを確認します。コマンドからエラーメッセージが表示される場合は、クラスターに aws-auth ConfigMap がないことが原因である可能性があ

ります。次のコマンドは ConfigMap にエントリを追加します。ConfigMap が存在しない場合は、コマンドによって作成されます。*111122223333* を IAM ロールの AWS アカウント ID に、*myAmazonEksNodeRole* をノードのロールの名前に置き換えます。

```
eksctl create iamidentitymapping --cluster my-cluster \
    --arn arn:aws:iam::111122223333:role/myAmazonEKSNodeRole --group
 system:bootstrappers,system:nodes \
    --username system:node:{{EC2PrivateDNSName}}
```

指定するロールの ARN には、/ 以外のパスを含めることはできません。例えば、ロールの名前が development/apps/my-role の場合、ロールの ARN を指定するときに my-role に変更する必要があります。ノード IAM ロール ARN (インスタンスプロファイルの ARN ではない) を指定していることを確認してください。

2. セルフマネージド型ノードが、アクセスエントリを管理する トピックの前提条件に記載されている最小バージョンのプラットフォームバージョンのクラスターにあるものの、ノードの IAM ロールのエントリが aws-auth ConfigMap (前の項目を参照) に記載されていないか、ロールのアクセスエントリが存在しません。この問題を解決するには、次のコマンドの *my-cluster* をクラスターの名前に置き換えて、変更したコマンド **aws eks list-access-entries --cluster-name *my-cluster*** を実行することで、現在のアクセスエントリを確認します。次のコマンドは、ノードの IAM ロールのアクセスエントリを追加します。*111122223333* を IAM ロールの AWS アカウント ID に、*myAmazonEksNodeRole* をノードのロールの名前に置き換えます。Windows ノードを使用している場合は、*EC2_Linux* を **EC2_Windows** に置き換えてください。ノード IAM ロール ARN (インスタンスプロファイルの ARN ではない) を指定していることを確認してください。

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/myAmazonEKSNodeRole --type EC2_Linux
```

TLS ハンドシェイクタイムアウト

ノードがパブリック API サーバーエンドポイントへの接続を確立できない場合、次のようなエラーが発生する可能性があります。

```
server.go:233] failed to run Kubelet: could not init cloud provider "aws": error
 finding instance i-1111f2222f333e44c: "error listing AWS instances: \"RequestError:
 send request failed\\ncaused by: Post  net/http: TLS handshake timeout\""
```

kubelet プロセスは、API サーバーエンドポイントを継続的に再生成およびテストします。このエラーは、コントロールプレーンでクラスターのローリング更新を行う手順 (設定の変更やバージョンの更新など) で一時的に発生することもあります。

この問題を解決するには、ルートテーブルとセキュリティグループを確認して、ノードからのトラフィックがパブリックエンドポイントに到達できることを確認します。

InvalidClientTokenId

中国 AWS リージョン でクラスターにデプロイされている Pod または DaemonSet 用サービスアカウントの IAM ロールを使用していて、spec に AWS_DEFAULT_REGION 環境可変を設定していない場合、Pod または DaemonSet に次のエラーが表示されることがあります。

```
An error occurred (InvalidClientTokenId) when calling the GetCallerIdentity operation:
 The security token included in the request is invalid
```

この問題を解決するには、次の Pod spec の例に示すように、AWS_DEFAULT_REGION 環境変数を Pod または DaemonSet spec に追加する必要があります。

```
apiVersion: v1
kind: Pod
metadata:
  name: envar-demo
  labels:
    purpose: demonstrate-envars
spec:
  containers:
  - name: envar-demo-container
    image: gcr.io/google-samples/node-hello:1.0
    env:
    - name: AWS_DEFAULT_REGION
      value: "region-code"
```

VPC アドミッションウェブフック証明書の有効期限切れ

VPC アドミッション Webhook の署名に使用された証明書の有効期限が切れた場合、新しい Windows Pod 展開のステータスは ContainerCreating のままになります。

データプレーンでレガシー Windows サポートがある場合に問題を解決するには、VPC アドミッションウェブフック証明書の更新 を参照してください。クラスタとプラットフォームのバージョンが

Windows サポートの前提条件 に記載されているバージョンよりも遅い場合は、データプレーンでレガシー Windows サポートを削除し、コントロールプレーンで有効にすることをお勧めします。一度行えば、ウェブフック証明書を管理する必要はありません。詳細については、「Amazon EKS クラスター の Windows サポートの有効化」を参照してください。

コントロールプレーンをアップグレードする前に、ノードグループが Kubernetes バージョンと一致する必要があります

コントロールプレーンを新しい Kubernetes バージョンにアップグレードする前に、クラスター内のマネージドノードと Fargate ノードのマイナーバージョンが、コントロールプレーンの現在のバージョンのバージョンと同じである必要があります。Amazon EKS `update-cluster-version` API は、すべての EKS マネージド型ノードが現在のクラスターバージョンにアップグレードされるまで、リクエストを拒否します。Amazon EKS は、マネージド型ノードをアップグレードするための API を提供します。マネージド型ノードグループの Kubernetes バージョンのアップグレードについては、マネージド型ノードグループの更新 を参照してください。Fargate ノードのバージョンをアップグレードするには、ノードによって表される pod を削除し、コントロールプレーンをアップグレードした後に pod を再デプロイします。詳細については、「Amazon EKS クラスターの Kubernetes バージョンの更新」を参照してください。

多数のノードを起動すると、`Too Many Requests` エラーが発生します。

多数のノードを同時に起動すると、Amazon EC2 ユーザーデータの実行ログに「`Too Many Requests`」というエラーメッセージが表示される場合があります。これは、コントロールプレーンが `describeCluster` 呼び出しで過負荷になっているために発生する可能性があります。過負荷が原因で、スロットリングが発生し、ノードがブートストラップスクリプトを実行できなくなり、ノードが完全にクラスターに参加できなくなります。

`--apiserver-endpoint`、`--b64-cluster-ca`、および `--dns-cluster-ip` 引数がノードブートストラップスクリプトに渡されていることを確認してください。これらの引数を含める場合、ブートストラップスクリプトが `describeCluster` 呼び出しを行う必要がなくなり、コントロールプレーンが過負荷になるのを防ぐのに役立ちます。詳細については、「Amazon EKS に最適化された Linux/Bottlerocket AMI に含まれる `bootstrap.sh` ファイルに引数を渡すためのユーザーデータを提供する」を参照してください。

Kubernetes API サーバーリクエストでの HTTP 401 不正エラーレスポンス

これらのエラーは、Pod のサービスアカウントトークンがクラスターで期限切れになった場合に表示されます。

Amazon EKS クラスターの Kubernetes API サーバーでは、90 日を超えるトークンのリクエストは拒否されます。以前の Kubernetes バージョンでは、トークンに有効期限がありませんでした。つまり、これらのトークンに依存しているクライアントは、1 時間以内にトークンを更新する必要があります。無効なトークンが原因で Kubernetes APIサーバーがリクエストを拒否しないようにするには、ワークロードで使用される Kubernetes クライアント SDK のバージョンが、次のバージョンと同じか、それ以降である必要があります。

- Go バージョン `0.15.7` 以降
- Python バージョン `12.0.0` 以降
- Java バージョン `9.0.0` 以降
- JavaScript バージョン `0.10.3` 以降
- Ruby `master` ブランチ
- Haskell バージョン`0.3.0.0`
- C# バージョン `7.0.5` 以降

以前のトークンを使用しているクラスター内の既存の Pods をすべて識別できます。詳細については、「Kubernetes サービス アカウント」を参照してください。

Amazon EKS プラットフォームのバージョンは、現在のプラットフォーム バージョンより 2 つ以上遅れています

これは、Amazon EKS がクラスターの プラットフォームバージョン を自動的に更新できない場合に発生する可能性があります。これには多くの原因がありますが、一般的な原因のいくつかが続きます。これらの問題のいずれかがクラスターに当てはまる場合、それはまだ機能する可能性がありますが、そのプラットフォームバージョンは Amazon EKS によって更新されません。

問題

クラスターIAMロール が削除されました - このロールは、クラスターの作成時に指定されました。次のコマンドで、どのロールが指定されたかを確認できます。*my-cluster* の部分は、自分のクラスター名に置き換えます。

```
aws eks describe-cluster --name my-cluster --query cluster.roleArn --output text | cut
 -d / -f 2
```

出力例は次のとおりです。

```
eksClusterRole
```

ソリューション

同じ名前で新しい クラスター IAM ロール を作成します。

問題

クラスターの作成中に指定されたサブネットが削除されました - クラスターで使用するサブネットは、クラスターの作成中に指定されました。次のコマンドで、指定されたサブネットを確認できます。*my-cluster* の部分は、自分のクラスター名に置き換えます。

```
aws eks describe-cluster --name my-cluster --query cluster.resourcesVpcConfig.subnetIds
```

出力例は次のとおりです。

```
[
"subnet-EXAMPLE1",
"subnet-EXAMPLE2"
]
```

ソリューション

アカウントにサブネット ID が存在するかどうかを確認します。

```
vpc_id=$(aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.vpcId --output text)
aws ec2 describe-subnets --filters "Name=vpc-id,Values=$vpc_id" --query
 "Subnets[*].SubnetId"
```

出力例は次のとおりです。

```
[
"subnet-EXAMPLE3",
"subnet-EXAMPLE4"
]
```

出力で返されたサブネット ID が、クラスターの作成時に指定されたサブネット ID と一致しない場合、Amazon EKS でクラスターを更新するには、クラスターで使用されるサブネットを変更する必要があります。これは、クラスターの作成時に 2 つ以上のサブネットを指定した場合、Amazon EKS は指定したサブネットをランダムに選択して新しいエラスティックネットワークインターフェイスを作成するためです。これらのネットワーク インターフェイスにより、コントロールプレーンはノードと通信できます。Amazon EKS は、選択したサブネットが存在しない場合、クラスターを更新しません。Amazon EKS が新しいネットワーク インターフェイスを作成するために選択する、クラスターの作成時に指定したサブネットをコントロールすることはできません。

クラスターの Kubernetes バージョンの更新を開始すると、同じ理由で更新が失敗する可能性があります。

問題

クラスターの作成中に指定されたセキュリティグループが削除されました - クラスターの作成中にセキュリティグループを指定した場合は、次のコマンドでそれらの ID を確認できます。*my-cluster* の部分は、自分のクラスター名に置き換えます。

```
aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.securityGroupIds
```

出力例は次のとおりです。

```
[
    "sg-EXAMPLE1"
]
```

[] が返された場合、クラスターの作成時にセキュリティグループが指定されておらず、セキュリティグループの欠落は問題ではありません。セキュリティグループが返された場合は、セキュリティグループがアカウントに存在することを確認します。

ソリューション

これらのセキュリティグループがアカウントに存在するかどうかを確認します。

```
vpc_id=$(aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.vpcId --output text)
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=$vpc_id" --query
 "SecurityGroups[*].GroupId"
```

出力例は次のとおりです。

```
[
"sg-EXAMPLE2"
]
```

出力で返されたセキュリティグループ ID が、クラスターの作成時に指定されたセキュリティグループ ID と一致しない場合、Amazon EKS でクラスターを更新するには、クラスターで使用されるセキュリティグループを変更する必要があります。クラスターの作成時に指定されたセキュリティグループ ID が存在しない場合、Amazon EKS はクラスターを更新しません。

クラスターの Kubernetes バージョンの更新を開始すると、同じ理由で更新が失敗する可能性があります。

Amazon EKS がクラスターのプラットフォームバージョンを更新しないその他の理由

- クラスターの作成時に指定した各サブネットで、少なくとも 6 つ (ただし、16 をお勧めします) の使用可能な IP アドレスがありません。サブネットに十分な使用可能な IP アドレスがない場合は、サブネット内の IP アドレスを解放するか、十分な使用可能な IP アドレスを持つサブネットを使用するように、クラスターで使用されるサブネットを変更する必要があります。
- クラスターの作成時に シークレットの暗号化 を有効にし、指定した AWS KMS キーが削除されました。Amazon EKS でクラスターを更新する場合は、新しいクラスターを作成する必要があります

クラスターの正常性に関するよくある質問およびエラーコードと解決パス

Amazon EKS は EKS クラスターとクラスターインフラストラクチャの問題を検出し、クラスターヘルスに保存します。クラスターヘルスの情報を利用することで、クラスターの問題をより迅速に検出、トラブルシューティング、対処できます。これにより、より安全で最新のアプリケーション環境を構築できます。さらに、必要なインフラストラクチャまたはクラスター設定に問題があるた

め、Kubernetes の新しいバージョンにアップグレードしたり、Amazon EKS が劣化したクラスターにセキュリティ更新をインストールしたりできない場合があります。Amazon EKS は、問題を検出したり、問題が解決されたことを検出したりするまでに 3 時間かかる場合があります。

Amazon EKS クラスターの状態は、Amazon EKS とそのユーザー間で共有される責任です。IAM ロールや Amazon VPC サブネットの前提となるインフラストラクチャ、および事前に提供する必要があるその他の必要なインフラストラクチャは、お客様の責任となります。Amazon EKS は、このインフラストラクチャとクラスターの設定の変更を検出します。

Amazon EKS コンソールでクラスターの状態にアクセスするには、Amazon EKS クラスター詳細ページの [概要] タブにある [ヘルスの問題] というセクションを探してください。このデータは、EKS API の `DescribeCluster` アクションを、例えば AWS Command Line Interface 内から呼び出すことでも確認できます。

この機能を使用する理由は何ですか?

Amazon EKS クラスターの状態を把握しやすくなり、問題を迅速に診断して修正できます。デバッグや AWS サポートケースの作成に時間を費やす必要はありません。例えば、Amazon EKS クラスターのサブネットを誤って削除した場合、Amazon EKS はクロスアカウントのネットワークインターフェイスや `kubectl exec` や `kubectl` ログなどの Kubernetes AWS CLI コマンドを作成できなくなります。この場合、次のエラーとともに処理が失敗します: `Error from server: error dialing backend: remote error: tls: internal error.` そして、次のような Amazon EKS ヘルスの問題が表示されます: `subnet-da60e280 was deleted: could not create network interface`。

この機能は他の AWS サービスとどのように関連し、機能しますか?

IAM ロールと Amazon VPC サブネットは、クラスターヘルスが問題を検出するための前提条件となるインフラストラクチャの 2 つの例です。この機能は、これらのリソースが正しく設定されていない場合に詳細情報を返します。

ヘルスに問題があるクラスターには料金がかかりますか?

はい。Amazon EKS クラスターはすべて標準の Amazon EKS 料金で請求されます。クラスターヘルス機能は追加料金なしで利用できます。

この機能は AWS Outposts の Amazon EKS クラスターで動作しますか?

はい。クラスターの問題は、AWS Outposts の拡張クラスターと AWS Outposts のローカルクラスターを含む AWS クラウド内の EKS クラスターで検出されます。クラスターヘルスでは、Amazon EKS Anywhere や Amazon EKS Distro (EKS-D) の問題は検出されません。

新しい問題が検出されたときに通知を受けることはできますか?

いいえ、Amazon EKS コンソールを確認するか、EKS `DescribeCluster` API を呼び出す必要があります。

コンソールにはヘルスの問題に関する警告が表示されますか?

はい。ヘルスに問題があるクラスターには、コンソールの上部にバナーが表示されます。

最初の 2 つの列は API レスポンス値に必要なものです。Health ClusterIssue オブジェクトの 3 番目のフィールドは resourceIds で、返される値は課題タイプによって異なります。

Code	メッセージ	ResourceIds	クラスターは回復可能ですか?	
SUBNET_NOT_FOUND	クラスターに現在関連付けられている 1 つ以上のサブネットが見つかりませんでした。Amazon EKS update-cluster-config API を呼び出してサブネットを更新します。	サブネット ID	はい	
SECURITY_GROUP_NOT_FOUND	クラスターに現在関連付けられている 1 つ以上のセキュリティグループが見つかりませんでした。Amazon EKS update-cluster-config API を呼び出して、セキュリティグループを更新します	セキュリティグループ ID	はい	
IP_NOT_AVAILABLE	クラスターに関連付けられている 1 つ以上のサブネットに、Amazon EKS がクラスター管理操作を実行するための十分な IP アドレスがありません。サブネット内のアドレスを解放するか、Amazon EKS update-cluster-config API を使用し	サブネット ID	はい	

Code	メッセージ	ResourceIds	クラスターは回復可能ですか?	
	て別のサブネットをクラスターに関連付けます。			
VPC_NOT_FOUND	クラスターに関連付けられている VPC が見つかりませんでした。クラスターを削除して再作成する必要があります。	VPC ID	いいえ	
ASSUME_ROLE_ACCESS_DENIED	クラスターは Amazon EKS サービスにリンクされたロールを使用していません。クラスターに関連するロールを引き受けて、必要な Amazon EKS 管理オペレーションを実行することができませんでした。ロールが存在し、必要な信頼ポリシーが適用されていることを確認してください。	クラスター IAM ロール	はい	
PERMISSION_ACCESS_DENIED	クラスターは Amazon EKS サービスにリンクされたロールを使用していません。クラスターに関連付けられているロールでは、Amazon EKS に必要な管理操作を実行するための十分なアクセス許可が付与されていません。クラスターロールにアタッチされているポリシーを確認し、別の拒否ポリシーが適用されているかどうかを確認してください。	クラスター IAM ロール	はい	

Code	メッセージ	ResourceIds	クラスターは回復可能ですか?	
ASSUME_ROLE_ACCESS_DENIED_USING_SLR	Amazon EKS クラスター管理サービスにリンクされたロールを想定することはできませんでした。ロールが存在し、必要な信頼ポリシーが適用されていることを確認してください。	Amazon EKS サービスにリンクされたロール	はい	
PERMISSION_ACCESS_DENIED_USING_SLR	Amazon EKS クラスター管理サービスにリンクされたロールは、Amazon EKS に必要な管理操作を実行するための十分なアクセス許可を付与しません。クラスターロールにアタッチされているポリシーを確認し、別の拒否ポリシーが適用されているかどうかを確認してください。	Amazon EKS サービスにリンクされたロール	はい	
OPT_IN_REQUIRED	あなたのアカウントには Amazon EC2 サービスのサブスクリプションがありません。アカウント設定ページでアカウントサブスクリプションを更新してください。	該当なし	はい	
STS_REGIONAL_ENDPOINT_DISABLED	STS リージョナルエンドポイントは無効になっています。Amazon EKS のエンドポイントで必要なクラスター管理操作を実行できるようにします。	該当なし	はい	

Code	メッセージ	ResourceIds	クラスターは回復可能ですか?	
KMS_KEY_DISABLED	クラスターに関連付けられている AWS KMS キーは無効になっています。クラスターを復元するには、キーを再度有効にします。	KMS Key Arn	はい	
KMS_KEY_NOT_FOUND	クラスターに関連付けられている AWS KMS キーが見つかりませんでした。クラスターを削除して再作成する必要があります。	KMS Key ARN	いいえ	
KMS_GRANT_REVOKED	クラスターに関連付けられた AWS KMS キーの付与は取り消されます。クラスターを削除して再作成する必要があります。	KMS Key Arn	いいえ	

Amazon EKS Connector

Amazon EKS Connector を使用して、準拠する Kubernetes クラスターを AWS に登録および接続し、Amazon EKS コンソールで可視化できます。クラスターが接続されると、Amazon EKS コンソールでクラスターのステータス、設定、およびワークロードを確認できます。この機能を使用して Amazon EKS コンソールで接続されたクラスターを表示できますが、管理することはできません。Amazon EKS Connector には、Github のオープンソースプロジェクトであるエージェントが必要です。よくある質問やトラブルシューティングなどのその他の技術コンテンツについては、「Amazon EKS Connector での問題のトラブルシューティング」を参照してください。

Amazon EKS Connector は、次のタイプの Kubernetes クラスターを Amazon EKS に接続できます。

- オンプレミス Kubernetes クラスター
- Amazon EC2 で実行されるセルフマネージド型クラスター
- 他のクラウドプロバイダーのマネージド型クラスター

Amazon EKS Connector の考慮事項

Amazon EKS Connector を使用する前に、次のことを理解してください。

- クラスターを Amazon EKS に接続するには、Kubernetes クラスターに対する管理者権限が必要です。
- Kubernetes クラスターには、接続する前に Linux 64 ビット (x86) ワーカーノードが存在する必要があります。ARM ワーカーノードはサポートされていません。
- `ssm.` および `ssmmessages.` Systems Manager エンドポイントへのアウトバウンドアクセスを持つ Kubernetes クラスター内のワーカーノードが必要です。詳細については、AWS 全般のリファレンス の「Systems Manager エンドポイント」を参照してください。
- デフォルトでは、1 つのリージョンで最大 10 個のクラスターを接続できます。Service Quotas コンソールを使用して引き上げをリクエストできます。詳細については、「クォータの引き上げのリクエスト」を参照してください。
- Amazon EKS `RegisterCluster`、`ListClusters`、`DescribeCluster`、および `DeregisterCluster` API のみが、外部 Kubernetes クラスターでサポートされています。
- クラスターを登録するには、次のアクセス許可が必要です。

- eks:RegisterCluster
- ssm:CreateActivation
- ssm:DeleteActivation
- iam:PassRole

- クラスターを登録解除するには、次のアクセス許可が必要です。
  - eks:DeregisterCluster
  - ssm:DeleteActivation
  - SSM: DeregisterManagedInstance

Amazon EKS コネクターの必要な IAM ロール

Amazon EKS Connector を使用するには、次の 2 つの IAM ロールが必要です。

- 初めてクラスターを登録すると、Amazon EKS Connector サービスにリンクされたロールが作成されます。
- Amazon EKS Connector エージェントの IAM ロールを作成する必要があります。詳細については、「Amazon EKS コネクタの IAM ロール」を参照してください。

IAM プリンシパルのクラスターおよびワークロードの表示許可を有効にするには、`eks-connector` および Amazon EKS Connector クラスターのロールをクラスターに適用します。「クラスター上の Kubernetes リソースを表示するためのアクセス権の IAM プリンシパルへの付与」の手順を実行します。

外部クラスターの接続

以下のプロセスで複数のメソッドを使用して、外部の Kubernetes クラスターを Amazon EKS に接続できます。このプロセスには 2 つのステップがあります。Amazon EKS へのクラスターの登録と、クラスターへの `eks-connector` エージェントのインストールです。

> **Important**
>
> 最初のステップを完了してから 3 日以内、つまり登録の有効期限が切れる前に 2 番目のステップを完了する必要があります。

コネクタメソッド

クラスターを登録する各メソッドの後で、エージェントをインストールするすべてのメソッドを使用できるわけではありません。次の表は、各登録メソッドと、使用できるエージェントのインストールメソッドのリストです。

[ステップ]	方法		
クラスターの登録	AWS Management Console	AWS Command Line Interface	`eksctl`
エージェントのインストール	Helm、YAML マニフェスト	Helm、YAML マニフェスト	YAML マニフェスト

前提条件

- Amazon EKS Connector エージェントロールが作成されたことを確認します。「Amazon EKS コネクタエージェントロールの作成」の手順を実行します。
- クラスターを登録するには、次のアクセス許可が必要です。
  - `eks:RegisterCluster`
  - `ssm:CreateActivation`
  - `ssm:DeleteActivation`
  - `iam:PassRole`

ステップ 1: クラスターの登録

AWS CLI

前提条件

- AWS CLIがインストールされている必要があります。インストールまたはアップグレードするには、AWS CLI のインストールを参照してください。

クラスターを AWS CLI に登録するには

- コネクタ設定には、Amazon EKS Connector エージェントの IAM ロールを指定します。詳細については、「Amazon EKS コネクターの必要な IAM ロール」を参照してください。

```
aws eks register-cluster \
     --name my-first-registered-cluster \
     --connector-config roleArn=arn:aws:iam::111122223333:role/
AmazonEKSConnectorAgentRole,provider="OTHER" \
     --region aws-region
```

出力例は次のとおりです。

```
{
    "cluster": {
        "name": "my-first-registered-cluster",
        "arn": "arn:aws:eks:region:111122223333:cluster/my-first-registered-
cluster",
        "createdAt": 1627669203.531,
        "ConnectorConfig": {
            "activationId": "xxxxxxxxACTIVATION_IDxxxxxxxx",
            "activationCode": "xxxxxxxxACTIVATION_CODExxxxxxxx",
            "activationExpiry": 1627672543.0,
            "provider": "OTHER",
            "roleArn": "arn:aws:iam::111122223333:role/
AmazonEKSConnectorAgentRole"
        },
        "status": "CREATING"
    }
}
```

次のステップで aws-region、activationId、activationCode の値を使用します。

AWS Management Console

コンソールに Kubernetes クラスターを登録するには。

1. https://console.aws.amazon.com/eks/home#/clusters で Amazon EKS コンソール を開きます。

2. [Add cluster] (クラスターの追加) を選択し、[Register] (登録) を選択して設定ページを表示します。
3. [Configure cluster (クラスターの設定)] セクションで、次のフィールドに入力します。
   - [Name (名前)] – クラスターの一意の名前。
   - プロバイダー - Kubernetes クラスタープロバイダーのドロップダウンリストを表示することを選択します。特定のプロバイダが不明な場合は、[その他] を選択します。
   - EKS Connector のロール — クラスターの接続に使用するロールを選択します。
4. クラスターの登録を選択します。
5. [Cluster] 概要ページが表示されます。Helm チャートを使用する場合は、`helm install` コマンドをコピーして次のステップに進みます。YAML マニフェストを使用する場合は、[YAML ファイルをダウンロード] を選択して、マニフェストファイルをローカルドライブにダウンロードします。

   > **Important**
   >
   > - これは、`helm install` コマンドをコピーする、またはこのファイルをダウンロードする唯一の機会です。このページから移動しないでください。リンクにアクセスできないので、クラスターの登録を解除し、最初から手順を開始する必要があります。
   > - コマンドまたはマニフェストファイルは、登録されたクラスターに対して一度だけ使用できます。Kubernetes クラスターからリソースを削除する場合は、クラスターを再登録し、新しいマニフェストファイルを取得する必要があります。

   次のステップに進み、Kubernetes クラスターにマニフェストファイルを適用します。

eksctl

前提条件

- `eksctl` バージョン `0.68` 以降がインストールされている必要があります。インストール、またはアップグレードをする場合は「Amazon EKS の開始方法 – `eksctl`」を参照してください。

クラスターを **eksctl** に登録するには

1. 名前、プロバイダー、およびリージョンを指定して、クラスターを登録します。

```
eksctl register cluster --name my-cluster --provider my-provider --
region region-code
```

出力例:

```
2021-08-19 13:47:26 [#]  creating IAM role "eksctl-20210819194112186040"
2021-08-19 13:47:26 [#]  registered cluster "<name>" successfully
2021-08-19 13:47:26 [#]  wrote file eks-connector.yaml to <current directory>
2021-08-19 13:47:26 [#]  wrote file eks-connector-clusterrole.yaml to <current
 directory>
2021-08-19 13:47:26 [#]  wrote file eks-connector-console-dashboard-full-access-
group.yaml to <current directory>
2021-08-19 13:47:26 [!]  note: "eks-connector-clusterrole.yaml" and "eks-
connector-console-dashboard-full-access-group.yaml" give full EKS Console access
 to IAM identity "<aws-arn>", edit if required; read https://eksctl.io/usage/
eks-connector for more info
2021-08-19 13:47:26 [#]  run `kubectl apply -f eks-connector.yaml,eks-connector-
clusterrole.yaml,eks-connector-console-dashboard-full-access-group.yaml` before
 expiry> to connect the cluster
```

これにより、ローカルコンピュータ上にファイルが作成されます。これらのファイルは 3 日以内に外部クラスターに適用する必要があります。そうしないと登録の有効期限が切れます。

2. クラスターにアクセスできるターミナルで、eks-connector-binding.yaml ファイルを適用します。

```
kubectl apply -f eks-connector-binding.yaml
```

ステップ 2: eks-connector エージェントのインストール

Helm chart

1. 前のステップで AWS CLI を使用した場合は、次のコマンドで ACTIVATION_CODE と ACTIVATION_ID をそれぞれ activationId と activationCode の値に置き換えます。aws-region を前のステップで使用した AWS リージョンに置き換えます。その後、次のコマンドを実行して、eks-connector エージェントを登録クラスターにインストールします。

   ```
   $ helm install eks-connector \
     --namespace eks-connector \
     oci://public.ecr.aws/eks-connector/eks-connector-chart \
     --set eks.activationCode=ACTIVATION_CODE \
     --set eks.activationId=ACTIVATION_ID \
     --set eks.agentRegion=aws-region
   ```

   前のステップで AWS Management Consoleを使用した場合は、前のステップからコピーした、以下の値が入力されたコマンドを使用します。
2. インストールされている eks-connector デプロイの正常性をチェックし、Amazon EKS に登録されたクラスターのステータスが ACTIVE になるのを待ちます。

YAML manifest

Kubernetes クラスターに Amazon EKS Connector のマニフェストファイルを適用して、接続を完了します。これを行うには、前述のメソッドを使用する必要があります。マニフェストが 3 日以内に適用されない場合、Amazon EKS Connector の登録の有効期限が切れます。クラスター接続が期限切れになった場合は、クラスターを再度接続する前にクラスターの登録を解除する必要があります。

1. Amazon EKS コネクタ YAML ファイルをダウンロードします。

   ```
   curl -O https://amazon-eks.s3.us-west-2.amazonaws.com/eks-connector/manifests/
   eks-connector/latest/eks-connector.yaml
   ```
2. Amazon EKS Connector の YAML ファイルを編集して、%AWS_REGION%、%EKS_ACTIVATION_ID%、%EKS_ACTIVATION_CODE% のすべてのリファレンスを、前のステップの出力から得られた aws-region、activationId、activationCode に置き換えます。

次のコマンド例では、これらの値を置き換えることができます。

```
sed -i "s~%AWS_REGION%~$aws-region~g; s~%EKS_ACTIVATION_ID
%~$EKS_ACTIVATION_ID~g; s~%EKS_ACTIVATION_CODE%~$(echo -n $EKS_ACTIVATION_CODE |
 base64)~g" eks-connector.yaml
```

> **Important**
>
> アクティベーションコードが base64 形式であることを確認します。

3. クラスターにアクセスできるターミナルで、次のコマンドを実行して、更新されたマニフェストファイルを適用できます。

```
kubectl apply -f eks-connector.yaml
```

4. Amazon EKS Connector のマニフェストとロールバインディング YAML ファイルが Kubernetes クラスターに適用されたら、クラスターが接続されたことを確認します。

```
aws eks describe-cluster \
     --name "my-first-registered-cluster" \
     --region AWS_REGION
```

出力には `status=ACTIVE` が含まれている必要があります。

5. (オプション) クラスターにタグを追加します。詳細については、「Amazon EKS リソースのタグ付け」を参照してください。

次のステップ

これらのステップで問題が発生した場合は、「Amazon EKS Connector での問題のトラブルシューティング」を参照してください。

追加の IAM プリンシパルに Amazon EKS コンソールへのアクセスを許可して、接続されているクラスター内の Kubernetes リソースを表示するには、「クラスター上の Kubernetes リソースを表示するためのアクセス権の IAM プリンシパルへの付与」を参照してください。

クラスター上の Kubernetes リソースを表示するためのアクセス権の IAM プリンシパルへの付与

IAM プリンシパルに Amazon EKS コンソールへのアクセスを許可して、接続されたクラスターで実行されている Kubernetes リソースに関する情報を表示します。

前提条件

AWS Management Console にアクセスするために使用する IAM プリンシパルは、次の要件を満たしている必要があります。

- `eks:AccessKubernetesApi` IAM アクセス許可が必要です。
- Amazon EKS Connector サービスアカウントが、クラスター内の IAM プリンシパルを偽装できる。これにより、Amazon EKS Connector は IAM プリンシパルを Kubernetes ユーザーにマッピングできます。

Amazon EKS クラスターロールを作成して適用するには

1. `eks-connector` クラスターのロールテンプレートをダウンロードします。

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/eks-connector/manifests/eks-connector-console-roles/eks-connector-clusterrole.yaml
```

2. クラスターロールテンプレート YAML ファイルを編集します。`%IAM_ARN%` のリファレンスを IAM プリンシパルの Amazon リソースネーム (ARN) に置き換えます。
3. Amazon EKS Connector クラスターロール YAML を Kubernetes クラスターに適用します。

```
kubectl apply -f eks-connector-clusterrole.yaml
```

IAM プリンシパルが Amazon EKS コンソールで Kubernetes リソースを確認するには、そのプリンシパルが、リソースを読み取るために必要な許可を持つ Kubernetes `role` または `clusterrole` に関連付けられている必要があります。詳細については、「Kubernetes ドキュメント」の「RBAC 認証の使用」を参照してください。

接続されたクラスターにアクセスするように IAM プリンシパルを設定するには

1. 以下のいずれかのサンプルマニフェストファイルをダウンロードして、それぞれ clusterrole および clusterrolebinding、または role および rolebinding を作成できます。

   すべての名前空間の Kubernetes リソースを表示する

   クラスターのロール eks-connector-console-dashboard-full-access-clusterrole は、コンソールで視覚化できるすべての名前空間とリソースへのアクセスを提供します。クラスターに適用する前に role、clusterrole、および対応するバインディングの名前を変更することができます。次のコマンドを使用して、サンプルファイルをダウンロードします。

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/eks-connector/manifests/eks-connector-console-roles/eks-connector-console-dashboard-full-access-group.yaml
   ```

   特定の名前空間内の Kubernetes リソースを表示する

   このファイルの名前空間は default です。別の名前空間を指定する場合は、クラスターに適用する前にファイルを編集します。次のコマンドを使用して、サンプルファイルをダウンロードします。

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/eks-connector/manifests/eks-connector-console-roles/eks-connector-console-dashboard-restricted-access-group.yaml
   ```

2. フルアクセスまたは制限付きアクセスの YAML ファイルを編集して、%IAM_ARN% のリファレンスを IAM プリンシパルの Amazon リソースネーム (ARN) に置き換えます。
3. フルアクセスまたは制限付きアクセスの YAML ファイルを Kubernetes クラスターに適用します。YAML ファイルの値は独自の値に置き換えます。

   ```
   kubectl apply -f eks-connector-console-dashboard-full-access-group.yaml
   ```

接続されたクラスター内の Kubernetes リソースを表示するには、「Kubernetes リソースを表示する」を参照してください。[Resources] (リソース) タブ上の一部のリソースタイプのデータは、接続されたクラスターでは使用できません。

クラスターの登録解除

接続されたクラスターの使用が終了したら、登録を解除できます。登録解除されると、クラスターは Amazon EKS コンソールに表示されなくなります。

deregisterCluster API を呼び出すには、次のアクセス許可が必要です。

- `eks:DeregisterCluster`
- `ssm:DeleteActivation`
- `ssm:DeregisterManagedInstance`

このプロセスには 2 つのステップがあります。Amazon EKS でのクラスターの登録解除と、クラスター内の eks-connector エージェントのアンインストールです。

Kubernetes クラスターの登録解除

AWS CLI

前提条件

- AWS CLIがインストールされている必要があります。インストールまたはアップグレードするには、AWS CLI のインストールを参照してください。
- Amazon EKS Connector エージェントロールが作成されたことを確認します。

接続されているクラスタの登録を解除します。

```
aws eks deregister-cluster \
    --name my-cluster \
    --region region-code
```

AWS Management Console

1. https://console.aws.amazon.com/eks/home#/clusters で Amazon EKS コンソール を開きます。
2. [Clusters] を選択します。
3. リポジトリの[クラスター]ページで、接続されているクラスターを選択し、[登録解除]を選択します。

4. クラスターの登録を解除することを確認します。

eksctl

前提条件

- eksctl バージョン 0.68 以降がインストールされている必要があります。インストール、またはアップグレードをする場合は「Amazon EKS の開始方法 – eksctl」を参照してください。
- Amazon EKS Connector エージェントロールが作成されたことを確認します。

クラスターを **eksctl** から登録解除するには

- コネクタ設定には、Amazon EKS Connector エージェントの IAM ロールを指定します。詳細については、「Amazon EKS コネクターの必要な IAM ロール」を参照してください。

```
eksctl deregister cluster --name my-cluster
```

Kubernetes クラスター内のリソースのクリーンアップ

Helm

- 次のコマンドを実行して、エージェントをアンインストールします。

```
helm -n eks-connector uninstall eks-connector
```

YAML manifest

1. Kubernetes クラスターから Amazon EKS コネクタの YAML ファイルを削除します。

```
kubectl delete -f eks-connector.yaml
```

2. 追加の IAM プリンシパルがクラスターにアクセスするために clusterrole または clusterrolebindings を作成した場合は、それらを Kubernetes クラスターから削除します。

Amazon EKS Connector での問題のトラブルシューティング

このトピックでは、Amazon EKS Connector の使用中に発生する可能性があるいくつかの一般的なエラーについて説明します。これには、解決方法および回避策も含まれています。

基本的なトラブルシューティング

このセクションでは、問題が不明な場合の診断手順について説明します。

Amazon EKS Connector の状態を確認する

Amazon EKS Connector の状態を確認します。

```
kubectl get pods -n eks-connector
```

Amazon EKS Connector のログを検査します

Amazon EKS Connector の Pod は 3 つのコンテナで構成されています。これらすべてのコンテナの完全なログを取得して検査できるようにするには、次のコマンドを実行します。

- `connector-init`

```
kubectl logs eks-connector-0 --container connector-init -n eks-connector
kubectl logs eks-connector-1 --container connector-init -n eks-connector
```

- `connector-proxy`

```
kubectl logs eks-connector-0 --container connector-proxy -n eks-connector
kubectl logs eks-connector-1 --container connector-proxy -n eks-connector
```

- `connector-agent`

```
kubectl exec eks-connector-0 --container connector-agent -n eks-connector -- cat /
var/log/amazon/ssm/amazon-ssm-agent.log
kubectl exec eks-connector-1 --container connector-agent -n eks-connector -- cat /
var/log/amazon/ssm/amazon-ssm-agent.log
```

有効なクラスター名を取得する

Amazon EKS クラスターは、単一の AWS アカウントおよび AWS リージョン 内で clusterName により一意に識別されます。Amazon EKS で接続されたクラスターが複数ある場合、現在の Kubernetes クラスターが登録されている Amazon EKS クラスターを確認できます。これを行うには、次のコマンドを入力して、現在のクラスターの clusterName を確認します。

```
kubectl exec eks-connector-0 --container connector-agent -n eks-connector \
  -- cat /var/log/amazon/ssm/amazon-ssm-agent.log | grep -m1 -oE "eks_c:[a-zA-Z0-9_-]+"
 | sed -E "s/^.*eks_c:([a-zA-Z0-9_-]+)_[a-zA-Z0-9]+.*$/\1/"
kubectl exec eks-connector-1 --container connector-agent -n eks-connector \
  -- cat /var/log/amazon/ssm/amazon-ssm-agent.log | grep -m1 -oE "eks_c:[a-zA-Z0-9_-]+"
 | sed -E "s/^.*eks_c:([a-zA-Z0-9_-]+)_[a-zA-Z0-9]+.*$/\1/"
```

その他のコマンド

次のコマンドは、問題のトラブルシューティングに必要な情報の取得に役立ちます。

- Amazon EKS Connector の Pods で使用されるイメージを収集するには、次のコマンドを使用します。

  ```
  kubectl get pods -n eks-connector -o jsonpath="{.items[*].spec.containers[*].image}"
   | tr -s '[[:space:]]' '\n'
  ```

- Amazon EKS Connector が実行されているノード名を確認するには、次のコマンドを使用します。

  ```
  kubectl get pods -n eks-connector -o jsonpath="{.items[*].spec.nodeName}" | tr -s
   '[[:space:]]' '\n'
  ```

- Kubernetes クライアントおよびサーバーのバージョンを取得するには、次のコマンドを実行します。

  ```
  kubectl version
  ```

- ノードに関する情報を取得するには、次のコマンドを実行します。

  ```
  kubectl get nodes -o wide --show-labels
  ```

Helm の問題: 403 Forbidden

Helm インストールコマンドの実行中に以下のエラーが表示された場合:

```
Error: INSTALLATION FAILED: unexpected status from HEAD request to https://
public.ecr.aws/v2/eks-connector/eks-connector-chart/manifests/0.0.6: 403 Forbidden
```

以下の行を実行して修正できます。

```
docker logout public.ecr.aws
```

コンソールのエラー: クラスターが Pending 状態でスタックしています

クラスターの登録後、クラスターが Amazon EKS コンソールで Pending 状態のままスタックしている場合、Amazon EKS Connector がクラスターを AWS に正常に接続しなかったことが原因である可能性があります。登録済みのクラスターの場合、Pending 状態は接続がまだ正常に確立されていないことを意味します。この問題を解決するには、ターゲットの Kubernetes クラスターにマニフェストを適用していることを確認します。クラスターにマニュフェストを適用したにもかかわらず、まだ Pending 状態である場合、eks-connector statefulset に異常がある可能性があります。この問題をトラブルシューティングするには、このトピックの Amazon EKS Connector の Pods がクラッシュループしている を参照してください。

コンソールエラー: クラススコープで User “system:serviceaccount:eks-connector:eks-connector” can't impersonate resource “users” in API group “”

Amazon EKS Connector は、Kubernetes ユーザー偽装を使用して AWS Management Console から IAM プリンシパルに代わってアクションを実行します。AWS eks-connector のサービスアカウントの Kubernetes API にアクセスする各プリンシパルについては、Kubernetes ユーザー名として IAM ARN で対応する Kubernetes ユーザーを偽装するために許可を付与する必要があります。次の例では、Kubernetes ユーザーに IAM ARN がマッピングされています。

- AWS アカウント *111122223333* からの IAM ユーザー *john* は、Kubernetes ユーザーにマッピングされています。IAM のベストプラクティスでは、ユーザーではなくロールに許可を付与することが推奨されています。

  ```
  arn:aws:iam::111122223333:user/john
  ```

- AWS アカウント *111122223333* からの IAM ロール *admin* は、Kubernetes ユーザーにマッピングされています。

  ```
  arn:aws:iam::111122223333:role/admin
  ```

  結果は AWS STS セッションの ARN ではなく、IAM ロールの ARN です。

マッピングされたユーザーを偽装するアカウント権限を eks-connector サービスに付与するために ClusterRole および ClusterRoleBinding を設定する方法については、「クラスター上の Kubernetes リソースを表示するためのアクセス権の IAM プリンシパルへの付与」を参照してください。テンプレートで、%IAM_ARN% が AWS Management Console IAM プリンシパルの IAM ARN に置き換えられていることを確認してください。

コンソールエラー: クラススコープで [...] is forbidden: User [...] cannot list resource “[...] in API group”

次の問題を検討します。Amazon EKS Connector により、ターゲットの Kubernetes クラスターでリクエストしている AWS Management Console IAM プリンシパルが正常に偽装されました。ただし、偽装されたプリンシパルには Kubernetes API オペレーション用の RBAC 許可がありません。

この問題を解決するには、追加のユーザーに権限を付与する方法が 2 つあります。以前に Helm チャート 経由で eks-connector をインストールしたことがある場合は、以下のコマンドを実行することでユーザーにアクセス権を簡単に付与できます。userARN1 と userARN2 を IAM ロールの ARN のリストに置き換えて、Kubernetes リソース表示のアクセス権を付与します。

```
helm upgrade eks-connector oci://public.ecr.aws/eks-connector/eks-connector-chart \
    --reuse-values \
    --set 'authentication.allowedUserARNs={userARN1,userARN2}'
```

または、クラスターの管理者として個々の Kubernetes ユーザーに適切なレベルの RBAC 権限を付与します。詳細な説明と例については、「クラスター上の Kubernetes リソースを表示するためのアクセス権の IAM プリンシパルへの付与」を参照してください。

コンソールエラー:「Amazon EKS が Kubernetes クラスターの API サーバーと通信できません。」 正常に接続するには、クラスターが ACTIVE 状態である必要があります。数分後にもう一度お試しください。

Amazon EKS サービスがターゲットクラスター内の Amazon EKS Connector と通信できない場合、次のいずれかの原因が考えられます。

- ターゲットクラスターの Amazon EKS Connector に異常があります。
- ターゲットクラスターと AWS リージョン 間の接続が悪い、または中断している。

この問題を解決するには、Amazon EKS Connector ログを確認してください。Amazon EKS Connector のエラーが表示されない場合、数分後に接続を再試行してください。ターゲットクラスターで高レイテンシーや断続的な接続が定期的に発生する場合は、近くにある AWS リージョン にクラスターを再登録することを検討してください。

Amazon EKS Connector の Pods がクラッシュループしている

Amazon EKS Connector の Pod が `CrashLoopBackOff` 状態になる原因は数多くあります。この問題は `connector-init` コンテナに関係している可能性があります。Amazon EKS コネクタの Pod ステータスを確認します。

```
kubectl get pods -n eks-connector
```

出力例は次のとおりです。

```
NAME              READY   STATUS                  RESTARTS   AGE
eks-connector-0   0/2     Init:CrashLoopBackOff   1          7s
```

出力が前の出力と似ている場合、Amazon EKS Connector のログを検査します を参照して問題をトラブルシューティングしてください。

`Failed to initiate eks-connector: InvalidActivation`

Amazon EKS Connector を初めて起動する場合、Amazon Web Services に `activationId` および `activationCode` を登録します。登録が失敗することがあり、次のエラーと同様なエラーで `connector-init` コンテナがクラッシュする原因になります。

```
F1116 20:30:47.261469       1 init.go:43] failed to initiate eks-connector:
 InvalidActivation:
```

この問題のトラブルシューティングを行うには、次の原因および推奨される修正を検討してください。

- activationId および activationCode がマニフェストファイルにないために、登録が失敗した可能性があります。このような場合、値が RegisterCluster API オペレーションから返された正しいものであり、activationCode がマニフェストファイルに含まれていることを確認してください。activationCode は Kubernetes シークレットに追加されるため、base64 でエンコードする必要があります。詳細については、「ステップ 1: クラスターの登録」を参照してください。
- アクティベーションの有効期限が切れたため、登録が失敗した可能性があります。これは、セキュリティ上の理由により、クラスターを登録してから 3 日以内に Amazon EKS Connector をアクティブ化する必要があるためです。この問題を解決するには、Amazon EKS Connector のマニフェストが有効期限の日時よりも前にターゲットの Kubernetes クラスターに適用されていることを確認してください。アクティブ化の有効期限を確認するには、DescribeCluster API オペレーションを呼び出します。

  ```
  aws eks describe-cluster --name my-cluster
  ```

  次のレスポンスの例では、有効期限の日時は 2021-11-12T22:28:51.101000-08:00 として記録されています。

  ```
  {
      "cluster": {
          "name": "my-cluster",
          "arn": "arn:aws:eks:region:111122223333:cluster/my-cluster",
          "createdAt": "2021-11-09T22:28:51.449000-08:00",
          "status": "FAILED",
          "tags": {
          },
          "connectorConfig": {
              "activationId": "00000000-0000-0000-0000-000000000000",
              "activationExpiry": "2021-11-12T22:28:51.101000-08:00",
              "provider": "OTHER",
              "roleArn": "arn:aws:iam::111122223333:role/my-connector-role"
          }
  ```

```
    }
}
```

activationExpiry が渡されたら、クラスターの登録を解除し、再度登録します。これにより、新しいアクティベーションが生成されます。

クラスターノードにアウトバウンド接続がありません

正常に動作するには、Amazon EKS Connector に複数の AWS エンドポイントへのアウトバウンド接続が必要です。ターゲットの AWS リージョン へのアウトバウンド接続が使用可能でないと、プライベートクラスターを接続することはできません。この問題を解決するには、必要なアウトバウンド接続を追加する必要があります。コネクタの要件については、「Amazon EKS Connector の考慮事項」を参照してください。

Amazon EKS Connector Pods が **ImagePullBackOff** 状態になっています

get pods コマンドを実行しているときに Pods が ImagePullBackOff 状態にある場合、正常に動作しません。Amazon EKS Connector Pods が ImagePullBackOff 状態の場合、正常に作動できません。Amazon EKS Connector Pods の状態を確認してください。

```
kubectl get pods -n eks-connector
```

出力例は次のとおりです。

```
NAME              READY   STATUS                  RESTARTS   AGE
eks-connector-0   0/2     Init:ImagePullBackOff   0          4s
```

デフォルトの Amazon EKS Connector のマニフェストファイルは「Amazon ECR Public Gallery」からイメージを参照します。ターゲットの Kubernetes クラスターが Amazon ECR Public Gallery からイメージをプルできない可能性があります。Amazon ECR Public Gallery のイメージのプルに関する問題を解決するか、選択したプライベートコンテナレジストリ内のイメージのミラーリングを検討してください。

よくある質問

Q: Amazon EKS Connector の基盤となるテクノロジーはどのように機能しますか?

A: Amazon EKS Connector は AWS Systems Manager (Systems Manager) エージェントに基づいています。Amazon EKS Connector は Kubernetes クラスター上で `StatefulSet` として実行されます。これにより接続が確立され、クラスターの API サーバーと Amazon Web Services 間の通信がプロキシされます。これは、AWS からクラスターが切断されるまで Amazon EKS コンソールにクラスターデータを表示しておくためです。Systems Manager エージェントはオープンソースのプロジェクトです。このプロジェクトについての詳細は、「GitHub GitHub project page」を参照してください。

質問: 接続したいオンプレミスの Kubernetes クラスターがあります。接続するにはファイアウォールポートを開く必要がありますか?

A: いいえ。ファイアウォールポートを開く必要はありません。Kubernetes クラスターは、AWS リージョン へのアウトバウンド接続のみを必要とします。AWS のサービスがオンプレミスネットワークのリソースにアクセスすることはありません。Amazon EKS Connector はクラスターで実行され、AWS への接続を開始します。クラスターの登録が完了すると、クラスター上の Kubernetes API サーバーからの情報を必要とする Amazon EKS コンソールからアクションを開始した後にのみ、AWS により Amazon EKS Connector にコマンドが発行されます。

Q: Amazon EKS Connector によってクラスターから AWS に送信されるデータはどのようなものですか?

A: Amazon EKS Connector により送信されるのは、AWS でのクラスターの登録に必要な技術情報です。また、お客様がリクエストする Amazon EKS コンソールにおける機能のクラスターとワークロードのメタデータも送信されます。Amazon EKS Connector は、AWS へデータの送信を必要とする Amazon EKS コンソールまたは Amazon EKS API からアクションを開始した場合にのみ、このデータを収集または送信します。デフォルトでは、AWS で Kubernetes のバージョン番号以外のデータが保存されることはありません。許可した場合にのみデータを保存します。

Q: AWS リージョン の外部のクラスターを接続できますか?

A: はい、任意の場所からクラスターを Amazon EKS に接続できます。さらに、Amazon EKS サービスはどんな AWS のパブリックの商用 AWS リージョン でもサポートされています。これは、クラスターからターゲットの AWS リージョン への有効なネットワーク接続により動作します。UI パフォーマンスを最適化するため、クラスターの場所に最も近い AWS リージョン を選択することをお勧めします。例えば、東京で実行しているクラスターがある場合、レイテンシーを低くするためにクラスターを東京の AWS リージョン (つまり `ap-northeast-1` AWS リージョン) に接続します。中国または GovCloud AWS リージョン を除き、クラスターを任意の場所からあらゆるパブリックの商用 AWS リージョン の Amazon EKS に接続できます。

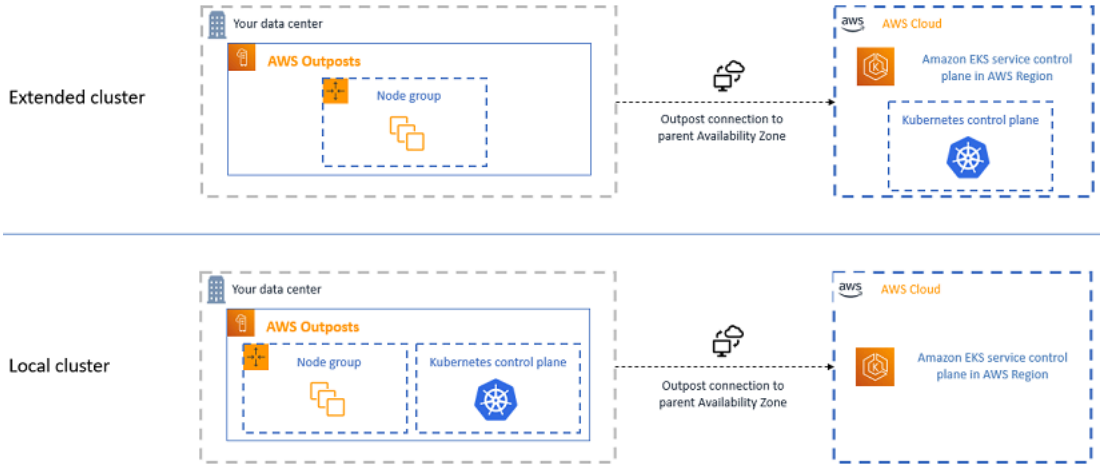
AWS Outposts における Amazon EKS

Amazon EKS を使用して、オンプレミス Kubernetes アプリケーションを AWS Outposts で実行できます。次の方法で Amazon EKS を Outposts にデプロイできます。

- 拡張クラスター - Outpost の AWS リージョン とノードで Kubernetes コントロールプレーンを実行します。
- ローカルクラスター - Outpost で Kubernetes コントロールプレーンとノードを実行します。

どちらのデプロイオプションでも、Kubernetes コントロールプレーンは、AWS によって完全に管理されています。クラウドで使用するのと同じ Amazon EKS API、ツール、コンソールを使用して Outposts で Amazon EKS を作成および実行することができます。

以下の図に、これらのデプロイのオプションを示します。



各デプロイオプションを使用する時

ローカルクラスターと拡張クラスターはどちらも汎用のデプロイオプションであり、さまざまなアプリケーションに使用できます。

ローカルクラスターでは、Amazon EKS クラスター全てを Outposts でローカルに実行できます。この方法により、クラウドへの一時的なネットワーク切断によって生じる可能性のあるアプリケーションダウンタイムのリスクを軽減できます。このようなネットワーク接断は、ファイバーの切断や天候によって発生する可能性があります。すべての Amazon EKS クラスター全体が Outposts でローカルに実行されるため、アプリケーションは引き続き使用できます。クラウドへのネットワーク切断中

にクラスターオペレーションを実行できます。詳細については、「ネットワーク切断の準備」を参照してください。Outposts から親 AWS リージョン へのネットワーク接続の品質が気になる場合、およびネットワークの切断による高可用性を必要とする場合は、ローカルクラスターデプロイオプションを使用してください。

拡張クラスターでは、Kubernetes コントロールプレーンが親 AWS リージョン で動作するため、Outpost の容量を節約できます。このオプションは、Outpost から AWS リージョン への信頼性が高く冗長なネットワーク接続に投資できる場合に適している可能性があります。このオプションでは、ネットワーク接続の品質が重要です。Kubernetes が Kubernetes コントロールプレーンとノード間のネットワーク切断を処理する方法によって、アプリケーションのダウンタイムが発生する可能性があります。Kubernetes の動作の詳細については、Kubernetes ドキュメントの「スケジューリング、プリエンプション、エビクション」を参照してください。

デプロイオプションの比較

以下の表は、これら 2 つのオプションの違いを比較したものです。

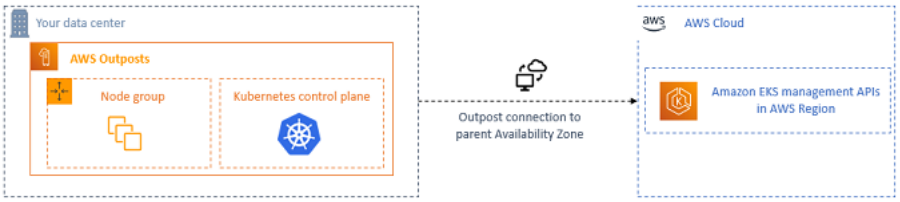
機能	拡張クラスター	ローカルクラスター
Kubernetes コントロールプレーンの位置	AWS リージョン	Outpost
Kubernetes コントロールプレーンアカウント	AWS アカウント	お客様のアカウント
リージョナルな可用性	「サービスエンドポイント」を参照してください	米国東部 (バージニア北部)、米国東部 (オハイオ)、米国西部 (北カリフォルニア)、米国西部 (オレゴン)、アジアパシフィック (ソウル)、アジアパシフィック (シンガポール)、アジアパシフィック (東京)、アジアパシフィック (シドニー)、カナダ (中部)、欧州 (フランクフルト)、欧州 (ロンドン)、中東 (バーレーン)、南米 (サンパウロ)

機能	拡張クラスター	ローカルクラスター
Kubernetes マイナーバージョン	サポートされている Amazon EKS バージョン。	サポートされている Amazon EKS バージョン。
プラットフォームのバージョン	「Amazon EKS のプラットフォームバージョン」を参照してください	「Amazon EKS ローカルクラスタープラットフォームのバージョン」を参照してください
Outpost フォームファクター	Outpost ラック	Outpost ラック
ユーザーインターフェイス	AWS Management Console、AWS CLI、Amazon EKS API、`eksctl`、AWS CloudFormation、および Terraform	AWS Management Console、AWS CLI、Amazon EKS API、`eksctl`、AWS CloudFormation、および Terraform
マネージドポリシー	AmazonEKSClusterPolicy および AmazonEKSServiceRolePolicy	AmazonEKSLocalOutpostClusterPolicy および AmazonEKSLocalOutpostServiceRolePolicy
クラスター VPC およびサブネット	「Amazon EKS VPC およびサブネットの要件と考慮事項」を参照してください	「Amazon EKS ローカルクラスター VPC およびサブネットの要件と考慮事項」を参照してください
クラスターエンドポイントのアクセス	公共または個人、あるいはその両方	非公開のみ
Kubernetes API サーバー認証	AWS Identity and Access Management (IAM) および OIDC	IAM および `x.509` 証明書
ノードタイプ	セルフマネージド型のみ	セルフマネージド型のみ
ノードのコンピュートタイプ	Amazon EC2 オンデマンド	Amazon EC2 オンデマンド

機能	拡張クラスター	ローカルクラスター
ノードストレージタイプ	Amazon EBS gp2 とローカル NVMe SSD	Amazon EBS gp2 とローカル NVMe SSD
Amazon EKS 最適化 AMI	Amazon Linux、Windows、Bottlerocket	Amazon Linux のみ
IP バージョン	IPv4 のみ	IPv4 のみ
アドオン	Amazon EKS アドオンまたはセルフマネージドアドオン	セルフマネージド型アドオンのみ
デフォルトの Container Network Interface	Amazon VPC CNI plugin for Kubernetes	Amazon VPC CNI plugin for Kubernetes
Kubernetes コントロールプレーンのログ	Amazon CloudWatch Logs	Amazon CloudWatch Logs
負荷分散	AWS Load Balancer Controller を使用して Application Load Balancer のみをプロビジョニングします (Network Load Balancer はプロビジョニングしない)	AWS Load Balancer Controller を使用して Application Load Balancer のみをプロビジョニングします (Network Load Balancer はプロビジョニングしない)
シークレットエンベロープ暗号化	「既存のクラスター上でシークレット暗号化を有効にする」を参照してください	サポートされません
サービスアカウントの IAM ロール	「サービスアカウントの IAM ロール」を参照してください	サポートされません
トラブルシューティング	「Amazon EKS のトラブルシューティング」を参照してください	「AWS Outposts での Amazon EKS のローカルクラスターのトラブルシューティング」を参照してください

AWS Outposts の Amazon EKS のローカルクラスター

ローカルクラスターを使用すると、Amazon EKS クラスターすべてを AWS Outposts でローカルに実行できます。これにより、クラウドへの一時的なネットワーク切断によって生じる可能性のあるアプリケーションダウンタイムのリスクを軽減できます。このような接断は、ファイバーの切断や天候によって発生する可能性があります。Kubernetes クラスター全体が Outposts でローカルに実行されるため、アプリケーションは引き続き使用できます。クラウドへのネットワーク切断中にクラスターオペレーションを実行できます。詳細については、「ネットワーク切断の準備」を参照してください。次の図は、ローカルクラスターのデプロイを示しています。



ローカルクラスターは一般的に Outposts ラックで使用できます。

サポートされている AWS リージョン

ローカルクラスターを作成できる AWS リージョン は、米国東部 (バージニア北部)、米国東部 (オハイオ)、米国西部 (北カリフォルニア)、米国西部 (オレゴン)、アジアパシフィック (ソウル)、アジアパシフィック (シンガポール)、アジアパシフィック (東京)、アジアパシフィック (シドニー)、カナダ (中部)、欧州 (フランクフルト)、欧州 (ロンドン)、中東 (バーレーン)、南米 (サンパウロ) です。サポートされる機能の詳細については、「デプロイオプションの比較」を参照してください。

トピック

- Outpost でのローカルクラスターの作成
- Amazon EKS ローカルクラスタープラットフォームのバージョン
- Amazon EKS ローカルクラスター VPC およびサブネットの要件と考慮事項
- ネットワーク切断の準備
- 容量に関する考慮事項
- AWS Outposts での Amazon EKS のローカルクラスターのトラブルシューティング

Outpost でのローカルクラスターの作成

このトピックでは、Outpost でローカルクラスターを実行する際に考慮すべき事項について、概要を説明します。また、Outpost でローカルクラスターをデプロイする方法についても説明します。

考慮事項

> **Important**
>
> - これらの考慮事項は、関連する Amazon EKS ドキュメントには記載されていません。Amazon EKS ドキュメントに記載されている他のトピックの情報が、ここに紹介する考慮事項と競合する場合は、こちらの考慮事項を優先してください。
> - これらの考慮事項は変更される可能性があり、変更は頻繁に行われる場合があります。そのため、このトピックは定期的に確認することをお勧めします。
> - 考慮事項の多くは、AWS クラウド でクラスターを作成する場合の考慮事項とは異なります。

- ローカルクラスターは Outpost ラックのみをサポートします。単一のローカルクラスターは、単一の論理 Outpost を構成する複数の物理 Outpost ラックにわたって実行できます。単一のローカルクラスターを複数の論理 Outposts にわたって実行することはできません。各論理 Outpost には、単一の Outpost ARN があります。
- ローカルクラスターは、Outpost のアカウントで Kubernetes コントロールプレーンを実行および管理します。Kubernetes コントロールプレーンインスタンスでワークロードを実行したり、Kubernetes コントロールプレーンのコンポーネントを変更することはできません。これらのノードは Amazon EKS サービスによって管理されます。Kubernetes コントロールプレーンへの変更は、パッチ適用などの自動 Amazon EKS 管理アクションでは存続しません。
- ローカルクラスターは、セルフマネージド型アドオンとセルフマネージド型 Amazon Linux ノードグループをサポートしています。Amazon VPC CNI plugin for Kubernetes、`kube-proxy`、および CoreDNS アドオンはローカルクラスターに自動的にインストールされます。
- ローカルクラスターでは、Outposts で Amazon EBS を使用する必要があります。お客様の Outpost では、Kubernetes コントロールプレーンストレージに使用できる Amazon EBS が必要です。
- ローカルクラスターは、Outposts で Amazon EBS を使用します。お客様の Outpost では、Kubernetes コントロールプレーンストレージに使用できる Amazon EBS が必要です。Outposts は、Amazon EBS gp2 ボリュームのみをサポートします。
- Amazon EBS のバックアップ対象である Kubernetes `PersistentVolumes` は、Amazon EBS CSI ドライバーを使用してサポートされています。

前提条件

- Outposts deployment options (Outposts デプロイオプション)、容量に関する考慮事項、および Amazon EKS ローカルクラスター VPC およびサブネットの要件と考慮事項 に精通していること。
- 既存の Outpost。詳細については、「AWS Outposts とは」を参照してください。
- コンピュータまたは AWS CloudShell に、`kubectl` コマンドラインツールがインストールされていること。バージョンは、ご使用のクラスターの Kubernetes バージョンと同じか、1 つ前のマイナーバージョン以前、あるいはそれより新しいバージョンが使用できます。例えば、クラスターのバージョンが `1.29` である場合、`kubectl` のバージョン `1.28`、`1.29`、または `1.30` が使用できます。`kubectl` をインストールまたはアップグレードする方法については、「kubectl のインストールまたは更新」を参照してください。
- ご使用のデバイスまたは AWS CloudShell で、バージョン `2.12.3` 以降、または AWS Command Line Interface (AWS CLI) のバージョン `1.27.160` 以降がインストールおよび設定されていること。現在のバージョンを確認するには、「**`aws --version | cut -d / -f2 | cut -d ' ' -f1`**」を参照してください。macOS の `yum`、`apt-get`、または Homebrew などのパッケージマネージャは、AWS CLI の最新バージョンより数バージョン遅れることがあります。最新バージョンをインストールするには、「AWS Command Line Interface ユーザーガイド」の「AWS CLI のインストール、更新、およびアンインストール」と「aws configure でのクイック設定」を参照してください。AWS CloudShell にインストールされている AWS CLI バージョンは、最新バージョンより数バージョン遅れている可能性もあります。更新するには、「AWS CloudShell ユーザーガイド」の「ホームディレクトリへの AWS CLI のインストール」を参照してください。
- Amazon EKS クラスターへの `create` および `describe` のアクセス許可を持つ IAM プリンシパル (ユーザーまたはロール)。詳細については、Outpost にローカル Kubernetes クラスターを作成しますおよびすべてのクラスターの一覧表示または説明を参照してください。

ローカルの Amazon EKS クラスターが作成される時点で、クラスターを作成する IAM プリンシパルが恒久的に追加されています。とりわけ、プリンシパルは管理者として、Kubernetes RBAC 承認テーブルに追加されます。このエンティティには `system:masters` アクセス許可が付与されています。このエンティティの ID は、クラスター設定には表示されません。したがって、クラスターを作成したエンティティに注意し、削除しないようにすることが重要です。初期状態では、サーバーを作成したプリンシパルのみが、`kubectl` を使用して Kubernetes API サーバーへのコールを実行することができます。コンソールを使用してクラスターを作成する場合は、クラスター上で `kubectl` コマンドを実行する際、同じ IAM 認証情報が AWS SDK 認証情報チェーンにあることを確認する必要があります。クラスターの作成が完了したら、他の IAM プリンシパルにクラスターへのアクセスを付与できます。

ローカルの Amazon EKS ローカルクラスターを作成するには

eksctl、AWS Management Console、AWS CLI、Amazon EKS API、AWS SDK、AWS CloudFormation、または Terraform を使用して、ローカルクラスターを作成できます。

1. ローカルクラスターを作成する。

   eksctl

   前提条件

   デバイスまたは AWS CloudShell にインストールされている eksctl コマンドラインツールのバージョン 0.183.0 以降。eksctl をインストールまたはアップグレードするには、eksctl ドキュメントの「インストール」を参照してください。

   eksctl を使用してクラスターを作成するには

   1. 次のコンテンツをデバイスにコピーします。次の値を置き換えたら、変更したコマンドを実行して outpost-control-plane.yaml ファイルを作成します。
      - *region-code* を、クラスターを作成するサポートされている AWS リージョン に置き換えます。
      - *my-cluster* をクラスターの名前に置き換えます。この名前には、英数字 (大文字と小文字が区別されます) とハイフンのみを使用できます。先頭の文字は英数字である必要があります。また、100 文字より長くすることはできません。名前は、クラスターを作成する AWS リージョン および AWS アカウント 内で一意である必要があります。名前は、クラスターを作成する AWS リージョン および AWS アカウント 内で一意である必要があります。
      - *vpc-ExampleID1* と *subnet-ExampleID1* を既存の VPC とサブネットの ID に置き換えます。VPC とサブネットは、Amazon EKS ローカルクラスター VPC およびサブネットの要件と考慮事項 に記載される要件を満たしている必要があります。
      - *uniqueid* を Outpost の ID に置き換えます。
      - *m5.large* を、Outpost で使用可能なインスタンスタイプに置き換えます。インスタンスタイプを選択する前に、「容量に関する考慮事項」を参照してください。3 つのコントロールプレーンインスタンスがデプロイされます。この番号を変更することはできません。

      ```
      cat >outpost-control-plane.yaml <<EOF
      apiVersion: eksctl.io/v1alpha5
      ```

```
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
  version: "1.24"

vpc:
  clusterEndpoints:
    privateAccess: true
  id: "vpc-vpc-ExampleID1"
  subnets:
    private:
      outpost-subnet-1:
        id: "subnet-subnet-ExampleID1"

outpost:
  controlPlaneOutpostARN: arn:aws:outposts:region-code:111122223333:outpost/
op-uniqueid
  controlPlaneInstanceType: m5.large
EOF
```

使用可能なすべてのオプションとデフォルトの完全なリストについては、eksctl ドキュメントの「AWS Outposts サポート」および「設定ファイルスキーマ」を参照してください。

2. 前のステップで作成した設定ファイルを使用してクラスターを作成します。eksctl は、クラスターをデプロイするために VPC と 1 つのサブネットを Outpost に作成します。

```
eksctl create cluster -f outpost-control-plane.yaml
```

クラスターのプロビジョニングには数分かかります。クラスターの作成中は、数行の出力が表示されます。出力の最後の行は、次のサンプル行のようになります。

```
[#]  EKS cluster "my-cluster" in "region-code" region is ready
```

> **Tip**
>
> eksctl を使用してクラスターを作成するときに指定できるほとんどのオプションを表示するには、**eksctl create cluster --help** コマンドを使用します。使用

可能なオプションをすべて表示するには、config ファイルを使用します。詳細については、「eksctl ドキュメント」の「Using config files」(設定ファイルの使用) と「設定ファイルのスキーマ」を参照してください。設定ファイルの例は、GitHub で見つけることができます。

Eksctl はクラスターを作成した IAM プリンシパル (ユーザーまたはロール) のアクセスエントリを自動的に作成し、クラスター上の Kubernetes オブジェクトに対して IAM プリンシパル管理者にアクセス許可を付与しました。クラスター作成者にクラスター上の Kubernetes オブジェクトへの管理者アクセス権を付与したくない場合は、前の設定ファイルに次のテキストを追加します: **bootstrapClusterCreatorAdminPermissions: false** (metadata、vpc および outpost と同じレベル)。オプションを追加した場合、クラスターの作成後に少なくとも 1 つの IAM プリンシパルのアクセスエントリを作成する必要があります。そうしないと、IAM プリンシパルはクラスター上の Kubernetes オブジェクトにアクセスできなくなります。

AWS Management Console

前提条件

Amazon EKS の要件を満たす既存の VPC とサブネット。詳細については、「Amazon EKS ローカルクラスター VPC およびサブネットの要件と考慮事項」を参照してください。

AWS Management Console を使用してクラスターを作成するには

1. 既にローカルクラスター IAM ロールがある場合、または eksctl を使用してクラスターを作成する場合は、このステップはスキップできます。デフォルトでは、eksctl により、ロールが自動的に作成されます。

   a. IAM 信頼ポリシー用の JSON ファイルを作成するには、次のコマンドを実行します。

```
cat >eks-local-cluster-role-trust-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
```

```
    }
  ]
}
EOF
```

b. Amazon EKS クラスターの IAM ロールを作成します。IAM ロールを作成するには、ロールを作成する IAM プリンシパルに `iam:CreateRole` アクション (許可) を割り当てる必要があります。

```
aws iam create-role --role-name myAmazonEKSLocalClusterRole --assume-role-policy-document file://"eks-local-cluster-role-trust-policy.json"
```

c. このロールに、Amazon EKS 管理の IAM ポリシー (AmazonEKSLocalOutpostClusterPolicy) をアタッチします。IAM ポリシーを IAM プリンシパルにアタッチするには、ポリシーのアタッチを行っているプリンシパルに、次のいずれかの IAM アクション (許可) を割り当てる必要があります: `iam:AttachUserPolicy` または `iam:AttachRolePolicy`。

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/AmazonEKSLocalOutpostClusterPolicy --role-name myAmazonEKSLocalClusterRole
```

2. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。
3. コンソール画面の上部で、サポートされている AWS リージョン を選択したことを確認します。
4. [クラスターを追加]、[作成] の順にクリックします。
5. [クラスターの設定] ページで、次のフィールドの値を入力または選択します。
   - [Kubernetes コントロールプレーンの位置] - AWS Outposts を選択します。
   - [Outpost ID] - コントロールプレーンを作成する Outpost の ID を選択します。
   - [インスタンスタイプ] - インスタンスタイプを選択します。Outpost で使用可能なインスタンスタイプのみが表示されます。ドロップダウンリストでは、各インスタンスタイプにそのインスタンスタイプに推奨されるノード数が示されています。インスタンスタイプを選択する前に、「容量に関する考慮事項」を参照してください。すべてのレプリカは、同じインスタンスタイプを使用してデプロイされます。クラスターの作成後にインスタンスタイプを変更することはできません。3 つのコントロールプレーンインスタンスがデプロイされます。この番号を変更することはできません。

- [名前] - クラスターの名前。AWS アカウント内で一意である必要があります。この名前には、英数字 (大文字と小文字が区別されます) とハイフンのみを使用できます。先頭の文字は英数字である必要があります。また、100 文字より長くすることはできません。名前は、クラスターを作成する AWS リージョン および AWS アカウント 内で一意である必要があります。名前は、クラスターを作成する AWS リージョン および AWS アカウント 内で一意である必要があります。
- Kubernetes バージョン – クラスターで使用する Kubernetes バージョンを選択します。以前のバージョンを使用する必要がない限り、最新バージョンを選択することをお勧めします。
- [クラスターサービスロール] - AWS リソースを管理することを Kubernetes コントロールプレーンに許可するために、前のステップで作成した Amazon EKS クラスター IAM ロールを選択します。
- Kubernetes クラスター管理者アクセス — クラスターを作成している IAM プリンシパル (ロールまたはユーザー) にクラスター上の Kubernetes オブジェクトへの管理者アクセス権を付与する場合は、デフォルト (許可) をそのまま使用します。Amazon EKS は IAM プリンシパルのアクセスエントリを作成し、クラスター管理者にそのアクセスエントリに対するアクセス許可を付与します。アクセスエントリの詳細については、「アクセスエントリを管理する」を参照してください。

  クラスターを作成するプリンシパルとは異なる IAM プリンシパルに Kubernetes クラスターオブジェクトへの管理者アクセス権を付与したい場合は、拒否のオプションを選択します。クラスターの作成後、アクセスエントリを作成する IAM アクセス許可を持つすべての IAM プリンシパルは、Kubernetes クラスターオブジェクトへのアクセスを必要とするすべての IAM プリンシパルのアクセスエントリを追加できます。必要な IAM アクセス許可の詳細については、サービス認証リファレンスの「Amazon Elastic Kubernetes Service で定義されるアクション」を参照してください。拒否のオプションを選択し、アクセスエントリを作成しない場合、IAM プリンシパルはクラスター上の Kubernetes オブジェクトにアクセスできなくなります。
- [タグ] - (オプション) クラスターにタグを追加します。詳細については、「Amazon EKS リソースのタグ付け」を参照してください。

このページを読み終えたら、[次へ] を選択します。

6. [ネットワーキングの指定] ページで、次のフィールドの値を選択します。
   - [VPC] - 既存の VPC を選択します。VPC には、作成するクラスター、ノード、およびその他の Kubernetes リソースで利用できる十分な数の IP アドレスが必要です。VPC は、VPC の要件と考慮事項 に記載されている要件を満たしている必要があります。

- [サブネット] - デフォルトで、前のフィールドで指定した VPC 内の利用可能なすべてのサブネットがあらかじめ選択されています。選択するサブネットは サブネットの要件と考慮事項 に記載される要件を満たしている必要があります。

  [セキュリティグループ] - (オプション) Amazon EKS が作成するネットワークインターフェイスに関連付ける 1 つまたは複数のセキュリティグループを指定します。Amazon EKS は、クラスターと VPC 間の通信を可能にするセキュリティグループを自動的に作成します。Amazon EKS は、このセキュリティグループおよびユーザーが選択したセキュリティグループを、作成するネットワークインターフェイスに関連付けます。Amazon EKS が作成するクラスターセキュリティグループの詳細については、「Amazon EKS セキュリティグループの要件および考慮事項」を参照してください。Amazon EKS が作成するクラスターセキュリティグループのルールを変更できます。独自のセキュリティグループを追加することを選択した場合、クラスターの作成後に選択したセキュリティグループを変更することはできません。オンプレミスホストがクラスターエンドポイントと通信するためには、クラスターセキュリティグループからのインバウンドトラフィックを許可する必要があります。入出力のインターネット接続がないクラスター (プライベートクラスターとも呼ばれる) の場合は、次のいずれかを行う必要があります。

  - 必要となる VPC エンドポイントに関連付けられたセキュリティグループを追加します。必要となるエンドポイントの詳細については、AWS のサービス へのサブネットアクセス の「インターフェイス VPC エンドポイント」を参照してください。
  - VPC エンドポイントに関連付けられたセキュリティグループからのトラフィックを許可するように、Amazon EKS が作成したセキュリティグループを変更します。

  このページを読み終えたら、[次へ] を選択します。

7. [オブザーバビリティの設定] ページでは、有効にする [メトリクス] と [コントロールプレーンのロギング] オプションをオプションで選択できます。デフォルトでは、それぞれのログタイプは無効化されています。

   - Prometheus メトリクスオプションの追加についての詳細は、「クラスターを作成するときは、Prometheus メトリクスを有効にしてください。」を参照してください。
   - 「コントロールプレーン」に関する詳細に関しては、「Amazon EKS コントロールプレーンのログ記録」を参照してください。

   このページを読み終えたら、[次へ] を選択します。

8. [確認と作成] ページで、前のページで入力または選択した情報を確認します。変更する必要がある場合は、[編集] を選択します。そのままでよければ、[作成] を選択します。クラ

スターがプロビジョニングされている間、[ステータス] フィールドに [作成中] と表示されます。

クラスターのプロビジョニングには数分かかります。

2. クラスターを作成すると、作成された Amazon EC2 コントロールプレーンのインスタンスを表示できます。

```
aws ec2 describe-instances --query 'Reservations[*].Instances[*].{Name:Tags[?
Key==`Name`]|[0].Value}' | grep my-cluster-control-plane
```

出力例は次のとおりです。

```
"Name": "my-cluster-control-plane-id1"
"Name": "my-cluster-control-plane-id2"
"Name": "my-cluster-control-plane-id3"
```

各インスタンスは node-role.eks-local.amazonaws.com/control-plane で汚染されているため、コントロールプレーンインスタンスでワークロードがスケジュールされることはありません。テイントの詳細については、Kubernetes ドキュメントの「テイントと容認」を参照してください。Amazon EKS はローカルクラスターの状態を継続的に監視します。セキュリティパッチや問題のあるインスタンスの修復などの自動管理アクションを実行します。ローカルクラスターがクラウドから切断されると、再接続時にクラスターが正常な状態に修復されるようアクションが実行されます。

3. eksctl を使用してクラスターを作成した場合、このステップはスキップできます。eksctl がこのステップを完了します。新しいコンテキストを kubectl config ファイルに追加して、kubectl がクラスターと通信できるようにします。ファイルを作成し更新する手順については、「Amazon EKS クラスターの kubeconfig ファイルを作成または更新する」を参照してください。

```
aws eks update-kubeconfig --region region-code --name my-cluster
```

出力例は次のとおりです。

```
Added new context arn:aws:eks:region-code:111122223333:cluster/my-cluster to /home/
username/.kube/config
```

4. ローカルクラスターの Kubernetes API サーバーに接続するには、サブネットのローカルゲートウェイにアクセスするか、VPC 内から接続します。Outpost ラックをオンプレミスネット

ワークに接続する方法の詳細については、「AWS Outposts ユーザーガイド」の「How local gateways for racks work」(ラックのローカルゲートウェイの仕組み) を参照してください。ダイレクト VPC ルーティングを使用しており、Outpost サブネットにローカルゲートウェイへのルートがある場合、Kubernetes コントロールプレーンインスタンスのプライベート IP アドレスがローカルネットワークを介して自動的にブロードキャストされます。ローカルクラスターの Kubernetes API サーバーエンドポイントは Amazon Route 53 (Route 53) でホストされます。API サービスエンドポイントは、パブリック DNS サーバーによって Kubernetes API サーバーのプライベート IP アドレスに解決されます。

ローカルクラスターの Kubernetes コントロールプレーンインスタンスは、クラスターのライフサイクルを通じて変更されない固定プライベート IP アドレスを静的なエラスティックネットワークインターフェイスで構成されます。Kubernetes API サーバーとやり取りするマシンは、ネットワーク接続が切断されている間は Route 53 に接続できない場合があります。このような場合は、操作の継続性を確保するために、静的プライベート IP アドレスを使用して `/etc/hosts` を構成することをお勧めします。また、ローカル DNS サーバーを設定して Outpost に接続することもお勧めします。詳細については、「AWS Outpostsドキュメント」を参照してください。次のコマンドを実行して、クラスターとの通信が確立されていることを確認します。

```
kubectl get svc
```

出力例は次のとおりです。

```
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   28h
```

5. (オプション) AWS クラウド から切断された状態のときにローカルクラスターへの認証をテストします。手順については、ネットワーク切断の準備 を参照してください。

内部リソース

Amazon EKS はクラスターで次のリソースを作成します。リソースは、Amazon EKS 内部で使用するためのものです。クラスターが適切に機能しなくなるため、これらのリソースは編集または変更しないでください。

- 次のミラー Pods:
  - `aws-iam-authenticator-node-hostname`
  - `eks-certificates-controller-node-hostname`

  - etcd-*node-hostname*
  - kube-apiserver-*node-hostname*
  - kube-controller-manager-*node-hostname*
  - kube-scheduler-*node-hostname*
- 次のセルフマネージド型アドオン:
  - kube-system/coredns
  - kube-system/kube-proxy (最初のノードを追加するまで作成されません)
  - kube-system/aws-node (最初のノードを追加するまで作成されません)。ローカルクラスターは、クラスターネットワークに Amazon VPC CNI plugin for Kubernetes プラグインを使用します。コントロールプレーンインスタンス (名前が aws-node-controlplane-* のポッド) の設定を変更しないでください。プラグインが新しいネットワークインターフェイスを作成したときに、デフォルト値を変更できる設定変数があります。詳細については、GitHub のドキュメントを参照してください。
- 次のサービス:
  - default/kubernetes
  - kube-system/kube-dns
- eks.system という名前の PodSecurityPolicy
- eks:system:podsecuritypolicy という名前の ClusterRole
- eks:system という名前の ClusterRoleBinding
- デフォルトの PodSecurityPolicy
- クラスターセキュリティグループに加えて、Amazon EKS は eks-local-internal-do-not-use-or-edit-*cluster-name*-*uniqueid* という名前の AWS アカウント にセキュリティグループを作成します。このセキュリティグループにより、コントロールプレーンインスタンスで実行されている Kubernetes コンポーネント間のトラフィックが自由に流れるようになります。

推奨される次の手順は以下の通りです。

- クラスターを作成した IAM プリンシパルに、AWS Management Console 内の Kubernetes リソースを表示するために必要な許可を付与します
- IAM エンティティにクラスターへのアクセスを許可する。エンティティに Amazon EKS コンソールで Kubernetes リソースを表示させる場合は、必要なアクセス許可 をエンティティに付与します。
- クラスターのログ記録を設定する

- ネットワークの切断中に何が起こるかをよく理解してください。
- クラスターにノードを追加する
- etcd のバックアップ計画を設定することを検討してください。Amazon EKS は、ローカルクラスターの etcd の自動バックアップと復元をサポートしていません。詳細については、「Kubernetes ドキュメント」の「etcd クラスターのバックアップ」を参照してください。2 つの主なオプションは、etcdctl を使用してスナップショットの作成を自動化する、または Amazon EBS ストレージボリュームのバックアップを使用することです。

Amazon EKS ローカルクラスタープラットフォームのバージョン

ローカルクラスタープラットフォームのバージョンは、AWS Outposts 上の Amazon EKS クラスターの機能を表します。バージョンには、Kubernetes コントロールプレーン上で実行されるコンポーネントが含まれており、Kubernetes API サーバーフラグが有効になっています。また、現在の Kubernetes パッチバージョンも含まれています。Kubernetes マイナーバージョンにはそれぞれ、プラットフォームのバージョンが 1 つ以上関連付けられています。別の Kubernetes マイナーバージョンのプラットフォームバージョンは独立しています。クラウド内のローカルクラスターと Amazon EKS クラスターのプラットフォームバージョンは独立しています。

1.28 など、新しい Kubernetes マイナーバージョンがローカルクラスターで使用可能になると、その Kubernetes マイナーバージョンの最初のプラットフォームバージョンは eks-local-outposts.1 から始まります。ただし、Amazon EKS では新しいプラットフォームバージョンを定期的にリリースすることで、新しい Kubernetes コントロールプレーン設定を有効化し、セキュリティ修正プログラムを提供します。

マイナーバージョンで新しいローカルクラスタープラットフォームのバージョンが使用可能になった場合:

- プラットフォームのバージョン番号がインクリメントされます (eks-local-outposts.n+1)。
- Amazon EKS は、既存のすべてのローカルクラスターを、対応する Kubernetes マイナーバージョン用の最新のプラットフォームバージョンに自動的にアップデートします。既存のプラットフォームバージョンの自動更新は段階的にロールアウトされます。ロールアウトプロセスには時間がかかる場合があります。最新のプラットフォームバージョンの機能がすぐに必要な場合は、新しいローカルクラスターを作成することをお勧めします。
- Amazon EKS は、対応するパッチバージョンを適用して新しいノード AMI を発行する可能性があります。すべてのパッチバージョンは、単一の Kubernetes マイナーバージョンに対して、Kubernetes コントロールプレーンとノード AMI との間で互換性があります。

新しい プラットフォームバージョンでは、重大な変更が発生したり、サービスが中断されたりすることはありません。

ローカルクラスターは常に、指定された Kubernetes バージョン用に入手可能な、最新のプラットフォームバージョン (eks-local-outposts.n) で作成されます。

現在および最新の プラットフォームバージョンを以下の表に示します。

Kubernetes バージョン 1.28

CertificateApproval、CertificateSigning、CertificateSubjectRestriction、DefaultIn および ValidatingAdmissionWebhook のアドミッションコントローラーは、すべての 1.28 プラットフォームバージョンに有効です。

Kubernetes バージョン	Amazon EKS プラットフォームバージョン	リリースノート	リリース日
1.28.6	eks-local-outposts.5	Bottlerocket バージョンを v1.19.3 に更新し、Outposts でのローカルブートをサポートする最新のバグ修正を追加しました。	2024 年 4 月 18 日
1.28.6	eks-local-outposts.4	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。Outposts でサポートまたはローカルブートを復元しました。互換性のために Bottlerocket バージョンを v1.15.1 にダウングレードしました。	2024 年 4 月 2 日
1.28.6	eks-local-outposts.3	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2024 年 3 月 22 日
1.28.6	eks-local-outposts.2	セキュリティの修正および機能強化が行われた新しいプラッ	2024 年 3 月 8 日

Kubernetes バージョン	Amazon EKS プラットフォームバージョン	リリースノート	リリース日
		トフォームバージョン。kube-proxy は v1.28.6 に更新されました。AWSIAM Authenticator が v0.6.17 に更新されました。互換性の理由から、Amazon VPC CNI plugin for Kubernetes が v1.13.2 にダウングレードされました。Bottlerocket バージョンが v1.19.2 に更新されました。	
1.28.1	eks-local-outposts.1	Outpost のローカル Amazon EKS クラスター用 Kubernetes バージョン v1.28 の初回リリース。	2023 年 10 月 4 日

Kubernetes バージョン **1.27**

CertificateApproval、CertificateSigning、CertificateSubjectRestriction、DefaultIn および ValidatingAdmissionWebhook のアドミッションコントローラーは、すべての 1.27 プラットフォームバージョンに有効です。

Kubernetes バージョン	Amazon EKS プラットフォームバージョン	リリースノート	リリース日
1.27.10	eks-local-outposts.5	セキュリティの修正および機能強化が行われた新しいプラットフォーム。	2024 年 4 月 2 日
1.27.10	eks-local-outposts.4	セキュリティの修正および機能強化が行われた新しいプラットフォーム。kube-proxy	2024 年 3 月 22 日

Kubernetes バージョン	Amazon EKS プラットフォームバージョン	リリースノート	リリース日
		は v1.27.10 に更新されました。AWSIAM Authenticator が v0.6.17 に更新されました。Bottlerocket バージョンが v1.19.2 に更新されました。	
1.27.3	eks-local-outposts.3	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。kube-proxy は v1.27.3 に更新されました。Kubernetes 向け Amazon VPC CNI プラグインは v1.13.2 に更新されました。	2023 年 7 月 14 日
1.27.1	eks-local-outposts.2	CoreDNS イメージを v1.10.1 に更新しました	2023 年 6 月 22 日
1.27.1	eks-local-outposts.1	Outposts のローカル Amazon EKS クラスター用 Kubernetes バージョン 1.27 の初回リリース。	2023 年 5 月 30 日

Kubernetes バージョン **1.26**

CertificateApproval、CertificateSigning、CertificateSubjectRestriction、DefaultIn および ValidatingAdmissionWebhook のアドミッションコントローラーは、すべての 1.26 プラットフォームバージョンに有効です。

Kubernetes バージョン	Amazon EKS プラットフォームバージョン	リリースノート	リリース日
1.26.13	eks-local-outposts.5	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。kube-proxy は v1.26.13 に更新されました。AWSIAM Authenticator が v0.6.17 に更新されました。Bottlerocket バージョンが v1.19.2 に更新されました。	2024 年 3 月 22 日

Kubernetes バージョン 1.25

次のアドミッションコントローラーは、すべての 1.25 プラットフォームバージョンで有効です: CertificateApproval、CertificateSigning、CertificateSubjectRestriction、DefaultIn および ValidatingAdmissionWebhook。

Kubernetes バージョン	Amazon EKS プラットフォームバージョン	リリースノート	リリース日
1.25.16	eks-local-outposts.7	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。kube-proxy は v1.25.16 に更新されました。AWSIAM Authenticator が v0.6.17 に更新されました。Bottlerocket バージョンが v1.19.2 に更新されました。	2024 年 3 月 22 日
1.25.11	eks-local-outposts.6	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。kube-proxy は v1.25.11 に更新	2023 年 7 月 14 日

Kubernetes バージョン	Amazon EKS プラットフォームバージョン	リリースノート	リリース日
		されました。Kubernetes 向け Amazon VPC CNI プラグインは v1.13.2 に更新されました。	
1.25.9	eks-local-outposts.5	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 7 月 13 日
1.25.6	eks-local-outposts.4	Bottlerocket のバージョンを 1.13.2 に更新しました。	2023 年 5 月 2 日
1.25.6	eks-local-outposts.3	Amazon EKS コントロールプレーンインスタンスのオペレーティングシステムが Bottlerocket バージョン v1.13.1 に更新され、Amazon VPC CNI plugin for Kubernetes はバージョン v1.12.6 に更新されました。	2023 年 4 月 14 日
1.25.6	eks-local-outposts.2	Kubernetes コントロールプレーンインスタンスの診断収集が改善されました。	2023 年 3 月 8 日
1.25.6	eks-local-outposts.1	Outposts のローカル Amazon EKS クラスター用 Kubernetes バージョン 1.25 の初回リリース。	2023 年 3 月 1 日

Kubernetes バージョン 1.24

次のアドミッションコントローラーは、すべての 1.24 プラットフォームバージョンで有効です: DefaultStorageClass、DefaultTolerationSeconds、LimitRanger、MutatingAdmissionWebh および DefaultIngressClass。

Kubernetes バージョン	Amazon EKS プラットフォームバージョン	リリースノート	リリース日
1.24.17	eks-local-outposts.7	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。kube-proxy は v1.25.16 に更新されました。AWSIAM Authenticator が v0.6.17 に更新されました。Bottlerocket バージョンが v1.19.2 に更新されました。	2024 年 3 月 22 日
1.24.15	eks-local-outposts.6	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。kube-proxy は v1.24.15 に更新されました。Kubernetes 向け Amazon VPC CNI プラグインは v1.13.2 に更新されました。	2023 年 7 月 14 日
1.24.13	eks-local-outposts.5	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 7 月 13 日
1.24.9	eks-local-outposts.4	Bottlerocket のバージョンを 1.13.2 に更新しました。	2023 年 5 月 2 日

Kubernetes バージョン	Amazon EKS プラットフォームバージョン	リリースノート	リリース日
1.24.9	eks-local-outposts.3	Amazon EKS コントロールプレーンインスタンスのオペレーティングシステムが Bottlerocket バージョン v1.13.1 に更新され、Amazon VPC CNI plugin for Kubernetes はバージョン v1.12.6 に更新されました。	2023 年 4 月 14 日
1.24.9	eks-local-outposts.2	Kubernetes コントロールプレーンインスタンスの診断収集が改善されました。	2023 年 3 月 8 日
1.24.9	eks-local-outposts.1	Outposts のローカル Amazon EKS クラスター用 Kubernetes バージョン 1.24 の初回リリース。	2023 年 1 月 17 日

Kubernetes バージョン **1.23**

次のアドミッションコントローラーは、すべての 1.23 プラットフォームバージョンで有効です: DefaultStorageClass、DefaultTolerationSeconds、LimitRanger、MutatingAdmissionWebh および DefaultIngressClass。

Kubernetes バージョン	Amazon EKS プラットフォームバージョン	リリースノート	リリース日
1.23.17	eks-local-outposts.6	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。	2023 年 7 月 13 日

Kubernetes バージョン	Amazon EKS プラットフォームバージョン	リリースノート	リリース日
1.23.17	eks-local-outposts.5	セキュリティの修正および機能強化が行われた新しいプラットフォームバージョン。kube-proxy は v1.23.17 に更新されました。Bottlerocket バージョンが v1.14.1 に更新されました。	2023 年 7 月 6 日
1.23.15	eks-local-outposts.4	Amazon EKS コントロールプレーンインスタンスのオペレーティングシステムが Bottlerocket バージョン v1.13.1 に更新され、Amazon VPC CNI plugin for Kubernetes はバージョン v1.12.6 に更新されました。	2023 年 4 月 14 日
1.23.15	eks-local-outposts.3	Kubernetes コントロールプレーンインスタンスの診断収集が改善されました。	2023 年 3 月 8 日
1.23.15	eks-local-outposts.2	Outposts のローカル Amazon EKS クラスター用 Kubernetes バージョン 1.23 の初回リリース。	2023 年 1 月 17 日

Amazon EKS ローカルクラスター VPC およびサブネットの要件と考慮事項

ローカルクラスターを作成する際には、VPC と、Outposts で実行されるプライベートサブネットを少なくとも 1 つ指定します。このトピックでは、ローカルクラスターの VPC およびサブネットの要件と考慮事項について概要を説明します。

VPC の要件と考慮事項

ローカルクラスターを作成する際には、指定する VPC が次の要件と考慮事項を満たす必要があります。

- VPC には、作成するローカルクラスター、すべてのノード、および作成するその他の Kubernetes リソースで利用できるだけの十分な数の IP アドレスが必要です。使用する VPC に十分な IP アドレスがない場合は、使用可能な IP アドレスの数を増やしてみてください。この操作を行うには、VPC への追加の Classless Inter-Domain Routing (CIDR) ブロックの関連付けが必要になります。クラスターの作成前または作成後に、プライベート (RFC 1918) とパブリック (非 RFC 1918) の CIDR ブロックを VPC に関連付けることができます。クラスターで VPC に関連付けた CIDR ブロックが認識されるまでに、最大 5 時間かかることがあります。
- VPC に IP プレフィックスを割り当てたり、IPv6 CIDR ブロックを持たせることはできません。これらの制約があるため、Amazon EC2 ノードで使用可能な IP アドレスの量を増やす および クラスター、Pods、services 用の IPv6 アドレス でカバーされた情報は VPC には適用されません。
- VPC では DNS ホスト名と DNS 解決が有効になっています。これらの機能がないと、ローカルクラスターの作成は失敗し、機能を有効にしてクラスターを再作成する必要があります。詳細については、「Amazon VPC ユーザーガイド」の「DNS attributes for your VPC」(VPC の DNS 属性) を参照してください。
- ローカルネットワーク経由でローカルクラスターにアクセスするには、VPC が Outpost のローカルゲートウェイルートテーブルに関連付けられている必要があります。詳細については、「AWS Outposts ユーザーガイド」の「VPC アソシエーション」を参照してください。

サブネットの要件と考慮事項

クラスターを作成する際には、少なくとも 1 つのプライベートサブネットを指定します。複数のサブネットを指定すると、Kubernetes コントロールプレーンインスタンスはサブネット全体に均等に分散されます。1 つ以上のサブネットを指定する場合、サブネットは同じ Outpost 上に存在する必要があります。また、相互に通信するために、サブネットには適切なルートとセキュリティグループ権限も必要です。ローカルクラスターの作成時に、指定するサブネットは次の要件を満たす必要があります。

- サブネットはすべて、同じロジカル Outpost 上に存在すること。
- Kubernetes コントロールプレーンインスタンスに対して、合計で少なくても 3 つ以上の使用可能な IP アドレスを用意してください。3 つのサブネットを指定する場合、各サブネットには少なくても 1 つ以上の使用可能な IP アドレスが必要です。2 つのサブネットを指定する場合、各サブ

ネットには少なくても 2 つ以上の利用可能な IP アドレスが必要です。1 つのサブネットを指定する場合、サブネットには少なくても 3 つ以上の利用可能な IP アドレスが必要です。

- ローカルネットワーク経由で Kubernetes API サーバーにアクセスするため、サブネットに Outpost ラックのローカルゲートウェイへのルートがあります。サブネットに Outpost ラックのローカルゲートウェイへのルートがない場合は、VPC 内から Kubernetes API サーバーと通信する必要があります。
- サブネットでは IP アドレスベースの命名を使用する必要があります。Amazon EC2 のリソースベースの命名は、Amazon EKS でサポートされていません。

AWS のサービス へのサブネットアクセス

Outposts 上のローカルクラスターのプライベートサブネットは、リージョンレベルの AWS のサービス と通信できる必要があります。アウトバウンドインターネットアクセスに NAT ゲートウェイを使用するか、VPC 内ですべてのトラフィックをプライベートに保ちたい場合は、インターフェイス VPC エンドポイントを使用することによって、これを実現できます。

NAT ゲートウェイの使用

Outposts 上のローカルクラスターのプライベートサブネットには、Outposts の親アベイラビリティーゾーンにあるパブリックサブネット内の NAT ゲートウェイへのルートを持つ、関連付けられたルートテーブルが必要です。パブリックサブネットには、インターネットゲートウェイへのルートが必要です。NAT ゲートウェイはアウトバウンドのインターネットアクセスを可能にし、インターネットから Outpost のインスタンスへの未承諾なインバウンド接続を防ぎます。

インターフェイス VPC エンドポイントの使用

Outposts のローカルクラスターのプライベートサブネットにアウトバウンドインターネット接続がない場合、または VPC 内ですべてのトラフィックをプライベートに保ちたい場合は、クラスターを作成する前に、リージョンレベルのサブネットで次のインターフェイス VPC エンドポイントとゲートウェイエンドポイントを作成する必要があります。

エンドポイント	エンドポイントタイプ
`com.amazonaws.` *`region-code`* `.ssm`	インターフェイス
`com.amazonaws.` *`region-code`* `.ssmmessages`	インターフェイス

エンドポイント	エンドポイントタイプ
com.amazonaws. *region-code* .ec2messages	インターフェイス
com.amazonaws. *region-code* .ec2	インターフェイス
com.amazonaws. *region-code* .secretsmanager	インターフェイス
com.amazonaws. *region-code* .logs	インターフェイス
com.amazonaws. *region-code* .sts	インターフェイス
com.amazonaws. *region-code* .ecr.api	インターフェイス
com.amazonaws. *region-code* .ecr.dkr	インターフェイス
com.amazonaws. *region-code* .s3	ゲートウェイ

エンドポイントは次の要件を満たしている必要があります。

- Outpost の親アベイラビリティーゾーンにあるプライベートサブネットに作成されている
- プライベートDNS 名が有効になっている
- プライベート Outpost サブネットの CIDR 範囲からのインバウンド HTTPS トラフィックを許可するセキュリティグループがアタッチされている

エンドポイントを作成すると料金が発生します。詳細については、「AWS PrivateLink 料金表」を参照してください。Pods が他の AWS のサービス にアクセスする必要がある場合は、追加のエンドポイントを作成する必要があります。エンドポイントの包括的なリストについては、「AWS PrivateLink と統合する AWS のサービス」を参照してください。

「VPC を作成する」

以下の AWS CloudFormation テンプレートのいずれかを使用して、前の要件を満たす VPC を作成できます。

- テンプレート 1 — このテンプレートは、Outpost に 1 つのプライベートサブネット、また AWS リージョン に 1 つのパブリックサブネットを持つ VPC を作成します。プライベートサブネットに

は、AWS リージョン のパブリックサブネットにある NAT Gateway 経由でのインターネットへのルートがあります。このテンプレートを使用して、出力インターネットアクセスを持つサブネットにローカルクラスターを作成できます。

- テンプレート 2 — このテンプレートは、Outpost に 1 つのプライベートサブネットと、入力または出力のインターネットアクセスがないサブネット (プライベートサブネットとも呼ばれる) にローカルクラスターを作成するために必要な最小限の VPC エンドポイントセットを含む VPC を作成します。

ネットワーク切断の準備

ローカルネットワークが AWS クラウド との接続を失った場合でも、Outpost にあるローカル Amazon EKS クラスターは引き続き使用できます。このトピックでは、ネットワークの切断とそれに関連する考慮事項に備えて、ローカルクラスターを準備する方法について説明します。

ネットワークの切断に備えてローカルクラスターを準備する際の考慮事項:

- ローカルクラスターにより、計画外の一時的なネットワーク切断中も安定してオペレーションを継続できます。AWS Outposts は、データセンター内の AWS クラウド の拡張機能として機能する完全に接続された製品であることに変わりはありません。Outpost と AWS クラウド 間のネットワークが切断された場合には、接続の復元を試みることをお勧めします。手順については、AWS Outposts ユーザーガイドの「AWS Outposts ラックネットワークのトラブルシューティングチェックリスト」を参照してください。ローカルクラスター上の問題をトラブルシューティングする方法については、「AWS Outposts での Amazon EKS のローカルクラスターのトラブルシューティング」を参照してください。
- Outposts は、Outpost の接続状態を監視するために使用できる `ConnectedStatus` 指標を出力します。詳細については、AWS Outposts ユーザーガイドの「Outposts メトリクス」を参照してください。
- ローカルクラスターでは、Kubernetes の AWS Identity and Access Management 認証を使用する、デフォルトの認証メカニズムとして IAM を使用します。ネットワークの切断中には、IAM は使用できません。ですから、ローカルクラスターが、ネットワークの切断中にクラスターの接続に使用できる `x.509` 証明書を使用する代替認証メカニズムをサポートします。クラスターの `x.509` 証明書を取得して使用する方法については、「ネットワーク切断中のローカルクラスターへの認証」を参照してください。
- ネットワークの切断中に Route 53 にアクセスできない場合は、オンプレミス環境上にあるローカル DNS サーバーを使用することを検討してください。Kubernetes コントロールプレーンインスタンスは静的 IP アドレスを使用します。ローカル DNS サーバーを使用する代わりに、エンドポ

イントのホスト名と IP アドレスを使用してクラスターに接続するホストを設定できます。詳細については、AWS Outposts ユーザーガイドの DNS を参照してください。

- ネットワークの切断中にアプリケーショントラフィックの増加が予想する場合は、クラウドに接続したときにクラスターに予備のコンピューティング容量をプロビジョニングできます。Amazon EC2 インスタンスは AWS Outposts の料金に含まれています。そのため、スペアインスタンスを実行しても AWS の使用コストには影響しません。
- ネットワークが切断されている間、ワークロードの作成、更新、スケーリングオペレーションを有効にするには、アプリケーションのコンテナイメージにローカルネットワークを介してアクセスでき、クラスターに十分な容量がある必要があります。ローカルクラスターはコンテナレジストリをホストしません。Pods がそれらのノードで以前に実行されていた場合、コンテナイメージはノードにキャッシュされます。アプリケーションのコンテナイメージをクラウドの Amazon ECR から通常にプルする場合は、ローカルキャッシュまたはレジストリの実行を検討してください。ローカルキャッシュまたはレジストリは、ネットワーク切断中にワークロードリソースの作成、更新、スケーリングオペレーションが必要になった場合に便利です。
- ローカルクラスターは、Amazon EBS を永続ボリュームのデフォルトストレージクラスとして使用し、Amazon EBS 永続ボリュームのライフサイクルを管理するために Amazon EBS CSI ドライバーを使用しています。ネットワークの切断中は、Amazon EBS によってバックアップされる Pods を作成、更新、またはスケーリングすることはできません。これらのオペレーションにはクラウド内の Amazon EBS API への呼び出しが必要だからです。ステートフルワークロードをローカルクラスターにデプロイしていて、ネットワークの切断中に作成、更新、またはスケーリングオペレーションが必要な場合は、別のストレージメカニズムの使用を検討してください。
- AWS Outposts が、関連する AWS リージョン内 API (Amazon EBS または Amazon S3 の API など) にアクセスできない場合、Amazon EBS スナップショットを作成または削除することはできません。
- ALB (Ingress) と AWS Certificate Manager (ACM) を統合すると、証明書がプッシュされ、AWS Outposts ALB コンピューティングインスタンスのメモリに保存されます。現在の TLS ターミネーションは、AWS リージョン との接続が切断された場合でも引き続き機能します。このコンテキストでの変更操作は失敗します (新しい入力定義、新しい ACM ベースの証明書 API 操作、ALB コンピューティングスケール、証明書のローテーションなど)。詳細については、「AWS Certificate Manager ユーザーガイド」の「マネージド型の証明書の更新のトラブルシューティング」を参照してください。
- Amazon EKS コントロールプレーンのログは、ネットワークの切断中に Kubernetes コントロールプレーンインスタンスでローカルにキャッシュされます。再接続すると、ログは親 AWS リージョン の CloudWatch Logs に送信されます。Prometheus、Grafana、または Amazon EKS パー

トナーソリューションを使用して、Kubernetes API サーバーのメトリクスエンドポイントを使用して、またはログに Fluent Bit を使用してクラスターをローカルで監視できます。

- アプリケーショントラフィックに Outposts で AWS Load Balancer Controller を使用している場合、AWS Load Balancer Controller が前面にある既存の Pods は、ネットワークの切断中もトラフィックを受信し続けます。ネットワーク切断中に作成された新しい Pods は、Outpost が AWS クラウド に再接続されるまでトラフィックを受信しません。ネットワーク切断中のスケーリングのニーズに対応するために、AWS クラウド に接続している間はアプリケーションのレプリカ数を設定することを検討してください。
- Amazon VPC CNI plugin for Kubernetes はセカンダリ IP モードのデフォルト設定に戻ります。`WARM_ENI_TARGET=1` で構成されているため、プラグインは使用可能な IP アドレスの「完全な伸縮性ネットワークインターフェイス」を維持できます。接続されていない状態でのスケーリングのニーズに応じて、`WARM_ENI_TARGET`、`WARM_IP_TARGET`、`MINIMUM_IP_TARGET` の値を変更することを検討してください。詳細については、「GitHub」の「プラグインの `readme` ファイル」を参照してください。各インスタンスタイプによりサポートされる Pods の最大数のリストについては、GitHub の `eni-max-pods.txt` ファイルを参照してください。

ネットワーク切断中のローカルクラスターへの認証

AWS Identity and Access Management (IAM) は、ネットワークの切断中は使用できません。切断されている間は、IAM 認証情報を使用してローカルクラスターを認証することはできません。しかしながら、切断時に `x509` 証明書を使用して、ローカルネットワークを介してクラスターに接続できます。切断時に使用するクライアント `X509` 証明書をダウンロードして保存する必要があります。このトピックでは、証明書を作成して使用し、クラスターが接続されていない状態のときにクラスターを認証する方法を説明します。

1. 証明書署名リクエストを作成します。

   a. 証明書署名リクエストを生成します。

   ```
   openssl req -new -newkey rsa:4096 -nodes -days 365 \
       -keyout admin.key -out admin.csr -subj "/CN=admin"
   ```

   b. Kubernetes で証明書署名リクエストを作成します。

   ```
   BASE64_CSR=$(cat admin.csr | base64 -w 0)
   cat << EOF > admin-csr.yaml
   apiVersion: certificates.k8s.io/v1
   kind: CertificateSigningRequest
   ```

```
metadata:
  name: admin-csr
spec:
  signerName: kubernetes.io/kube-apiserver-client
  request: ${BASE64_CSR}
  usages:
  - client auth
EOF
```

2. kubectl を使用して証明書署名リクエストを作成します。

```
kubectl create -f admin-csr.yaml
```

3. 証明書登録リクエストのステータスを確認します。

```
kubectl get csr admin-csr
```

出力例は次のとおりです。

```
NAME       AGE   REQUESTOR                      CONDITION
admin-csr  11m   kubernetes-admin               Pending
```

Kubernetes が証明書署名リクエストを作成しました。

4. 証明書署名リクエストを承認します。

```
kubectl certificate approve admin-csr
```

5. 証明書署名リクエストの承認のステータスを再確認します。

```
kubectl get csr admin-csr
```

出力例は次のとおりです。

```
NAME       AGE   REQUESTOR                    CONDITION
admin-csr  11m   kubernetes-admin             Approved
```

6. 証明書を取得して検証します。

a. 証明書を取得する。

```
kubectl get csr admin-csr -o jsonpath='{.status.certificate}' | base64 --decode
 > admin.crt
```

b. 証明書を検証する。

```
cat admin.crt
```

7. admin ユーザーのクラスターロールバインディングを作成します。

```
kubectl create clusterrolebinding admin --clusterrole=cluster-admin \
    --user=admin --group=system:masters
```

8. 接続されていない状態のユーザースコープの kubeconfig を生成します。

ダウンロードした admin 証明書を使用して kubeconfig ファイルを生成できます。次のコマンドの *my-cluster* と *apiserver-endpoint* を置き換えます。

```
aws eks describe-cluster --name my-cluster \
    --query "cluster.certificateAuthority" \
    --output text | base64 --decode > ca.crt
```

```
kubectl config --kubeconfig admin.kubeconfig set-cluster my-cluster \
    --certificate-authority=ca.crt --server apiserver-endpoint --embed-certs
```

```
kubectl config --kubeconfig admin.kubeconfig set-credentials admin \
    --client-certificate=admin.crt --client-key=admin.key --embed-certs
```

```
kubectl config --kubeconfig admin.kubeconfig set-context admin@my-cluster \
    --cluster my-cluster --user admin
```

```
kubectl config --kubeconfig admin.kubeconfig use-context admin@my-cluster
```

9. kubeconfig ファイルを表示する。

```
kubectl get nodes --kubeconfig admin.kubeconfig
```

10. Outpost で既にサービスが稼働している場合は、この手順をスキップしてください。Outpost で実行されているサービスが Amazon EKS のみで、その Outpost が現在本番環境ではない場合

は、ネットワークの切断をシミュレートできます。ローカルクラスターで本番環境に入る前に、切断をシミュレートして、接続されていない状態でもクラスターにアクセスできることを確認できます。

a. Outpostを AWS リージョン に接続するネットワークデバイスにファイアウォールルールを適用します。これにより、Outpost のサービスリンクが切断されます。新しいインスタンスを作成することはできません。現在実行中のインスタンスは、AWS リージョン およびインターネットへの接続を失います。

b. x509 証明書を使用して、切断時にローカルクラスターへの接続をテストできます。kubeconfig を前の手順で作成した admin.kubeconfig に必ず変更してください。*my-cluster* の部分は、お客様のローカルクラスター名に置き換えます。

```
kubectl config use-context admin@my-cluster --kubeconfig admin.kubeconfig
```

ローカルクラスターが接続されていない状態で問題が発生した場合は、サポートチケットを開くことをお勧めします。

容量に関する考慮事項

このトピックでは、Kubernetes コントロールプレーンインスタンスタイプを選択し、(オプションで) プレースメントグループを使用して Outpost のローカル Amazon EKS クラスターの高可用性要件を満たすためのガイダンスを提供します。

Outposts でローカルクラスターの Kubernetes コントロールプレーンに使用するインスタンスタイプ (m5、c5、r5 など) を選択する前に、Outpost 設定で使用可能なインスタンスタイプを確認する必要があります。使用可能なインスタンスタイプを特定したら、ワークロードに必要なノード数に基づいてインスタンスサイズ (large、xlarge、2xlarge など) を選択します。次のテーブルは、インスタンスサイズを選択する際のレコメンデーションを示しています。

> **Note**
>
> インスタンスサイズは Outposts にスロットされている必要があります。ローカルクラスターの存続期間中、Outposts で使用可能なサイズの 3 つのインスタンス用に十分な容量を確保してください。使用可能な Amazon EC2 インスタンスタイプのリストについては、「AWS Outposts ラック機能」の「コンピューティングとストレージ」セクションを参照してください。

ノードの数。	Kubernetes コントロールプレーンのインスタンスサイズ
1 ~ 20	large
21 ~ 100	xlarge
101 ~ 250	2xlarge
251 ~ 500	4xlarge

Kubernetes コントロールプレーンのストレージには、etcd の要求される IOPS を満たすために、ローカルクラスターごとに 246 GB の Amazon EBS ストレージが必要です。ローカルクラスターの作成時に、Amazon EBS ボリュームは自動的にプロビジョニングされます。

コントロールプレーンの配置

OutpostConfig.ControlPlanePlacement.GroupName プロパティでプレイスメントグループを指定しない場合、Kubernetes コントロールプレーン用にプロビジョニングされた Amazon EC2 インスタンスは、Outpost で利用可能な基盤となる容量全体にわたって、特定のハードウェア配置の強制を受けません。

プレイスメントグループを使用すると、Outpost のローカル Amazon EKS クラスターの高可用性要件を満たすことができます。クラスターの作成中にプレイスメントグループを指定することで、Kubernetes コントロールプレーンインスタンスの配置に影響を与えます。インスタンスは基盤となる独立したハードウェア (ラックまたはホスト) に分散されるため、ハードウェア障害発生時のインスタンス間の影響が最小限に抑えられます。

要件

設定できるスプレッドのタイプは、デプロイの Outpost ラックの数によって異なります。

- 単一の論理的な Outpost に 1 つまたは 2 つの物理ラックを使用するデプロイ - Kubernetes コントロールプレーンインスタンス用に選択したインスタンスタイプで構成されたホストが少なくとも 3 つ必要です。ホストレベルのスプレッドを使用するスプレッドプレイスメントグループにより、すべての Kubernetes コントロールプレーンインスタンスが、Outpost デプロイで使用可能な基盤となるラック内の個別のホストで実行されるようになります。
- 単一の論理的な Outpost に 3 つ以上の物理ラックを使用するデプロイ - Kubernetes コントロールプレーンインスタンス用に選択したインスタンスタイプで構成されたホストが少なくとも 3 つ必

要です。ラックレベルスプレッドを使用するスプレッドプレイスメントグループにより、すべての Kubernetes コントロールプレーンインスタンスが、Outpost デプロイ内の個別のラックで実行されるようになります。または、前のオプションで説明したように、ホストレベルのスプレッドプレイスメントグループを使用することもできます。

希望するプレイスメントグループの作成はお客様の責任となります。`CreateCluster` API を呼び出すときにプレイスメントグループを指定します。プレイスメントグループとその作成方法の詳細については、「Amazon EC2 ユーザーガイド」の「プレイスメントグループ」を参照してください。

考慮事項

- プレイスメントグループを指定する場合、ローカルの Amazon EKS クラスターを正常に作成するには、Outpost に使用可能なスロット容量が必要です。容量は、ホストタイプとラックスプレッドタイプのどちらを使用するかによって異なります。十分な容量がない場合、クラスターは `Creating` 状態のままになります。`DescribeCluster` API レスポンスのヘルスフィールドで `Insufficient Capacity Error` を確認できます。作成プロセスを進めるには、容量を解放する必要があります。
- Amazon EKS ローカルクラスタープラットフォームとバージョンの更新中に、クラスターの Kubernetes コントロールプレーンインスタンスはローリング更新戦略を使用して新しいインスタンスに置き換えられます。この交換プロセス中に、各コントロールプレーンインスタンスは終了し、それぞれのスロットが解放されます。代わりに新しい更新済みインスタンスがプロビジョニングされます。更新されたインスタンスは、リリースされたスロットに配置される可能性があります。スロットが別の無関係なインスタンスによって消費され、必要なスプレッドトポロジー要件を満たす容量が残っていない場合、クラスターは `Updating` 状態のままになります。`DescribeCluster` API レスポンスのヘルスフィールドでそれぞれの `Insufficient Capacity Error` を確認できます。更新プロセスを進めて以前の高可用性レベルを再設定できるように、容量を解放する必要があります。
- AWS リージョン ごとにアカウントあたり、最大 500 個のプレイスメントグループを作成できます。詳細については、「Amazon EC2 ユーザーガイド」の「一般的なルールと制限」を参照してください。

AWS Outposts での Amazon EKS のローカルクラスターのトラブルシューティング

このトピックでは、ローカルクラスターの使用中に表示される可能性がある一般的なエラーとそのトラブルシューティング方法について説明します。ローカルクラスターはクラウド内の Amazon EKS クラスターと似ていますが、Amazon EKS による管理する方法にはいくつかの違いがあります。

API の動作

ローカルクラスターは Amazon EKS API を通して作成されますが、非同期で実行されます。つまり、Amazon EKS API へのリクエストはローカルクラスターに対してすぐに返されます。ただし、これらのリクエストは成功し、入力検証エラーにより迅速に失敗するか、あるいは失敗して説明入りの検証エラーが発生する可能性があります。この動作は、Kubernetes API のと似ています。

ローカルクラスターは `FAILED` ステータスに移行しません。Amazon EKS は、クラスターの状態をユーザーが要求した望ましい状態と一致させようと継続的に試みます。その結果、根本的な問題が解決されるまで、長期間にわたってローカルクラスターが `CREATING` 状態のままになる可能性があります。

クラスターヘルスフィールドについて説明します

`describe-cluster` Amazon EKS AWS CLI コマンドを使用して、ローカルクラスターの問題は検出できます。ローカルクラスターの問題は、`describe-cluster` コマンドの応答の `cluster.health` フィールドによって明らかになります。このフィールドに含まれるメッセージには、エラーコード、説明メッセージ、および関連するリソース ID が含まれます。この情報は、Amazon EKS API および AWS CLI のみから利用可能です。次の例では、*`my-cluster`* をローカルクラスターの名前に置き換えます。

```
aws eks describe-cluster --name my-cluster --query 'cluster.health'
```

出力例は次のとおりです。

```
{
    "issues": [
        {
            "code": "ConfigurationConflict",
            "message": "The instance type 'm5.large' is not supported in Outpost 'my-
outpost-arn'.",
            "resourceIds": [
                "my-cluster-arn"
```

```
            ]
        }
    ]
}
```

問題が修復できない場合は、ローカルクラスターを削除して新しいクラスターを作成する必要がある場合があります。たとえば、Outpost で利用できないインスタンスタイプでクラスターをプロビジョニングしようとしている場合などです。下表は、一般的なヘルス関連のエラーを示しています。

エラーシナリオ	Code	メッセージ	ResourceIds
指定されたサブネットが見つかりませんでした。	ResourceNotFound	The subnet ID *subnet-id* does not exist	指定されたすべてのサブネット ID
指定されたサブネットが同じ VPC に属していません。	ConfigurationConflict	Subnets specified must belong to the same VPC	指定されたすべてのサブネット ID
指定されたサブネットの一部が、指定された Outpost に属していません。	ConfigurationConflict	Subnet *subnet-id* expected to be in *outpost-arn* , but is in *other-outpost-arn*	問題のあるサブネット ID
指定されたサブネットの中には、どの Outpost にも属していないものがあります。	ConfigurationConflict	Subnet *subnet-id* is not part of any Outpost	問題のあるサブネット ID
指定されたサブネットの中には、コントロールプレーンインスタンス用のエラスティックネットワークインターフェイス	ResourceLimitExceeded	The specified subnet does not have enough free addresses to satisfy the request.	問題のあるサブネット ID

エラーシナリオ	Code	メッセージ	ResourceIds
を作成するだけの十分な空きアドレスがないものがあります。			
指定されたコントロールプレーンインスタンスタイプは、お使いの Outpost でサポートされていません。	ConfigurationConflict	The instance type *type* is not supported in Outpost *outpost-arn*	クラスター ARN
コントロールプレーン Amazon EC2 インスタンスを終了したか、run-instance が成功し、観察された状態が Terminated に変わりました。これは、Outpost が再接続され、Amazon EBS の内部エラーが原因で Amazon EC2 内部ワークフローが失敗した後、一定期間発生する可能性があります。	InternalFailure	EC2 instance state "Terminated" is unexpected	クラスター ARN

エラーシナリオ	Code	メッセージ	ResourceIds
Outpost の容量が不足しています。これは、クラスターの作成中に Outpost が AWS リージョン から切断された場合にも発生する可能性があります。	`ResourceLimitExceeded`	`There is not enough capacity on the Outpost to launch or start the instance.`	クラスター ARN
アカウントがセキュリティグループの制限を超えています。	`ResourceLimitExceeded`	Amazon EC2 API から返されたエラーメッセージ	ターゲット VPC ID
アカウントがエラスティックネットワークインターフェイスの制限を超えています。	`ResourceLimitExceeded`	Amazon EC2 API から返されたエラーメッセージ	ターゲットサブネット ID
AWS Systems Manager を介してコントロールプレーンインスタンスにアクセスできません。解決策については、「AWS Systems Manager を通してコントロールプレーンインスタンスにアクセスできません」を参照してください。	`ClusterUnreachable`	Amazon EKS コントロールプレーンインスタンスには SSM 経由ではアクセスできません。SSM とネットワーク設定を確認し、EKS on Outposts のトラブルシューティングドキュメントを参照してください。	Amazon EC2 インスタンス ID

エラーシナリオ	Code	メッセージ	ResourceIds
マネージドセキュリティグループまたはエラスティックネットワークインターフェイスの詳細を取得中に、エラーが発生しました。	Amazon EC2 クライアントエラーコードに基づきます。	Amazon EC2 API から返されたエラーメッセージ	すべてのマネージドセキュリティグループ ID
セキュリティグループのイングレスルールを承認または取り消す際にエラーが発生しました。これは、クラスターとコントロールプレーンの両方のセキュリティグループに適用されます。	Amazon EC2 クライアントエラーコードに基づきます。	Amazon EC2 API から返されたエラーメッセージ	問題のあるセキュリティグループの ID
コントロールプレーンインスタンスのエラスティックネットワークインターフェイスの削除中にエラーが発生しました。	Amazon EC2 クライアントエラーコードに基づきます。	Amazon EC2 API から返されたエラーメッセージ	問題のあるエラスティックネットワークインターフェイス ID

次の表は、`describe-cluster` 応答のヘルスフィールドに表示される他の AWS のサービス からのエラーを示しています。

Amazon EC2 エラーコード	クラスターヘルス問題コード	説明
`AuthFailure`	`AccessDenied`	このエラーは、さまざまな理由で発生する可能性があります。最も一般的な理由は、サービスにリンクされたロー

Amazon EC2 エラーコード	クラスターヘルス問題コード	説明
		ルポリシーの範囲を狭めるためにサービスが使用するタグが、コントロールプレーンインスタンスから誤って削除された場合に発生するのです。この場合、Amazon EKS はこれらの AWS リソースを管理および監視できなくなります。
UnauthorizedOperation	AccessDenied	このエラーは、さまざまな理由で発生する可能性があります。最も一般的な理由は、サービスにリンクされたロールポリシーの範囲を狭めるためにサービスが使用するタグが、コントロールプレーンインスタンスから誤って削除された場合に発生するのです。この場合、Amazon EKS はこれらの AWS リソースを管理および監視できなくなります。
InvalidSubnetID.NotFound	ResourceNotFound	このエラーは、セキュリティグループのイングレスルールのサブネット ID が見つからない場合に発生します。
InvalidPermission.NotFound	ResourceNotFound	このエラーは、セキュリティグループのイングレスルールの権限が正しくない場合に発生します。

Amazon EC2 エラーコード	クラスターヘルス問題コード	説明
InvalidGroup.NotFound	ResourceNotFound	このエラーは、セキュリティグループのイングレスルールのグループが見つからない場合に発生します。
InvalidNetworkInterfaceID.NotFound	ResourceNotFound	このエラーは、セキュリティグループのイングレスルールのネットワークインターフェイス ID が見つからない場合に発生します。
InsufficientFreeAddressesInSubnet	ResourceLimitExceeded	このエラーは、サブネットリソースのクォータを超えたときに発生します。
InsufficientCapacityOnOutpost	ResourceLimitExceeded	このエラーは、outpost のクォータを超えたときに発生します。
NetworkInterfaceLimitExceeded	ResourceLimitExceeded	このエラーは、エラスティックネットワークインターフェイスのクォータを超えた場合に発生します。
SecurityGroupLimitExceeded	ResourceLimitExceeded	このエラーは、セキュリティグループのクォータを超えたときに発生します。

Amazon EC2 エラーコード	クラスターヘルス問題コード	説明
VcpuLimitExceeded	ResourceLimitExceeded	Amazon EC2 インスタンスを新規アカウントで作成するときに発生します。エラーは次のようなものになります:「You have requested more vCPU capacity than your current vCPU limit of 32 allows for the instance bucket that the specified instance type belongs to. Please visit http://aws.amazon.com/contact-us/ec2-request to request an adjustment to this limit."
InvalidParameterValue	ConfigurationConflict	Amazon EC2 は、指定されたインスタンスタイプが Outpost でサポートされていない場合、このエラーコードを返します。
その他のすべての障害	InternalFailure	なし

クラスターを作成または変更できません

ローカルクラスターには、クラウド内にホストされている Amazon EKS クラスターとは異なるアクセス権限とポリシーが必要です。クラスターの作成に失敗し InvalidPermissions エラーが表示される場合は、使用しているクラスター ロールに AmazonEKSLocalOutpostClusterPolicy マネージドポリシーがアタッチされているかどうかを確認します。その他すべての API 呼び出しには、クラウド内の Amazon EKS クラスターと同じ権限セットが必要です。

クラスターが **CREATING** の状態でスタックしています

ローカルクラスターの作成にかかる時間は、いくつかの要因によって異なります。ネットワーク設定、Outpost の設定、およびクラスターの設定などが要因として考えられます。通常、ローカルクラスターは 15 ~ 20 分以内に作成され、ACTIVE ステータスに変わります。ローカルクラスターが CREATING 状態を維持する場合は、`describe-cluster` を呼び出すと `cluster.health` 出力フィールドに原因に関する情報が表示されます。

次は、最も一般的な問題を示しています。

AWS Systems Manager (Systems Manager) が次の問題に直面している:

- クラスターが Systems Manager のある AWS リージョン からコントロールプレーンインスタンスに接続できません。地域内の踏み台ホストから `aws ssm start-session --target` *`instance-id`* を呼び出すことでこれを検証できます。このコマンドがうまくいかない場合は、Systems Manager がコントロールプレーンインスタンスで実行されているかどうかを確認します。または、別の回避策として、クラスターを削除して再度作成することもできます。
- Systems Manager のコントロールプレーンインスタンスが、インターネットにアクセスできない可能性があります。クラスターの作成時に指定したサブネットに NAT ゲートウェイとインターネットゲートウェイを備えた VPC があるかどうかを確認します。VPC 到達可能性アナライザーを使用して、コントロールプレーンインスタンスがインターネットゲートウェイに到達できることを確認します。詳細については、「VPC Reachability Analyzer の開始方法」を参照してください。
- 指定したロール ARN にポリシーがありません。AWS マネージドポリシー: AmazonEKSLocalOutpostClusterPolicy がロールから削除されかどうかを確認します。これは、AWS CloudFormation スタックの設定が間違っている場合にも発生する可能性があります。

クラスターの作成時に、複数のサブネットの設定と指定が誤っていた場合:

- 指定されているすべてのサブネットが同じ Outpost に関連付けられており、相互に到達できる必要があります。クラスターの作成時に複数のサブネットを指定すると、Amazon EKS はコントロールプレーンインスタンスを複数のサブネットに分散させようとします。
- Amazon EKS マネージド セキュリティグループは、エラスティックネットワークインターフェイスに適用されます。しかしながら、NACL ファイアウォールルールなどの他の設定要素が、エラスティックネットワークインターフェイスのルールと競合する可能性があります。

VPC とサブネット DNS の設定が誤っているか、欠落しています

確認Amazon EKS ローカルクラスター VPC およびサブネットの要件と考慮事項。

ノードをクラスターに結合できません

一般的な原因:

- AMI に関する問題:
  - サポートされていない AMI を使用しています。Amazon EKS 最適化 Amazon Linux AMI Amazon EKS 最適化 Amazon Linux では、v20220620 以降を使用する必要があります。
  - AWS CloudFormation テンプレートを使用してノードを作成した場合、サポートされていない AMI を使用していないことを確認します。
- AWS IAM Authenticator `ConfigMap` が見つからない - 見つからない場合は、作成する必要があります。詳細については、「`aws-auth ConfigMap` をクラスターに適用する」を参照してください。
- 間違ったセキュリティグループが使用されている - ワーカーノードのセキュリティグループには、必ず `eks-cluster-sg-`*`cluster-name`*`-`*`uniqueid`* を使用してください。スタックが使用されるたびに、選択したセキュリティグループは AWS CloudFormation によって変更され、新しいセキュリティグループを使用できるようになります。
- 予期しないプライベートリンクの VPC 手順に従う - CA データが間違っている (`--b64-cluster-ca`) または API エンドポイント (`--apiserver-endpoint`) が渡されました。
- Pod のセキュリティポリシーの設定ミス:
  - ノードがクラスターに結合して通信できるようにするために、CoreDNS と Amazon VPC CNI plugin for Kubernetes Daemonset はノード上で実行する必要があります。
  - Amazon VPC CNI plugin for Kubernetes を正常に動作するには、いくつかの特権ネットワーク機能が必要です。次のコマンドを使用して特権ネットワーク機能を表示できます: `kubectl describe psp eks.privileged`。

  デフォルトのポッドセキュリティポリシーを変更することはお勧めしません。詳細については、「ポッドのセキュリティポリシー」を参照してください。

ログの収集

Outpost が関連付けられている AWS リージョン から切断された場合でも、Kubernetes クラスターは依然として動作し続ける可能性があります。ただし、クラスターが正常に機能しない場合は、ネットワーク切断の準備 にあるトラブルシューティング手順に従ってください。他の問題が発生した場合は、AWS Support にお問い合わせください。AWS Support からログ収集ツールをダウンロードして実行する方法を説明します。この方法により、Kubernetes クラスターコントロールプレーンインスタンスからログを収集し、詳細な調査のために AWS Support サポートに送信できます。

AWS Systems Manager を通してコントロールプレーンインスタンスにアクセスできません

Amazon EKS コントロールプレーンインスタンスが AWS Systems Manager (Systems Manager) を通してアクセスでない場合、Amazon EKS はクラスターに対して次のエラーを表示します。

```
Amazon EKS control plane instances are not reachable through SSM. Please verify your
 SSM and network configuration, and reference the EKS on Outposts troubleshooting
 documentation.
```

この問題を解決するには、VPC とサブネットが Amazon EKS ローカルクラスター VPC およびサブネットの要件と考慮事項 の要件を満たしており、AWS Systems Manager ユーザーガイドの「Session Manager の設定」にある手順が完了していることを確認してください。

Outpost 上のセルフマネージド型の Amazon Linux ノードの起動

このトピックは、Amazon EKS クラスターに登録されている Outpost 上の Amazon Linux ノードの Auto Scaling グループを起動する方法を説明します。クラスターは、AWS クラウド または Outpost に置くことができます。

前提条件

- 既存の Outpost。詳細については、「AWS Outposts とは」を参照してください。
- 既存の Amazon EKS クラスター。AWS クラウド にクラスターをデプロイするには、「Amazon EKS クラスターの作成」を参照してください。Outpost にクラスターをデプロイするには、「AWS Outposts の Amazon EKS のローカルクラスター」を参照してください。
- AWS クラウド のクラスターにノードを作成しており、AWS Outposts、AWS Wavelength、または AWS Local Zones が有効になっている AWS リージョン にサブネットがあるとします。この場合、クラスターを作成したときに、これらのサブネットが渡されていない必要があります。Outpost 上のクラスターにノードを作成する場合は、クラスターの作成時に Outpost サブネットを渡しておく必要があります。
- (AWS クラウド 上のクラスターに推奨) 必要な IAM ポリシーがアタッチされた独自の IAM ロールで設定された Amazon VPC CNI plugin for Kubernetes アドオン。詳細については、「サービスアカウントの IAM ロールを使用する Amazon VPC CNI plugin for Kubernetes の設定」を参照してください。ローカルクラスターは、サービスアカウントの IAM ロールをサポートしていません。

`eksctl` または AWS Management Console (AWS CloudFormation テンプレートを使用) で、セルフマネージド型の Amazon Linux ノードグループを作成できます。Terraform を使用することもできます。

eksctl

前提条件

デバイスまたは AWS CloudShell にインストールされている `eksctl` コマンドラインツールのバージョン `0.183.0` 以降。`eksctl` をインストールまたはアップグレードするには、`eksctl` ドキュメントの「インストール」を参照してください。

`eksctl` を使用してセルフマネージド型の Linux ノードを起動する

1. クラスターが AWS クラウド にあり、[AmazonEKS_CNI_Policy] マネージド IAM ポリシーが Amazon EKS ノードの IAM ロール へアタッチされている場合は、代わりに Kubernetes `aws-node` サービスアカウントに関連付けた IAM ロールに割り当てることをお勧めします。詳細については、「サービスアカウントの IAM ロールを使用する Amazon VPC CNI plugin for Kubernetes の設定」を参照してください。クラスターが Outpost にある場合は、ポリシーをノードロールにアタッチする必要があります。
2. 次のコマンドは、既存のクラスターにノードグループを作成します。クラスターは、`eksctl` を使用して作成されている必要があります。*`al-nodes`* をノードグループの名前に置き換えます。ノードグループ名は 63 文字以下である必要があります。先頭は文字または数字でなければなりませんが、残りの文字にはハイフンおよびアンダースコアを含めることもできます。*`my-cluster`* を自分のクラスター名に置き換えます。この名前には、英数字 (大文字と小文字が区別されます) とハイフンのみを使用できます。先頭の文字は英数字である必要があります。また、100 文字より長くすることはできません。名前は、クラスターを作成する AWS リージョン および AWS アカウント 内で一意である必要があります。クラスターが Outpost に存在する場合は、*`id`* を Outpost サブネットの ID に置き換えます。AWS クラウド にクラスターが存在する場合、*`id`* をクラスターの作成時に指定しなかったサブネットの ID に置き換えます。*`instance-type`* を、Outpost でサポートされているインスタンスタイプに置き換えます。残りの *`example values`* を独自の値に置き換えてください。デフォルトでは、ノードはコントロールプレーンと同じ Kubernetes バージョンで作成されます。

   `instance-type` を、Outpost で使用可能なインスタンスタイプに置き換えます。

   `my-key` を Amazon EC2 キーペアまたはパブリックキーの名前に置き換えます。このキーは、起動後のノードに SSH 接続するために使用されます。Amazon EC2 キーペアをまだ持っていない場合は、AWS Management Console で作成できます。詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EC2 キーペア」を参照してください。

   次のコマンドを使用して、ノードグループを作成します。

```
eksctl create nodegroup --cluster my-cluster --name al-nodes --node-
type instance-type \
    --nodes 3 --nodes-min 1 --nodes-max 4 --managed=false --node-volume-type gp2
 --subnet-ids subnet-id
```

クラスターを AWS クラウド 上にデプロイしている場合:

- デプロイするノードグループには、インスタンスのブロックとは異なる CIDR ブロックから IPv4 アドレスを Pods に割り当てることができます。詳細については、「ポッド用のカスタムネットワーク」を参照してください。
- デプロイするノードグループは、アウトバウンドインターネットアクセスを必要としません。詳細については、「プライベートクラスターの要件」を参照してください。

利用できるすべてのオプションとデフォルトの詳細なリストについては、`eksctl` ドキュメントの「AWS Outposts サポート」を参照してください。

ノードがクラスターに参加できない場合は、Amazon EKS のトラブルシューティング の ノードをクラスターに結合できません と AWS Outposts での Amazon EKS のローカルクラスターのトラブルシューティング の ノードをクラスターに結合できません を参照してください。

出力例は次のとおりです。ノードの作成中に、複数の行が出力されます。出力の最後の行は、次のサンプル行が表示されます。

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. (オプション) サンプルアプリケーション をデプロイして、クラスターと Linux ノードをテストします。

AWS Management Console

ステップ 1: AWS Management Console を使用してセルフマネージド型の Amazon Linux ノードを起動します

1. AWS CloudFormation テンプレートの最新バージョンをダウンロードする

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2022-12-23/
amazon-eks-nodegroup.yaml
```

2. https://console.aws.amazon.com/cloudformation で AWS CloudFormation コンソール を開きます。
3. [スタックの作成] を選択し、[新しいリソースを使用 (標準)] を選択します。
4. [テンプレートの指定] で、[テンプレートファイルのアップロード] を選択し、[ファイルを選択] を選択します。前の手順でダウンロードした `amazon-eks-nodegroup.yaml` ファイルを選択し、[次へ] を選択します。
5. [スタックの詳細の指定] ページで、必要に応じて次のパラメータを入力し、[次へ] を選択します。

   - [スタック名]: AWS CloudFormation スタックのスタック名を選択します。例えば、***al-nodes*** という名前にすることができます。この名前には、英数字 (大文字と小文字が区別されます) とハイフンのみを使用できます。先頭の文字は英数字である必要があります。また、100 文字より長くすることはできません。名前は、クラスターを作成する AWS リージョン および AWS アカウント 内で一意である必要があります。
   - [ClusterName]: クラスターの名前を入力します。この名前が、クラスター名と一致しない場合、ノードはクラスターに参加できません。
   - [ClusterControlPlaneSecurityGroup]: VPC の作成時に生成した AWS CloudFormation 出力の [SecurityGroups] 値を選択します。

     次のステップでは、該当するグループを取得する 1 つのオペレーションを説明します。

     1. Amazon EKS コンソール (https://console.aws.amazon.com/eks/home#/clusters) を開きます。
     2. クラスターの名前を選択します。
     3. [ネットワーキング] タブを選択します。
     4. [ClusterControlPlaneSecurityGroup] ドロップダウンリストから選択する場合は、[追加のセキュリティグループ] の値をリファレンスとして使用します。
   - [NodeGroupName]: ノードグループの名前を入力します。この名前は、ノードに対して作成される Auto Scaling ノードグループを識別するために後で使用できます。
   - [NodeAutoScalingGroupMinSize]: ノードの Auto Scaling グループがスケールインできる最小ノード数を入力します。

- NodeAutoScalingGroupDesiredCapacity: スタック作成時にスケーリングする必要のあるノード数を入力します。
- [NodeAutoScalingGroupMaxSize]: ノードの Auto Scaling グループがスケールアウトできる最大ノード数を入力します。
- [NodeInstanceType]: ノードのインスタンスタイプを選択します。クラスターが AWS クラウド で動作している場合は、詳細については、「Amazon EC2 インスタンスタイプを選択する」を参照してください。クラスターが Outpost で実行されている場合、Outpost で使用できるインスタンスタイプのみを選択できます。
- [NodeImageIdSSMParam]: 最新の Amazon EKS 最適化 AMI の Amazon EC2 Systems Manager のパラメータが、可変 Kubernetes バージョン用に事前設定されています。Amazon EKS でサポートされている別の Kubernetes マイナーバージョンを使用するには、*1.XX* を別のサポートされているバージョンに置き換えます。クラスターと同じ Kubernetes バージョンを指定することをお勧めします。

  Amazon EKS 最適化高速 AMI を使用するには、*amazon-linux-2* を **amazon-linux-2-gpu** に置き換えます。Amazon EKS 最適化 Arm AMI を使用するには、*amazon-linux-2* を **amazon-linux-2-arm64** に置き換えます。

  > **Note**
  >
  > Amazon EKS ノード AMI は Amazon Linux をベースとしています。Amazon Linux セキュリティセンターで Amazon Linux 2 のセキュリティもしくはプライバシーに関するイベントを追跡したり、関連する RSS フィードをサブスクライブできます。セキュリティおよびプライバシーイベントには、問題の概要、影響を受けるパッケージ、および問題を修正するためにインスタンスを更新する方法などがあります。

- [NodeImageId]: (オプション) (Amazon EKS 最適化 AMI の代わりに) 独自のカスタム AMI を使用している場合は、AWS リージョン のノード AMI ID を入力します。ここで値を指定すると、[NodeImageIdSSMParam] フィールドの値はすべて上書きされます。
- [NodeVolumeSize]: ノードのルートボリュームのサイズを GiB 単位で指定します。
- [NodeVolumeType]: ノードのルートボリュームタイプを指定します。
- [KeyName]: 起動後に、SSH を使用してノードに接続するときに使用できる Amazon EC2 SSH キーペアの名前を入力します。Amazon EC2 キーペアをまだ持っていない場合は、AWS Management Console で作成できます。詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EC2 キーペア」を参照してください。

> **Note**
>
> ここでキーペアを指定しないと、AWS CloudFormation スタックの作成は失敗します。

- [BootstrapArguments]: ノードに渡すことができるオプションの引数がいくつかあります。詳細については、「GitHub」の「ブートストラップスクリプトの使用状況」を参照してください。AWS Outposts の Amazon EKS ローカルクラスター (Kubernetes コントロールプレーンインスタンスが AWS Outposts で稼働) と入出力のインターネット接続がないクラスター (プライベートクラスターとも呼ばれる) にノードを追加する場合は、次のブートストラップ引数を (1 行で) 指定する必要があります。

```
--b64-cluster-ca ${CLUSTER_CA} --apiserver-endpoint https://
${APISERVER_ENDPOINT} --enable-local-outpost true --cluster-id ${CLUSTER_ID}
```

- [DisableIMDSv1]: 各ノードは、デフォルトでインスタンスメタデータサービスバージョン 1 (IMDSv1) および IMDSv2 をサポートします。IMDSv1 は無効にできます。ノードグループ内の将来のノードおよび Pods が MDSv1 を使用しないようにするには、[DisableIMDsv1] を [true] に設定します。IDMS の詳細については、「インスタンスメタデータサービスの設定」を参照してください。ノードでのそれへのアクセス制限について詳しくは、ワーカーノードに割り当てられたインスタンスプロファイルへのアクセスを制限するを参照してください。
- [VpcId]: 作成した VPC の ID を入力します。VPC を選択する前に、VPC の要件と考慮事項を確認します。
- [サブネット]: クラスターが Outpost にある場合、VPC 内で少なくとも 1 つのプライベートサブネットを選択します。サブネットを選択する前に、「サブネットの要件と考慮事項」を確認してください。クラスターの [ネットワーキング] タブから、各サブネットリンクを開き、プライベートのサブネットを確認できます。

6. [スタックオプションの設定] ページで、希望する設定を選択し、[次へ] を選択します。
7. [AWS CloudFormation が IAM リソースを作成する可能性を認識しています] の左にあるチェックボックスを選択して、[スタックの作成] を選択します。
8. スタックの作成が完了したら、コンソールで選択し、[出力] を選択します。
9. 作成されたノードグループの [NodeInstanceRole] を記録します。これは、Amazon EKS ノードを設定する際、必要になります。

ステップ 2: ノードを有効にしてクラスターに参加させるには

1. aws-auth ConfigMap がすでにあるかどうかを確認します。

```
kubectl describe configmap -n kube-system aws-auth
```

2. aws-auth ConfigMap が表示されている場合は、必要に応じて更新してください。

   a. 編集する ConfigMap を開きます。

```
kubectl edit -n kube-system configmap/aws-auth
```

   b. 必要に応じて新しい mapRoles エントリを追加します。rolearn 値を、前の手順で記録した [NodeInstanceRole] 値に設定します。

```
[...]
data:
  mapRoles: |
    - rolearn: <ARN of instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
[...]
```

   c. ファイルを保存し、テキストエディタを終了します。

3. 「Error from server (NotFound): configmaps "aws-auth" not found」というエラーが表示されたら、ストック ConfigMap を適用してください。

   a. 設定マップをダウンロードします。

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-
eks/cloudformation/2020-10-29/aws-auth-cm.yaml
```

   b. aws-auth-cm.yaml ファイルで、rolearn を前の手順で記録した [NodeInstanceRole] 値に設定します。これを行うには、テキストエディタを使用するか、*my-node-instance-role* を置き換えて次のコマンドを実行します。

```
sed -i.bak -e 's|<ARN of instance role (not instance profile)>|my-node-
instance-role|' aws-auth-cm.yaml
```

c. 設定を適用します。このコマンドが完了するまで数分かかることがあります。

```
kubectl apply -f aws-auth-cm.yaml
```

4. ノードのステータスを監視し、Ready ステータスになるまで待機します。

```
kubectl get nodes --watch
```

Ctrl+C を入力して、シェルプロンプトに戻ります。

> **Note**
>
> 認証またはリソースタイプのエラーが発生した場合は、トラブルシューティングトピックの「許可されていないか、アクセスが拒否されました (kubectl)」を参照してください。

ノードがクラスターに参加できない場合は、Amazon EKS のトラブルシューティング の ノードをクラスターに結合できません と AWS Outposts での Amazon EKS のローカルクラスターのトラブルシューティング の ノードをクラスターに結合できません を参照してください。

5. Amazon EBS CSI ドライバーをインストールします。詳細については、GitHub の Installation を参照してください。[ドライバーのアクセス許可を設定] セクションでは、[IAM インスタンスプロファイルの使用] オプションの指示に従うことを確認します。gp2 ストレージクラスを使用する必要があります。gp3 ストレージクラスは、サポートされていません。

クラスターの gp2 ストレージクラスを作成するには、以下のステップを実行します。

1. 次のコマンドを実行して、gp2-storage-class.yaml ファイルを作成します。

```
cat >gp2-storage-class.yaml <<EOF
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
  name: ebs-sc
provisioner: ebs.csi.aws.com
```

```
volumeBindingMode: WaitForFirstConsumer
parameters:
  type: gp2
  encrypted: "true"
allowVolumeExpansion: true
EOF
```

2. マニフェストをクラスターに適用します。

```
kubectl apply -f gp2-storage-class.yaml
```

6. (GPU ノードのみ) GPU インスタンスタイプと Amazon EKS 最適化アクセラレーション AMI を選択した場合は、クラスター上の DaemonSet として Kubernetes 用の NVIDIA デバイスプラグインを適用する必要があります。次のコマンドを実行する前に、*vX.X.X* を必要となる NVIDIA/k8s-device-plugin バージョンに置き換えます。

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-
plugin/vX.X.X/nvidia-device-plugin.yml
```

ステップ 3: その他のアクション

1. (オプション) サンプルアプリケーションをデプロイして、クラスターと Linux ノードをテストします。
2. クラスターが Outpost にデプロイされている場合は、このステップをスキップしてください。クラスターが AWS クラウド にデプロイされている場合、次の情報はオプションです。[AmazonEKS_CNI_Policy] マネージド IAM ポリシーが Amazon EKS ノードの IAM ロール にアタッチされている場合は、代わりに Kubernetes aws-node サービスアカウントに関連付けた IAM ロールに割り当てることをお勧めします。詳細については、「サービスアカウントの IAM ロールを使用する Amazon VPC CNI plugin for Kubernetes の設定」を参照してください。

関連プロジェクト

これらのオープンソース プロジェクトは、Amazon EKS によって管理されるクラスターを含む、AWS の内外で実行される Kubernetes クラスターの機能を拡張します。

管理ツール

Amazon EKS および Kubernetes クラスターの関連する管理ツール。

eksctl

`eksctl` は、Amazon EKS 上にクラスターを作成するためのシンプルな CLI ツールです。

- プロジェクト URL
- プロジェクトドキュメント
- AWS オープンソースブログ:「eksctl: 1 つのコマンドによる Amazon EKS クラスター」

Kubernetes の AWS コントローラ

Kubernetes 用 AWS コントローラーを使用すると、Kubernetes クラスターから直接 AWS リソースを作成および管理できます。

- プロジェクト URL
- AWS オープン ソース ブログ:「Kubernetes 向けの AWS サービス オペレーターが利用可能になりました」

Flux CD

Flux は Git を使用して、クラスター設定の管理に使用できるツールです。クラスター内のオペレータを使用して、Kubernetes 内のデプロイをトリガーします。オペレーターの詳細については、GitHub の「OperatorHub.io」を参照してください。

- プロジェクト URL
- プロジェクトドキュメント

Kubernetes 用の CDK

Kubernetes 用の CDK (cdk8s) を使用すると、使い慣れたプログラミング言語を使用して Kubernetes アプリとコンポーネントを定義できます。cdk8s アプリは標準の Kubernetes マニフェストに合成され、Kubernetes クラスターにも適用できます。

- プロジェクトURL
- プロジェクトドキュメント
- AWS コンテナのブログ:「cdk8s+ の紹介: Kubernetes オブジェクト用のインテント駆動型 API」

ネットワーク

Amazon EKS および Kubernetes クラスターに関連するネットワークプロジェクト。

Amazon VPC CNI plugin for Kubernetes

Amazon EKS は、Amazon VPC CNI plugin for Kubernetes を介して従来の VPC ネットワークをサポートしています。プラグインは、VPC から各 Pod に IP アドレスを割り当てます。

- プロジェクト URL
- プロジェクトドキュメント

Kubernetes 用の AWS Load Balancer Controller

AWS Load Balancer Controller は Kubernetes クラスター向けの AWS Elastic Load Balancer の管理に役立ちます。AWS Application Load Balancer をプロビジョニングすることで、Kubernetes Ingress リソースの条件を満たします。AWS Network Load Balancer をプロビジョニングすることで、Kubernetes サービスリソースの条件を満たします。

- プロジェクト URL
- プロジェクトドキュメント

ExternalDNS

ExternalDNS は、公開された Kubernetes サービスとそのイングレスを、Amazon Route 53 や AWS Service Discovery を含む DNS プロバイダーと同期させます。

- プロジェクト URL
- プロジェクトドキュメント

Machine Learning

Amazon EKS および Kubernetes クラスターに関連する機械学習プロジェクト。

Kubeflow

Kubernetes 用の機械学習ツールキット。

- プロジェクト URL
- プロジェクトドキュメント
- AWS オープンソースブログ: 「Amazon EKS での Kubeflow」

Auto Scaling

Amazon EKS および Kubernetes クラスターに関連する自動スケーリングプロジェクト。

Cluster Autoscaler

Cluster Autoscaler は、CPU とメモリーの負荷に基づいて Kubernetes クラスターのサイズを自動的に調整するツールです。

- プロジェクト URL
- プロジェクトドキュメント
- Amazon EKS ワークショップ: https://www.eksworkshop.com/

Escalator

Escalator は、Kubernetes 用のバッチまたはジョブ最適化水平オートスケーラーです。

- プロジェクト URL
- プロジェクトドキュメント

モニタリング

Amazon EKS および Kubernetes クラスターに関連するモニタリングプロジェクト。

Prometheus

Prometheus はオープンソースのシステムモニタリングおよび警告ツールキットです。

- プロジェクト URL
- プロジェクトドキュメント
- Amazon EKS ワークショップ: https://eksworkshop.com/intermediate/240_monitoring/

継続的インテグレーション/継続的デプロイメント

Amazon EKS および Kubernetes クラスターに関連する CI/CD プロジェクト。

Jenkins X

Amazon EKS および Kubernetes クラスター上の最新のクラウドアプリケーション用 CI/CD ソリューション。

- プロジェクト URL
- プロジェクトドキュメント

Amazon EKS 新機能とロードマップ

Amazon EKS の新機能については、AWS の最新情報ページの新機能フィードをスクロールし、確認してください。GitHub で ロードマップ を確認することもできます。これにより、今後の機能と優先事項について知ることができるため、Amazon EKS を将来どのように使用するかを計画できます。ロードマップの優先度について、直接フィードバックをお寄せください。

Amazon EKS のドキュメント履歴

以下の表は、Amazon EKS ユーザーガイドの主な更新や新機能を示しています。また、お客様からいただいたフィードバックに対応するために、ドキュメントを頻繁に更新しています。

変更	説明	日付
Kubernetes バージョン 1.30	Kubernetes バージョン 1.30 の新しいクラスターとバージョンアップグレードのサポートを追加。	2024 年 5 月 23 日
Amazon EKS プラットフォームのバージョンを更新しました	これは、セキュリティの修正および機能強化が行われた新しいプラットフォームバージョンです。これには、Kubernetes 1.29.4、1.28.9、1.27.13 の新しいパッチバージョンが含まれます。	2024 年 5 月 14 日
CoreDNS 自動スケーリング	CoreDNS オートスケーラーは、ノードと CPU コアの数に基づいて、EKS クラスター内の CoreDNS デプロイのレプリカの数を動的に調整します。この機能は、CoreDNS v1.9 および EKS リリースバージョン 1.25 以降の最新プラットフォームバージョンで機能します。	2024 年 5 月 14 日
CloudWatch Container Insights が Windows をサポート	Amazon CloudWatch Observability Operator アドオンを使って、クラスター内の Windows ワーカーノードで	2024 年 4 月 10 日

	Container Insights を使用できるようになりました。	
Kubernetes コンセプト	Kubernetes コンセプトのトピックが新たに追加されました。	2024 年 4 月 5 日
アクセスと IAM コンテンツを再構築する	認証設定マップ、アクセスエントリ、ポッド ID、IRSA など、アクセスと IAM トピックに関連する既存のページが新しいセクションに移動します。概要の内容を改訂しました。	2024 年 4 月 2 日
Amazon S3 CSI ドライバーの Bottlerocket OS サポート	Mountpoint for Amazon S3 CSI ドライバーが Bottlerocket と互換性を持つようになりました。	2024 年 3 月 13 日
AWS 管理ポリシーの更新 – 既存のポリシーへの更新	Amazon EKS において、既存の AWS 管理ポリシーの更新を行いました。	2024 年 3 月 4 日
Amazon Linux 2023	Amazon Linux 2023 (AL2023) は、クラウドアプリケーションに安全かつ安定した高パフォーマンス環境を提供するように設計された、新しい Linux ベースのオペレーティングシステムです。	2024 年 2 月 29 日

Kubernetes1.29 で EKS Pod Identity と IRSA がサイドカーをサポート	Kubernetes 1.29 では、サイドカーコンテナは Amazon EKS クラスターで使用できます。サイドカーコンテナは、サービスアカウントまたは EKS Pod Identity の IAM ロールでサポートされています。サイドカーの詳細については、Kubernetes ドキュメントの「サイドカーコンテナ」を参照してください。	2024 年 2 月 26 日
Kubernetes バージョン 1.29	Kubernetes バージョン 1.29 の新しいクラスターとバージョンアップグレードのサポートを追加。	2024 年 1 月 23 日
フルリリース: Kubernetes バージョンに対する Amazon EKS の延長サポート	拡張 Kubernetes バージョンサポートにより、特定の Kubernetes バージョンを 14 か月よりも長く使用することができます。	2024 年 1 月 16 日

AWS クラウドでの Amazon EKS クラスターヘルスの検出	Amazon EKS はクラスターヘルス内の、Amazon EKS クラスターとクラスターの前提条件のインフラストラクチャの問題を検出します。EKS API では、AWS Management Console 内およびクラスターの `health` 内の EKS クラスターの問題を確認できます。これらの問題は、コンソールによって検出され表示される問題に追加されます。以前は、クラスターヘルスは AWS Outposts 上のローカルクラスターのみで利用可能でした。	2023 年 12 月 28 日
Amazon EKS AWS リージョン 拡張	Amazon EKS がカナダ西部 (カルガリー) (`ca-west-1`) AWS リージョン で使用可能になりました。	2023 年 12 月 20 日
クラスターのインサイト	定期的なチェックに基づいて、クラスターに関する推奨事項を取得できるようになりました。	2023 年 12 月 20 日

IAM ロールとユーザーに、アクセスエントリを使用してクラスターへのアクセス権を付与できるようになりました。	アクセスエントリを導入する前は、aws-auth ConfigMap にエントリを追加して IAM ロールとユーザーにクラスターへのアクセス権を付与していました。これで、各クラスターにアクセスモードが用意され、スケジュールに従ってアクセスエントリを使用するように切り替えることができるようになりました。モードを切り替えたら、AWS CLI、AWS CloudFormation、AWS SDK にアクセスエントリを追加してユーザーを追加できます。	2023 年 12 月 18 日
Amazon EKS プラットフォームのバージョンの更新	これは、セキュリティの修正および機能強化が行われた新しいプラットフォームバージョンです。これには、Kubernetes 1.28.4、1.27.8、1.26.11、1.25.16 の新しいパッチバージョンが含まれます。	2023 年 12 月 12 日
Mountpoint for Amazon S3 CSI ドライバー	Amazon EKS クラスターで Mountpoint for Amazon S3 CSI ドライバーをインストールできるようになりました。	2023 年 11 月 27 日

クラスターの作成時に Prometheus メトリクスを有効にする	AWS Management Console では、クラスターの作成時に Prometheus メトリクスをオンにできるようになりました。Prometheus スクレイパーの詳細は [オブザーバビリティ] タブでも表示できます。	2023 年 11 月 26 日
Amazon EKS Pod Identity	Amazon EKS Pod Identity は IAM ロールを Kubernetes サービスアカウントに関連付けます。この機能を使用すると、ノード IAM ロールに拡張されたアクセス許可を提供する必要がなくなります。この方法で、そのノード上の Pods は AWS API を呼び出すことができます。サービスアカウントの IAM ロールとは異なり、EKS ポッド ID は完全に EKS 内部にあるため、OIDC ID プロバイダーは必要ありません。	2023 年 11 月 26 日
AWS 管理ポリシーの更新 – 既存のポリシーへの更新	Amazon EKS において、既存の AWS 管理ポリシーの更新を行いました。	2023 年 11 月 26 日
CSI スナップショットコントローラー	CSI スナップショットコントローラーをインストールして、Amazon EBS CSI ドライバーなどの互換性のある CSI ドライバーで使用できるようになりました。	2023 年 11 月 17 日

ADOT オペレータートピックリライト	ADOT Operator セクションの Amazon EKS アドオンサポートは、OpenTelemetry ドキュメント用の AWS Distro と重複していました。古い情報や一貫性のない情報を減らすために、残りの重要な情報をそのリソースに移行しました。	2023 年 11 月 14 日
Prometheus メトリクスの CoreDNS EKS アドオンサポート	CoreDNS 用の EKS アドオンの `v1.10.1-eksbuild.5` 、`v1.9.3-eksbuild.9` 、`v1.8.7-eksbuild.8` バージョンでは、メトリクスを CoreDNS 公開したポートが `kube-dns` サービス内で公開されます。これにより、モニタリングシステムに CoreDNS メトリクスを組み込むことが容易になります。	2023 年 11 月 10 日
Amazon EKS CloudWatch オブザーバビリティオペレーターアドオン	Amazon EKS CloudWatch オブザーバビリティオペレーターページを追加しました。	2023 年 11 月 6 日
米国東部 (オハイオ) のセルフマネージド型 P5 インスタンスのキャパシティブロック	米国東部 (オハイオ) では、米国東部 (オハイオ) のセルフマネージド型 P5 インスタンスにキャパシティブロックを使用できるようになりました。	2023 年 10 月 31 日

クラスターでのサブネットおよびセキュリティグループの変更のサポート	クラスターを更新して、クラスターが使用するサブネットとセキュリティグループを変更できます。AWS Management Console、AWS CLI の最新バージョン、AWS CloudFormation、および `eksctl` のバージョン `v0.164.0-rc.0` 以降から更新できます。クラスターバージョンを正常にアップグレードするには、これを実行して、サブネットに利用可能な IP アドレスを増やす必要がある場合があります。	2023 年 10 月 24 日
クラスターロールおよびマネージドノードグループロールでのカスタマーマネージド AWS Identity and Access Management ポリシーのサポート	クラスターロールでは、`AmazonEKSClusterPolicy` AWS マネージドポリシーの代わりにカスタム IAM ポリシーを使用できます。また、マネージドノードグループのノードロールでは、`AmazonEKSWorkerNodePolicy` AWS マネージドポリシーの代わりにカスタム IAM ポリシーを使用できます。これにより、厳格なコンプライアンス要件を満たす最小特権でポリシーを作成できます。	2023 年 10 月 23 日
eksctl インストールへのリンクの修正	ページが移動された後の eksctl のインストールリンクを修正しました。	2023 年 10 月 6 日

プレビューリリース: Kubernetes バージョンに対する Amazon EKS の延長サポート	拡張 Kubernetes バージョンサポートにより、特定の Kubernetes バージョンを 14 か月よりも長く使用することができます。	2023 年 10 月 4 日
AWS App Mesh インテグレーションへのリファレンスを削除する	Amazon EKS と AWS App Mesh の統合は、引き続き App Mesh の既存のお客様のみを対象としています。	2023 年 9 月 29 日
Kubernetes バージョン 1.28	Kubernetes バージョン 1.28 の新しいクラスターとバージョンアップグレードのサポートを追加。	2023 年 9 月 26 日
既存のクラスターは、Amazon VPC CNI plugin for Kubernetes で Kubernetes ネットワークポリシーの適用をサポートしています	サードパーティのソリューションの代わりに、既存のクラスターにある Kubernetes のネットワークポリシーと Amazon VPC CNI plugin for Kubernetes を使用できます。	2023 年 9 月 15 日
CoreDNS Amazon EKS アドオンは PDB の変更をサポートします	バージョン v1.9.3-eksbuild.7 以降および v1.10.1-eksbuild.4 以降では、CoreDNS の EKS アドオンの PodDisruptionBudget を変更できます。	2023 年 9 月 15 日
Amazon EKS での共有サブネットのサポート	共有サブネット内に Amazon EKS クラスターを作成するための、新しい「共有サブネットの要件と考慮事項」。	2023 年 9 月 7 日

「Amazon EKS とは」の更新	新規トピック「一般的なユースケース」と「アーキテクチャ」の追加。その他に複数のトピックの更新。	2023 年 9 月 6 日
Amazon VPC CNI plugin for Kubernetes での Kubernetes ネットワークポリシーの適用	サードパーティのソリューションの代わりに、Kubernetes ネットワークポリシーと Amazon VPC CNI plugin for Kubernetes を使用できます。	2023 年 8 月 29 日
Amazon EKS AWS リージョン の拡張	Amazon EKS がイスラエル (テルアビブ) でご利用いただけるようになりました (`il-central-1`) AWS リージョン。	2023 年 8 月 1 日
設定可能な Fargate 用エフェメラルストレージ	Amazon EKS Fargate で実行される各 Pod 用エフェメラルストレージの総量を増やすことができます。	2023 年 7 月 31 日
Amazon EFS CSI ドライバーのアドオンサポート	AWS Management Console、AWS CLI、API を使用して Amazon EFS CSI ドライバーを管理できるようになりました。	2023 年 7 月 26 日
AWS 管理ポリシーの更新 – 新しいポリシー	Amazon EKS に、新しい AWS 管理ポリシーが追加されました。	2023 年 7 月 26 日

1.27、1.26、1.25、1.24 向けの Kubernetes バージョンアップデートが、AWS Outposts のローカルクラスターで利用できるようになりました。	1.27.3、1.26.6、1.25.11、1.24.15 向けの Kubernetes バージョンアップデートが、AWS Outposts のローカルクラスターで利用できるようになりました。	2023 年 7 月 20 日
Windows ノードの IP プレフィックスサポート	ノードに IP プレフィックスを割り当てると、個別のセカンダリ IP アドレスをノードに割り当てる場合よりもはるかに多くの数の Pods をノードでホストできます。	2023 年 7 月 6 日
Amazon FSx for OpenZFS CSI ドライバー	Amazon EKS クラスターで Amazon FSx for OpenZFS CSI ドライバーをインストールできるようになりました。	2023 年 6 月 30 日
IPv4 クラスター内の Linux ノード上の Pods が、IPv6 エンドポイントと通信できるようになりました。	IPv6 アドレスがノードに割り当てられた後、Pods の IPv4 アドレスが実行されているノードの IPv6 アドレスに変換されるネットワークアドレスです。	2023 年 6 月 19 日
AWS GovCloud (US) Regions の Windows マネージド型ノードグループ	AWS GovCloud (US) Regions では、Amazon EKS マネージド型ノードグループが Windows コンテナを実行できるようになりました。	2023 年 5 月 30 日
Kubernetes バージョン 1.27	Kubernetes バージョン 1.27 の新しいクラスターとバージョンアップグレードのサポートを追加。	2023 年 5 月 24 日

Kubernetes バージョン 1.26	Kubernetes バージョン 1.26 の新しいクラスターとバージョンアップグレードのサポートを追加。	2023 年 4 月 11 日
「ドメインレス gMSA」	Windows Pods でドメインレス gMSA を使用できます。	2023 年 3 月 27 日
Amazon EKS AWS リージョン の拡張	Amazon EKS がアジアパシフィック (メルボルン) (ap-southeast-4) AWS リージョン で利用可能になりました。	2023 年 3 月 10 日
Amazon File Cache CSI ドライバー	Amazon EKS クラスターに Amazon File Cache CSI ドライバーをインストールできるようになりました。	2023 年 3 月 3 日
Kubernetes バージョン 1.25 が AWS Outposts のローカルクラスターで使用できるようになりました	Kubernetes バージョン 1.22 – 1.25 を使用して、Outpost に Amazon EKS ローカルクラスターを作成できるようになりました。	2023 年 3 月 1 日
Kubernetes バージョン 1.25	Kubernetes バージョン 1.25 の新しいクラスターとバージョンアップグレードのサポートを追加。	2023 年 2 月 22 日
AWS 管理ポリシーの更新 – 既存のポリシーへの更新	Amazon EKS において、既存の AWS 管理ポリシーの更新を行いました。	2023 年 2 月 7 日

Amazon EKS AWS リージョン の拡張	Amazon EKS は現在、アジアパシフィック (ハイデラバード) (ap-south-2)、欧州 (チューリッヒ) (eu-central-2)、欧州 (スペイン) (eu-south-2) AWS リージョンでご利用いただけます。	2023 年 2 月 6 日
Kubernetes バージョン 1.21 — 1.24 が、AWS Outposts 上のローカルクラスターで使用可能になりました。	Kubernetes バージョン 1.21 – 1.24 を使用して、Outpost に Amazon EKS ローカルクラスターを作成できるようになりました。以前は、1.21 バージョンのみが使用可能でした。	2023 年 1 月 17 日
Amazon EKS が AWS PrivateLink をサポートするようになりました	AWS PrivateLink を使用して、VPC と Amazon EKS 間にプライベート接続を作成できます。	2022 年 12 月 16 日
マネージド型ノードグループの Windows サポート	Amazon EKS マネージド型ノードグループに、Windows を使用できるようになりました。	2022 年 12 月 15 日
独立系ソフトウェアベンダーの Amazon EKS アドオンが、AWS Marketplace で利用できるようになりました。	AWS Marketplace を介して、独立系ソフトウェアベンダーの Amazon EKS アドオンを閲覧および購読できるようになりました。	2022 年 11 月 28 日
AWS 管理ポリシーの更新 – 既存のポリシーへの更新	Amazon EKS において、既存の AWS 管理ポリシーの更新を行いました。	2022 年 11 月 17 日

Kubernetes バージョン 1.24	Kubernetes バージョン 1.24 の新しいクラスターとバージョンアップグレードのサポートを追加。	2022 年 11 月 15 日
Amazon EKS AWS リージョン の拡張	Amazon EKS が中東 (UAE) (me-central-1) AWS リージョン で利用可能になりました。	2022 年 11 月 3 日
AWS 管理ポリシーの更新 – 既存のポリシーへの更新	Amazon EKS において、既存の AWS 管理ポリシーの更新を行いました。	2022 年 10 月 24 日
AWS 管理ポリシーの更新 – 既存のポリシーへの更新	Amazon EKS において、既存の AWS 管理ポリシーの更新を行いました。	2022 年 10 月 20 日
AWS Outposts のローカルクラスターが使用可能になりました	Outpost に Amazon EKS ローカルクラスターを作成できるようになりました。	2022 年 9 月 19 日
Fargate vCPU ベースのクォータ	Fargate は、Pod ベースのクォータから vCPU ベースのクォータに移行しています。	2022 年 9 月 8 日
AWS 管理ポリシーの更新 – 既存のポリシーへの更新	Amazon EKS において、既存の AWS 管理ポリシーの更新を行いました。	2022 年 8 月 31 日
コストモニタリング	Amazon EKS が Kubecost をサポートするようになりました。これにより、Pods、ノード、名前空間、ラベルなどの Kubernetes リソースごとにコストを分類して監視することができます。	2022 年 8 月 24 日

AWS 管理ポリシーの更新 – 新しいポリシー	Amazon EKS に、新しい AWS 管理ポリシーが追加されました。	2022 年 8 月 24 日
AWS 管理ポリシーの更新 – 新しいポリシー	Amazon EKS に、新しい AWS 管理ポリシーが追加されました。	2022 年 8 月 23 日
請求用のタグリソース	すべてのクラスターで、`aws:eks:cluster-name` 生成のコスト配分タグがサポートされるようになりました。	2022 年 8 月 16 日
Fargate プロファイルのワイルドカード	名前空間、ラベルキー、およびラベル値のセレクター基準で、Fargate プロファイルのワイルドカードがサポートされるようになりました。	2022 年 8 月 16 日
Kubernetes バージョン `1.23`	Kubernetes バージョン `1.23` の新しいクラスターとバージョンアップグレードのサポートを追加。	2022 年 8 月 11 日
AWS Management Console 内の Kubernetes リソースを表示	これで、AWS Management Console を使用してクラスターにデプロイされた Kubernetes リソースの情報を表示できます。	2022 年 5 月 3 日
Amazon EKS AWS リージョン の拡張	Amazon EKS がアジアパシフィック (ジャカルタ) (`ap-southeast-3`) AWS リージョン で利用可能になりました。	2022 年 5 月 2 日

観測可能性ページとページと ADOT アドオンのサポート	追加された [可観測性] ページと、AWS Distro for OpenTelemetry (ADOT)。	2022 年 4 月 21 日
Kubernetes バージョン 1.22	Kubernetes バージョン 1.22 の新しいクラスターとバージョンアップグレードのサポートを追加。	2022 年 4 月 4 日
AWS 管理ポリシーの更新 – 新しいポリシー	Amazon EKS に、新しい AWS 管理ポリシーが追加されました。	2022 年 4 月 4 日
追加された Fargate Pod 適用の詳細	Fargate Pods をアップグレードする場合、Amazon EKS は最初に Pod 中断バジェットに基づいて Pods を削除しようとします。Pods が削除される前に、失敗したエビクションに対応するイベントルールを作成できます。	2022 年 4 月 1 日
フルリリース: Amazon EBS CSI ドライバーのアドオンサポート	AWS Management Console、AWS CLI、API を使用して Amazon EBS CSI ドライバーを管理できるようになりました。	2022 年 3 月 31 日
AWS Outposts コンテンツの更新	AWS Outposts で Amazon EKS クラスターをデプロイする手順。	2022 年 3 月 22 日
AWS 管理ポリシーの更新 – 既存のポリシーへの更新	Amazon EKS において、既存の AWS 管理ポリシーの更新を行いました。	2022 年 3 月 21 日

Windows containerd サポート	これで、Windows ノードの containerd ランタイムを選択できます。	2022 年 3 月 14 日
Amazon EKS Connector の考慮事項をセキュリティドキュメントに追加	接続されたクラスターに関連する責任共有モデルについて説明しています。	2022 年 2 月 25 日
Pods とサービスへの IPv6 アドレスの割り当てます	Pods とサービスに IPv6 アドレスを割り当てる 1.21 以降のクラスターを作成できるようになりました。	2022 年 1 月 6 日
AWS 管理ポリシーの更新 – 既存のポリシーへの更新	Amazon EKS において、既存の AWS 管理ポリシーの更新を行いました。	2021 年 12 月 13 日
プレビューリリース: Amazon EBS CSI ドライバーのアドオンサポート	AWS Management Console、AWS CLI、および API を使用して Amazon EBS CSI ドライバーを管理できるようになりました。	2021 年 12 月 9 日
Karpenter オートスケーラーのサポート	Karpenter オープンソースプロジェクトを使用してノードを自動スケールできるようになりました。	2021 年 11 月 29 日
Fargate ログ記録での Fluent Bit Kubernetes フィルターのサポート	Fargate ログ記録で Fluent Bit Kubernetes フィルターを使用できるようになりました。	2021 年 11 月 10 日
コントロールプレーンで利用可能な Windows サポート	Windows サポートがコントロールプレーンで利用可能になりました。データプレーンで有効にする必要はなくなりました。	2021 年 11 月 9 日

マネージド型ノードグループの AMI タイプとして Bottlerocket を追加	以前は、Bottlerocket はセルフマネージド型のノードオプションとしてのみ利用可能でした。マネージド型ノードグループとして設定できるようになったため、ノードのコンプライアンス要件を満たすために必要な労力が軽減されます。	2021 年 10 月 28 日
DL1 ドライバーのサポート	カスタム Amazon Linux AMI が、Amazon Linux 2 の深層学習ワークロードをサポートするようになりました。この有効化により、一般的なオンプレミスまたはクラウドのベースライン設定が可能になります。	2021 年 10 月 25 日
VT1 ビデオのサポート	カスタム Amazon Linux AMI では、一部のディストリビューションで VT1 がサポートされるようになりました。この有効化により、Amazon EKS クラスター上の Xilinx U30 デバイスがアドバタイズされます。	2021 年 9 月 13 日
Amazon EKS Connector が利用可能になりました	Amazon EKS Connector を使用して、準拠する Kubernetes クラスターを AWS に登録および接続し、Amazon EKS コンソールで可視化できます。	2021 年 9 月 8 日

Amazon EKS Anywhere 利用可能になりました	Amazon EKS Anywhere は、オンプレミスの Kubernetes クラスターの作成と運用に使用できる Amazon EKS の新しいデプロイオプションです。	2021 年 9 月 8 日
Amazon FSx for NetApp ONTAP CSI ドライバ	Amazon FSx for NetApp ONTAP CSI ドライバーの概要と、その他のリファレンスへのリンクを提供するトピックを追加しました。	2021 年 9 月 2 日
マネージド型ノードグループは、ノードの Amazon EKS 推奨最大 Pods を自動計算するようになりました	マネージド型ノードグループは、起動テンプレートなしで、または AMI ID を指定していない起動テンプレートを使用して、デプロイするノードの Amazon EKS 最大 Pods を自動的に計算するようになりました。	2021 年 8 月 30 日
Amazon EKS アドオンソフトウェアを削除せずに、アドオン設定の Amazon EKS 管理を削除します。	クラスターからアドオンソフトウェアを削除せずに Amazon EKS アドオンを削除できるようになりました。	2021 年 8 月 20 日
Multus を使用してマルチホーム Pods を作成する	Multus を使用して Pod に複数のネットワークインターフェイスを追加できるようになりました。	2021 年 8 月 2 日
Linux Amazon EC2 ノードに複数の IP アドレスを追加する	Linux Amazon EC2 ノードに大量の IP アドレスを追加できるようになりました。これはつまり、各ノードでより高い密度の Pods を実行できるということです。	2021 年 7 月 27 日

containerd ランタイムブートストラップ	Amazon EKS 最適化されアクセラレートされた Amazon Linux Amazon マシンイメージ (AMI) に、Amazon EKS 最適化された Bottlerocket AMI で containerd ランタイムの有効化に使用できる、ブートストラップフラグが含まれるようになりました。このフラグは、AMI のすべてのサポートされている Kubernetes バージョンで使用できます。	2021 年 7 月 19 日
Kubernetes バージョン 1.21	Kubernetes バージョン 1.21 のサポートを追加。	2021 年 7 月 19 日
管理ポリシーのトピックを追加	Amazon EKS IAM 管理ポリシーと、2021 年 6 月 17 日以降に行われたすべての変更のリスト。	2021 年 6 月 17 日
Fargate で Pods のセキュリティグループを使用する	Amazon EC2 ノードに加えて、Fargate でも Pods のセキュリティグループを使用できるようになりました。	2021 年 6 月 1 日
CoreDNS と kube-proxy Amazon EKS アドオンの追加	Amazon EKS は、クラスターの CoreDNS および kube-proxy Amazon EKS アドオンの管理に役立つようになりました。	2021 年 5 月 19 日
Kubernetes バージョン 1.20	Kubernetes バージョン 1.20 の新しいクラスターとバージョンアップグレードのサポートを追加。	2021 年 5 月 18 日

AWS Load Balancer Controller2.2.0 のリリース	AWS Load Balancer Controller 使用して、インスタンスまたは IP ターゲットを使用した Elastic Load Balancer を作成できるようなりました。	2021 年 5 月 14 日
マネージド型ノードグループのノードテイント	Amazon EKS では、マネージド型ノードグループへのノードテイントの追加がサポートされるようになりました。	2021 年 5 月 11 日
既存クラスターのシークレット暗号化	Amazon EKS では、既存クラスターへのシークレット暗号化の追加がサポートされるようになりました。	2021 年 2 月 26 日
Kubernetes バージョン 1.19	Kubernetes バージョン 1.19 の新しいクラスターとバージョンアップグレードのサポートを追加。	2021 年 2 月 16 日
Amazon EKS では、バージョン 1.16 以降のクラスターに対してユーザーを認証する方法として、OpenID Connect (OIDC) ID プロバイダーがサポートされるようになりました。	OIDC ID プロバイダーは、AWS Identity and Access Management (IAM) と併用、または代替として使用できます。	2021 年 2 月 12 日
AWS Management Console でのノードとワークロードのリソースの表示	マネージド型ノード、セルフマネージド型ノード、Fargate ノード、およびデプロイされた Kubernetes ワークロードに関する詳細を AWS Management Console で表示できるようになりました。	2020 年 12 月 1 日

マネージド型ノードグループへのスポットインスタンスタイプのデプロイ	複数のスポットまたはオンデマンドインスタンスタイプを、マネージド型ノードグループにデプロイできるようになりました。	2020 年 12 月 1 日
Amazon EKS がクラスターの特定のアドオンを管理できるようになりました	アドオンをお客様自身で管理することもできますが、Amazon EKS API を介してアドオンの起動とバージョンを Amazon EKS に制御させることもできます。	2020 年 12 月 1 日
複数の Ingress 間で ALB を共有する	AWS Application Load Balancer (ALB) を複数の Kubernetes Ingress 間で共有できるようになりました。以前は、各 Ingress に個別の ALB をデプロイする必要がありました。	2020 年 10 月 23 日
NLB IP ターゲットのサポート	IP ターゲットを使用して、Network Load Balancer (NLB) をデプロイできるようになりました。これにより、NLB を使用して、Fargate Pods へのネットワークトラフィック、および Amazon EC2 ノードで実行されている Pods への直接的なネットワークトラフィックの負荷分散ができます。	2020 年 10 月 23 日
Kubernetes バージョン 1.18	Kubernetes バージョン 1.18 の新しいクラスターとバージョンアップグレードのサポートを追加。	2020 年 10 月 13 日

Kubernetes サービス IP アドレス割り当て用のカスタム CIDR ブロックを指定します。	Kubernetes がサービス IP アドレスを割り当てるカスタム CIDR ブロックを指定できるようになりました。	2020 年 9 月 29 日
セキュリティグループを個別の Pods に割り当てる	多くの Amazon EC2 インスタンスタイプで実行されている個々の Pods の一部にさまざまなセキュリティグループを関連付けることができるようになりました。	2020 年 9 月 9 日
Bottlerocket をノードにデプロイする	Bottlerocket を実行中のノードをデプロイできるようになりました。	2020 年 8 月 31 日
Arm ノードを起動する機能は、一般的に利用可能です	マネージド型ノードグループ、およびセルフマネージド型ノードグループで、Arm ノードを起動できるようになりました。	2020 年 8 月 17 日
マネージド型ノードグループの起動テンプレートとカスタム AMI	Amazon EC2 起動テンプレートを使用して、マネージド型ノードグループをデプロイできるようになりました。起動テンプレートで、カスタム AMI を指定できます。	2020 年 8 月 17 日
AWS Fargate の EFS サポート	AWS Fargate で Amazon EFS を使用できるようになりました。	2020 年 8 月 17 日

Amazon EKS プラットフォームのバージョンの更新	これは、セキュリティの修正および機能強化が行われた新しいプラットフォームバージョンです。これには、Kubernetes バージョン 1.15 以降で Network Load Balancer を使用する場合、LoadBalancer タイプのサービスに対する UDP サポートが含まれます。詳細については、「GitHub」の「Allow UDP for AWS Network Load Balancer」(Network Load Balancer に UDP を許可する) という問題を参照してください。	2020 年 8 月 12 日
Amazon EKS AWS リージョン の拡張	Amazon EKS がアフリカ (ケープタウン) (af-south-1) および欧州 (ミラノ) (eu-south-1) AWS リージョンで利用可能になりました。	2020 年 8 月 6 日
Fargate 使用状況メトリクス	AWS Fargate は、Fargate オンデマンドリソースのアカウント使用状況を可視化する CloudWatch 使用状況メトリクスを提供します。	2020 年 8 月 3 日
Kubernetes バージョン 1.17	Kubernetes バージョン 1.17 の新しいクラスターとバージョンアップグレードのサポートを追加。	2020 年 7 月 10 日

Kubernetes 用 App Mesh コントローラーを使用して、Kubernetes 内から App Mesh リソースを作成し管理する	Kubernetes 内から App Mesh リソースを作成し管理できるようになりました。また、コントローラーは、デプロイする Pods に Envoy プロキシと init コンテナを自動的に挿入します。	2020 年 6 月 18 日
Amazon EKS が Amazon EC2 Inf1 ノードをサポートするようになりました	Amazon EC2 Inf1 ノードをクラスターに追加できます。	2020 年 6 月 4 日
Amazon EKS AWS リージョン の拡張	Amazon EKS が AWS GovCloud (米国東部) (`us-gov-east-1`) および AWS GovCloud (米国西部) (`us-gov-west-1`) AWS リージョン で利用可能になりました。	2020 年 5 月 13 日
Kubernetes1.12 は Amazon EKS でサポートされなくなりました	Kubernetes バージョン `1.12` は Amazon EKS でサポートされなくなりました。サービスの中断を防ぐため、`1.12` クラスターはバージョン `1.13` 以降に更新してください。	2020 年 5 月 12 日
Kubernetes バージョン `1.16`	Kubernetes バージョン `1.16` の新しいクラスターとバージョンアップグレードのサポートを追加。	2020 年 4 月 30 日
サービスにリンクされたロール AWSServiceRoleForAmazonEKS を追加	サービスにリンクされたロール [AWSServiceRoleForAmazonEKS] を追加しました。	2020 年 4 月 16 日

Kubernetes バージョン 1.15	Kubernetes バージョン 1.15 の新しいクラスターとバージョンアップグレードのサポートを追加。	2020 年 3 月 10 日
Amazon EKS AWS リージョン の拡張	Amazon EKS が北京 (cn-north-1) および寧夏 (cn-northwest-1) AWS リージョン で利用可能になりました。	2020 年 2 月 26 日
FSx for Lustre CSI ドライバー	Kubernetes 1.14 Amazon EKS クラスターに FSx for Lustre CSI ドライバーをインストールするトピックを追加しました。	2019 年 12 月 23 日
クラスターのパブリックアクセスエンドポイントへのネットワークアクセスを制限	この更新では、Amazon EKS を使用して、Kubernetes API サーバーのパブリックアクセスエンドポイントと通信できる CIDR 範囲を制限できるようになりました。	2019 年 12 月 20 日
VPC 外部からクラスターのプライベートアクセスエンドポイントアドレスを解決	この更新では、Amazon EKS を使用して、VPC 外から Kubernetes API サーバーのプライベートアクセスエンドポイントを解決できるようになりました。	2019 年 12 月 13 日
(ベータ) Amazon EC2 A1 Amazon EC2 インスタンスノード	Amazon EKS クラスターに登録する Amazon EC2 A1 Amazon EC2 インスタンスノードを起動します。	2019 年 12 月 4 日

AWS Outposts でのクラスターの作成	Amazon EKS では、AWS Outposts でのクラスターの作成がサポートされるようになりました。	2019 年 12 月 3 日
Amazon EKS の AWS Fargate	Amazon EKS Kubernetes クラスターは、Fargate での Pods の実行をサポートするようになりました。	2019 年 12 月 3 日
Amazon EKS AWS リージョン の拡張	Amazon EKS がカナダ (中部) (`ca-central-1`) AWS リージョン で使用可能になりました。	2019 年 11 月 21 日
マネージド型ノードグループ	Amazon EKS マネージド型ノードグループは、Amazon EKS Kubernetes クラスターのノード (Amazon EC2 インスタンス) のプロビジョニングとライフサイクル管理を自動化します。	2019 年 11 月 18 日
Amazon EKS プラットフォームのバージョンの更新	`CVE-2019-11253` に対応する新しいプラットフォームバージョン。	2019 年 11 月 6 日
Kubernetes1.11 は Amazon EKS でサポートされなくなりました	Kubernetes バージョン `1.11` は Amazon EKS でサポートされなくなりました。サービスの中断を防ぐため、`1.11` クラスターはバージョン `1.12` 以降に更新してください。	2019 年 11 月 4 日
Amazon EKS AWS リージョン の拡張	Amazon EKS が南米 (サンパウロ) (`sa-east-1`) AWS リージョン で利用可能になりました。	2019 年 10 月 16 日

Windows のサポート	Kubernetes のバージョン 1.14 を実行する Amazon EKS クラスターが Windows ワークロードをサポートするようになりました。	2019 年 10 月 7 日
Autoscaling	Amazon EKS クラスターでサポートされる異なるタイプの Kubernetes autoscaling に関する章を追加。	2019 年 9 月 30 日
Kubernetes ダッシュボードの更新	ベータ 2.0 バージョンを使用するため、Amazon EKS クラスターに Kubernetes ダッシュボードをインストールするトピックを更新しました。	2019 年 9 月 28 日
Amazon EFS CSI ドライバー	Kubernetes 1.14 Amazon EKS クラスターに Amazon EFS CSI ドライバーをインストールするトピックを追加しました。	2019 年 9 月 19 日
Amazon EKS 最適化された AMI ID の Amazon EC2 Systems Manager パラメータ	Amazon EC2 Systems Manager パラメータを使用して、Amazon EKS 最適化された AMI ID を取得するトピックを追加しました。このパラメータを使用すると、AMI ID を検索する必要がなくなります。	2019 年 9 月 18 日
Amazon EKS リソースのタグ付け	Amazon EKS クラスターのタグ付けを管理できます。	2019 年 9 月 16 日

Amazon EBS CSI ドライバー	Kubernetes 1.14 Amazon EKS クラスターに Amazon EBS CSI ドライバーをインストールするトピックを追加しました。	2019 年 9 月 9 日
新しい Amazon EKS 最適化 AMI に CVE-2019-9512 および CVE-2019-9514 用のパッチを適用	Amazon EKS は CVE-2019-9512 および CVE-2019-9514 に対応するため、Amazon EKS 最適化された AMI を更新しました。	2019 年 9 月 6 日
Amazon EKS の Kubernetes1.11 の廃止を発表	Amazon EKS は 2019 年 11 月 4 日に Kubernetes バージョン 1.11 のサポートを終了しました。	2019 年 9 月 4 日
Kubernetes バージョン 1.14	Kubernetes バージョン 1.14 の新しいクラスターとバージョンアップグレードのサポートを追加。	2019 年 9 月 3 日
サービスアカウントの IAM ロール	Amazon EKS クラスターのサービスアカウントの IAM ロールを使用すると、IAM ロールを Kubernetes サービスアカウントに関連付けることができます。この機能を使用すると、ノード IAM ロールに拡張されたアクセス許可を提供する必要がなくなります。この方法で、そのノード上の Pods は AWS API を呼び出すことができます。	2019 年 9 月 3 日

Amazon EKS AWS リージョン の拡張	Amazon EKS が中東 (バーレーン) (me-south-1) AWS リージョン で利用可能になりました。	2019 年 8 月 29 日
Amazon EKS プラットフォームのバージョンの更新	CVE-2019-9512 および CVE-2019-9514 に対応する新しいプラットフォームバージョン。	2019 年 8 月 28 日
Amazon EKS プラットフォームのバージョンの更新	CVE-2019-11247 および CVE-2019-11249 に対応する新しいプラットフォームバージョン。	2019 年 8 月 5 日
Amazon EKS リージョンの拡張	Amazon EKS がアジアパシフィック (香港) (ap-east-1) AWS リージョン で利用可能になりました。	2019 年 7 月 31 日
Kubernetes1.10 は Amazon EKS でサポートされなくなりました	Kubernetes バージョン 1.10 は Amazon EKS でサポートされなくなりました。サービスの中断を防ぐため、1.10 クラスターはバージョン 1.11 以降に更新してください。	2019 年 7 月 30 日
ALB Ingress Controller に関するトピックの追加	AWS ALB Ingress Controller for Kubernetes は、Ingress リソースが作成されたときに ALB を作成するコントローラーです。	2019 年 7 月 11 日
新しい Amazon EKS 最適化された AMI	不要な kubectl バイナリを AMI から削除します。	2019 年 7 月 3 日

Kubernetes バージョン 1.13	Kubernetes バージョン 1.13 の新しいクラスターとバージョンアップグレードのサポートを追加。	2019 年 6 月 18 日
新しい Amazon EKS 最適化 AMI に AWS-2019-005 用のパッチを適用	Amazon EKS では、AWS-2019-005 で説明されている脆弱性に対応するため、Amazon EKS 最適化された AMI が更新されました。	2019 年 6 月 17 日
Amazon EKS での Kubernetes1.10 のサポート終了を発表	Amazon EKS は、2019 年 7 月 22 日に Kubernetes バージョン 1.10 のサポートを停止しました。	2019 年 5 月 21 日
Amazon EKS プラットフォームのバージョンの更新	kubelet 証明書のカスタム DNS 名をサポートし、etcd パフォーマンスを向上させる Kubernetes 1.11 および 1.10 クラスター用の新しいプラットフォームバージョン。	2019 年 5 月 21 日

AWS CLIget-token コマンド	aws eks get-token コマンドが AWS CLI に追加されました。クラスター API サーバー通信用のクライアントセキュリティトークンを作成するために、AWS IAM Authenticator for Kubernetes をインストールする必要がなくなりました。この新しい機能を使用するには、インストール済みの AWS CLI を最新バージョンにアップグレードしてください。詳細については、「AWS Command Line Interface ユーザーガイド」の「AWS Command Line Interface のインストール」を参照してください。	2019 年 5 月 10 日
eksctl の開始方法	この入門ガイドは、eksctl を使用して Amazon EKS を開始するために必要な、すべてのリソースの作成に役立ちます。これは、Amazon EKS で Kubernetes クラスターを作成および管理するためのシンプルなコマンドラインユーティリティです。	2019 年 5 月 10 日

Amazon EKS プラットフォームのバージョンの更新	kubelet 証明書のカスタム DNS 名をサポートし、etcd パフォーマンスを向上させる Kubernetes 1.12 クラスター用の新しいプラットフォームバージョン。これにより、ノードの kubelet デーモンによって、数秒ごとに新しい証明書が要求されるバグが修正されました。	2019 年 5 月 8 日
Prometheus チュートリアル	Amazon EKS クラスターに Prometheus のデプロイに関するトピックが追加されました。	2019 年 4 月 5 日
Amazon EKS コントロールプレーンのログ記録	今回の更新では、監査および診断ログは Amazon EKS コントロールペインから直接取得できるようになりました。これらの CloudWatch ログは、クラスターをセキュリティで保護および実行するための参照としてアカウントで使用できます。	2019 年 4 月 4 日
Kubernetes バージョン 1.12	Kubernetes バージョン 1.12 の新しいクラスターとバージョンアップグレードのサポートを追加。	2019 年 3 月 28 日
App Mesh 入門ガイドの追加	App Mesh と Kubernetes の入門用ドキュメントが追加されました。	2019 年 3 月 27 日

Amazon EKS API サーバーエンドポイントのプライベートアクセス	Amazon EKS クラスターの Kubernetes API サーバーエンドポイントのパブリックアクセスを無効にする方法に関するドキュメントが追加されました。	2019 年 3 月 19 日
Kubernetes メトリクスサーバーのインストールに関するトピックの追加	Kubernetes メトリクスサーバーは、クラスター内のリソース使用状況データを集約します。	2019 年 3 月 18 日
関連するオープンソースプロジェクトのリストの追加	これらのオープンソースプロジェクトは、Amazon EKS によって管理されているクラスターを含め、AWS で実行されている Kubernetes クラスターの機能を拡張します。	2019 年 3 月 15 日
Helm のローカルインストールに関するトピックの追加	Kubernetes 用 `helm` パッケージマネージャーは、Kubernetes クラスターでアプリケーションをインストールおよび管理するために役立ちます。このトピックでは、`helm` および `tiller` バイナリをローカルにインストールして実行する方法について説明します。これにより、ローカルシステムで Helm CLI を使用してチャートをインストールおよび管理することができます。	2019 年 3 月 11 日

Amazon EKS プラットフォームのバージョンの更新	CVE-2019-1002100 に対応するために Amazon EKS Kubernetes 1.11 クラスターをパッチレベル 1.11.8 に更新する新しいプラットフォームバージョン。	2019 年 3 月 8 日
クラスターの上限の引き上げ	Amazon EKS では、AWS リージョン 内で作成できるクラスターの数が 3 から 50 に増えました。	2019 年 2 月 13 日
Amazon EKS AWS リージョン の拡張	Amazon EKS が欧州 (ロンドン) (eu-west-2)、欧州 (パリ) (eu-west-3)、アジアパシフィック (ムンバイ) (ap-south-1) AWS リージョンで利用可能になりました。	2019 年 2 月 13 日
新しい Amazon EKS 最適化 AMI に ALAS-2019-1156 用のパッチを適用	Amazon EKS では、ALAS-2019-1156 で説明されている脆弱性に対応するため、Amazon EKS 最適化された AMI が更新されました。	2019 年 2 月 11 日
新しい Amazon EKS 最適化 AMI に ALAS2-2019-1141 用のパッチを適用	Amazon EKS では、ALAS2-2019-1141 で参照されている CVE に対応するため、Amazon EKS 最適化された AMI が更新されました。	2019 年 1 月 9 日

Amazon EKS AWS リージョン の拡張	Amazon EKS がアジアパシフィック (ソウル) (ap-northeast-2) AWS リージョン で利用可能になりました。	2019 年 1 月 9 日
Amazon EKS リージョンの拡張	Amazon EKS が、欧州 (フランクフルト) (eu-central-1)、アジアパシフィック (東京) (ap-northeast-1)、アジアパシフィック (シンガポール) (ap-southeast-1)、およびアジアパシフィック (シドニー) (ap-southeast-2) AWS リージョン で追加で利用可能になりました。	2018 年 12 月 19 日
Amazon EKS クラスターの更新	「Amazon EKS クラスターの Kubernetes バージョン更新」および「セルフマネージド型ノードの更新」のドキュメントが追加されました。	2018 年 12 月 12 日
Amazon EKS AWS リージョン の拡張	Amazon EKS が欧州 (ストックホルム) (eu-north-1) AWS リージョン で利用可能になりました。	2018 年 12 月 11 日
Amazon EKS プラットフォームのバージョンの更新	CVE-2018-1002105 に対応するために Kubernetes をパッチレベル 1.10.11 に更新する新しいプラットフォームバージョン。	2018 年 12 月 4 日

ALB Ingress Controller のバージョン 1.0.0 のサポートを追加	ALB Ingress Controller で、AWS の公式サポートを備えたバージョン 1.0.0 がリリースされました。	2018 年 11 月 20 日
CNI ネットワーク設定のサポートを追加	Amazon VPC CNI plugin for Kubernetes のバージョン 1.2.1 が、セカンダリ Pod ネットワークインターフェイスのカスタムネットワーク設定をサポートするようになりました。	2018 年 10 月 16 日
MutatingAdmissionWebhook と ValidatingAdmissionWebhook のサポートが追加されました	Amazon EKS プラットフォームバージョン 1.10-eks.2 が MutatingAdmissionWebhook および ValidatingAdmissionWebhook アドミッションコントローラーをサポートするようになりました。	2018 年 10 月 10 日
パートナー AMI 情報の追加	Canonical 社は Amazon EKS と提携し、クラスターで使用できるノード AMI を作成しました。	2018 年 10 月 3 日
AWS CLIupdate-kubeconfig のコマンドの手順が追加されました	Amazon EKS で update-kubeconfig が AWS CLI に追加され、クラスターにアクセスするための kubeconfig ファイルの作成プロセスが簡単になりました。	2018 年 9 月 21 日

新しい Amazon EKS 最適化された AMI	Amazon EKS は、さまざまなセキュリティ修正と AMI 最適化を提供する Amazon EKS 最適化された AMI (GPU サポートあり/なし) を更新しました。	2018 年 9 月 13 日
Amazon EKS AWS リージョン の拡張	Amazon EKS が欧州 (アイルランド) (`eu-west-1`) リージョンで利用可能になりました。	2018 年 9 月 5 日
Amazon EKS プラットフォームのバージョンの更新	Kubernetes アグリゲーションレイヤーと Horizontal Pod Autoscaler (HPA) をサポートする新しいプラットフォームバージョン。	2018 年 8 月 31 日
新しい Amazon EKS 最適化された AMI と GPU サポート	Amazon EKS は、新しい AWS CloudFormation ノードテンプレートと bootstrap スクリプトを使用するため、Amazon EKS 最適化された AMI を更新しました。さらに、新しい GPU サポートを備えた Amazon EKS 最適化された AMI が利用可能になりました。	2018 年 8 月 22 日
新しい Amazon EKS 最適化 AMI に `ALAS2-2018-1058` 用のパッチを適用	Amazon EKS では、`ALAS2-2018-1058` で参照されている CVE に対応するため、Amazon EKS 最適化された AMI が更新されました。	2018 年 8 月 14 日

Amazon EKS 最適化された AMI のビルドスクリプト	Amazon EKS で、Amazon EKS 最適化された AMI のビルドに使用されているビルドスクリプトをオープンソース化されました。これらのビルドスクリプトは、現在 GitHub で利用可能です。	2018 年 7 月 10 日
Amazon EKS 初回リリース	サービス開始時の初版ドキュメント	2018 年 5 月 6 日