aws

ユーザーガイド

AWS Batch

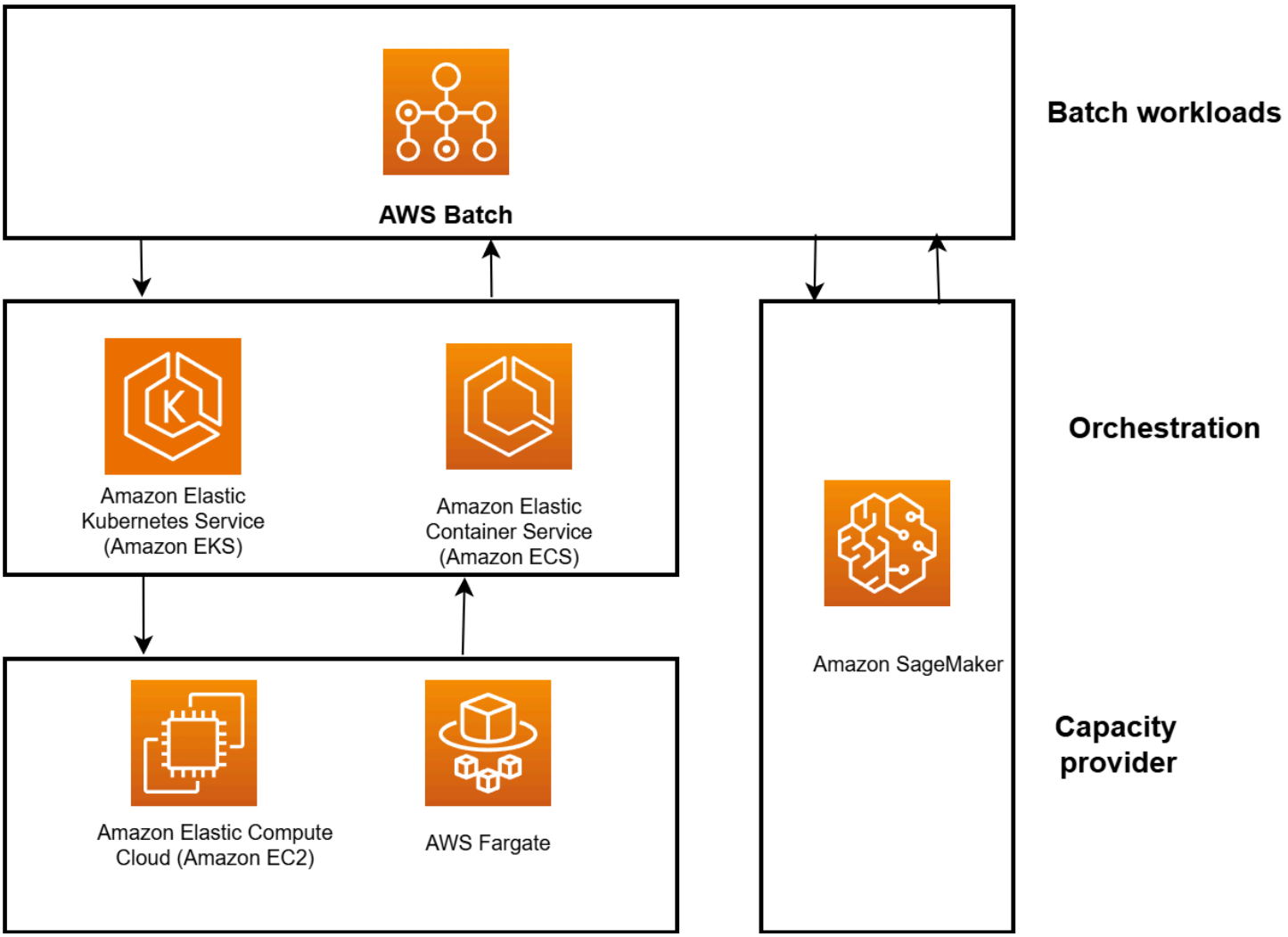
AWS Batch: ユーザーガイド

Table of Contents

AWS Batch とは

AWS Batch を使用すると、AWS クラウド でバッチコンピューティングワークロードを実行できます。バッチコンピューティングは、開発者、科学者、エンジニアが大量のコンピューティングリソースにアクセスするための一般的な方法です。AWS Batch は、従来のバッチコンピューティングソフトウェアと同様に、必要なインフラストラクチャの設定および管理に伴う、差別化につながらない力仕事を排除します。このサービスでは、送信されたジョブに応じてリソースを効率的にプロビジョニングし、キャパシティ制限の排除、コンピューティングコストの削減、および結果の迅速な提供を行うことができます。

AWS Batch はフルマネージドサービスであり、あらゆるスケールのバッチコンピューティングワークロードを実行できます。AWS Batch は、コンピューティングリソースを自動的にプロビジョニングし、ワークロードの量と規模に基づいてワークロードのディストリビューションを最適化します。AWS Batch を使うと、バッチコンピューティングソフトウェアのインストールや管理が不要になり、結果の分析と問題の解決に集中できます。

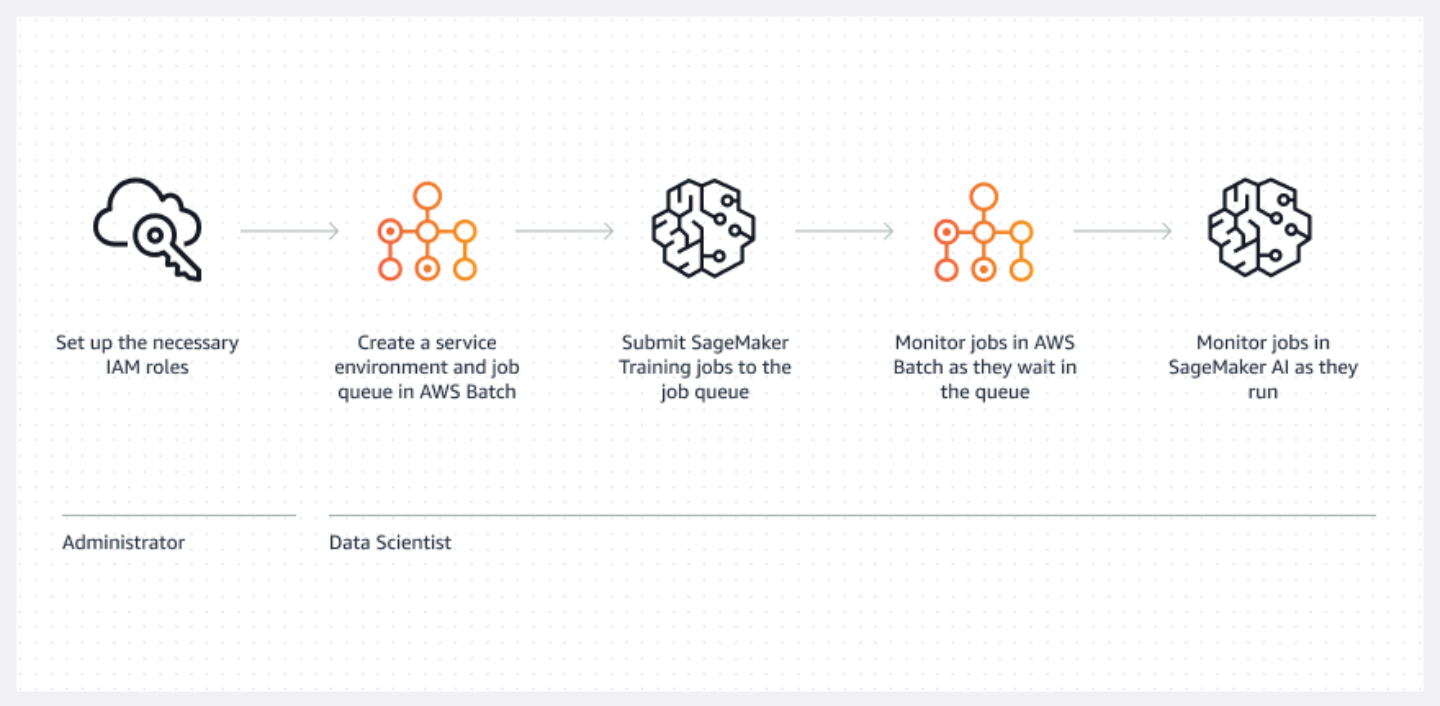


AWS Batch は、AWS マネージドコンテナオーケストレーションサービス、Amazon ECS、Amazon EKS に加えて、大規模でコンピューティング集約型のワークロードを実行するために必要なすべての機能を提供します。 AWS Batch は、Amazon EC2 インスタンスと Fargate リソースでコンピューティング容量をスケールできます。

AWS Batch は、バッチワークロード用のフルマネージドサービスを提供し、スループット、速度、リソース効率、コストのためにこれらのタイプのワークロードを最適化する運用機能を提供します。

また、AWS Batch は SageMaker トレーニングジョブキューイングを有効にし、データサイエンティストと ML エンジニアが優先順位を付けてトレーニングジョブを設定可能なキューに送信できるようにします。この機能を使用すると、リソースが利用可能になるとすぐに ML ワークロードが自動的に実行されるため、手動で調整する必要がなくなり、リソース使用率が向上します。

機械学習ワークロードの場合、AWS Batch は SageMaker トレーニングジョブのキューイング機能を提供します。特定のポリシーでキューを設定して、ML トレーニングワークロードのコスト、パフォーマンス、リソース割り当てを最適化できます。



これにより、管理者がインフラストラクチャとアクセス許可を設定し、データサイエンティストが ML トレーニングワークロードの送信とモニタリングに集中できる責任共有モデルが提供されます。ジョブは自動的にキューに入れられ、設定された優先順位とリソースの可用性に基づいて実行されます。

AWS Batch を初めてお使いになる方向けの情報

AWS Batch を初めて使用する方には、以下のセクションを初めに読むことをお勧めします。

- AWS Batch のコンポーネント
- IAM のアカウントと管理ユーザーを作成する
- AWS Batch を設定する
- AWS Batch チュートリアルの開始方法
- SageMaker AI AWS Batch での の開始方法

関連サービス

AWS Batch はフルマネージド型のバッチコンピューティングサービスで、Amazon ECS、Amazon EKS、AWS Fargate、スポットインスタンスやオンデマンドインスタンスなど、あらゆる AWS コンピューティングサービスでコンテナ化されたバッチ ML、シミュレーション、分析ワークロードを計画、スケジュール、実行します。それぞれのマネージドコンピューティングサービスの詳細については、以下を参照してください。

- Amazon EC2 ユーザーガイド
- AWS Fargate デベロッパーガイド
- Amazon EKS ユーザーガイド
- Amazon SageMaker AI デベロッパーガイド

AWS Batch へのアクセス

次の方法で AWS Batch にアクセスできます。

AWS Batch コンソール

リソースを作成および管理するウェブインターフェイス。

AWS Command Line Interface

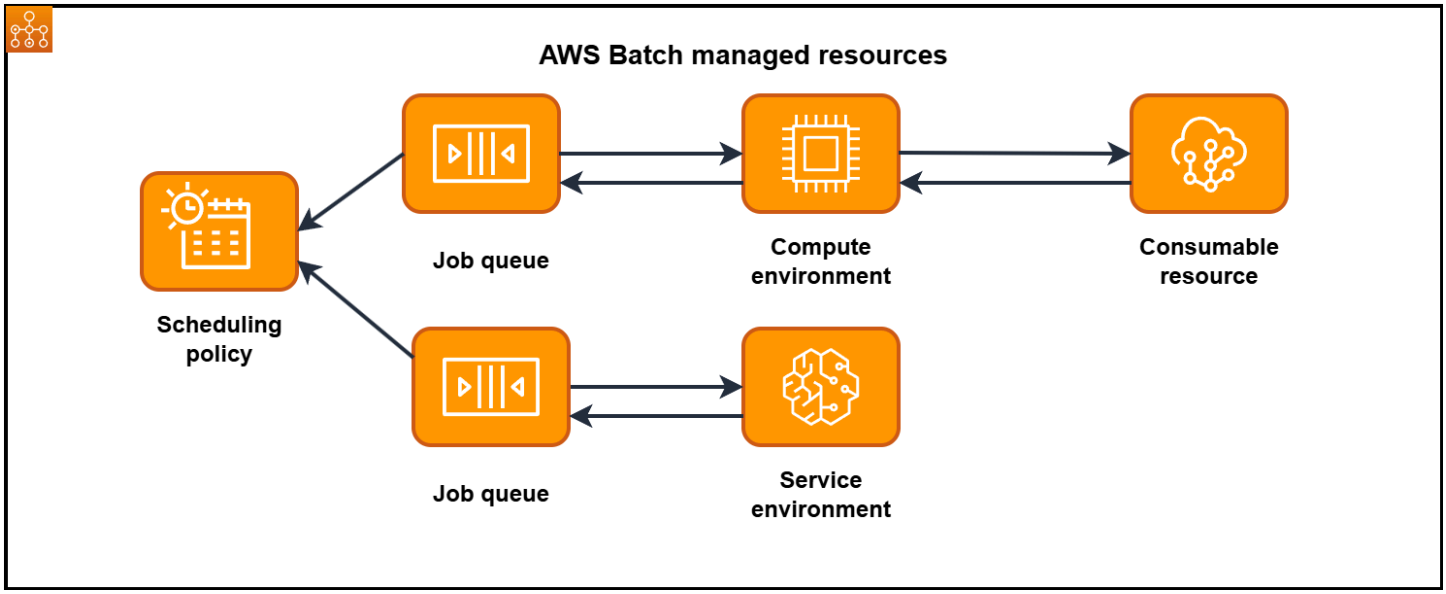
コマンドラインシェルでコマンドを使用して AWS のサービス を操作できます。AWS Command Line Interface は、Windows、macOS、Linux でサポートされています。AWS CLI の詳細については「AWS Command Line Interface ユーザーガイド」を参照してください。「AWS CLI コマンドリファレンス」で AWS Batch コマンドを確認できます。

AWS SDK

HTTP または HTTPS を介してリクエストを送信する代わりに、言語固有の API を使用してアプリケーションを構築することを希望する場合、AWS が提供するライブラリ、サンプルコード、チュートリアルなどのリソースを使用できます。これらのライブラリには、リクエストの暗号化署名、リクエストの再試行、エラーレスポンスの処理などのタスクを自動化する基本的な機能が用意されています。これらの関数を使用すると、作業をより効率的に開始できます。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。

AWS Batch のコンポーネント

AWS Batch は、リージョン内の複数のアベイラビリティーゾーン間で実行中のバッチジョブを簡略化します。新規または既存の VPC 内に AWS Batch コンピューティング環境を作成できます。コンピューティング環境が稼働し、ジョブキューに関連付けられた後で、ジョブを実行する Docker コンテナイメージを指定するジョブ定義を指定できます。コンテナのイメージは、コンテナレジストリに保存され引き出されます。これは AWS インフラストラクチャの内にある場合も外にある場合もあります。



コンピューティング環境

コンピューティング環境は、ジョブを実行するために使用されるマネージドまたはアンマネージドコンピューティングリソースのセットです。マネージド型のコンピューティング環境では、複数の詳細レベルで目的のコンピューティングタイプ (Fargate または EC2) を指定できます。コンピューティング環境は、特定の EC2 インスタンスタイプや特定のモデル (`c5.2xlarge` や `m5.10xlarge` など) を使用するように設定できます。または、最新のインスタンスタイプを使用するように指定することのみを選択できます。また、環境の vCPU の最小数、希望数、最大数を、スポットインスタンスに対して支払う金額と共に、オンデマンドインスタンスの価格と一連のターゲット VPC サブネットの割合として指定できます。AWS Batch によって必要に応じてコンピューティングタイプの起動、管理、終了が効率的に行われます。お客様独自のコンピューティング環境を管理することもできます。その場合は、ユーザー自身が AWS Batch によって作成される Amazon ECS クラスターでインスタンスの設定とスケーリングを行う必要があります。詳細については、[のコンピューティング環境 AWS Batch] を参照してください。

ジョブキュー

AWS Batch ジョブを送信するときは、コンピューティング環境にスケジュールされるまでそのジョブが存在する、特定のジョブキューに送信します。1 つのジョブキューには、1 つ以上のコンピューティング環境を関連付けることができます。これらのコンピューティング環境には優先度の値を割り当てることができ、複数のジョブキュー自体にまたがって割り当てることもできます。例えば、時間が重要なジョブを送信する高優先度キューや、コンピューティングリソースが安価であるときにいつでも実行できるジョブ用の低優先度キューを持つことができます。詳細については、ジョブキューを参照してください。

ジョブ定義

ジョブ定義は、ジョブの実行方法を指定します。ジョブ定義は、ジョブのリソースのブループリントであると考えることができます。他の AWS リソースへのアクセスを提供するために、ジョブに IAM ロールを指定することができます。また、メモリ要件と CPU 要件の両方を指定できます。また、ジョブ定義では、永続的ストレージのコンテナのプロパティ、環境変数、マウントポイントを制御できます。ジョブ定義の多くの仕様は、個別のジョブを送信するときに新しい値を指定してオーバーライドできます。詳細については、ジョブ定義 を参照してください。

ジョブ

AWS Batch に送信する作業単位 (シェルスクリプト、Linux 実行可能ファイル、Docker コンテナイメージなど)。ジョブには名前を付けます。ジョブは、ジョブ定義で指定したパラメーターを使用して、コンピューティング環境でコンテナ化されたアプリケーションとして AWS Fargate または Amazon EC2 リソースで実行されます。ジョブは、他のジョブを名前または ID で参照できます。また、他のジョブの正常な完了または指定したリソースのアベイラビリティに依存する場合があります。詳細については、ジョブ を参照してください。

スケジューリングポリシー

スケジューリングポリシーを使用して、ジョブキュー内のコンピューティングリソースをユーザーやワークロード間でどのように配分するかを設定できます。公平配分スケジューリングポリシーを使用すると、ワークロードまたはユーザーに異なる共有識別子を割り当てることができます。AWS Batch ジョブスケジューラのデフォルトは、先入れ先出し (FIFO) 戦略です。詳細については、公平配分スケジューリングポリシー を参照してください。

消費型リソース

消費型リソースは、サードパーティーのライセンストークン、データベースアクセス帯域幅、サードパーティー API への呼び出しを調整する必要性など、ジョブの実行に必要なリソースです。ジョブの実行に必要な消費型リソースを指定すると、Batch はジョブをスケジュールするときにこれらのリソースの依存関係を考慮します。必要なすべてのリソースが利用可能なジョブのみを割り当てることで、コンピューティングリソースの使用率の低下を抑制できます。詳細については、リソース認識型スケジューリング を参照してください。

サービス環境

サービス環境は、AWS Batch がジョブ実行のために SageMaker と統合する方法を定義します。サービス環境により、AWS Batch は AWS Batch のキューイング、スケジューリング、優先度管理機能を提供しながら、SageMaker のジョブを送信および管理できます。サービス環境は、SageMaker トレーニングジョブなどの特定のサービスタイプの容量制限を定義します。容量制限は、環境内のサービスジョブで使用できる最大リソースを制御します。詳細については、のサービス環境 AWS Batch を参照してください。

サービスジョブ

サービスジョブは、サービス環境で実行するために AWS Batch に送信する作業単位です。サービスジョブは、実際の実行を外部サービスに委任しながら、AWS Batch のキューイングおよびスケジューリング機能を活用します。例えば、サービスジョブとして送信された SageMaker トレーニングジョブは AWS Batch によってキューに入れられ、優先順位が付けられますが、SageMaker トレーニングジョブの実行は SageMaker AI インフラストラクチャ内で行われます。この統合により、データサイエンティストと ML エンジニアは、SageMaker AI トレーニングワークロードに対して、AWS Batch の自動ワークロード管理と優先度キューイングを活用できます。サービスジョブは、名前または ID で他のジョブを参照し、ジョブの依存関係をサポートできます。詳細については、のサービスジョブ AWS Batch を参照してください。

AWS Batch を設定する

すでに Amazon Web Services (AWS) にサインアップしていて、Amazon Elastic Compute Cloud (Amazon EC2)または Amazon Elastic Container Service (Amazon ECS) を使用している場合は、まもなく AWS Batch を使用できるようになります。これらのサービスのセットアッププロセスは似ています。これは、AWS Batch がコンピューティング環境で Amazon ECS コンテナインスタンスを使用するためです。AWS CLI と AWS Batch をともに使用するには、最新の AWS CLI 機能をサポートしているバージョンの AWS Batch を使用する必要があります。AWS CLI で AWS Batch 機能のサポートがない場合は、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、http://aws.amazon.com/cli/ を参照してください。

> **Note**
>
> AWS Batch では Amazon EC2 のコンポーネントを使用するため、これらの手順の多くで Amazon EC2 コンソールを使用します。

AWS Batch のセットアップを行うには、以下のタスクを完了します。

トピック

- IAM のアカウントと管理ユーザーを作成する
- コンピューティング環境およびコンテナインスタンスのIAM ロールの作成
- インスタンスのキーペアを作成する
- VPC を作成する
- セキュリティグループの作成
- AWS CLI のインストール

IAM のアカウントと管理ユーザーを作成する

開始するには、通常管理権限が付与されている AWS アカウントと 1 人のユーザーを作成する必要があります。これを完了するには、以下のチュートリアルを実行してください。

にサインアップする AWS アカウント

がない場合は AWS アカウント、次の手順を実行して作成します。

にサインアップするには AWS アカウント

1. https://portal.aws.amazon.com/billing/signup を開きます。
2. オンラインの手順に従います。

   サインアップ手順の一環として、電話またはテキストメッセージを受け取り、電話キーパッドで検証コードを入力します。

   にサインアップすると AWS アカウント、 AWS アカウントのルートユーザー が作成されます。ルートユーザーには、アカウントのすべての AWS のサービス とリソースへのアクセス権があります。セキュリティベストプラクティスとして、ユーザーに管理アクセス権を割り当て、ルートユーザーアクセスが必要なタスクの実行にはルートユーザーのみを使用するようにしてください。

AWS サインアッププロセスが完了すると、 から確認メールが送信されます。https://aws.amazon.com/ の [マイアカウント] をクリックして、いつでもアカウントの現在のアクティビティを表示し、アカウントを管理することができます。

管理アクセスを持つユーザーを作成する

にサインアップしたら AWS アカウント、日常的なタスクにルートユーザーを使用しないように AWS アカウントのルートユーザー、 を保護し AWS IAM アイデンティティセンター、 を有効にして管理ユーザーを作成します。

を保護する AWS アカウントのルートユーザー

1. ルートユーザーを選択し、 AWS アカウント E メールアドレスを入力して、アカウント所有者AWS マネジメントコンソールとして にサインインします。次のページでパスワードを入力します。

   ルートユーザーを使用してサインインする方法については、「AWS サインイン ユーザーガイド」の「ルートユーザーとしてサインインする」を参照してください。
2. ルートユーザーの多要素認証 (MFA) を有効にします。

   手順については、IAM ユーザーガイドの AWS アカウント 「ルートユーザー (コンソール) の仮想 MFA デバイスを有効にする」を参照してください。

管理アクセスを持つユーザーを作成する

1. IAM アイデンティティセンターを有効にします。

   手順については、「AWS IAM アイデンティティセンター ユーザーガイド」の「AWS IAM アイデンティティセンターの有効化」を参照してください。

2. IAM アイデンティティセンターで、ユーザーに管理アクセスを付与します。

   を ID ソース IAM アイデンティティセンターディレクトリ として使用する方法のチュートリアルについては、AWS IAM アイデンティティセンター 「 ユーザーガイド」の「デフォルトを使用してユーザーアクセスを設定する IAM アイデンティティセンターディレクトリ」を参照してください。

管理アクセス権を持つユーザーとしてサインインする

- IAM アイデンティティセンターのユーザーとしてサインインするには、IAM アイデンティティセンターのユーザーの作成時に E メールアドレスに送信されたサインイン URL を使用します。

  IAM Identity Center ユーザーを使用してサインインする方法については、AWS サインイン 「 ユーザーガイド」の AWS 「 アクセスポータルにサインインする」を参照してください。

追加のユーザーにアクセス権を割り当てる

1. IAM アイデンティティセンターで、最小特権のアクセス許可を適用するというベストプラクティスに従ったアクセス許可セットを作成します。

   手順については、「AWS IAM アイデンティティセンター ユーザーガイド」の「アクセス許可セットを作成する」を参照してください。

2. グループにユーザーを割り当て、そのグループにシングルサインオンアクセス権を割り当てます。

   手順については、「AWS IAM アイデンティティセンター ユーザーガイド」の「グループの追加」を参照してください。

コンピューティング環境およびコンテナインスタンスのIAM ロールの作成

AWS Batch のコンピューティング環境およびコンテナインスタンスでは、他の AWS API を呼び出すために AWS アカウント 認証情報が必要です。AWS Identity and Access Management ロールを作成してこれらの認証情報をコンピューティング環境およびコンテナインスタンスに提供し、そのロールをコンピューティング環境に関連付けます。

> **Note**
>
> AWS アカウント に必要なアクセス許可があることを確認するには、「Initial IAM service set up for your account」を参照してください。
> AWS Batch コンピューティング環境およびコンテナインスタンスのロールは、コンソールの初回実行時に自動的に作成されます。したがって、AWS Batch コンソールを使用する場合は、次のセクションに進むことができます。代わりに AWS CLI を使用する場合は、最初のコンピューティング環境を作成する前に AWS Batchのサービスにリンクされたロールの使用、Amazon ECS インスタンスロール、チュートリアル: IAM 実行ロールを作成する の手順を実行してください。

インスタンスのキーペアを作成する

AWS では公開キー暗号化を使用して、お客様のインスタンスのログイン情報の安全性を保護します。AWS Batch コンピューティング環境のコンテナインスタンスなどの Linux インスタンスでは、SSH アクセスにパスワードを使用しません。キーペアを使用してインスタンスに安全にログインします。コンピューティング環境の作成時にキーペアの名前を指定し、SSH を使ってログインするときにプライベートキーを指定します。

キーペアを既に作成していない場合は、Amazon EC2 コンソールを使用して作成できます。複数の AWS リージョン でインスタンスを起動する予定がある場合は、各リージョンでキーペアを作成します。リージョンの詳細については、「Amazon EC2 ユーザーガイド」の「リージョンとゾーン」を参照してください。

キーペアを作成するには

1. Amazon EC2 コンソールの https://console.aws.amazon.com/ec2/ を開いてください。

2. ナビゲーションバーで、キーペアを生成する AWS リージョン を選択します。使用可能なリージョンは場所に関係なく選択できますが、キーペアはリージョンに固有のものです。例えば、米国西部 (オレゴン) リージョンでインスタンスを起動する予定の場合は、その同じリージョン内のインスタンス用にキーペアを作成します。
3. ナビゲーションペインで キーペア] を選択し、キーペアの作成] を選択します。
4. キーペア作成 ダイアログボックスの キーペア名 フィールドで、新しいキーペアの名前を入力し、作成 を選択します。自分のユーザー名の後に、-key-pair とリージョン名を続けたものなど、覚えやすいものにしましょう。例えば、me-key-pair-uswest2 などです。
5. ブラウザによって秘密キーファイルが自動的にダウンロードされます。ベースファイル名はキーペア名として指定した名前であり、ファイル名の拡張子は .pem です。ダウンロードしたプライベートキーのファイルを安全な場所に保存します。

   > **Important**
   >
   > プライベートキーのファイルを保存できるのはこのタイミングだけです。インスタンスの起動時に鍵ペアの名前を指定し、インスタンスに接続するたびに対応する秘密鍵を指定する必要があります。

6. Mac または Linux コンピュータの SSH クライアントを使用して Linux インスタンスに接続する場合は、次のコマンドを使用してプライベートキーファイルの権限を設定します。これにより、お客様だけが読むことができます。

   ```
   $ chmod 400 your_user_name-key-pair-region_name.pem
   ```

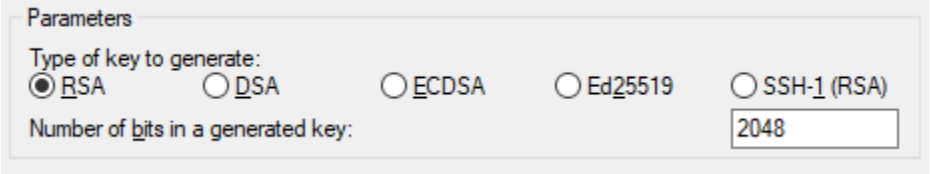
詳細については「Amazon EC2 ユーザーガイド」の「Amazon EC2 キーペア」を参照してください。

キーペアを使用してインスタンスに接続するには

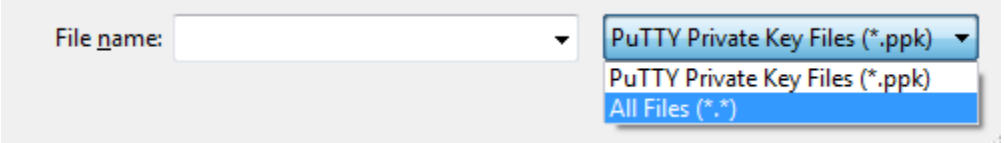
Mac または Linux を実行しているコンピュータから Linux インスタンスに接続するには、.pem オプションとプライベートキーへのパスを指定して、SSH クライアントに対する -i ファイルを指定します。Windows を実行しているコンピュータから Linux インスタンスに接続する場合は、MindTerm または PuTTY のどちらかを使用します。PuTTY を使用する場合は、PuTTY をインストールしてから、次の手順に従って .pem ファイルを .ppk ファイルに変換します。

(オプション) PuTTY を使用して Windows から Linux インスタンスに接続するには

1. http://www.chiark.greenend.org.uk/~sgtatham/putty/ から PuTTY をダウンロードしてインストールします。必ずスイート全体をインストールします。
2. PuTTYgen を起動します (例: Start] (スタート) メニューで All Programs] (すべてのプログラム)、PuTTY]、PuTTYgen] の順に選択します)。
3. Type of key to generate] (生成するキーのタイプ) で、RSA] を選択します。以前のバージョンの PuTTYgen を使用している場合は、SSH-2 RSA] を選択します。



4. ロード] を選択します。PuTTYgen では、デフォルトでは .ppk 拡張子を持つファイルだけが表示されます。.pem ファイルの場所を特定するには、すべてのタイプのファイルを表示するオプションを選択します。



5. 前の手順で作成したプライベートキーファイルを選択してから、開く を選択します。OK] を選択して、確認ダイアログボックスを閉じます。
6. [Save private key] (プライベートキーの保存) を選択します。PuTTYgen に、パスフレーズなしでキーを保存することに関する警告が表示されます。Yes] を選択します。
7. キーペアに使用した名前と同じ名前をキーに指定します。PuTTY によって、.ppk ファイルに拡張子が自動的に追加されます。

VPC を作成する

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。コンテナインスタンスは、VPC で起動することを強くお勧めします。

デフォルトの VPC がある場合は、このセクションもスキップして、次のタスク セキュリティグループの作成 に移動できます。デフォルトの VPC があるかどうかを判断するには、「Amazon EC2 ユーザーガイド」の「Supported Platforms in the Amazon EC2 Console」を参照してください。

Amazon VPC の作成方法については、「Amazon VPC ユーザーガイド」の「VPC のみを作成する」を参照してください。次の表を参考にして、選択するオプションを決定します。

オプション	値
作成するためのリソース	VPC 専用
名前	オプションで、VPC の名前を指定します。
IPv4 CIDR ブロック	IPv4 CIDR 手動入力 CIDR ブロックサイズは /16 から /28 の間である必要があります。
IPv6 CIDR ブロック	IPv6 CIDR ブロックなし
テナンシー	デフォルト

Amazon VPC の詳細については、Amazon VPC ユーザーガイドのAmazon VPC とはを参照してください。

セキュリティグループの作成

セキュリティグループは、関連付けられたコンピューティング環境コンテナインスタンスのファイアウォールとして動作し、インバウンドトラフィックとアウトバウンドトラフィックの両方をコンテナインスタンスレベルで制御します。セキュリティグループは、それが対象としている VPC 内でのみ使用が可能です。

SSH を使用して IP アドレスからコンテナインスタンスに接続するためのルールをセキュリティグループに追加できます。さらに、任意の場所からのインバウンドおよびアウトバウンドの HTTP アクセスおよび HTTPS アクセスを可能にするルールを追加できます。タスクで使用するポートを開くためのルールを追加します。

複数のリージョンでコンテナインスタンスを起動する予定がある場合は、各リージョンでセキュリティグループを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「リージョンとアベイラビリティゾーン」を参照してください。

> **Note**
>
> ローカルコンピュータのパブリック IP アドレスが必要になります。このアドレスはサービスを使って取得できます。例えば、次のサービスが提供されています。http://checkip.amazonaws.com/ または https://checkip.amazonaws.com/。IP アドレスを提供する別のサービスを検索するには、検索フレーズwhat is my IP addressを使用します。インターネットサービスプロバイダー (ISP) 経由で、またはファイアウォールの背後から静的 IP アドレスなしで接続している場合は、クライアントコンピュータで使用されている IP アドレスの範囲を調べる必要があります。

コンソールを使用してセキュリティグループを作成するには

1. Amazon VPC コンソール (https://console.aws.amazon.com/vpc/) を開きます。
2. ナビゲーションペインで、セキュリティグループ] を選択します。
3. セキュリティグループの作成 を選択します。
4. セキュリティグループの名前と説明を入力します。セキュリティグループの作成後に名前と説明を変更することはできません。
5. VPC で、VPC を選択します。
6. (オプション) デフォルトでは、新しいセキュリティグループにはすべてのトラフィックがリソースを離れることを許可するアウトバウンドルールのみが設定されています。任意のインバウンドトラフィックを許可するには、またはアウトバウンドトラフィックを制限するには、ルールを追加する必要があります。

   AWS Batch コンテナインスタンスでは、インバウンドポートが開いている必要はありません。ただし、SSH ルールを追加することもできます。これにより、コンテナインスタンスにログインして Docker コマンドでジョブのコンテナを確認できるようになります。また、ウェブサーバーを実行するジョブをコンテナインスタンスでホストする場合は、HTTP ルールを追加できます。これらのオプションのセキュリティグループルールを追加するには、以下のステップを実行します。

   [インバウンド] タブで以下のルールを作成し、作成] を選択します。

   - [ルールの追加] を選択します。[タイプ] で HTTP] を選択します。[ソース] では、[任意の場所] (`0.0.0.0/0`) を選択します。
   - [ルールの追加] を選択します。[タイプ] で SSH] を選択します。ソース で、カスタム IP を選択し、コンピュータまたはネットワークのパブリック IP アドレスを Classless Inter-Domain

Routing (CIDR) 表記で指定します。会社が特定の範囲からアドレスを割り当てている場合、203.0.113.0/24 などの範囲全体を指定します。CIDR 表記で個々の IP アドレスを指定するには、My IP を選択します。これにより、パブリック IP アドレスに /32 ルーティングプレフィックスが追加されます。

> **Note**
>
> セキュリティ上の理由で、すべての IP アドレス (0.0.0.0/0) からインスタンスへの SSH アクセスを許可することはお勧めしません。ただし、それがテスト目的で短期間の場合は例外です。

7. タグはここで追加することも、後で追加することもできます。タグを追加するには、新しいタグを追加 をクリックし、タグのキーと値を入力します。
8. セキュリティグループの作成 を選択します。

コマンドラインを使用してセキュリティグループを作成するには、「>create-security-group (AWS CLI)」を参照してください。

セキュリティグループの詳細については、セキュリティグループの操作を参照してください。

AWS CLI のインストール

AWS Batch で AWS CLI を使用するには、最新バージョンの AWS CLI をインストールします。AWS CLI のインストールまたは最新バージョンへの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 コマンドラインインターフェイスのインストールを参照してください。

AWS Batch チュートリアルの開始方法

AWS Batch 初回実行ウィザードを使用すると、すぐに使用を開始できます AWS Batch。前提条件を確認したら、初回実行ウィザードを使用してコンピューティング環境、ジョブ定義、およびジョブキューを作成できます。

AWS Batch 初回実行ウィザードを使用してサンプル「Hello World」ジョブを送信し、設定をテストすることもできます。起動する Docker イメージが既にある場合は AWS Batch、そのイメージを使用してジョブ定義を作成できます。

その後、 AWS Batch 初回実行ウィザードを使用してコンピューティング環境、ジョブキューを作成し、サンプルの Hello World ジョブを送信できます。

ウィザードを使用した Amazon EC2 オーケストレーションの開始方法

Amazon Elastic Compute Cloud (Amazon EC2) は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。Amazon EC2 の使用により、ハードウェアに事前投資する必要がなくなり、アプリケーションをより速く開発およびデプロイできます。

Amazon EC2 を使用すると、必要な数 (またはそれ以下) の仮想サーバーの起動、セキュリティおよびネットワーキングの構成、ストレージの管理ができます。Amazon EC2 は、要件変更や需要増に応じてスケールアップまたはスケールダウンできるため、トラフィック予測の必要性を軽減できます。

概要

このチュートリアルでは、ウィザードで AWS Batch をセットアップして Amazon EC2 を設定し、`Hello World` を実行する方法について説明します。

対象者

このチュートリアルは、AWS Batch のセットアップ、テスト、デプロイを担当するシステム管理者とデベロッパーを対象としています。

使用される機能

このチュートリアルでは、AWS Batch コンソールウィザードを使用して以下を行う方法について説明します。

- Amazon EC2 コンピューティング環境を作成して設定する
- ジョブキューの作成。
- ジョブ定義の作成
- ジョブを作成して実行する
- CloudWatch でジョブ出力を表示する

所要時間

このチュートリアルは完了までに約 10 ~ 15 分かかります。

リージョン別制限

このソリューションの使用に関連する国やリージョン別の制限はありません。

リソース使用量のコスト

AWS アカウントを作成するための料金はかかりません。ただし、このソリューションを実装することにより、次の表に記載されるコストの一部またはすべてが発生する可能性があります。

説明	コスト (USD)
Amazon EC2 インスタンス	作成された各 Amazon EC2 インスタンスに対して料金が発生します。料金の詳細については、「Amazon EC2 料金表」を参照してください。

前提条件

開始する前に、以下を確認してください。

- まだ作成していない場合は、AWS アカウント を作成します。
- `ecsInstanceRole` インスタンスロールを作成する

ステップ 1: コンピューティング環境を作成する

> **Important**
>
> 可能な限りシンプルかつ迅速に使用を開始するため、このチュートリアルでは、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。本番での使用に向けて作成する前に、すべての設定内容に習熟した上で、要件を満たす設定でデプロイすることをお勧めします。

Amazon EC2 オーケストレーション用のコンピューティング環境を作成するには、以下の操作を実行します。

1. AWS Batch コンソールの初回実行ウィザード を開きます。
2. [ジョブとオーケストレーションタイプの設定] には、[Amazon Elastic Compute Cloud (Amazon EC2)] を選択します。
3. [次へ] を選択します。
4. コンピューティング環境設定 の 名前 で、コンピューティング環境の一意な名前を指定します。名前の最大長は 128 文字です。大文字、小文字、数字、ハイフン (-)、アンダースコア (_) を含めることができます。
5. [インスタンスロール] では、必要な IAM アクセス許可がアタッチされた既存のインスタンスロールを選択します。このインスタンスロールを使用すると、コンピューティング環境にある Amazon ECS コンテナインスタンスにより、必要な AWS API オペレーションを呼び出すことができます。詳細については、「Amazon ECS インスタンスロール」を参照してください。

   [インスタンスロール] のデフォルト名は `ecsInstanceRole` です。
6. [インスタンス設定] では、デフォルト設定のままにすることができます。
7. [ネットワーク設定] では、AWS リージョン にデフォルトの VPC を使用します。
8. [次へ] を選択します。

ステップ 2: ジョブキューを作成する

ジョブキューに送信したジョブは、AWS Batch スケジューラによってコンピューティング環境内のコンピューティングリソースで実行されるまで、ジョブキューに格納されます。詳細については、ジョブキューを参照してください。

Amazon EC2 オーケストレーション用のジョブキューを作成するには、以下の操作を実行します。

1. [名前] の [ジョブキュー設定] で、ジョブキューの一意の名前を指定します。名前の最大長は 128 文字です。大文字、小文字、数字、ハイフン (-)、アンダースコア (_) を含めることができます。
2. 他のすべての設定オプションでは、デフォルト値のままにすることができます。
3. [次へ] を選択します。

ステップ 3: ジョブ定義を作成する

AWS Batch のジョブ定義は、ジョブの実行方法を指定します。各ジョブはジョブ定義を参照しなければならないが、ジョブ定義で指定されたパラメーターの多くは実行時に上書きできます。

ジョブ定義を作成するには

1. [ジョブ定義の作成] では以下を行います。
   a. [名前] に、一意のジョブキュー名を入力します。名前の最大長は 128 文字です。大文字、小文字、数字、ハイフン (-)、アンダースコア (_) を含めることができます。
   b. [コマンド - オプション] で、`hello world` をカスタムメッセージに変更することも、そのままにすることもできます。
2. 他のすべての設定オプションでは、デフォルト値のままにすることができます。
3. [次へ] を選択します。

ステップ 4: ジョブを作成する

ジョブを作成するには、以下の手順を実行します。

1. ジョブの設定 セクションの 名前 で、ジョブの一意の名前を指定します。名前の最大長は 128 文字です。大文字、小文字、数字、ハイフン (-)、アンダースコア (_) を含めることができます。
2. 他のすべての設定オプションでは、デフォルト値のままにすることができます。
3. [次へ] を選択します。

ステップ 5: 確認して作成する

レビューと作成では、設定手順を確認してください。変更する必要がある場合は、編集 を選択します。完了したら、リソースを作成 を選択します。

1. [レビューと作成] で、[リソースの作成]　を選択します。

2. AWS Batch がリソースの割り当てを開始すると、ウィンドウが開きます。完了したら、[ダッシュボードに移動] を選択します。ダッシュボードには、割り当てられたすべてのリソースと、ジョブが `Runnable` 状態になっていることが表示されます。ジョブの実行がスケジュールされ、2～3 分で完了します。

ステップ 6: ジョブの出力を表示する

ジョブの出力を表示するには、以下を実行します。

1. ナビゲーションペインで [ジョブ] を選択します。
2. [ジョブキュー] ドロップダウンで、チュートリアル用に作成したジョブキューを選択します。
3. [ジョブ] テーブルには、すべてのジョブとその現在のステータスが一覧表示されます。ジョブの [ステータス] が [成功] であれば、ジョブの [名前] を選択してジョブの詳細を表示します。
4. [詳細] ペインで、ログストリームの名前を選択します。ジョブの CloudWatch コンソールが開き、`hello world` の [メッセージ] またはカスタムメッセージを含むイベントが 1 つあるはずです。

ステップ 7: チュートリアルのリソースをクリーンアップする

Amazon EC2 インスタンスが有効になっている間は課金されます。インスタンスを削除して、料金の発生を停止できます。

作成したリソースを削除するには、次の作業を行います。

1. ナビゲーションペインで [ジョブキュー] を選択します。
2. [ジョブキュー] テーブルで、チュートリアル用に作成したジョブキューを選択します。
3. [無効化] を選択します。ジョブキューの [状態] が無効になったら、[削除] を選択できます。
4. ジョブキューが削除されたら、ナビゲーションペインで [コンピューティング環境] を選択します。
5. このチュートリアル用に作成したコンピューティング環境を選択し、[無効化] を選択します。コンピューティング環境が無効になるまでに 1～2 分かかる場合があります。
6. コンピューティング環境の [状態] が無効になったら、[削除] を選択します。コンピューティング環境が削除されるまでに 1～2 分かかる場合があります。

その他のリソース

チュートリアルが完了したら、次のトピックを試すことができます。

- AWS Batch コアコンポーネントについて説明します。詳細については、「AWS Batch のコンポーネント」を参照してください。
- AWS Batch で使用できるさまざまなコンピューティング環境について説明します。
- [ジョブキュー] とそのさまざまなスケジューリングオプションについて詳しく説明します。
- [ジョブ定義] とそのさまざまな設定オプションについて詳しく説明します。
- さまざまなタイプの [ジョブ] について詳しく説明します。

ウィザードを使用した AWS Batch と Fargate オーケストレーションの開始方法

AWS Fargate は、コンテナに指定したリソース要件に厳密に一致するようにコンピューティングを起動してスケーリングします。Fargate を使用すると、追加のサーバーに対してオーバープロビジョニングまたは料金を支払う必要はありません。詳細については、Fargate を参照してください。

概要:

このチュートリアルでは、ウィザード AWS Batch で をセットアップして AWS Fargate を設定し、を実行する方法を示しますHello World。

対象者

このチュートリアルは、 AWS Batchのセットアップ、テスト、デプロイを担当するシステム管理者とデベロッパーを対象としています。

使用される機能

このチュートリアルでは、 AWS Batch コンソールウィザードを使用して以下を行う方法を示します。

- AWS Fargate コンピューティング環境を作成して設定する
- ジョブキューの作成。
- ジョブ定義の作成
- ジョブを作成して実行する
- CloudWatch でジョブ出力を表示する

所要時間

このチュートリアルの所要時間は約 10～15 分です。

リージョンの制限

このソリューションの使用に関連する国やリージョンの制限はありません。

リソースの使用コスト

AWS アカウントの作成には料金はかかりません。ただし、このソリューションを実装することにより、次の表に記載されるコストの一部またはすべてが発生する可能性があります。

説明	コスト (USD)
料金は、タスクまたはポッドのリクエストされた vCPU、メモリ、オペレーティングシステム、CPU アーキテクチャ、ストレージリソースに基づいています。	料金の詳細については、「Fargate の料金表」を参照してください。

前提条件

開始する前に:

- AWS アカウント がない場合は、 を作成します。
- タスク実行ロールを作成します。タスク実行ロールをまだ作成していない場合は、このチュートリアルの一部として作成できます。

ステップ 1: コンピューティング環境を作成する

> **Important**
>
> 可能な限りシンプルかつ迅速に使用を開始するため、このチュートリアルでは、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。本番での使用に向けて作成する前に、すべての設定内容に習熟した上で、要件を満たす設定でデプロイすることをお勧めします。

Fargate オーケストレーション用のコンピューティング環境を作成するには、以下の操作を実行します。

1. AWS Batch コンソールの初回実行ウィザード を開きます。
2. [ジョブとオーケストレーションタイプを設定する] で、[Fargate] を選択します。
3. [次へ] を選択します。
4. コンピューティング環境設定 の 名前 で、コンピューティング環境の一意な名前を指定します。名前の最大長は 128 文字です。大文字、小文字、数字、ハイフン (-)、アンダースコア (_) を含めることができます。
5. 他のすべての設定オプションでは、デフォルト値のままにすることができます。
6. [次へ] を選択します。

ステップ 2: ジョブキューを作成する

ジョブキューは、ス AWS Batch ケジューラがコンピューティング環境のリソースでジョブを実行するまで、送信されたジョブを保存します。ジョブキューを作成するには:

Fargate オーケストレーションのジョブキューを作成するには、以下の操作を実行します。

1. ジョブキュー設定 セクションの 名前 で、コンピューティング環境の一意の名前を指定します。名前の最大長は 128 文字です。大文字、小文字、数字、ハイフン (-)、アンダースコア (_) を含めることができます。
2. [優先度] には、このジョブキュー向けに 900 と入力します。
3. 他のすべての設定オプションでは、デフォルト値のままにすることができます。
4. [次へ] を選択します。

ステップ 3: ジョブ定義を作成する

ジョブ定義を作成するには

1. 一般設定 セクションで:
   - 一般設定 セクションの 名前 で、コンピューティング環境に固有の名前を指定します。名前の最大長は 128 文字です。大文字、小文字、数字、ハイフン (-)、アンダースコア (_) を含めることができます。
2. Fargate プラットフォーム設定 セクションで:

a. [パブリック IP アドレスの割り当て] をオンにして、パブリック IP アドレスを割り当てます。プライベートイメージリポジトリをセットアップしない限り、コンテナイメージをダウンロードするにはパブリック IP が必要です。

b. 実行ロール で、Amazon Elastic Container Service (Amazon ECS) エージェントにユーザーに代わって AWS 呼び出しを許可するタスク実行ロールを選択します。[ecsTaskExecutionRole] または [BatchEcsTaskExecutionRole] を選択します。

   [実行ロール] を作成するには、[新しいロールを作成] を選択します。[IAM ロールを作成] モーダルで、[IAM ロールを作成] を選択します。

   i. IAM コンソールには、実行ロールを作成するためのアクセス許可設定が既に設定されています。

   ii. [信頼されたエンティティタイプ] で、[AWS サービス] が選択されていることを確認します。

   iii. [サービスまたはユーザーケース] で、[Elastic Container Service] が選択されていることを確認します。

   iv. [次へ] を選択します。

   v. [アクセス許可ポリシー] セクションで、[AmazonECSTaskExecutionRolePolicy] ポリシーが選択されていることを確認します。

   vi. [次へ] を選択します。

   vii. [名前、レビュー、作成] で、ロール名が [BatchEcsTaskExecutionRole] であることを確認します。

   viii. [ロールの作成] を選択してください。

   ix. AWS Batch コンソールで、実行ロールの横にある更新ボタンを選択します。[BatchEcsTaskExecutionRole] 実行ロールを選択します。

3. コンテナの設定 セクションで次の操作を行います。

   - [コマンド] では、`hello world` をカスタムメッセージに変更することも、そのままにすることもできます。

4. 他のすべての設定オプションでは、デフォルト値のままにすることができます。

5. [次へ] を選択します。

ステップ 4: ジョブを作成する

Fargate ジョブを作成するには、以下の手順を実行します。

1. ジョブの設定 セクションの 名前 で、ジョブの一意の名前を指定します。名前の最大長は 128 文字です。大文字、小文字、数字、ハイフン (-)、アンダースコア (_) を含めることができます。
2. 他のすべての設定オプションでは、デフォルト値のままにすることができます。
3. [次へ] を選択します。

ステップ 5: 確認して作成する

レビューと作成では、設定手順を確認してください。変更する必要がある場合は、編集 を選択します。完了したら、リソースを作成 を選択します。

ステップ 6: ジョブの出力を表示する

ジョブの出力を表示するには、以下を実行します。

1. ナビゲーションペインで [ジョブ] を選択します。
2. [ジョブキュー] ドロップダウンで、チュートリアル用に作成したジョブキューを選択します。
3. [ジョブ] テーブルには、すべてのジョブとその現在のステータスが一覧表示されます。ジョブの [ステータス] が [成功] であれば、ジョブの [名前] を選択してジョブの詳細を表示します。
4. [詳細] ペインで、ログストリームの名前を選択します。ジョブの CloudWatch コンソールが開き、`hello world` の [メッセージ] またはカスタムメッセージを含むイベントが 1 つあるはずです。

ステップ 7: チュートリアルのリソースをクリーンアップする

Amazon EC2 インスタンスが有効になっている間は課金されます。インスタンスを削除して、料金の発生を停止できます。

作成したリソースを削除するには、次の作業を行います。

1. ナビゲーションペインで [ジョブキュー] を選択します。
2. [ジョブキュー] テーブルで、チュートリアル用に作成したジョブキューを選択します。
3. [Disable] (無効化) を選択します。ジョブキューの [状態] が無効になったら、[削除] を選択できます。

4. ジョブキューが削除されたら、ナビゲーションペインで [コンピューティング環境] を選択します。
5. このチュートリアル用に作成したコンピューティング環境を選択し、[無効化] を選択します。コンピューティング環境が無効になるまでに 1～2 分かかる場合があります。
6. コンピューティング環境の [状態] が無効になったら、[削除] を選択します。コンピューティング環境が削除されるまでに 1～2 分かかる場合があります。

その他のリソース

チュートリアルが完了したら、次のトピックを試すことができます。

- ベストプラクティスについて詳しく説明します。
- AWS Batch コアコンポーネントについて説明します。詳細については、「AWS Batch のコンポーネント」を参照してください。
- AWS Batchで使用できるさまざまなコンピューティング環境について説明します。
- [ジョブキュー] とそのさまざまなスケジューリングオプションについて詳しく説明します。
- [ジョブ定義] とそのさまざまな設定オプションについて詳しく説明します。
- さまざまなタイプの [ジョブ] について詳しく説明します。

を使用した AWS Batch と Fargate の開始方法 AWS CLI

このチュートリアルでは、 AWS Fargate オーケストレーション AWS Batch を使用して をセットアップし、 AWS Command Line Interface () を使用して単純な「Hello World」ジョブを実行する方法を示しますAWS CLI。コンピューティング環境、ジョブキュー、ジョブ定義を作成し、 AWS Batchにジョブを送信する方法を学びます。

トピック

- 前提条件
- IAM 実行ロールを作成する
- コンピューティング環境を作成する
- ジョブキューを作成する
- ジョブ定義の作成
- ジョブを送信し、モニタリングする
- ジョブ出力を表示する

- リソースをクリーンアップする
- 本番環境への移行
- 次の手順

前提条件

このチュートリアルを開始する前に、以下の要件を満たしていることを確認してください。

1. AWS CLI。インストールする必要がある場合は、「AWS CLI の最新バージョンのインストールまたは更新」に従ってください。を含む を使用 AWS CloudShellすることもできます AWS CLI。
2. 適切な認証情報 AWS CLI を使用して を設定しました。認証情報をまだ設定していない場合は、`aws configure` を実行します。
3. コマンドラインインターフェイスとコンテナ化の概念に関する基本的な知識。
4. が IAM と AWS Batch 連携する方法 を使用して、 の AWS Batch リソース、IAM ロール、VPC リソースを作成および管理します AWS アカウント。
5. の VPC からのサブネット ID とセキュリティグループ ID AWS アカウント。VPC がない場合は作成できます。を使用してこれらのリソース ID AWS CLI を取得する方法の詳細については、AWS CLI 「 コマンドリファレンス」の「describe-subnets」とdescribe-security-groups」を参照してください。 IDs

所要時間: このチュートリアルの所要時間は約 15～20 分です。

コスト: このチュートリアルでは、Fargate コンピューティングリソースを使用します。このチュートリアルの完了にかかる推定コストは、クリーンアップの指示に従って完了直後にリソースを削除すると想定した場合、0.01 USD 未満です。Fargate の料金は、消費された vCPU リソースとメモリリソースに基づいており、1 秒あたりで課金されます。最低課金時間は 1 分です。最新の料金については、「AWS Fargate の料金」を参照してください。

IAM 実行ロールを作成する

AWS Batch には、Amazon Elastic Container Service (Amazon ECS) エージェントがユーザーに代わって AWS API コールを実行できるようにする実行ロールが必要です。このロールは、Fargate タスクがコンテナイメージをプルし、Amazon CloudWatch にログを書き込む上で必要です。

信頼ポリシードキュメントを作成する

最初に、Amazon ECS タスクがロールを引き受けることを許可する信頼ポリシーを作成します。

```
cat > batch-execution-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ecs-tasks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

実行ロールを作成する

前のステップで作成した信頼ポリシーを使用して、BatchEcsTaskExecutionRoleTutorial という名前の IAM ロールを作成するには、次のコマンドを使用します。

```
aws iam create-role \
    --role-name BatchEcsTaskExecutionRoleTutorial \
    --assume-role-policy-document file://batch-execution-role-trust-policy.json
```

必要なポリシーをアタッチする

Amazon ECS タスク実行に必要なアクセス許可を提供する AWS マネージドポリシーをアタッチします。

```
aws iam attach-role-policy \
    --role-name BatchEcsTaskExecutionRoleTutorial \
    --policy-arn arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy
```

これで、 AWS Batch Fargate タスク実行のために がロールを使用する準備ができました。

コンピューティング環境を作成する

コンピューティング環境は、バッチジョブを実行するコンピューティングリソースを定義します。このチュートリアルでは、ジョブの要件に基づいてリソースを自動的にプロビジョニングおよびスケールするマネージド Fargate コンピューティング環境を作成します。

コンピューティング環境を作成する

次のコマンドを使用して Fargate コンピューティング環境を作成します。例のサブネット ID とセキュリティグループ ID を 前提条件 ごとに実際に使用する値に置き換えます。

```
aws batch create-compute-environment \
    --compute-environment-name my-fargate-compute-env \
    --type MANAGED \
    --state ENABLED \
    --compute-resources type=FARGATE,maxvCpus=128,subnets=subnet-
a123456b,securityGroupIds=sg-a12b3456
```

コマンドが正常に実行されたときの出力の様子を以下に示します。

```
{
    "computeEnvironmentName": "my-fargate-compute-env",
    "computeEnvironmentArn": "arn:aws:batch:us-west-2:123456789012:compute-environment/
my-fargate-compute-env"
}
```

コンピューティング環境の準備が整うまで待ちます。

先に進む前に、コンピューティング環境のステータスをチェックして、準備が整っていることを確認します。

```
aws batch describe-compute-environments \
    --compute-environments my-fargate-compute-env \
    --query 'computeEnvironments[0].status'
```

```
"VALID"
```

ステータスが VALID と表示されると、コンピューティング環境はジョブを受け入れる準備が整っています。

ジョブキューを作成する

ジョブキューは、ス AWS Batch ケジューラがコンピューティング環境のリソースで実行するまで、送信されたジョブを保存します。ジョブはキュー内の優先順位に従って処理されます。

ジョブキューを作成する

次のコマンドは、Fargate コンピューティング環境を使用する優先度 900 のジョブキューを作成します。

```
aws batch create-job-queue \
    --job-queue-name my-fargate-job-queue \
    --state ENABLED \
    --priority 900 \
    --compute-environment-order order=1,computeEnvironment=my-fargate-compute-env
```

コマンドが正常に実行されたときの出力の様子を以下に示します。

```
{
    "jobQueueName": "my-fargate-job-queue",
    "jobQueueArn": "arn:aws:batch:us-west-2:123456789012:job-queue/my-fargate-job-
queue"
}
```

ジョブキューの準備が整っていることを確認します

ジョブキューが ENABLED 状態であり、ジョブを受け入れる準備が整っていることを確認します。

```
aws batch describe-job-queues \
    --job-queues my-fargate-job-queue \
    --query 'jobQueues[0].state' "ENABLED"
```

ジョブ定義の作成

ジョブ定義は、使用する Docker イメージ、リソース要件、その他のパラメータなど、ジョブの実行方法を指定します。Fargate では、従来の vCPU およびメモリパラメータの代わりにリソース要件を使用します。

ジョブ定義を作成する

次のコマンドは、ビジーボックスコンテナイメージを使用して単純な「hello world」コマンドを実行するジョブ定義を作成します。を実際の AWS アカウント ID 123456789012に置き換え、例を独自の ID AWS リージョン に置き換えます。

```
aws batch register-job-definition \
    --job-definition-name my-fargate-job-def \
    --type container \
```

```
    --platform-capabilities FARGATE \
    --container-properties '{
        "image": "busybox",
        "resourceRequirements": [
            {"type": "VCPU", "value": "0.25"},
            {"type": "MEMORY", "value": "512"}
        ],
        "command": ["echo", "hello world"],
        "networkConfiguration": {
            "assignPublicIp": "ENABLED"
        },
        "executionRoleArn": "arn:aws:iam::123456789012:role/
BatchEcsTaskExecutionRoleTutorial"
    },
{
    "jobDefinitionName": "my-fargate-job-def",
    "jobDefinitionArn": "arn:aws:batch:us-west-2:123456789012:job-definition/my-
fargate-job-def:1",
    "revision": 1
}'
```

ジョブ定義は、Fargate タスクの最小リソースである 0.25 vCPU と 512 MB のメモリを指定します。assignPublicIp の設定が有効になっているため、コンテナは Docker Hub からビジーボックスイメージをプルできます。

ジョブを送信し、モニタリングする

必要なコンポーネントがすべて揃ったので、ジョブをキューに送信し、その進行状況をモニタリングできます。

ジョブを送信する

次のコマンドは、作成したジョブ定義を使用してジョブをキューに送信します。

```
aws batch submit-job \
    --job-name my-hello-world-job \
    --job-queue my-fargate-job-queue \
    --job-definition my-fargate-job-def
```

コマンドが正常に実行されたときの出力の様子を以下に示します。

```
{
```

```
    "jobArn": "arn:aws:batch:us-west-2:123456789012:job/my-hello-world-job",
    "jobName": "my-hello-world-job",
    "jobId": "1509xmpl-4224-4da6-9ba9-1d1acc96431a"
}
```

レスポンスで返された jobId を書き留めておきます。これを使用してジョブの進行状況をモニタリングします。

ジョブステータスをモニタリングする

ジョブ ID を使用して、ジョブのステータスを確認します。ジョブは、SUBMITTED、PENDING、RUNNABLE、STARTING、RUNNING、および最後に SUCCEEDED または FAILED のいくつかの状態を進行します。

```
aws batch describe-jobs --jobs 1509xmpl-4224-4da6-9ba9-1d1acc96431a
```

コマンドが正常に実行されたときの出力の様子を以下に示します。

```
{
    "jobs": [
        {
            "jobArn": "arn:aws:batch:us-west-2:123456789012:job/my-hello-world-job",
            "jobName": "my-hello-world-job",
            "jobId": "1509xmpl-4224-4da6-9ba9-1d1acc96431a",
            "jobQueue": "arn:aws:batch:us-west-2:123456789012:job-queue/my-fargate-job-
queue",
            "status": "SUCCEEDED",
            "createdAt": 1705161908000,
            "jobDefinition": "arn:aws:batch:us-west-2:123456789012:job-definition/my-
fargate-job-def:1"
        }
    ]
}
```

ステータスが SUCCEEDED と表示されると、ジョブは正常に完了しています。

ジョブ出力を表示する

ジョブが完了したら、Amazon CloudWatch Logs でその出力を表示できます。

ログストリーム名を取得する

まず、ジョブの詳細からログストリーム名を取得します。ジョブ ID の例を実際に使用する値に置き換えます。

```
aws batch describe-jobs --jobs 1509xmpl-4224-4da6-9ba9-1d1acc96431a \
    --query 'jobs[0].attempts[0].containers[0].logStreamName' \
    --output text
```

```
my-fargate-job-def/default/1509xmpl-4224-4da6-9ba9-1d1acc96431a
```

ジョブのログを表示する

ログストリーム名を使用して、CloudWatch Logs からジョブの出力を取得します。

```
aws logs get-log-events \
    --log-group-name /aws/batch/job \
    --log-stream-name my-fargate-job-def/default/1509xmpl-4224-4da6-9ba9-1d1acc96431a \
    --query 'events[*].message' \
    --output text
```

出力には「hello world」と表示され、ジョブが正常に実行されたことを確認できます。

リソースをクリーンアップする

このチュートリアルで作成したリソースは、継続的な料金の発生を回避するためにクリーンアップしてください。依存関係の理由により、正しい順序でリソースを削除する必要があります。

ジョブキューを無効化して削除する

まず、ジョブキューを無効にして削除します。

```
aws batch update-job-queue \
    --job-queue my-fargate-job-queue \
    --state DISABLED
```

```
aws batch delete-job-queue \
    --job-queue my-fargate-job-queue
```

コンピューティング環境を無効化して削除する

ジョブキューが削除されたら、コンピューティング環境を無効化して削除します。

```
aws batch update-compute-environment \
    --compute-environment my-fargate-compute-env \
    --state DISABLED
```

```
aws batch delete-compute-environment \
    --compute-environment my-fargate-compute-env
```

IAM ロールをクリーンアップする

ポリシーアタッチメントを削除し、IAM ロールを削除します。

```
aws iam detach-role-policy \
    --role-name BatchEcsTaskExecutionRoleTutorial \
    --policy-arn arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy
```

```
aws iam delete-role \
    --role-name BatchEcsTaskExecutionRoleTutorial
```

一時ファイルを削除する

作成した信頼ポリシーファイルを削除します。

```
rm batch-execution-role-trust-policy.json
```

すべてのリソースが正常にクリーンアップされました。

本番環境への移行

このチュートリアルは、 が Fargate と AWS Batch どのように連携するかを理解するのに役立つように設計されています。本番デプロイでは、次の追加要件を考慮してください。

セキュリティに関する考慮事項:

- デフォルトのセキュリティグループを使用する代わりに、アクセスを必要最低限に抑えた専用セキュリティグループを作成する
- コンテナのパブリック IP 割り当ての代わりに NAT Gateway でプライベートサブネットを使用する

- パブリックリポジトリを使用する代わりに Amazon ECR にコンテナイメージを保存する
- インターネットトラフィックを回避するために AWS 、サービス通信用の VPC エンドポイントを実装する

アーキテクチャ上の考慮事項:

- 可用性を高めるため、複数のアベイラビリティーゾーンにデプロイする
- エラー処理のためにジョブの再試行戦略とデッドレターキューを実装する
- ワークロード管理の優先順位が異なる複数のジョブキューを使用する
- キューの深さとリソース使用率に基づいて自動スケーリングポリシーを設定する
- ジョブの失敗とリソース使用率のモニタリングとアラートを実装する

運用上の考慮事項:

- モニタリング用の CloudWatch ダッシュボードとアラームを設定する
- 適切なログ記録と監査証跡を実装する
- Infrastructure as Code AWS CDK に CloudFormation または を使用する
- バックアップとディザスタリカバリの手順を策定する

本番環境に対応したアーキテクチャに関する包括的なガイダンスについては、「AWS Well-Architected フレームワーク」と「AWS 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。

次の手順

このチュートリアルを完了したので、より高度な AWS Batch 機能を調べることができます。

- ジョブキュー – ジョブキューのスケジュールと優先度管理について説明します。
- ジョブ定義 – 環境変数、ボリューム、再試行戦略など、高度なジョブ定義設定について説明します。
- のコンピューティング環境 AWS Batch – さまざまなコンピューティング環境タイプとスケーリングオプションを理解できます。
- マルチノード並列ジョブ – 複数のコンピューティングノードにまたがるジョブを実行します。
- 配列ジョブ – 多数の類似ジョブを効率的に送信します。
- AWS Batch のベストプラクティス – 本番ワークロードの最適化手法について説明します。

Amazon EKS で AWS Batch のご利用開始にあたって

Amazon EKS における AWS Batch は、既存の Amazon EKS クラスターにバッチワークロードをスケジュールおよびスケーリングするためのマネージドサービスです。AWS Batch が、ユーザーに代わって Amazon EKS クラスターの作成、管理、ライフサイクル操作を行うことはありません。AWS Batch オーケストレーションは、AWS Batch が管理するノードのスケールアップとスケールダウンを行い、それらのノード上でポッドを実行します。

AWS Batch は、Amazon EKS クラスター内の AWS Batch コンピューティング環境に関連付けられていないノード、自動スケーリングノードグループ、またはポッドライフサイクルには影響しません。AWS Batch を効果的に動作させるには、そのサービスにリンクされたロールに、既存の Amazon EKS クラスター内における Kubernetes ロールベースのアクセス制御 (RBAC) 権限が必要になります。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。

AWS Batch には、ポッドを Kubernetes ジョブとしてスコープできる AWS Batch 名前空間が必要です。AWS Batch ポッドを他のクラスターワークロードから分離するために、専用の名前空間を使用することをお勧めします。

AWS Batch に RBAC アクセス権が付与され、名前空間が確立されたら、CreateComputeEnvironment API オペレーションを使用してその Amazon EKS クラスターを AWS Batch コンピューティング環境に関連付けることができます。ジョブキューをこの新しい Amazon EKS コンピューティング環境に関連付けることができます。AWS Batch ジョブは、SubmitJob API オペレーションを使用して Amazon EKS ジョブ定義に基づいてジョブキューに送信されます。次に、AWS Batch は AWS Batch マネージドノードを起動し、ジョブキューにあるジョブを Kubernetes ポッドとして AWS Batch コンピューティング環境に関連付けられた EKS クラスターに配置します。

以下のセクションでは、Amazon EKS で AWS Batch を設定する方法について説明します。

目次

概要

このチュートリアルでは、AWS CLI、`kubectl`、`eksctl` を使用して Amazon EKS で AWS Batch を設定する方法について説明します。

対象者

このチュートリアルは、AWS Batch のセットアップ、テスト、デプロイを担当するシステム管理者とデベロッパーを対象としています。

使用される機能

このチュートリアルでは、AWS CLI を使用して以下を行う方法について説明します。

- Amazon EKS コンピューティング環境を作成して設定する
- ジョブキューの作成。
- ジョブ定義の作成
- ジョブを作成して実行する
- オーバーライドを含むジョブを送信する

所要時間

このチュートリアルは完了までに約 30～40 分かかります。

リージョン別制限

このソリューションの使用に関連する国やリージョン別の制限はありません。

リソース使用量のコスト

AWS アカウントを作成するための料金はかかりません。ただし、このソリューションを実装することにより、次の表に記載されるコストの一部またはすべてが発生する可能性があります。

説明	コスト (USD)
クラスター時間ごとに課金されます	インスタンスによって異なります。「Amazon EKS の料金」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、AWS Batch と Amazon EKS リソースの両方を作成して管理する上で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。このガイドでは、バージョン 2.8.6 以降または 1.26.0 以降の使用を想定しています。詳細については、「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」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、AWS Command Line Interface ユーザーガイドの `aws configure` を使用したクイック設定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。このガイドでは、バージョン `1.23` 以降の使用を想定しています。詳細については、「Amazon EKS ユーザーガイド」の「kubectl のインストールまたは更新」を参照してください。
- **`eksctl`** – Amazon 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。このガイドでは、バージョン `0.115.0` 以降の使用を想定しています。詳細については、「Amazon EKS ユーザーガイド」の「eksctl のインストールまたは更新」を参照してください。
- IAM からの必要なアクセス許可 – ここで使用する IAM セキュリティプリンシパルには、Amazon EKS の IAM ロールおよびサービスにリンクされたロール、CloudFormation、ならびに VPC とその関連リソースを操作するための権限が必要となります。詳細については、「IAM ユーザーガイド」の「Actions, resources, and condition keys for Amazon Elastic Kubernetes Service」と「サービスにリンクされたロールの作成」を参照してください。このガイドのすべてのステップは 1 つのユーザーとして実行する必要があります。
- 権限 — CreateComputeEnvironment API オペレーションを呼び出して Amazon EKS リソースを使用するコンピューティング環境を作成するユーザーには、`eks:DescribeCluster` API オペレーションに対する権限が必要です。
- AWS アカウント 番号 – AWS アカウント ID を把握する必要があります。「AWS アカウント ID の表示」の指示に従います。

- (オプション) CloudWatch – (オプション) [オーバーライドを含むジョブを送信する] の詳細を確認するには、ログ記録を設定する必要があります。詳細については、「CloudWatch Logs を使用して Amazon EKS ジョブにおける AWS Batch をモニタリングする」を参照してください。

ステップ 1: AWS Batch の 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、このチュートリアルでは、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。本番での使用に向けて作成する前に、すべての設定内容に習熟した上で、要件を満たす設定でデプロイすることをお勧めします。

前提条件をインストールしたら、eksctl を使用してクラスターを作成する必要があります。クラスターの作成には 10 ~ 15 分かかる場合があります。

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

上記のコマンドでは:

- *my-cluster-name* は、使用したいクラスターの名前に置き換えます。
- *region-code* は AWS リージョン に置き換え、us-west-2 などでクラスターを作成します。

このチュートリアルの後半では、クラスター名とリージョンが必要です。

AWS Batch の Amazon EKS クラスターを準備する

すべての手順を実行する必要があります。

1. AWS Batch ジョブ専用の名前空間を作成する

   kubectl を使用して新しい名前空間を作成します。

   ```
   $ namespace=my-aws-batch-namespace
   ```

   ```
   $ cat - <<EOF | kubectl create -f -
   {
   ```

```
  "apiVersion": "v1",
  "kind": "Namespace",
  "metadata": {
    "name": "${namespace}",
    "labels": {
      "name": "${namespace}"
    }
  }
}
EOF
```

出力:

```
namespace/my-aws-batch-namespace created
```

2. ロールベースアクセス制御 (RBAC) を有効にします。

   kubectl を使用して、クラスターの Kubernetes ロールを作成すると、AWS Batch はノードとポッドを監視したり、ロールをバインドしたりできるようになります。これを EKS クラスターごとに 1 回実行する必要があります。

```
$ cat - <<EOF | kubectl apply -f -
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: aws-batch-cluster-role
rules:
  - apiGroups: [""]
    resources: ["namespaces"]
    verbs: ["get"]
  - apiGroups: [""]
    resources: ["nodes"]
    verbs: ["get", "list", "watch"]
  - apiGroups: [""]
    resources: ["pods"]
    verbs: ["get", "list", "watch"]
  - apiGroups: [""]
    resources: ["events"]
    verbs: ["list"]
  - apiGroups: [""]
    resources: ["configmaps"]
    verbs: ["get", "list", "watch"]
```

```
  - apiGroups: ["apps"]
    resources: ["daemonsets", "deployments", "statefulsets", "replicasets"]
    verbs: ["get", "list", "watch"]
  - apiGroups: ["rbac.authorization.k8s.io"]
    resources: ["clusterroles", "clusterrolebindings"]
    verbs: ["get", "list"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: aws-batch-cluster-role-binding
subjects:
- kind: User
  name: aws-batch
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: aws-batch-cluster-role
  apiGroup: rbac.authorization.k8s.io
EOF
```

出力:

```
clusterrole.rbac.authorization.k8s.io/aws-batch-cluster-role created
clusterrolebinding.rbac.authorization.k8s.io/aws-batch-cluster-role-binding created
```

3. 名前空間を対象範囲とする Kubernetes ロールを作成すると、AWS Batch はポッドを管理およびライフサイクルしたり、バインドできるようになります。これは固有の名前空間ごとに 1 回行う必要があります。

```
$ namespace=my-aws-batch-namespace
```

```
$ cat - <<EOF | kubectl apply -f - --namespace "${namespace}"
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: aws-batch-compute-environment-role
  namespace: ${namespace}
rules:
  - apiGroups: [""]
    resources: ["pods"]
    verbs: ["create", "get", "list", "watch", "delete", "patch"]
```

```
  - apiGroups: [""]
    resources: ["serviceaccounts"]
    verbs: ["get", "list"]
  - apiGroups: ["rbac.authorization.k8s.io"]
    resources: ["roles", "rolebindings"]
    verbs: ["get", "list"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: aws-batch-compute-environment-role-binding
  namespace: ${namespace}
subjects:
- kind: User
  name: aws-batch
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: aws-batch-compute-environment-role
  apiGroup: rbac.authorization.k8s.io
EOF
```

出力:

```
role.rbac.authorization.k8s.io/aws-batch-compute-environment-role created
rolebinding.rbac.authorization.k8s.io/aws-batch-compute-environment-role-binding
 created
```

4. Kubernetes aws-auth 設定マップを更新して、前述の RBAC 権限を AWS Batch サービスにリンクされたロールにマップします。

   以下のコマンドで、下記のような置き換えを行います。

   - *<your-account-number>* を AWS アカウント 番号に置き換えます。

```
$ eksctl create iamidentitymapping \
    --cluster my-cluster-name \
    --arn "arn:aws:iam::<your-account-number>:role/AWSServiceRoleForBatch" \
    --username aws-batch
```

出力:

```
2022-10-25 20:19:57 [#]  adding identity "arn:aws:iam::<your-account-number>:role/
AWSServiceRoleForBatch" to auth ConfigMap
```

> **Note**
>
> パス aws-service-role/batch.amazonaws.com/ が、サービスにリンクされたロールの ARN から削除されました。これは aws-auth 設定マップに問題があるためです。詳細については、「Roles with paths don't work when the path is included in their ARN in the aws-authconfigmap」を参照してください。

ステップ 3: Amazon EKS コンピューティング環境を作成する

AWS Batch コンピューティング環境は、バッチワークロードのニーズを満たすコンピューティングリソースパラメータを定義します。マネージドコンピューティング環境では、AWS Batch は Amazon EKS クラスター内のコンピューティングリソース (Kubernetes ノード) の容量とインスタンスタイプを管理します。これは、コンピューティング環境の作成時に定義するコンピューティングリソースの仕様に基づいています。EC2 オンデマンドインスタンスまたは EC2 スポットインスタンスを選択できます。

AWSServiceRoleForBatch サービスにリンクされたロールが Amazon EKS クラスターにアクセスできるようになったので、AWS Batch リソースを作成できます。まず、Amazon EKS クラスターを指すコンピューティング環境を作成します。

- subnets で eksctl get cluster *my-cluster-name* を実行し、クラスターで使用されるサブネットを取得します。
- securityGroupIds パラメータには、Amazon EKS クラスターと同じセキュリティグループを使用できます。このコマンドは、クラスターのセキュリティグループ ID を取得します。

```
$ aws eks describe-cluster \
    --name my-cluster-name \
    --query cluster.resourcesVpcConfig.clusterSecurityGroupId
```

- instanceRole は、クラスターの作成時に作成されます。AmazonEKSWorkerNodePolicy ポリシーを使用するすべてのエンティティをリストした instanceRole を検索するには:

```
$  aws iam list-entities-for-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSWorkerNodePolicy
```

ポリシーロールの名前には、eksctl-*my-cluster-name*-nodegroup-example を作成したクラスターの名前が含まれます。

instanceRole の ARN を検索するには、以下のコマンドを実行します。

```
$  aws iam list-instance-profiles-for-role --role-name eksctl-my-cluster-name-
nodegroup-example
```

出力:

```
INSTANCEPROFILES        arn:aws:iam::<your-account-number>:instance-profile/
eks-04cb2200-94b9-c297-8dbe-87f12example
```

詳しくは、「Amazon EKS ユーザーガイド」の「Amazon EKS ノード IAM ロールの作成」と「クラスターへの IAM プリンシパルアクセスを有効にする」を参照してください。ポッドネットワークを使用している場合は、「Amazon EKS ユーザーガイド」の「サービスアカウントで IAM ロールを使用するための Kubernetes の Amazon VPC CNI プラグインを設定する」を参照してください。

```
$ cat <<EOF > ./batch-eks-compute-environment.json
{
  "computeEnvironmentName": "My-Eks-CE1",
  "type": "MANAGED",
  "state": "ENABLED",
  "eksConfiguration": {
    "eksClusterArn": "arn:aws:eks:region-code:your-account-number:cluster/my-cluster-
name",
    "kubernetesNamespace": "my-aws-batch-namespace"
  },
  "computeResources": {
    "type": "EC2",
    "allocationStrategy": "BEST_FIT_PROGRESSIVE",
    "minvCpus": 0,
    "maxvCpus": 128,
    "instanceTypes": [
```

```
        "m5"
    ],
    "subnets": [
        "<eks-cluster-subnets-with-access-to-internet-for-image-pull>"
    ],
    "securityGroupIds": [
        "<eks-cluster-sg>"
    ],
    "instanceRole": "<eks-instance-profile>"
  }
}
EOF
```

```
$ aws batch create-compute-environment --cli-input-json file://./batch-eks-compute-environment.json
```

メモ

- Amazon EKS コンピューティング環境のメンテナンスは共同責任です。詳細については、「Kubernetes ノードの責任分担」を参照してください。

ステップ 4: ジョブキューを作成してコンピューティング環境をアタッチする

> **Important**
>
> 処理を進める前に、コンピューティング環境が正常であることを確認することが重要です。これには DescribeComputeEnvironments API オペレーションを使用できます。
>
> ```
> $ aws batch describe-compute-environments --compute-environments My-Eks-CE1
> ```
>
> `status` パラメータが `INVALID` ではないことを確認してください。そうであれば、`statusReason` パラメータを調べて原因を調べてください。詳細については、「トラブルシューティング AWS Batch」を参照してください。

この新しいジョブキューに送信されたジョブは、コンピューティング環境に関連付けられた Amazon EKS クラスターに結合された AWS Batch マネージドノードでポッドとして実行されます。

```
$ cat <<EOF > ./batch-eks-job-queue.json
 {
    "jobQueueName": "My-Eks-JQ1",
    "priority": 10,
    "computeEnvironmentOrder": [
      {
        "order": 1,
        "computeEnvironment": "My-Eks-CE1"
      }
    ]
  }
EOF
```

```
$ aws batch create-job-queue --cli-input-json file://./batch-eks-job-queue.json
```

ステップ 5: ジョブ定義を作成する

次のジョブ定義は、ポッドに 60 秒間スリープするように指示します。

```
$ cat <<EOF > ./batch-eks-job-definition.json
{
  "jobDefinitionName": "MyJobOnEks_Sleep",
  "type": "container",
  "eksProperties": {
    "podProperties": {
      "hostNetwork": true,
      "containers": [
        {
          "image": "public.ecr.aws/amazonlinux/amazonlinux:2",
          "command": [
            "sleep",
            "60"
          ],
          "resources": {
            "limits": {
              "cpu": "1",
              "memory": "1024Mi"
            }
          }
        }
      ],
      "metadata": {
```

```
        "labels": {
          "environment": "test"
        }
      }
    }
  }
}
EOF
```

```
$ aws batch register-job-definition --cli-input-json file://./batch-eks-job-definition.json
```

メモ

- cpu および memory パラメータには考慮事項があります。詳細については、「Amazon EKS の AWS Batch におけるメモリと vCPU に関する考慮事項」を参照してください。

ステップ 6: ジョブを送信する

新しいジョブを送信するには、以下の AWS CLI コマンドを実行します。

```
$ aws batch submit-job --job-queue My-Eks-JQ1 \
    --job-definition MyJobOnEks_Sleep --job-name My-Eks-Job1
```

ジョブのステータスを確認するには:

```
$ aws batch describe-jobs --job <jobId-from-submit-response>
```

メモ

- Amazon EKS リソースでのジョブ実行の詳細については、Amazon EKS ジョブ を参照してください。

ステップ 7: ジョブの出力を表示する

ジョブの出力を表示するには、以下を実行します。

1. AWS Batch コンソールを https://console.aws.amazon.com/batch/ で開きます。

2. ナビゲーションペインで [ジョブ] を選択します。
3. [ジョブキュー] ドロップダウンで、チュートリアル用に作成したジョブキューを選択します。
4. [ジョブ] テーブルには、すべてのジョブとその現在のステータスが一覧表示されます。ジョブの [ステータス] が [成功] したら、ジョブの [名前]、*[My-Eks-JQ1]* を選択してジョブの詳細を表示します。
5. [詳細] ペインでは、[開始時刻] と [停止時刻] を 1 分間隔で指定する必要があります。

ステップ 8: (オプション) オーバーライドを含むジョブを送信する

このジョブは、コンテナに渡されたコマンドを上書きします。AWS Batch は、ジョブの完了後にポッドを積極的にクリーンアップして、Kubernetes に対する負荷を軽減します。ジョブの詳細を確認するには、ログ記録を設定する必要があります。詳細については、「CloudWatch Logs を使用して Amazon EKS ジョブにおける AWS Batch をモニタリングする」を参照してください。

```
$ cat <<EOF > ./submit-job-override.json
{
  "jobName": "EksWithOverrides",
  "jobQueue": "My-Eks-JQ1",
  "jobDefinition": "MyJobOnEks_Sleep",
  "eksPropertiesOverride": {
    "podProperties": {
      "containers": [
        {
          "command": [
            "/bin/sh"
          ],
          "args": [
            "-c",
            "echo hello world"
          ]
        }
      ]
    }
  }
}
EOF
```

```
$ aws batch submit-job --cli-input-json file://./submit-job-override.json
```

メモ

- 操作の詳細を把握しやすくするには、Amazon EKS コントロールプレーンのログ記録を有効にします。詳細については、「Amazon EKS ユーザーガイド」の「Amazon EKS コントロールプレーンのログ」を参照してください。
- Daemonsets と kubelets オーバーヘッドは、使用可能な vCPU とメモリのリソース、特にスケーリングとジョブの配置に影響します。詳細については、「Amazon EKS の AWS Batch におけるメモリと vCPU に関する考慮事項」を参照してください。

ジョブの出力を表示するには、以下を実行します。

1. AWS Batch コンソールを https://console.aws.amazon.com/batch/ で開きます。
2. ナビゲーションペインで [ジョブ] を選択します。
3. [ジョブキュー] ドロップダウンで、チュートリアル用に作成したジョブキューを選択します。
4. [ジョブ] テーブルには、すべてのジョブとその現在のステータスが一覧表示されます。ジョブの [ステータス] が [成功] であれば、ジョブの [名前] を選択してジョブの詳細を表示します。
5. [詳細] ペインで、ログストリームの名前を選択します。ジョブの CloudWatch コンソールが開き、`hello world` の [メッセージ] またはカスタムメッセージを含むイベントが 1 つあるはずです。

ステップ 9: チュートリアルのリソースをクリーンアップする

Amazon EC2 インスタンスが有効になっている間は課金されます。インスタンスを削除して、料金の発生を停止できます。

作成したリソースを削除するには、次の作業を行います。

1. AWS Batch コンソールを https://console.aws.amazon.com/batch/ で開きます。
2. ナビゲーションペインで [ジョブキュー] を選択します。
3. [ジョブキュー] テーブルで、チュートリアル用に作成したジョブキューを選択します。
4. [無効化] を選択します。ジョブキューの [状態] が無効になったら、[削除] を選択できます。
5. ジョブキューが削除されたら、ナビゲーションペインで [コンピューティング環境] を選択します。
6. このチュートリアル用に作成したコンピューティング環境を選択し、[無効化] を選択します。コンピューティング環境が無効になるまでに 1~2 分かかる場合があります。

7. コンピューティング環境の [状態] が無効になったら、[削除] を選択します。コンピューティング環境が削除されるまでに 1～2 分かかる場合があります。

その他のリソース

チュートリアルが完了したら、次のトピックを試すことができます。

- ベストプラクティスについて詳しく説明します。
- AWS Batch コアコンポーネントについて説明します。詳細については、「AWS Batch のコンポーネント」を参照してください。
- AWS Batch で使用できるさまざまなコンピューティング環境について説明します。
- [ジョブキュー] とそのさまざまなスケジューリングオプションについて詳しく説明します。
- [ジョブ定義] とそのさまざまな設定オプションについて詳しく説明します。
- さまざまなタイプの [ジョブ] について詳しく説明します。

Amazon EKS プライベートクラスターで AWS Batch の使用を開始する

AWS Batch は Amazon Elastic Kubernetes Service (Amazon EKS) クラスターでのバッチワークロードをオーケストレーションするマネージドサービスです。このサービスにはキューイング、依存関係の追跡、ジョブの再試行と優先順位の管理、ポッド管理、ノードスケーリングが含まれます。この機能により、既存のプライベート Amazon EKS クラスターを AWS Batch に接続して、ジョブを大規模に実行できます。`eksctl` (Amazon EKS のコマンドラインインターフェイス)、AWS コンソール、AWS Command Line Interface を使用して、プライベート Amazon EKS クラスターを他のすべての必要なリソースと共に作成できます。

Amazon EKS プライベート専用クラスターには、デフォルトではインバウンド/アウトバウンドインターネットアクセスがなく、VPC 内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。他の AWS サービスへのプライベートアクセスを有効にするには Amazon VPC エンドポイントを使用します。`eksctl` は既存の Amazon VPC とサブネットを使用した完全なプライベートクラスターの作成をサポートします。`eksctl` はまた、提供された Amazon VPC に Amazon VPC エンドポイントを作成し、提供されたサブネットのルートテーブルを変更します。

`eksctl` はメインルートテーブルを変更しないため、各サブネットには明示的に関連付けられたルートテーブルが必要です。クラスターは Amazon 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。Amazon 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もできます。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。Amazon ECR のプライベートリポジトリの使用を開始するには、「Amazon ECR private repositories」を参照してください。

必要に応じて、Amazon ECR を使用してプルスルーキャッシュルールを作成することもできます。外部パブリックレジストリのプルスルーキャッシュルールを作成したら、Amazon ECR プライベートレジストリの URI を使用して、その外部パブリックレジストリからイメージをプルできます。その後、Amazon ECR でリポジトリが作成され、イメージがキャッシュされます。キャッシュされたイメージが Amazon ECR プライベートレジストリ URI を使用してプルされると、Amazon ECR はリモートレジストリをチェックしてイメージの新しいバージョンがあるかどうかを確認し、24 時間ごとに最大 1 回、プライベートレジストリを更新します。

目次

概要

このチュートリアルでは、AWS CloudShell、`kubectl`、`eksctl` を使用してプライベート Amazon EKS で AWS Batch を設定する方法を示します。

対象者

このチュートリアルは、AWS Batch のセットアップ、テスト、デプロイを担当するシステム管理者とデベロッパーを対象としています。

使用される機能

このチュートリアルでは、AWS CLI を使用して以下を行う方法について説明します。

- Amazon Elastic Container Registry (Amazon ECR) を使用してコンテナイメージを保存する
- Amazon EKS コンピューティング環境を作成して設定する
- ジョブキューの作成。
- ジョブ定義の作成
- ジョブを作成して実行する
- オーバーライドを含むジョブを送信する

所要時間

このチュートリアルは完了までに約 40～50 分かかります。

リージョン別制限

このソリューションの使用に関連する国やリージョン別の制限はありません。

リソース使用量のコスト

AWS アカウントを作成するための料金はかかりません。ただし、このソリューションを実装することにより、次の表に記載されるコストの一部またはすべてが発生する可能性があります。

説明	コスト (USD)
クラスター時間ごとに課金されます	インスタンスによって異なります。「Amazon EKS の料金」を参照してください。
Amazon EC2 インスタンス	作成された各 Amazon EC2 インスタンスに対して料金が発生します。料金の詳細については、「Amazon EC2 料金表」を参照してください。

前提条件

このチュートリアルでは、AWS マネジメントコンソール から直接起動できる、ブラウザベースの事前認証されたシェルである AWS CloudShell を取り上げます。これにより、パブリックインターネットアクセスがなくなると、クラスターにアクセスできます。AWS CLI、`kubectl`、`eksctl` は、AWS CloudShell の一部として既にインストールされている場合があります。AWS CloudShell の詳細については、「AWS CloudShell ユーザーガイド」を参照してください。代替策として、AWS CloudShell をクラスターの VPC または接続済みのネットワークに接続することもできます。

kubectl コマンドを実行するには Amazon EKS クラスターへのプライベートアクセスが必要です。つまり、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。

- 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。このガイドでは、バージョン 2.8.6 以降または 1.26.0 以降の使用を想定しています。詳細については、「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」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、AWS Command Line Interface ユーザーガイドの `aws configure` を使用したクイック設定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。このガイドでは、バージョン `1.23` 以降の使用を想定しています。詳細については、「Amazon EKS ユーザーガイド」の「kubectl のインストールまたは更新」を参照してください。
- **`eksctl`** – Amazon 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。このガイドでは、バージョン `0.115.0` 以降の使用を想定しています。詳細については、「Amazon EKS ユーザーガイド」の「eksctl のインストールまたは更新」を参照してください。
- 権限 – CreateComputeEnvironment API オペレーションを呼び出して Amazon EKS リソースを使用するコンピューティング環境を作成するユーザーには、`eks:DescribeCluster` および `eks:ListClusters` API オペレーションに対する権限が必要です。AWSBatchFullAccess マネージドポリシーをユーザーアカウントにアタッチするには、「IAM ユーザーガイド」の「IAM ID アクセス許可の追加と削除」の指示に従います。
- InstanceRole – `AmazonEKSWorkerNodePolicy` および `AmazonEC2ContainerRegistryPullOnly` ポリシーを持つ Amazon EKS ノードの `InstanceRole` を作成する必要があります。`InstanceRole` の作成方法については、「Amazon EKS ノード IAM ロールの作成」を参照してください。`InstanceRole` の ARN が必要です。

- AWS アカウント ID – AWS アカウント ID を把握する必要があります。「AWS アカウント ID の表示」の指示に従います。
- (オプション) CloudWatch – (オプション) [オーバーライドを含むジョブを送信する] の詳細を確認するには、ログ記録を設定する必要があります。詳細については、「CloudWatch Logs を使用して Amazon EKS ジョブにおける AWS Batch をモニタリングする」を参照してください。

ステップ 1: AWS Batch の 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、このチュートリアルでは、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。本番での使用に向けて作成する前に、すべての設定内容に習熟した上で、要件を満たす設定でデプロイすることをお勧めします。

クラスターを作成するには、`eksctl` と次の設定ファイルを使用することをお勧めします。クラスターを手動でセットアップするには、「Amazon EKS ユーザーガイド」の「インターネットアクセスが制限されたプライベートクラスターをデプロイする」の指示に従います。

1. [AWS CloudShell コンソール] を開き、リージョンを [us-east-1] に設定します。チュートリアルの残りの部分では、`us-east-1` を使用していることを確認してください。
2. サンプルの `eksctl` 設定ファイルを使用して `us-east-1` リージョンにプライベート EKS クラスターを作成します。yaml ファイルを AWS CloudShell 環境に保存し、`clusterConfig.yaml` という名前を付けます。*`[my-test-cluster]`* は、クラスターに使用する名前で変更できます。

```
kind: ClusterConfig
apiVersion: eksctl.io/v1alpha5
metadata:
    name: my-test-cluster
    region: us-east-1
availabilityZones:
    - us-east-1a
    - us-east-1b
    - us-east-1c
managedNodeGroups:
    - name: ng-1
      privateNetworking: true
```

```
privateCluster:
    enabled: true
    skipEndpointCreation: false
```

3. `eksctl create cluster -f clusterConfig.yaml` のコマンドを使用してリソースを作成します。クラスターの作成には、10 ~ 15 分かかる場合があります。
4. クラスターの作成が完了したら、AWS CloudShell IP アドレスを許可リストに追加する必要があります。AWS CloudShell IP アドレスを検索するには、次のコマンドを実行します。

```
curl http://checkip.amazonaws.com
```

パブリック IP アドレスを取得したら、許可リストルールを作成する必要があります。

```
aws eks update-cluster-config \
  --name my-test-cluster \
  --region us-east-1 \
  --resources-vpc-config
 endpointPublicAccess=true,endpointPrivateAccess=true,publicAccessCidrs=["<Public
 IP>/32"]
```

次に、kubectl 設定ファイルに更新を適用します。

```
aws eks update-kubeconfig --name my-test-cluster --region us-east-1
```

5. ノードにアクセスできることをテストするには、次のコマンドを実行します。

```
kubectl get nodes
```

コマンドの出力:

```
NAME                                STATUS   ROLES    AGE     VERSION
ip-192-168-107-235.ec2.internal   Ready    none     1h   v1.32.3-eks-473151a
ip-192-168-165-40.ec2.internal    Ready    none     1h   v1.32.3-eks-473151a
ip-192-168-98-54.ec2.internal     Ready    none     1h   v1.32.1-eks-5d632ec
```

ステップ 2: AWS Batch 用の EKS クラスターを準備する

すべてのステップが必要であり、AWS CloudShell で実行する必要があります。

1. AWS Batch ジョブ専用の名前空間を作成する

   kubectl を使用して新しい名前空間を作成します。

```
$ namespace=my-aws-batch-namespace
```

```
$ cat - <<EOF | kubectl create -f -
{
  "apiVersion": "v1",
  "kind": "Namespace",
  "metadata": {
    "name": "${namespace}",
    "labels": {
      "name": "${namespace}"
    }
  }
}
EOF
```

   出力:

```
namespace/my-aws-batch-namespace created
```

2. ロールベースアクセス制御 (RBAC) を有効にします。

   kubectl を使用して、クラスターの Kubernetes ロールを作成すると、AWS Batch はノードとポッドを監視したり、ロールをバインドしたりできるようになります。これを Amazon EKS クラスターごとに 1 回実行する必要があります。

```
$ cat - <<EOF | kubectl apply -f -
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: aws-batch-cluster-role
rules:
  - apiGroups: [""]
    resources: ["namespaces"]
    verbs: ["get"]
  - apiGroups: [""]
    resources: ["nodes"]
    verbs: ["get", "list", "watch"]
```

```
  - apiGroups: [""]
    resources: ["pods"]
    verbs: ["get", "list", "watch"]
  - apiGroups: [""]
    resources: ["events"]
    verbs: ["list"]
  - apiGroups: [""]
    resources: ["configmaps"]
    verbs: ["get", "list", "watch"]
  - apiGroups: ["apps"]
    resources: ["daemonsets", "deployments", "statefulsets", "replicasets"]
    verbs: ["get", "list", "watch"]
  - apiGroups: ["rbac.authorization.k8s.io"]
    resources: ["clusterroles", "clusterrolebindings"]
    verbs: ["get", "list"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: aws-batch-cluster-role-binding
subjects:
- kind: User
  name: aws-batch
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: aws-batch-cluster-role
  apiGroup: rbac.authorization.k8s.io
EOF
```

出力:

```
clusterrole.rbac.authorization.k8s.io/aws-batch-cluster-role created
clusterrolebinding.rbac.authorization.k8s.io/aws-batch-cluster-role-binding created
```

名前空間を対象範囲とする Kubernetes ロールを作成すると、AWS Batch はポッドを管理およびライフサイクルしたり、バインドできるようになります。これは固有の名前空間ごとに 1 回行う必要があります。

```
$ namespace=my-aws-batch-namespace
```

```
$ cat - <<EOF | kubectl apply -f - --namespace "${namespace}"
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: aws-batch-compute-environment-role
  namespace: ${namespace}
rules:
  - apiGroups: [""]
    resources: ["pods"]
    verbs: ["create", "get", "list", "watch", "delete", "patch"]
  - apiGroups: [""]
    resources: ["serviceaccounts"]
    verbs: ["get", "list"]
  - apiGroups: ["rbac.authorization.k8s.io"]
    resources: ["roles", "rolebindings"]
    verbs: ["get", "list"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: aws-batch-compute-environment-role-binding
  namespace: ${namespace}
subjects:
- kind: User
  name: aws-batch
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: aws-batch-compute-environment-role
  apiGroup: rbac.authorization.k8s.io
EOF
```

出力:

```
role.rbac.authorization.k8s.io/aws-batch-compute-environment-role created
rolebinding.rbac.authorization.k8s.io/aws-batch-compute-environment-role-binding
 created
```

Kubernetes `aws-auth` 設定マップを更新して、前述の RBAC 権限を AWS Batch サービスにリンクされたロールにマップします。

```
$ eksctl create iamidentitymapping \
    --cluster my-test-cluster \
    --arn "arn:aws:iam::<your-account-ID>:role/AWSServiceRoleForBatch" \
    --username aws-batch
```

出力:

```
2022-10-25 20:19:57 [#]  adding identity "arn:aws:iam::<your-account-ID>:role/
AWSServiceRoleForBatch" to auth ConfigMap
```

> **Note**
>
> パス aws-service-role/batch.amazonaws.com/ が、サービスにリンクされたロールの ARN から削除されました。これは aws-auth 設定マップに問題があるためです。詳細については、「Roles with paths don't work when the path is included in their ARN in the aws-authconfigmap」を参照してください。

ステップ 3: Amazon EKS コンピューティング環境を作成する

AWS Batch コンピューティング環境は、バッチワークロードのニーズを満たすコンピューティングリソースパラメータを定義します。マネージドコンピューティング環境では、AWS Batch は Amazon EKS クラスター内のコンピューティングリソース (Kubernetes ノード) の容量とインスタンスタイプを管理します。これは、コンピューティング環境の作成時に定義するコンピューティングリソースの仕様に基づいています。EC2 オンデマンドインスタンスまたは EC2 スポットインスタンスを選択できます。

AWSServiceRoleForBatch サービスにリンクされたロールが Amazon EKS クラスターにアクセスできるようになったので、AWS Batch リソースを作成できます。まず、Amazon EKS クラスターを指すコンピューティング環境を作成します。

- subnets で `eksctl get cluster my-test-cluster` を実行し、クラスターで使用されるサブネットを取得します。
- securityGroupIds パラメータには、Amazon EKS クラスターと同じセキュリティグループを使用できます。このコマンドは、クラスターのセキュリティグループ ID を取得します。

```
$ aws eks describe-cluster \
```

```
    --name my-test-cluster \
    --query cluster.resourcesVpcConfig.clusterSecurityGroupId
```

- 前提条件で作成した instanceRole の ARN を使用します。

```
$ cat <<EOF > ./batch-eks-compute-environment.json
{
  "computeEnvironmentName": "My-Eks-CE1",
  "type": "MANAGED",
  "state": "ENABLED",
  "eksConfiguration": {
    "eksClusterArn": "arn:aws:eks:us-east-1:<your-account-ID>:cluster/my-test-cluster",
    "kubernetesNamespace": "my-aws-batch-namespace"
  },
  "computeResources": {
    "type": "EC2",
    "allocationStrategy": "BEST_FIT_PROGRESSIVE",
    "minvCpus": 0,
    "maxvCpus": 128,
    "instanceTypes": [
        "m5"
    ],
    "subnets": [
        "<eks-cluster-subnets-with-access-to-the-image-for-image-pull>"
    ],
    "securityGroupIds": [
        "<eks-cluster-sg>"
    ],
    "instanceRole": "<eks-instance-profile>"
  }
}
EOF
```

```
$ aws batch create-compute-environment --cli-input-json file://./batch-eks-compute-
environment.json
```

メモ

- Amazon EKS コンピューティング環境のメンテナンスは共同責任です。詳細については「Amazon EKS のセキュリティ」をご参照ください。

ステップ 4: ジョブキューを作成してコンピューティング環境をアタッチする

> **Important**
>
> 処理を進める前に、コンピューティング環境が正常であることを確認することが重要です。これには DescribeComputeEnvironments API オペレーションを使用できます。
>
> ```
> $ aws batch describe-compute-environments --compute-environments My-Eks-CE1
> ```
>
> status パラメータが INVALID ではないことを確認してください。そうであれば、statusReason パラメータを調べて原因を調べてください。詳細については、「トラブルシューティング AWS Batch」を参照してください。

この新しいジョブキューに送信されたジョブは、コンピューティング環境に関連付けられた Amazon EKS クラスターに結合された AWS Batch マネージドノードでポッドとして実行されます。

```
$ cat <<EOF > ./batch-eks-job-queue.json
 {
    "jobQueueName": "My-Eks-JQ1",
    "priority": 10,
    "computeEnvironmentOrder": [
      {
        "order": 1,
        "computeEnvironment": "My-Eks-CE1"
      }
    ]
  }
EOF
```

```
$ aws batch create-job-queue --cli-input-json file://./batch-eks-job-queue.json
```

ステップ 5: プルスルーキャッシュを使用して Amazon ECR を作成する

クラスターにはパブリックインターネットアクセスがないため、コンテナイメージ用の Amazon ECR を作成する必要があります。以下の手順では、イメージを保存するプルスルーキャッシュルールを使用して Amazon ECR を作成します。

1. 次のコマンドを使用してプルスルーキャッシュルールを作成します。*[tutorial-prefix]* は別のプレフィックスに置き換えることができます。

```
aws ecr create-pull-through-cache-rule \
    --ecr-repository-prefix "my-prefix" \
    --upstream-registry-url "public.ecr.aws" \
    --region us-east-1
```

2. パブリック ECR で認証します。

```
aws ecr get-login-password --region us-east-1 | docker login --username AWS --
password-stdin <your-account-ID>.dkr.ecr.us-east-1.amazonaws.com
```

これで、イメージをプルできます。

```
docker pull <your-account-ID>.dkr.ecr.us-east-1.amazonaws.com/my-prefix/
amazonlinux/amazonlinux:2
```

3. リポジトリとイメージを確認するには、次のコマンドを実行します。

```
aws ecr describe-repositories
```

```
aws ecr describe-images --repository-name my-prefix/amazonlinux/amazonlinux
```

4. コンテナのプルに使用するイメージ文字列は、次の形式です。

```
<your-account-ID>.dkr.ecr.us-east-1.amazonaws.com/my-prefix/amazonlinux/
amazonlinux:2
```

ステップ 6: ジョブ定義を登録する

次のジョブ定義は、ポッドに 60 秒間スリープするように指示します。

ジョブ定義のイメージフィールドに、パブリック ECR リポジトリのイメージへのリンクではなく、プライベート ECR リポジトリに保存されているイメージへのリンクを指定します。以下はジョブ定義の例です。

```
$ cat <<EOF > ./batch-eks-job-definition.json
{
```

```
  "jobDefinitionName": "MyJobOnEks_Sleep",
  "type": "container",
  "eksProperties": {
    "podProperties": {
      "hostNetwork": true,
      "containers": [
        {
          "image": "<your-account-ID>.dkr.ecr.us-east-1.amazonaws.com/my-prefix/
amazonlinux/amazonlinux:2",
          "command": [
            "sleep",
            "60"
          ],
          "resources": {
            "limits": {
              "cpu": "1",
              "memory": "1024Mi"
            }
          }
        }
      ],
      "metadata": {
        "labels": {
          "environment": "test"
        }
      }
    }
  }
}
EOF
```

```
$ aws batch register-job-definition --cli-input-json file://./batch-eks-job-
definition.json
```

メモ

- cpu および memory パラメータには考慮事項があります。詳細については、「Amazon EKS の AWS Batch におけるメモリと vCPU に関する考慮事項」を参照してください。

ステップ 7: ジョブ実行を送信する

AWS CloudShell で次の AWS CLI コマンドを実行して新しいジョブを送信し、一意の JobID を返します。

```
$ aws batch submit-job --job-queue My-Eks-JQ1 \
    --job-definition MyJobOnEks_Sleep - -job-name My-Eks-Job1
```

メモ

- Amazon EKS リソースでのジョブ実行の詳細については、Amazon EKS ジョブ を参照してください。

ステップ 8: ジョブの出力を表示する

ジョブのステータスを確認するには:

```
$ aws batch describe-jobs --job <JobID-from-submit-response>
```

`startedAt` と `stoppedAt` は 1 分間隔で配置する必要があります。

ステップ 9: (オプション) オーバーライドを含むジョブを送信する

このジョブは、コンテナに渡されたコマンドをオーバーライドします。

```
$ cat <<EOF > ./submit-job-override.json
{
  "jobName": "EksWithOverrides",
  "jobQueue": "My-Eks-JQ1",
  "jobDefinition": "MyJobOnEks_Sleep",
  "eksPropertiesOverride": {
    "podProperties": {
      "containers": [
        {
          "command": [
            "/bin/sh"
          ],
          "args": [
            "-c",
            "echo hello world"
          ]
```

```
        }
      ]
    }
  }
}
EOF
```

```
$ aws batch submit-job - -cli-input-json file://./submit-job-override.json
```

メモ

- 操作の詳細を把握しやすくするには、Amazon EKS コントロールプレーンのログ記録を有効にします。詳細については、「Amazon EKS ユーザーガイド」の「Amazon EKS コントロールプレーンのログ」を参照してください。
- Daemonsets と kubelets オーバーヘッドは、使用可能な vCPU とメモリのリソース、特にスケーリングとジョブの配置に影響します。詳細については、「Amazon EKS の AWS Batch におけるメモリと vCPU に関する考慮事項」を参照してください。

ステップ 10: チュートリアルのリソースをクリーンアップする

Amazon EC2 インスタンスが有効になっている間は課金されます。インスタンスを削除して、料金の発生を停止できます。

作成したリソースを削除するには、次の作業を行います。

1. AWS Batch コンソールを https://console.aws.amazon.com/batch/ で開きます。
2. ナビゲーションペインで [ジョブキュー] を選択します。
3. [ジョブキュー] テーブルで、チュートリアル用に作成したジョブキューを選択します。
4. [アクション] で、[無効化] を選択します。ジョブキューの [状態] が無効になったら、[削除] を選択できます。
5. ジョブキューが削除されたら、ナビゲーションペインで [コンピューティング環境] を選択します。
6. このチュートリアル用に作成したコンピューティング環境を選択し、[アクション] から [無効化] を選択します。コンピューティング環境が無効になるまでに 1～2 分かかる場合があります。
7. コンピューティング環境の [状態] が無効になったら、[削除] を選択します。コンピューティング環境が削除されるまでに 1～2 分かかる場合があります。

その他のリソース

チュートリアルが完了したら、次のトピックを試すことができます。

- ベストプラクティスについて詳しく説明します。
- AWS Batch コアコンポーネントについて説明します。詳細については、「AWS Batch のコンポーネント」を参照してください。
- AWS Batch で使用できるさまざまなコンピューティング環境について説明します。
- [ジョブキュー] とそのさまざまなスケジューリングオプションについて詳しく説明します。
- [ジョブ定義] とそのさまざまな設定オプションについて詳しく説明します。
- さまざまなタイプの [ジョブ] について詳しく説明します。

トラブルシューティング

AWS Batch によって起動されたノードが、イメージを保存する Amazon ECR リポジトリ (またはその他の任意のリポジトリ) にアクセスできない場合、ジョブは STARTING の状態のままになる可能性があります。これは、ポッドがイメージをダウンロードして AWS Batch のジョブを実行できないことによります。AWS Batch によって起動されたポッド名をクリックするとエラーメッセージが表示され、問題を確認できます。エラーメッセージは次のようになります。

```
Failed to pull image "public.ecr.aws/amazonlinux/amazonlinux:2": rpc error: code =
Unknown desc = failed to pull and unpack image
"public.ecr.aws/amazonlinux/amazonlinux:2": failed to resolve reference
"public.ecr.aws/amazonlinux/amazonlinux:2": failed to do request: Head
"https://public.ecr.aws/v2/amazonlinux/amazonlinux/manifests/2": dial tcp: i/o timeout
```

その他の一般的なトラブルシューティングのシナリオについては、「AWS Batch のトラブルシューティング」を参照してください。ポッドステータスに基づくトラブルシューティングについては、「How do I troubleshoot the pod status in Amazon EKS?」を参照してください。

SageMaker AI AWS Batch での の開始方法

AWS Batch サービスジョブを使用すると、スケジューリング、優先順位付け、キューイング機能を備えた AWS Batch ジョブキューを介して SageMaker トレーニングジョブを送信できます。このチュートリアルでは、 AWS Batch サービスジョブを使用してシンプルな SageMaker トレーニングジョブをセットアップして実行する方法を示します。

目次

概要:

このチュートリアルでは、 を使用して SageMaker トレーニングジョブ AWS Batch のサービスジョブを設定する方法を示します AWS CLI。

対象者

このチュートリアルは、大規模な機械学習トレーニングジョブのセットアップと実行を担当するデータサイエンティストと開発者を対象としています。

使用される機能

このチュートリアルでは、 を使用して以下 AWS CLI を行う方法を示します。

- SageMaker トレーニングジョブのサービス環境を作成する
- SageMaker トレーニングジョブのキューを作成する
- `SubmitServiceJob` API を使用してサービスジョブを送信する
- ジョブステータスをモニタリングし、出力を表示する
- トレーニングジョブに関する CloudWatch Logs にアクセスする

所要時間

このチュートリアルは完了までに約 15 分かかります。

リージョンの制限

このチュートリアルは、 AWS Batch と SageMaker AI の両方が利用可能な任意の AWS リージョンで完了できます。

リソースの使用コスト

AWS アカウントの作成には料金はかかりません。ただし、このソリューションを実装することによって、以下のリソースにコストが発生する可能性があります。

説明	コスト (USD)
SageMaker AI トレーニングのインスタンス	使用した SageMaker AI トレーニングインスタンスごとに料金が発生します。料金の詳細については、「Amazon SageMaker の料金」を参照してください。
Amazon S3 ストレージ	トレーニングジョブの出力を保存するための最小限のコスト。詳細については、Amazon S3 の料金 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、 と SageMaker AI リソースの両方を作成および管理するために必要な以下のツール AWS Batch とリソースをインストールして設定する必要があります。

- AWS CLI – AWS Batch や SageMaker AI などの AWS サービスを操作するためのコマンドラインツールです。このガイドでは、バージョン 2.8.6 以降を使用する必要があります。詳細については、「AWS Command Line Interface ユーザーガイド」の「AWS CLIのインストール、更新、およびアンインストール」を参照してください。をインストールしたら AWS CLI、これも設定することをお勧めします。詳細については、AWS Command Line Interface ユーザーガイドの `aws configure` を使用したクイック設定を参照してください。

ステップ 1: SageMaker AI 実行ロールを作成する

SageMaker AI は実行ロールを使用して、他の AWS サービスを使用するユーザーに代わって操作を実行します。実行ロールを作成し、トレーニングジョブに必要なサービスとリソースを使用するため

のアクセス許可を SageMaker AI に付与する必要があります。Amazon S3 のアクセス許可が含まれているため、AmazonSageMakerFullAccess マネージドポリシーを使用します。

> **Note**
>
> 次の手順に従って、このチュートリアルの SageMaker AI 実行ロールを作成します。本番環境の実行ロールを作成する前に、「SageMaker AI デベロッパーガイド」の「SageMaker AI 実行ロールの使用方法」を確認することをお勧めします。

1. IAM ロールの作成

   以下の信頼ポリシーを持つ、sagemaker-trust-policy.json という名前の JSON ファイルを作成します。

   JSON

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

   信頼ポリシーを使用して IAM ロールを作成します。

   ```
   aws iam create-role \
       --role-name SageMakerExecutionRole \
       --assume-role-policy-document file://sagemaker-trust-policy.json \
       --description "Execution role for SageMaker training jobs"
   ```

2. マネージドポリシーをアタッチする

   必要なマネージドポリシーをロールにアタッチします。

```
aws iam attach-role-policy \
    --role-name SageMakerExecutionRole \
    --policy-arn arn:aws:iam::aws:policy/AmazonSageMakerFullAccess
```

```
aws iam attach-role-policy \
    --role-name SageMakerExecutionRole \
    --policy-arn arn:aws:iam::aws:policy/AmazonS3FullAccess
```

3. ロールの ARN を書き留めておきます。

   ロール ARN を取得します。これは後のステップで必要になります。

```
aws iam get-role --role-name SageMakerExecutionRole --query 'Role.Arn' --output
 text
```

   トレーニングジョブペイロードを作成するときに使用するので、この ARN を保存します。

ステップ 2: サービス環境を作成する

サービス環境は、SageMaker トレーニングジョブの容量制約を定義します。サービス環境は、同時に実行できるトレーニングインスタンスの最大数をカプセル化します。

> **Important**
>
> SageMaker トレーニング用の最初のサービス環境を作成すると、 AWS Batch はアカウントで AWSServiceRoleForAWSBatchWithSagemaker という名前の、サービスにリンクされたロールを自動的に作成します。このロールにより、 AWS Batch はユーザーに代わって SageMaker トレーニングジョブをキューに入れ、管理できます。サービスにリンクされたロールおよびその許可の詳細については、「the section called “SageMaker 統合ロール”」を参照してください。

最大 5 つのインスタンスを処理できるサービス環境を作成します。

```
aws batch create-service-environment \
    --service-environment-name TutorialServiceEnvironment \
    --service-environment-type SAGEMAKER_TRAINING \
    --capacity-limits capacityUnit=NUM_INSTANCES,maxCapacity=5
```

出力:

```
{
    "serviceEnvironmentName": "TutorialServiceEnvironment",
    "serviceEnvironmentArn": "arn:aws:batch:your-region:your-account-id:service-
environment/TutorialServiceEnvironment"
}
```

サービス環境が正常に作成されたことを確認します。

```
aws batch describe-service-environments --service-
environments TutorialServiceEnvironment
```

出力:

```
{
    "serviceEnvironments": [
        {
            "serviceEnvironmentName": "TutorialServiceEnvironment",
            "serviceEnvironmentArn": "arn:aws:batch:your-region:your-account-
id:service-environment/TutorialServiceEnvironment",
            "serviceEnvironmentType": "SAGEMAKER_TRAINING",
            "state": "ENABLED",
            "status": "VALID",
            "capacityLimits": [
                {
                    "maxCapacity": 5,
                    "capacityUnit": "NUM_INSTANCES"
                }
            ],
            "tags": {}
        }
    ]
}
```

サービス環境の詳細については、「サービス環境」を参照してください。

ステップ 3: SageMaker ジョブキューを作成する

SageMaker ジョブキューは、サービスジョブのスケジューリングと実行を管理します。このキューに送信されたジョブは、使用可能な容量に基づいてサービス環境にディスパッチされます。

SageMaker トレーニングジョブのキューを作成します。

```
aws batch create-job-queue \
    --job-queue-name my-sm-training-fifo-jq \
    --job-queue-type SAGEMAKER_TRAINING \
    --priority 1 \
    --service-environment-order order=1,serviceEnvironment=TutorialServiceEnvironment
```

出力:

```
{
    "jobQueueName": "my-sm-training-fifo-jq",
    "jobQueueArn": "arn:aws:batch:your-region:your-account-id:job-queue/my-sm-training-fifo-jq"
}
```

ジョブのキューが正常に作成されたことを確認します。

```
aws batch describe-job-queues --job-queues my-sm-training-fifo-jq
```

出力:

```
{
    "jobQueues": [
        {
            "jobQueueName": "my-sm-training-fifo-jq",
            "jobQueueArn": "arn:aws:batch:your-region:your-account-id:job-queue/my-sm-training-fifo-jq",
            "state": "ENABLED",
            "status": "VALID",
            "statusReason": "JobQueue Healthy",
            "priority": 1,
            "computeEnvironmentOrder": [],
            "serviceEnvironmentOrder": [
                {
                    "order": 1,
                    "serviceEnvironment": "arn:aws:batch:your-region:your-account-id:service-environment/TutorialServiceEnvironment"
                }
            ],
            "jobQueueType": "SAGEMAKER_TRAINING",
```

```
            "tags": {}
        }
    ]
}
```

SageMaker ジョブのキューの詳細については、「the section called “SageMaker ジョブのキューを作成する”」を参照してください。

ステップ 4: トレーニングジョブを作成して送信する

次に、シンプルなトレーニングジョブを作成し、ジョブキューに送信します。この例では、サービスジョブの機能を示す基本的な「hello world」トレーニングジョブを使用しています。

my_training_job.json という名前のファイルを作成し、次の内容を記述します。*your-account-id* は、自分の AWS アカウント ID に置き換えます。

> **Note**
>
> S3OutputPath は SageMaker トレーニングジョブの作成に必要ですが、このチュートリアルの結果は Amazon S3 バケットに保存されず、次の JSON でパスを使用できます。本番環境では、必要に応じて出力を保存するために有効な Amazon S3 バケットが必要です。

```
{
    "TrainingJobName": "my-simple-training-job",
    "RoleArn": "arn:aws:iam::your-account-id:role/SageMakerExecutionRole",
    "AlgorithmSpecification": {
        "TrainingInputMode": "File",
        "TrainingImage": "763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-
training:2.0.0-cpu-py310",
        "ContainerEntrypoint": [
            "echo",
            "hello world"
        ]
    },
    "ResourceConfig": {
        "InstanceType": "ml.c5.xlarge",
        "InstanceCount": 1,
        "VolumeSizeInGB": 1
    },
    "OutputDataConfig": {
```

```
        "S3OutputPath": "s3://your-s3-bucket/output"
    },
    "StoppingCondition": {
        "MaxRuntimeInSeconds": 30
    }
}
```

SubmitServiceJob API を使用してトレーニングジョブを送信します。

```
aws batch submit-service-job \
    --job-queue my-sm-training-fifo-jq \
    --job-name my-batch-sm-job \
    --service-job-type SAGEMAKER_TRAINING \
    --retry-strategy attempts=1 \
    --timeout-config attemptDurationSeconds=60 \
    --service-request-payload file://my_training_job.json
```

出力:

```
{
    "jobArn": "arn:aws:batch:your-region:your-account-id:service-job/your-job-id",
    "jobName": "my-batch-sm-job",
    "jobId": "your-job-id"
}
```

サービスジョブのペイロードの詳細については、「the section called “サービスジョブのペイロード”」を参照してください。サービスジョブの送信の詳細については、「the section called “サービスジョブを送信する”」を参照してください。

ステップ 5: ジョブステータスをモニタリングする

DescribeServiceJob、GetJobQueueSnapshot の AWS Batch APIs を使用してトレーニングジョブをモニタリングできます。 ListServiceJobs このセクションでは、ジョブのステータスとキュー情報を確認するさまざまな方法を示します。

キューで実行中のジョブを表示します。

```
aws batch list-service-jobs \
    --job-queue my-sm-training-fifo-jq --job-status RUNNING
```

出力:

```
{
    "jobSummaryList": [
        {
            "latestAttempt": {
                "serviceResourceId": {
                    "name": "TrainingJobArn",
                    "value": "arn:aws:sagemaker:your-region:your-account-id:training-
job/AWSBatch<my-simple-training-job><your-attempt-id>"
                }
            },
            "createdAt": 1753718760,
            "jobArn": "arn:aws:batch:your-region:your-account-id:service-job/your-job-
id",
            "jobId": "your-job-id",
            "jobName": "my-batch-sm-job",
            "serviceJobType": "SAGEMAKER_TRAINING",
            "status": "RUNNING",
            "startedAt": 1753718820
        }
    ]
}
```

RUNNABLE 状態にあるジョブを表示します。

```
aws batch list-service-jobs \
    --job-queue my-sm-training-fifo-jq --job-status RUNNABLE
```

キュー内の今後のジョブのスナップショットを取得します。

```
aws batch get-job-queue-snapshot --job-queue my-sm-training-fifo-jq
```

出力:

```
{
    "frontOfQueue": {
        "jobs": [
            {
                "jobArn": "arn:aws:batch:your-region:your-account-id:service-job/your-
job-id",
                "earliestTimeAtPosition": 1753718880
```

```
            },
            {
                "jobArn": "arn:aws:batch:your-region:your-account-id:service-job/your-job-id-2",
                "earliestTimeAtPosition": 1753718940
            }
        ],
        "lastUpdatedAt": 1753718970
    }
}
```

名前でジョブを検索します。

```
aws batch list-service-jobs \
    --job-queue my-sm-training-fifo-jq \
    --filters name=JOB_NAME,values="my-batch-sm-job"
```

出力:

```
{
    "jobSummaryList": [
        {
            "latestAttempt": {
                "serviceResourceId": {
                    "name": "TrainingJobArn",
                    "value": "arn:aws:sagemaker:your-region:your-account-id:training-job/AWSBatch<my-simple-training-job><your-attempt-id>"
                }
            },
            "createdAt": 1753718760,
            "jobArn": "arn:aws:batch:your-region:your-account-id:service-job/your-job-id",
            "jobId": "your-job-id",
            "jobName": "my-batch-sm-job",
            "serviceJobType": "SAGEMAKER_TRAINING",
            "status": "RUNNING"
        }
    ]
}
```

ジョブ状態のマッピングの詳細については、「the section called “サービスジョブのステータス”」を参照してください。

ステップ 6: ジョブ出力を表示する

ジョブが完了すると、 と SageMaker AI APIs の両方を通じて、その出力 AWS Batch と口グを表示できます。

ジョブの詳細については、以下を参照してください AWS Batch。

```
aws batch describe-service-job \
    --job-id your-job-id
```

出力:

```
{
    "attempts": [
        {
            "serviceResourceId": {
                "name": "TrainingJobArn",
                "value": "arn:aws:sagemaker:your-region:your-account-id:training-job/
AWSBatch<my-simple-training-job><your-attempt-id>"
            },
            "startedAt": 1753718820,
            "stoppedAt": 1753718880,
            "statusReason": "Received status from SageMaker: Training job completed"
        }
    ],
    "createdAt": 1753718760,
    "jobArn": "arn:aws:batch:your-region:your-account-id:service-job/your-job-id",
    "jobId": "your-job-id",
    "jobName": "my-batch-sm-job",
    "jobQueue": "arn:aws:batch:your-region:your-account-id:job-queue/my-sm-training-
fifo-jq",
    "latestAttempt": {
        "serviceResourceId": {
            "name": "TrainingJobArn",
            "value": "arn:aws:sagemaker:your-region:your-account-id:training-job/
AWSBatch<my-simple-training-job><your-attempt-id>"
        }
    },
    "retryStrategy": {
        "attempts": 1,
        "evaluateOnExit": []
    },
    "serviceRequestPayload": "your-training-job-request-json",
```

```
    "serviceJobType": "SAGEMAKER_TRAINING",
    "startedAt": 1753718820,
    "status": "SUCCEEDED",
    "statusReason": "Received status from SageMaker: Training job completed",
    "stoppedAt": 1753718880,
    "tags": {},
    "timeoutConfig": {
        "attemptDurationSeconds": 60
    }
}
```

このコマンドは、SageMaker AI を介してジョブに直接アクセスするために使用できる SageMaker トレーニングジョブ ARN を含む包括的なジョブ情報を返します。

```
aws sagemaker describe-training-job \
    --training-job-name AWSBatch<my-simple-training-job><your-attempt-id>
```

トレーニングジョブの CloudWatch ログを表示するには、まずログストリーム名を取得します。

```
aws logs describe-log-streams \
    --log-group-name /aws/sagemaker/TrainingJobs \
    --log-stream-name-prefix AWSBatchmy-simple-training-job
```

出力:

```
{
    "logStreams": [
        {
            "logStreamName": "your-log-stream-name",
            "creationTime": 1753718830,
            "firstEventTimestamp": 1753718840,
            "lastEventTimestamp": 1753718850,
            "lastIngestionTime": 1753718860,
            "uploadSequenceToken": upload-sequence-token,
            "arn": "arn:aws:logs:your-region:your-account-id:log-group:/aws/sagemaker/
TrainingJobs:log-stream:AWSBatch<my-simple-training-job><your-attempt-id>/algo-1-algo-
id",
            "storedBytes": 0
        }
    ]
}
```

次に、前のレスポンスのログストリーム名を使用してログを取得します。

```
aws logs get-log-events \
    --log-group-name /aws/sagemaker/TrainingJobs \
    --log-stream-name your-log-stream-name
```

出力:

```
{
    "events": [
        {
            "timestamp": 1753718845,
            "message": "hello world",
            "ingestionTime": 1753718865
        }
    ],
    "nextForwardToken": "next-forward-token",
    "nextBackwardToken": "next-backward-token"
}
```

ログ出力には、トレーニングジョブからの「hello world」メッセージが表示され、ジョブが正常に実行されたことを確認します。

ステップ 7: チュートリアルのリソースをクリーンアップする

このチュートリアルを完了したら、不要な課金が発生しないよう、作成したリソースをクリーンアップします。

まず、ジョブキューを無効にして削除します。

```
aws batch update-job-queue \
    --job-queue my-sm-training-fifo-jq \
    --state DISABLED
```

ジョブキューが無効になるまで待ってから、それを削除します。

```
aws batch delete-job-queue \
    --job-queue my-sm-training-fifo-jq
```

次に、サービス環境を無効にして削除します。

```
aws batch update-service-environment \
    --service-environment TutorialServiceEnvironment \
    --state DISABLED
```

サービス環境が無効になるまで待ってから、それを削除します。

```
aws batch delete-service-environment \
    --service-environment TutorialServiceEnvironment
```

その他のリソース

チュートリアルが完了したら、次のトピックを試すことができます。

- PySDK にはヘルパークラスとユーティリティがあるため、サービスジョブの作成とジョブキューへの送信に PySDK を使用することをお勧めします。PySDK の使用例については、GitHub の「SageMaker AI の例」を参照してください。
- the section called “サービスジョブ” の詳細を確認してください。
- より複雑なトレーニングジョブ設定については、「のサービスジョブペイロード AWS Batch」を参照してください。
- でサービスジョブを送信する AWS Batch と `SubmitServiceJob` API について説明します。
- ジョブ状態の遷移を理解するには、「AWS Batch サービスジョブのステータスを SageMaker AI ステータスにマッピングする」を参照してください。
- Python を使用して SageMaker トレーニングジョブを作成および送信するその他の機能豊富な方法については、「SageMaker AI Python SDK ドキュメント」を参照してください。
- より複雑な機械学習ワークフローについては、「SageMaker サンプルノートブック」をご覧ください。

AWS Batch ウィジェットダッシュボード

AWS Batch ダッシュボードで最近のジョブ、ジョブキュー、コンピューティング環境を監視できます。デフォルトでは、以下のダッシュボードウィジェットが表示されます。

- ジョブの概要 – AWS Batchジョブの詳細については、ジョブを参照してください。
- ジョブキューの概要 – AWS Batch ジョブキューの詳細については、ジョブキュー を参照してください。
- コンピューティング環境の概要 — AWS Batch コンピューティング環境の詳細については、のコンピューティング環境 AWS Batch を参照してください。

ダッシュボードページに表示されるウィジェットをカスタマイズできます。以下のセクションでは、追加できるその他のウィジェットについて説明します。

トピック

- AWS Batch ダッシュボードに単一ジョブキューのウィジェットを追加する方法
- CloudWatch Container Insights のウィジェットを AWS Batch ダッシュボードに追加する方法
- AWS Batch ダッシュボードにジョブログのウィジェットを追加する方法

AWS Batch ダッシュボードに単一ジョブキューのウィジェットを追加する方法

単一ジョブキューのウィジェットは、単一ジョブキューの詳細情報を表示します。

このウィジェットを追加するには、次の手順に従います。

1. AWS Batch コンソールを開きます。
2. ナビゲーションバーから、ご希望の AWS リージョン を選択します。
3. ナビゲーションペインで、ダッシュボードを選択します。
4. ウィジェットの追加 を選択します。
5. 単一ジョブキュー では、ウィジェットを追加 を選択します。
6. ジョブキュー では、表示するジョブキューを選択します。
7. ジョブステータス では、表示するジョブステータスを選択します。

8. (オプション) 接続されたコンピューティング環境を表示 をオフにすると、コンピューティング環境のプロパティを非表示にできます。
9. コンピューティング環境のプロパティ で、必要なプロパティを選択します。
10. [Add] (追加) を選択します。

CloudWatch Container Insights のウィジェットを AWS Batch ダッシュボードに追加する方法

このウィジェットには、AWS Batch コンピューティング環境とジョブの集計メトリックが表示されます。コンテナインサイトの詳細については、the section called “CloudWatch コンテナインサイト”を参照してください。

このウィジェットを追加するには、次の手順に従います。

1. AWS Batch コンソールを開きます。
2. ナビゲーションバーから、ご希望の AWS リージョン を選択します。
3. ナビゲーションペインで、ダッシュボードを選択します。
4. ウィジェットの追加 を選択します。
5. Container insights では、ウィジェットを追加 を選択します。
6. コンピューティング環境 では、表示するコンピューティング環境を選択します。
7. [Add] (追加) を選択します。

AWS Batch ダッシュボードにジョブログのウィジェットを追加する方法

このウィジェットは、ジョブからのさまざまなログを一箇所の便利な場所に表示します。ジョブのログの詳細については、the section called “CloudWatch Logs でジョブログを表示する”を参照してください。

このウィジェットを追加するには、次の手順に従います。

1. AWS Batch コンソールを開きます。
2. ナビゲーションバーから、ご希望の AWS リージョン を選択します。
3. ナビゲーションペインで、ダッシュボードを選択します。

4. ウィジェットの追加 を選択します。
5. ジョブのログでは、ウィジェットを追加を選択します。
6. ジョブ ID で、必要なジョブのジョブ ID を入力します。
7. 追加を選択します。

のコンピューティング環境 AWS Batch

ジョブキューは、1 つ以上のコンピューティング環境にマッピングされます。コンピューティング環境には、コンテナ化されたバッチジョブを実行するための Amazon ECS コンテナインスタンスが含まれています。特定のコンピューティング環境を 1 つ以上のジョブキューにマッピングすることもできます。ジョブキュー内では、関連するコンピュート環境はそれぞれ順番を持っており、スケジューラは実行準備が整ったジョブをどこで実行するかを決定するのに使用します。最初のコンピュート環境のVALIDステータスが、利用可能なリソースがある場合、ジョブはそのコンピュート環境内のコンテナインスタンスにスケジューリングされる。最初のコンピュート環境のINVALIDステータスが、適切なコンピュートリソースを提供できないか、または提供できない場合、スケジューラは次のコンピュート環境でジョブを実行しようとします。

トピック

マネージド型のコンピューティング環境

マネージド型のコンピューティング環境を使用して、 が環境内のコンピューティングリソースの容量とインスタンスタイプ AWS Batch を管理できます。これは、コンピュート環境の作成時に定義したコンピュートリソースの仕様に基づいています。Amazon EC2 オンデマンドインスタンスと Amazon EC2 スポットインスタンスを使用するかを選択できます。または、マネージド型のコンピューティング環境で Fargate および Fargate スポット容量を使用することもできます。スポットイ

ンスタンスを使用する場合、オプションで上限価格を設定できます。こうすることで、スポット・インスタンスは、スポットインスタンス価格がオンデマンド価格の指定されたパーセンテージを下回った場合にのみ起動する。

> **Important**
>
> Fargate スポットインスタンスは ではサポートされていません Windows containers on AWS Fargate。FargateWindows のジョブが、Fargate Spot のコンピュート環境のみを利用するジョブキューに投入された場合、ジョブキューはブロックされます。

> **Important**
>
> AWS Batch は、Amazon EC2 起動テンプレート、Amazon EC2 Auto Scaling グループ、Amazon EC2 スポットフリート、Amazon ECS クラスターなど、ユーザーに代わってアカウント内で複数の AWS リソースを作成および管理します。これらのマネージドリソースは、最適な AWS Batch オペレーションを確保するために特別に設定されています。これらのバッチマネージドリソースを手動で変更すると、 AWS Batch ドキュメントに明示的に記載されていない限り、予期しない動作が発生し、`INVALID` コンピューティング環境、最適ではないインスタンススケーリング動作、ワークロード処理の遅延、予期しないコストが発生する可能性があります。これらの手動変更を AWS Batch サービスで決定的にサポートすることはできません。コンピューティング環境を管理するには、サポートされている Batch API または Batch コンソールを必ず使用してください。

マネージドコンピューティング環境は、指定した VPC とサブネットに Amazon EC2 インスタンスを起動し、Amazon ECS クラスターに登録します。Amazon EC2インスタンスは、Amazon ECS サービスエンドポイントと通信するために外部ネットワークアクセスが必要です。一部のサブネットでは、Amazon EC2インスタンスにパブリックIPアドレスを提供していない。Amazon EC2インスタンスがパブリックIPアドレスを持っていない場合、このアクセスを得るためにネットワークアドレス変換 (NAT) を使用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NAT ゲートウェイを参照してください。VPC の作り方の詳細については、仮想プライベートクラウドを作成するを参照してください。

デフォルトでは、 AWS Batch マネージド型コンピューティング環境は、コンピューティングリソース用に最新の承認済みバージョンの Amazon ECS 最適化 AMI を使用します。ただし、さまざまな理由により、マネージド型のコンピューティング環境で使用する AMI を独自に作成する場合もあります。詳細については、「コンピューティングリソースの AMI」を参照してください。

Note

AWS Batch は、作成後にコンピューティング環境の AMIs を自動的にアップグレードしません。例えば、Amazon ECS最適化AMIの新しいバージョンがリリースされても、コンピュート環境のAMIは更新されません。ゲストオペレーティングシステムの管理はユーザーの責任です。これには、アップデートとセキュリティパッチが含まれます。また、コンピューティングリソースにインストールするその他のアプリケーションソフトウェアやユーティリティについても責任を負うものとします。 AWS Batch ジョブに新しい AMI を使用する方法は 2 つあります。オリジナルの方法は、次のステップを完了することです。

1. 新しい AMI を使用して新しいコンピューティング環境を作成します。
2. コンピューティング環境を既存のジョブキューに追加します。
3. 古いコンピューティング環境をジョブキューから削除します。
4. 以前のコンピューティング環境を削除します。

2022 年 4 月に、コンピューティング環境の更新に対する拡張サポート AWS Batch が追加されました。詳細については、「AWS Batch でコンピューティング環境を更新する」を参照してください。コンピューティング環境の拡張アップデートを使用して AMI を更新するには、次のルールに従います。

- サービスロール(serviceRole)パラメータを設定しないか、AWSServiceRoleForBatch サービス連動ロールに設定します。
- 割り当て戦略 (allocationStrategy) パラメータを、BEST_FIT_PROGRESSIVE、SPOT_CAPACITY_OPTIMIZED または SPOT_PRICE_CAPACITY_OPTIMIZEDに設定します。
- 最新のイメージバージョンへの更新 (updateToLatestImageVersion) パラメータを true に設定します。
- imageId、imageIdOverride(ec2Configuration) または起動テンプレート (launchTemplate) には AMI ID を指定しないでください。この場合、 はインフラストラクチャの更新が開始された AWS Batch 時点で でサポートされている最新の Amazon ECS 最適化 AMI AWS Batch を選択します。または、imageIdまたはimageIdOverrideパラメータで AMI ID を指定するか、LaunchTemplateプロパティで識別されるローンチテンプレートを指定することもできます。これらのプロパティのいずれかを変更すると、インフラストラクチャの更新が開始されます。AMI ID が起動テンプレートで指定されている場合、imageId または imageIdOverride パラメータで AMI ID を指定して置き換えることはできません。別の起動テンプレートを指定することでのみ置き換えることができま

す。$Default または $Latest、起動テンプレートのバージョンがまたはに設定されている場合は、起動テンプレートの新しいデフォルトバージョンを設定するか(設定されている場合 $Default)、起動テンプレートに新しいバージョンを追加(ある場合 $Latest)します。

これらのルールに従うと、インフラストラクチャの更新をトリガーする更新により、AMI IDが再選択されます。起動テンプレートversionの (launchTemplate)設定が$Latestまたは$Defaultに設定されている場合、launchTemplateを更新していない場合でも、インフラストラクチャの更新時に起動テンプレートの最新バージョンまたはデフォルトバージョンが評価されます。

マルチノード並列ジョブを作成する際の考慮事項

AWS Batch では、マルチノード並列 (MNP) ジョブと非 MNP ジョブを実行するための専用のコンピューティング環境を作成することをお勧めします。これは、マネージドコンピュート環境におけるコンピューティングキャパシティの作り方によるものです。新しいマネージドコンピューティング環境を作成するときに、ゼロより大きいminvCpu値を指定すると、 は非 MNP ジョブでのみ使用するインスタンスプール AWS Batch を作成します。マルチノード並列ジョブが送信されると、 はマルチノード並列ジョブを実行する新しいインスタンス容量 AWS Batch を作成します。単一ノードとマルチノードの両方の並列ジョブが、 minvCpusまたは maxvCpus値が設定された同じコンピューティング環境で実行されている場合、必要なコンピューティングリソースが使用できない場合 AWS Batch 、 は現在のジョブが終了するのを待ってから、新しいジョブを実行するために必要なコンピューティングリソースを作成します。

アンマネージド型のコンピューティング環境

アンマネージド型のコンピューティング環境では、独自のコンピューティングリソースを管理します。 は Amazon ECS と Amazon EKS の両方のアンマネージド型のコンピューティング環境 AWS Batch をサポートしているため、Batch のジョブスケジューリング機能を活用しながら、インフラストラクチャの制御を維持できます。

> **Note**
>
> AWS Fargate リソースは、アンマネージド型のコンピューティング環境ではサポートされていません。

アンマネージド Amazon ECS コンピューティング環境

アンマネージド Amazon ECS コンピューティング環境では、コンピューティングリソースに使用する AMI が Amazon ECS コンテナインスタンス AMI 仕様を満たしていることを確認する必要があります。詳細については、「コンピューティングリソースの AMI 仕様」および「チュートリアル: コンピューティングリソース AMI を作成する」を参照してください。

アンマネージ型のコンピューティング環境を作成したら、DescribeComputeEnvironments API オペレーションを使用してコンピューティング環境の詳細を確認します。環境に関連付けられている Amazon ECS クラスターを見つけ、その Amazon ECS クラスター内で手動でコンテナインスタンスを起動します。

次の AWS CLI コマンドは、Amazon ECS クラスター ARN も提供します。

```
$ aws batch describe-compute-environments \
    --compute-environments unmanagedCE \
    --query "computeEnvironments[].ecsClusterArn"
```

詳細については、Amazon Elastic Container Service デベロッパーガイドのAmazon ECS コンテナインスタンスの起動を参照してください。コンピュートリソースを起動する際、リソースが登録するAmazon ECSクラスターARNを以下のAmazon EC2ユーザーデータで指定します。*ecsClusterArn* を前のコマンドで取得したクラスター ARN に置き換えます。

```
#!/bin/bash
echo "ECS_CLUSTER=ecsClusterArn" >> /etc/ecs/ecs.config
```

アンマネージド Amazon EKS コンピューティング環境

アンマネージド型の Amazon EKS コンピューティング環境では、 がジョブのスケジュールと配置 AWS Batch を処理しながら、独自の Kubernetes ノードを管理します。これにより、Kubernetes インフラストラクチャのセキュリティ、コンプライアンス、または運用要件を直接制御できます。は既存の Amazon EKS クラスターと AWS Batch 統合してジョブをスケジュールして実行しながら、Amazon EKS ノードのプロビジョニングと設定を行う責任があります。

詳細については、「チュートリアル: Amazon EKS リソースを使用してアンマネージド型のコンピューティング環境を作成する」を参照してください。

コンピューティング環境を作成する

でジョブを実行する前に AWS Batch、コンピューティング環境を作成する必要があります。が仕様に基づいて環境内の Amazon EC2 インスタンスまたは AWS Fargate リソース AWS Batch を管理するマネージドコンピューティング環境を作成できます。あるいは、アンマネージド型のコンピューティング環境を構築し、Amazon EC2 インスタンスの設定を環境内で処理することも可能です。

> **Important**
>
> Fargate Spot インスタンスは、以下のシナリオではサポートされません。
>
> - Windows containers on AWS Fargate
>
> このようなシナリオでは、Fargate Spot コンピューティング環境のみを使用するジョブキューにジョブが送信されると、ジョブキューはブロックされます。

トピック

チュートリアル: Fargate リソースを使用してマネージド型のコンピューティング環境を作成する

以下のステップを実行し、AWS Fargate リソースを使用してマネージドコンピューティング環境を作成します。

1. AWS Batch コンソールを https://console.aws.amazon.com/batch/ で開きます。
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. ナビゲーションペインで、コンピューティング環境] を選択します。
4. [作成] を選択します。
5. コンピューティング環境を設定します。

   > **Note**
   >
   > Windows containers on AWS Fargate ジョブ用のコンピューティング環境では、少なくとも 1 つの vCPU を指定する必要があります。

   a. コンピュート環境設定 で、Fargate を選択します。
   b. 名前 で、コンピューティング環境の一意な名前を指定します。名前の長さは最大 128 文字です。大文字、小文字、数字、ハイフン (-)、アンダースコア (_) を含めることができます。
   c. サービスロール で、AWS Batch サービスがユーザーに代わって、必要な AWS APIオペレーションを呼び出せるようにするサービスにリンクされたロールを選択します。例えば、AWSServiceRoleForBatch を選択します。詳細については、AWS Batch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。
   d. (オプション) タグ を展開します。タグを追加するには、タグの追加 を選択します。次に、キー 名を入力し、オプションで 値 を入力します。タグを追加] を選択します。
   e. 次のページ を選択します。
6. インスタンス設定 セクション:

   a. (オプション) Fargate Spot キャパシティを使用 では、Fargate Spot をオンにします。Fargate Spot については、Amazon EC2 スポットと Fargate_Spot を使用するを参照してください。
   b. 最大 vCPU] では、ジョブキューの需要にかかわらず、コンピューティング環境でスケールアウトできる vCPU の最大数を選択します。
   c. 次のページ を選択します。
7. ネットワーキングを設定します。

> **Important**
>
> コンピューティングリソースには、Amazon ECS サービスエンドポイントと通信するために外部ネットワークアクセスが必要です。これは、インターフェイス VPC エンドポイントを介して、またはパブリック IP アドレスを持つコンピュートリソースを通じて可能になります。
> インターフェイス VPC エンドポイントの詳細については、Amazon Elastic Container Service 開発者ガイドのAmazon ECS インターフェイス VPC エンドポイント (AWS PrivateLink) を参照してください。
> インターフェイス VPC エンドポイントが設定されておらず、コンピューティングリソースがパブリック IP アドレスを持たない場合は、ネットワークアドレス変換 (NAT) を使用してこのアクセスを提供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 NAT ゲートウェイを参照してください。詳細については、the section called “VPC を作成する”を参照してください。

a. 仮想プライベートクラウド (VPC) ID では、インスタンスを起動する VPC を選択します。

b. サブネット では、使用するサブネットを選択します。デフォルトでは、選択した VPC 内のすべてのサブネットを使用できます。

> **Note**
>
> Fargate の AWS Batch では、ローカルゾーンがサポートされていません。詳細については、Amazon Elastic Container Service 開発者ガイドのAmazon ECS クラスターのローカルゾーン、波長ゾーン、および AWS Outpostsを参照してください。

c. セキュリティグループ] では、インスタンスにアタッチするセキュリティグループを選択します。デフォルトでは、VPC のデフォルトのセキュリティグループが選択されます。

d. 次のページ を選択します。

8. レビュー では、設定手順を確認します。変更する必要がある場合は、[編集] を選択します。完了したら、コンピューティング環境の作成 を選択します。

チュートリアル: Amazon EC2 リソースを使用して、マネージド型のコンピューティング環境を作成する

Amazon Elastic Compute Cloud (Amazon EC2) リソースを使用してマネージドコンピューティング環境を作成するには、次の手順を実行します。

1. AWS Batch コンソールを https://console.aws.amazon.com/batch/ で開きます。
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. ナビゲーションペインで [Environments (環境)] を選択します。
4. [環境の作成] を選択した後、[コンピューティング環境] を選択します。
5. 環境を設定します。

    a. コンピューティング環境設定 で、Amazon Elastic Compute Cloud (Amazon EC2) を選択します。

    b. オーケストレーションタイプ で、マネージド を選択します。

    c. 名前 で、コンピューティング環境の一意な名前を指定します。名前の長さは最大 128 文字です。大文字、小文字、数字、ハイフン (-)、アンダースコア (_) を含めることができます。

    d. サービスロールで、AWS Batch サービスがユーザーに代わって必要な AWS API オペレーションを呼び出せるようにする、サービスにリンクされたロールを選択します。例えば、AWSServiceRoleForBatch を選択します。詳細については、AWS Batch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。

    e. インスタンスロールの場合、新しいインスタンスプロファイルを作成するか、必要な IAM アクセス許可がアタッチされた既存のプロファイルを使用するかを選択します。このインスタンスプロファイルを使用すると、コンピューティング環境用に作成した Amazon ECS コンテナインスタンスにより、必要な AWS API オペレーションを呼び出すことができます。詳細については、Amazon ECS インスタンスロールを参照してください。新しいインスタンスプロファイルを作成することを選択した場合は、必要なロール (`ecsInstanceRole`) が作成されます。

    f. (オプション) タグ を展開します。

        i. (オプション) EC2 タグ で、タグを追加 を選択して、コンピューティング環境で起動されるリソースにタグを追加します。次に、キー 名を入力し、オプションで 値 を入力します。[タグを追加] を選択します。

        ii. (オプション) タグ には タグを追加 を選択します。次に、キー 名を入力し、オプションで 値 を入力します。[タグを追加] を選択します。

詳細については、AWS Batch リソースのタグ付け を参照してください。

g. [次へ] を選択します。

6. インスタンス設定 セクション:

a. (オプション) スポットインスタンスの使用を有効にする でスポットをオンにします。詳細については、スポットインスタンスを参照してください。

b. (スポットの場合のみ) 上限のオンデマンド価格の割合 (%) で、インスタンス起動前のインスタンスタイプのオンデマンド価格と対比したスポットインスタンス価格の最大パーセンテージを選択します。例えば、上限価格が 20% の場合、その EC2 インスタンスのスポット料金は現在のオンデマンド料金の 20% 未満にする必要があります。支払い額は常に最低 (市場料金) となり、最大パーセンテージを超えることはありません。このフィールドを空のままにした場合、デフォルト値はオンデマンド料金の 100% です。

c. (スポットの場合のみ) スポットフリートロール で、既存の Amazon EC2 スポットフリート IAM ロールを選択してスポットコンピューティング環境に適用します。既存の Amazon EC2 スポットフリート IAM ロールが存在しない場合には、まずこのロールを作成する必要があります。詳細については、Amazon EC2 スポットフリートロールを参照してください。

> **Important**
>
> 作成時にスポットインスタンスにタグを付けるには、Amazon EC2 スポットフリートの IAM ロールに、より新しい AmazonEC2SpotFleetTaggingRole 管理ポリシーを使用する必要があります。AmazonEC2SpotFleetRole 管理ポリシーには、スポットインスタンスにタグを付けるために必要なアクセス許可はありません。詳細については、「作成時にタグが付けられていないスポットインスタンス」および「the section called “リソースのタグ付け”」を参照してください。

d. 最小 vCPU で、ジョブキューの需要にかかわらず、コンピューティング環境で維持する vCPU の最小数を選択します。

e. 必要な vCPU で、コンピューティング環境の起動に必要な vCPU の数を選択します。ジョブキューの需要が増えると、AWS Batch はコンピューティング環境で必要な vCPU の数を増やし、vCPU の最大数まで EC2 インスタンスを追加できます。需要が減ると、AWS Batch はコンピューティング環境で必要な vCPU の数を減らし、vCPU の最小数までインスタンスを削減できます。

f. [最大 vCPU] では、ジョブキューの需要にかかわらず、コンピューティング環境でスケールアウトできる vCPU の最大数を選択します。

g. 許可されたインスタンスタイプ] では、起動できる Amazon EC2 インスタンスタイプを選択します。インスタンスファミリーを指定してそのファミリー内のいずれかのインスタンスタイプ (c5、c5n、p3など) を起動できます。または、ファミリー内の特定のサイズを指定することもできます (c5.8xlarge)。メタルインスタンスタイプはインスタンスファミリーに含まれていません。例えば、c5 は c5.metal を含んでいません。

次のいずれかを選択した場合、AWS Batch はインスタンスタイプを選択できます。

- optimal を選択すると、ジョブキューの需要に見合ったインスタンスタイプを (c4、m4、r4、c5、m5 および r5 インスタンスファミリーから) 選択します。
- default_x86_64 を選択すると、ジョブキューのリソース需要に見合った x86 ベースのインスタンスタイプを (m6i、c6i、r6i、c7i インスタンスファミリーから) 選択します。
- default_arm64 を選択すると、ジョブキューのリソース需要に見合った x86 ベースのインスタンスタイプを (m6g、c6g、r6g、c7g インスタンスファミリーから) 選択します。

> **Note**
>
> 2025 年 11 月 1 日以降、optimal の動作は default_x86_64 と一致するように変更されます。変更中、インスタンスファミリーは新しい世代に更新される可能性があります。アップグレードを実行するためにアクションを実行する必要はありません。変更に関する詳細については、「
>
> > **Note**
> >
> > 2025 年 11 月 1 日以降、optimal の動作は default_x86_64 と一致するように変更されます。変更
>
> 中、インスタンスファミリーは新しい世代に更新される
>
> 可能性があります。アップグレードを実行するためにアクションを実行する必要はありません。
>
> AWS Batch は、ジョブキューの需要optimalに合わせて、の instanceTypes で 1 つのオプションをサポートしまし

た。default_x86_64 と default_arm64 の 2 つの新しいインスタンスタイプのオプションが導入されました。インスタンスタイプを選択しない場合、default_x86_64 が使用されます。これらの新しいオプションでは、ジョブキューの要件に基づいて、さまざまなファミリーや世代にわたって費用対効果の高いインスタンスタイプが自動的に選択されるため、ワークロードをすばやく実行できます。

で新しいインスタンスタイプの十分な容量が利用可能になると AWS リージョン、対応するデフォルトプールは新しいインスタンスタイプで自動的に更新されます。既存の optimal オプションは引き続きサポートされ、今後更新されたインスタンスを提供するために基盤となるデフォルトプールでサポートされるため、廃止されません。'optimal を使用している場合、ユーザー側でアクションは必要ありません。

ただし、ENABLED および VALID コンピューティング環境 (CEs) のみが新しいインスタンスタイプで自動的に更新されることに注意してください。DISABLED または INVALID CE がある場合は、再有効化されて VALID状態に設定されると、更新を受け取ります。

」を参照してください。

Note

- インスタンスファミリーの可用性は、AWS リージョン によって異なります。例えば、一部の AWS リージョン には第 4 世代のインスタンスファミリーがないものの、第 5 世代と第 6 世代のインスタンスファミリーがある場合があります。
- default_x86_64 または default_arm64 インスタンスバンドルを使用する場合、AWS Batch はコスト効率とパフォーマンスのバランスに基づいてインスタンスファミリーを選択します。新世代のインスタンスでは多くの場合、価格パフォーマンスが向上しますが、ワークロードの可用性、コスト、パフォーマンスの最適な組み合わせを提供する場合、AWS Batch は旧世代のインスタンスファミリーを選択することがあります。例えば、c6i インスタンスと c7i インスタンスの両方が利用可能な AWS リージョン では、特定のジョブ要件に対してコスト効率が向上する場合、AWS Batch は c6i インスタンスを選択することがありま

す。AWS Batch インスタンスタイプと AWS リージョン の可用性の詳細については、「インスタンスタイプのコンピューティングテーブル」を参照してください。

- AWS Batch は、デフォルトのバンドル内のインスタンスを、よりコスト効率の高い新しいオプションに定期的に更新します。更新は、ユーザーからのアクションを必要とせずに自動的に行われます。ワークロードは、更新中も中断せずに実行され続けます。

> **Note**
>
> コンピューティング環境を作成する際、そのコンピューティング環境で選択するインスタンスタイプで同じアーキテクチャを使用する必要があります。例えば、x86 と ARM インスタンスを同じコンピューティング環境で使用することはできません。

> **Note**
>
> AWS Batch は、ジョブキューに必要な容量に基づいて GPU をスケールします。GPU スケジューリングを使用するには、コンピューティング環境に p6、p3、p4、p5、g3、g3s、g4、g5、または g6 ファミリーのインスタンスタイプを含める必要があります。

h. [配分戦略] では、許可されるインスタンスタイプのリストからインスタンスタイプを選択するときに使用する配分戦略を選択します。EC2 のオンデマンドコンピューティング環境では BEST_FIT_PROGRESSIVE を、EC2 のスポットコンピューティング環境では SPOT_CAPACITY_OPTIMIZED または SPOT_PRICE_CAPACITY_OPTIMIZED を選択するのが一般的です。詳細については、the section called “インスタンスタイプの配分戦略” を参照してください。

i. [Additional configuration] (追加設定) を展開します。

   i. (オプション) プレイスメントグループ では、プレイスメントグループ名を入力して、コンピューティング環境内のリソースをグループ化します。

ii. (オプション) EC2 キーペア で、インスタンス接続時のセキュリティ認証情報としてパブリックキーとプライベートキーのペアを選択します。Amazon EC2 キーペアの詳細については、Amazon EC2 キーペアおよび Linux インスタンスを参照してください。

iii. (オプション) [EC2 設定] では、AWS Batch がコンピューティング環境のインスタンスにAmazon Machine Images (AMI) を選択するための情報を提供するために、[イメージタイプ] とイメージ ID オーバーライド] の値を選択します。各 イメージタイプ] で イメージ ID のオーバーライド] が指定されていない場合、AWS Batch は Amazon ECS に最適化された AMI を選択します。イメージタイプ を指定しない場合、デフォルトは非 GPU、非 AWS Graviton インスタンス用の Amazon Linux 2 になります。

> **Important**
>
> カスタム AMI を使用するには、イメージタイプを選択し、イメージ ID のオーバーライド ボックスにカスタム AMI ID を入力します。

Amazon Linux 2

すべての AWS Graviton ベースのインスタンスファミリー (C6g、M6g、R6gおよびT4gなど) のデフォルトで、すべての非 GPU インスタンスタイプに使用できます。

Amazon Linux 2 (GPU)

すべての GPU インスタンスファミリー (P4 および G4 など) のデフォルトで、非 AWS Graviton ベースのすべてのインスタンスタイプに使用できます。

Amazon Linux 2023

AWS Batch は、Amazon Linux 2023 をサポートします。

> **Note**
>
> Amazon Linux 2023 は A1 インスタンスをサポートしていません。

Amazon Linux 2023 (GPU)

すべての GPU インスタンスファミリー (P4 および G4 など) のデフォルトで、非 AWS Graviton ベースのすべてのインスタンスタイプに使用できます。

> **Note**
>
> コンピューティング環境用に選択する AMI は、そのコンピューティング環境で使用するインスタンスタイプのアーキテクチャと一致している必要があります。例えば、コンピューティング環境で A1 インスタンスタイプを使用する場合、選択するコンピューティングリソース AMI で ARM インスタンスをサポートしている必要があります。Amazon ECS は、Amazon ECS に最適化された Amazon Linux 2 AMI の、x86 と ARM の両バージョンを提供しています。詳細については、Amazon Elastic Container Service デベロッパーガイドの「Amazon ECS に最適化された Amazon Linux 2 AMI」を参照してください。

j. (オプション) 起動テンプレートを展開する

   i. (オプション) [デフォルトの起動テンプレート] では、既存の Amazon EC2 起動テンプレートを選択して、コンピューティングリソースを設定します。テンプレートのデフォルトバージョンは自動的に設定されます。詳細については、で Amazon EC2 起動テンプレートを使用する AWS Batchを参照してください。

   > **Note**
   >
   > 起動テンプレートで、作成したカスタム AMI を指定できます。

   ii. (オプション) [デフォルトのバージョン] では、`$Default` または `$Latest` を入力するか、使用する特定のバージョン番号を指定します。

   > **Note**
   >
   > 注意: 置換変数 ($Default または $Latest) のいずれかを使用すると、この設定が保存された時点での現在のデフォルトまたは最新のバージョン番号が適用されます。今後、デフォルトバージョンまたは最新バージョンが変更された場合は、情報をアップデートする必要があります。自動的にはアップデートされません。

> **Important**
>
> 起動テンプレートのバージョンパラメータが `$Default` または `$Latest` の場合、指定された起動テンプレートのデフォルトまたは最新バージョンがインフラストラクチャの更新時に評価されます。デフォルトで別の AMI ID が選択されている場合、または起動テンプレートの最新バージョンが選択されている場合、その AMI ID が更新に使用されます。詳細については、the section called "インフラストラクチャの更新中の AMI の選択" を参照してください。

iii. (オプション) [起動テンプレートのオーバーライド] で、[起動テンプレートのオーバーライドを追加] を選択します。

A. (オプション) [起動テンプレート] で、特定のインスタンスタイプとファミリーに使用する既存の Amazon EC2 起動テンプレートを選択します。

B. (オプション) [デフォルトのバージョン] では、特定のバージョン番号を入力して `$Default` または `$Latest` を使用します。

> **Note**
>
> `$Default` または `$Latest`変数を使用する場合、AWS Batch はコンピューティング環境の作成時に現在の情報を適用します。今後、デフォルトバージョンまたは最新バージョンが変更された場合は、UpdateComputeEnvironment または AWS マネジメントコンソール - AWS Batch で情報をアップデートする必要があります。

C. (オプション) [ターゲットインスタンスタイプ] で、起動テンプレートのオーバーライドを適用するインスタンスタイプまたはファミリーを選択します。

> **Note**
>
> 起動テンプレートのオーバーライドを指定する場合は、[ターゲットインスタンスタイプ] が必要です。詳細については、「LaunchTemplateSpecificationOverride.targetInstanceTypes」を参照してください。

> **Note**
>
> 選択するインスタンスタイプまたはファミリーがこのリストに表示されない場合は、`Allowed instance types` で行った選択を確認します。

k. [次へ] を選択します。

7. ネットワーク設定 セクションで:

> **Important**
>
> コンピューティングリソースには、Amazon ECS サービスエンドポイントと通信するために外部ネットワークアクセスが必要です。これは、インターフェイス VPC エンドポイントを介して、またはパブリック IP アドレスを持つコンピュートリソースを通じて可能になります。
> インターフェイス VPC エンドポイントの詳細については、Amazon Elastic Container Service 開発者ガイドのAmazon ECS インターフェイス VPC エンドポイント (AWS PrivateLink) を参照してください。
> インターフェイス VPC エンドポイントが設定されておらず、コンピューティングリソースがパブリック IP アドレスを持たない場合は、ネットワークアドレス変換 (NAT) を使用してこのアクセスを提供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 NAT ゲートウェイを参照してください。詳細については、the section called “VPC を作成する”を参照してください。

a. 仮想プライベートクラウド (VPC) ID では、インスタンスを起動する先の VPC を選択します。

b. サブネット では、使用するサブネットを選択します。デフォルトでは、選択した VPC 内のすべてのサブネットを使用できます。

> **Note**
>
> Amazon EC2 における AWS Batch では、ローカルゾーンをサポートしています。詳細については、「Amazon EC2 ユーザーガイド」の「ローカルゾーン」、および「Amazon Elastic Container Service 開発者ガイド」の「Amazon ECS クラスターのローカルゾーン、波長ゾーン、および AWS Outposts」を参照してください。

c. (オプション) セキュリティグループ で、インスタンスにアタッチするセキュリティグループを選択します。デフォルトでは、VPC のデフォルトのセキュリティグループが選択されます。

> **Note**
>
> 注意: 置換変数 ($Default または $Latest) のいずれかを使用すると、この設定が保存された時点での現在のデフォルトまたは最新のバージョン番号が適用されます。今後、デフォルトバージョンまたは最新バージョンが変更された場合は、情報をアップデートする必要があります。自動的にはアップデートされません。

8. 次のページ を選択します。
9. レビュー では、設定手順を確認します。変更する必要がある場合は、[編集] を選択します。完了したら、コンピューティング環境の作成 を選択します。

チュートリアル: Amazon EC2 リソースを使用して、アンマネージド型のコンピューティング環境を作成する

Amazon Elastic Compute Cloud (Amazon EC2) リソースを使用してアンマネージドコンピューティング環境を作成するには、次のステップを実行します。

1. AWS Batch コンソールを https://console.aws.amazon.com/batch/ で開きます。
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. [コンピューティング環境] ページで、 [作成] を選択します。
4. 環境を設定します。
   a. コンピューティング環境設定 で、Amazon Elastic Compute Cloud (Amazon EC2) を選択します。
   b. オーケストレーションタイプ で、アンマネージド を選択します。
5. 名前 で、コンピューティング環境の一意な名前を指定します。名前の最大長は 128 文字です。大文字、小文字、数字、ハイフン (-)、アンダースコア (_) を含めることができます。
6. サービスロールで、AWS Batch サービスがユーザーに代わって、必要な AWS API オペレーションを呼び出せるようにするロールを選択します。

> **Note**
>
> アンマネージド型のコンピューティング環境では `AWSServiceRoleForBatch` を使用できません。

7. [最大 vCPU] では、ジョブキューの需要にかかわらず、コンピューティング環境でスケールアウトできる vCPU の最大数を選択します。
8. (オプション) タグ を展開します。タグを追加するには、タグの追加 を選択します。次に、キー名を入力し、オプションで 値 を入力します。[タグを追加] を選択します。詳細については、AWS Batch リソースのタグ付けを参照してください。
9. 次のページ を選択します。
10. レビュー では、設定手順を確認します。変更する必要がある場合は、[編集] を選択します。完了したら、コンピューティング環境の作成 を選択します。

チュートリアル: Amazon EKS リソースを使用してマネージド型のコンピューティング環境を作成する

Amazon Elastic Kubernetes Service (Amazon EKS) リソースを使用してマネージドコンピューティング環境を作成するには、次の手順を実行します。

1. AWS Batch コンソールを https://console.aws.amazon.com/batch/ で開きます。
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. ナビゲーションペインで、コンピューティング環境] を選択します。
4. 作成] を選択します。
5. コンピューティング環境の設定 で、Amazon Elastic Kubernetes Service (Amazon EKS) を選択します。
6. 名前 で、コンピューティング環境の一意な名前を指定します。名前の最大長は 128 文字です。大文字、小文字、数字、ハイフン (-)、アンダースコア (_) を含めることができます。
7. インスタンスロール では、必要な IAM アクセス許可がアタッチされた既存のインスタンスプロファイルを使用することを選択します。

> **Note**
>
> AWS Batch コンソールでコンピューティング環境を作成するには、`eks:ListClusters` および `eks:DescribeCluster` 権限を持つインスタンスプロファイルを選択します。

8. EKS クラスター で、既存の Amazon EKS クラスターを選択します。
9. 名前空間 で、クラスター内の Kubernetes プロセスをグループ化する AWS Batch 名前空間を入力します。
10. (オプション) タグ を展開します。タグの追加 を選択し、キーと値のペアを入力します。
11. 次のページ を選択します。
12. (オプション) EC2 スポットインスタンスを使用 で、スポットインスタンスの使用を有効にするをオンにして Amazon EC2 スポットインスタンスを使用します。
13. (スポットの場合のみ) 上限のオンデマンド価格の割合 (%) で、インスタンス起動前のインスタンスタイプのオンデマンド価格と対比したスポットインスタンス価格の最大パーセンテージを選択します。例えば、上限価格が 20% の場合、その EC2 インスタンスのスポット料金は現在のオンデマンド料金の 20% 未満にする必要があります。支払い額は常に最低 (市場料金) となり、最大パーセンテージを超えることはありません。このフィールドを空のままにした場合、デフォルト値はオンデマンド料金の 100% です。
14. (スポットのみ) スポットフリートロール で、SPOT コンピューティング環境用の Amazon EC2 スポットフリート IAM ロールを選択します。

> **Important**
>
> このロールは、配分戦略が `BEST_FIT` に設定されている場合、または指定されていない場合に必要です。

15. (オプション) 最小 vCPU で、ジョブキューの需要にかかわらず、コンピューティング環境で維持する vCPU の最小数を選択します。
16. (オプション) 最大 vCPU で、ジョブキューの需要にかかわらず、コンピューティング環境でスケールアウトできる vCPU の最大数を選択します。
17. 許可されたインスタンスタイプ] では、起動できる Amazon EC2 インスタンスタイプを選択します。インスタンスファミリーを指定してそのファミリー内のいずれかのインスタンスタイプ (c5、c5n、p3など) を起動できます。または、ファミリー内の特定のサイズを指定することも

できます (c5.8xlarge)。メタルインスタンスタイプはインスタンスファミリーに含まれていません。例えば、c5 は c5.metal を含んでいません。

次のいずれかを選択した場合、AWS Batch はインスタンスタイプを選択できます。

- optimal を選択すると、ジョブキューの需要に見合ったインスタンスタイプを (c4、m4、r4、c5、m5 および r5 インスタンスファミリーから) 選択します。
- default_x86_64 を選択すると、ジョブキューのリソース需要に見合った x86 ベースのインスタンスタイプを (m6i、c6i、r6i、c7i インスタンスファミリーから) 選択します。
- default_arm64 を選択すると、ジョブキューのリソース需要に見合った x86 ベースのインスタンスタイプを (m6g、c6g、r6g、c7g インスタンスファミリーから) 選択します。

> **Note**
>
> 2025 年 11 月 1 日以降、optimal の動作は default_x86_64 と一致するように変更されます。変更中、インスタンスファミリーは新しい世代に更新される可能性があります。アップグレードを実行するためにアクションを実行する必要はありません。変更に関する詳細については、「インスタンスファミリーの自動更新を受信するための最適なインスタンスタイプの設定」を参照してください。

> **Note**
>
> - インスタンスファミリーの可用性は、AWS リージョン によって異なります。例えば、一部の AWS リージョン には第 4 世代のインスタンスファミリーがないものの、第 5 世代と第 6 世代のインスタンスファミリーがある場合があります。
> - default_x86_64 または default_arm64 インスタンスバンドルを使用する場合、AWS Batch はコスト効率とパフォーマンスのバランスに基づいてインスタンスファミリーを選択します。新世代のインスタンスでは多くの場合、価格パフォーマンスが向上しますが、ワークロードの可用性、コスト、パフォーマンスの最適な組み合わせを提供する場合、AWS Batch は旧世代のインスタンスファミリーを選択することがあります。例えば、c6i インスタンスと c7i インスタンスの両方が利用可能な AWS リージョン では、特定のジョブ要件に対してコスト効率が向上する場合、AWS Batch は c6i インスタンスを選択することがあります。AWS Batch インスタンスタイプと AWS リージョン の可用性の詳細については、「インスタンスタイプのコンピューティングテーブル」を参照してください。

- AWS Batch は、デフォルトのバンドル内のインスタンスを、よりコスト効率の高い新しいオプションに定期的に更新します。更新は、ユーザーからのアクションを必要とせずに自動的に行われます。ワークロードは、更新中も中断せずに実行され続けます。

> **Note**
>
> コンピューティング環境を作成する際、そのコンピューティング環境で選択するインスタンスタイプで同じアーキテクチャを使用する必要があります。例えば、x86 と ARM インスタンスを同じコンピューティング環境で使用することはできません。

> **Note**
>
> AWS Batch は、ジョブキューに必要な容量に基づいて GPU をスケールします。GPU スケジューリングを使用するには、コンピューティング環境に p6、p3、p4、p5、g3、g3s、g4、g5、または g6 ファミリーのインスタンスタイプを含める必要があります。

18. (オプション) [追加設定] を展開します。

    a. (オプション) プレイスメントグループ では、プレイスメントグループ名を入力して、コンピューティング環境内のリソースをグループ化します。

    b. 配分戦略 で、BEST_FIT_PROGRESSIVE を選択します。

    c. (オプション) Amazon マシンイメージ (AMI) 設定 で、Amazon マシンイメージ (AMI) 設定の追加 を選択します。

       Amazon EKS 最適化 Amazon Linux AMI またはカスタム AMI のいずれかを使用できます。

       i. Amazon EKS 最適化 Amazon Linux AMI を使用するには:

          A. [イメージタイプ] で、以下のいずれかを選択します。

             - Amazon Linux 2 - すべての AWS Graviton ベースのインスタンスファミリー (C6g、M6g、R6g、T4g など) のデフォルトで、すべての非 GPU インスタンスタイプに使用できます。

- Amazon Linux 2 (accelerated): すべての GPU インスタンスファミリー (P4 および G4 など) のデフォルトで、非 AWS Graviton ベースのすべてのインスタンスタイプに使用できます。
- Amazon Linux 2023: AWS Batch は Amazon Linux 2023 (AL2023) をサポートしています。
- Amazon Linux 2023 (accelerated): GPU インスタンスファミリーで、非 AWS Graviton ベースのすべてのインスタンスタイプに使用できます。

B. [Kubernetesバージョン] に、Kubernetes バージョン番号を入力します。

ii. カスタム AMI を使用するには:

A. [イメージタイプ] で、カスタム AMI の基となる AMI タイプを選択します。

- Amazon Linux 2 - すべての AWS Graviton ベースのインスタンスファミリー (C6g、M6g、R6g、T4g など) のデフォルトで、すべての非 GPU インスタンスタイプに使用できます。
- Amazon Linux 2 (accelerated): すべての GPU インスタンスファミリー (P4 および G4 など) のデフォルトで、非 AWS Graviton ベースのすべてのインスタンスタイプに使用できます。
- Amazon Linux 2023: AWS Batch は AL2023 をサポートしています。
- Amazon Linux 2023 (accelerated): GPU インスタンスファミリーで、非 AWS Graviton ベースのすべてのインスタンスタイプに使用できます。

B. [イメージ ID オーバーライド] には、カスタム AMI ID を入力します。

C. [Kubernetesバージョン] に、Kubernetes バージョン番号を入力します。

d. (オプション) [起動テンプレート] で、既存の [起動テンプレート] を選択します。

e. (オプション) 起動テンプレートのバージョン では、**$Default** または **$Latest** を使用するか、あるいは起用するバージョン番号を指定します。

f. (オプション) [起動テンプレートのオーバーライド] でオーバーライドを追加するには、[起動テンプレートのオーバーライドを追加] を選択します。

i. (オプション) [起動テンプレート] で、オーバーライドを追加する起動テンプレートを選択します。

ii. (オプション) [起動テンプレートのバージョン] で、起動テンプレートのバージョン番号、$Default または $Latest を選択します。

iii. (オプション) [ターゲットインスタンスタイプ] で、このオーバーライドを適用するインスタンスタイプまたはファミリーを選択します。これにより、許可されたインスタンスタイプに含まれるインスタンスタイプとファミリーのみをターゲットにできます。

iv. (オプション) userdataType で EKS ノードの初期化を選択します。起動テンプレートまたは起動テンプレートのオーバーライドとして AMI が指定されている場合にのみ、このフィールドを使用します。`EKS_AL2023`、`EKS_AL2023_NVIDIA` に基づくカスタム AMI には EKS_NODEADM、`EKS_AL2` および `EKS_AL_NVIDIA` には EKS_BOOSTRAP_SH を選択します。デフォルト値は EKS_BOOSTRAP_SH です。

同じコンピューティング環境で AL2 と AL2023 ベースのカスタム AMI の両方を使用している混合環境の場合は、userdataType を使用します。

19. 次のページ を選択します。
20. 仮想プライベートクラウド (VPC) ID で、インスタンスを起動する先の VPC を選択します。
21. サブネット では、使用するサブネットを選択します。デフォルトでは、選択した VPC 内のすべてのサブネットを使用できます。

> **Note**
>
> Amazon EKS における AWS Batch では、ローカルゾーンをサポートしています。詳細については、「Amazon EKS ユーザーガイド」の「Amazon EKS と AWS ローカルゾーン」を参照してください。

22. (オプション) セキュリティグループ で、インスタンスにアタッチするセキュリティグループを選択します。デフォルトでは、VPC のデフォルトのセキュリティグループが選択されます。
23. 次のページ を選択します。
24. レビュー では、設定手順を確認します。変更する必要がある場合は、[編集] を選択します。完了したら、コンピューティング環境の作成 を選択します。

チュートリアル: Amazon EKS リソースを使用してアンマネージド型のコンピューティング環境を作成する

Amazon Elastic Kubernetes Service (Amazon EKS) リソースを使用してアンマネージド型のコンピューティング環境を作成するには、次の手順を実行します。

1. https://console.aws.amazon.com/batch/ で AWS Batch コンソールを開きます。

2. ページ上部のナビゲーションバーから、 AWS リージョン 使用する を選択します。
3. ナビゲーションペインで、コンピューティング環境] を選択します。
4. [作成] を選択します。
5. 環境を設定します。
   a. コンピューティング環境の設定 で、Amazon Elastic Kubernetes Service (Amazon EKS) を選択します。
   b. オーケストレーションタイプ で、アンマネージド を選択します。
6. 名前 で、コンピューティング環境の一意な名前を指定します。名前の最大長は 128 文字です。大文字、小文字、数字、ハイフン (-)、アンダースコア (_) を含めることができます。
7. EKS クラスター で、既存の Amazon EKS クラスターを選択します。新しい EKS クラスターを作成するには、「Amazon EKS クラスターの作成」ページのステップに従います。
8. 名前空間 で、クラスター内の AWS Batch プロセスをグループ化する Kubernetes 名前空間を入力します。
9. (オプション) 最大 vCPUs、プロビジョンドキャパシティからジョブスケジューリングに使用できる vCPUs の最大数を指定します。
10. (オプション) タグ を展開します。タグの追加 を選択し、キーと値のペアを入力します。
11. 次のページ を選択します。
12. レビュー では、設定手順を確認します。変更する必要がある場合は、[編集] を選択します。完了したら、コンピューティング環境の作成 を選択します。

アンマネージド型のコンピューティング環境に Amazon EKS クラスターノードを割り当てる

アンマネージド型のコンピューティング環境を作成したら、Amazon EKS ノードにコンピューティング環境 UUID のラベルを付ける必要があります。
まず、`DescribeComputeEnvironments`API 結果からコンピューティング環境 UUID を取得します。

```
$ aws batch describe-compute-environments \
    --compute-environments unmanagedEksCE \
    --query "computeEnvironments[].{name: computeEnvironmentName, uuid: uuid}"
```

ノード情報を取得します。

```
kubectl get nodes -o name
```

ノードに AWS Batch コンピューティング環境 UUID のラベルを付けます。

```
kubectl label <node-name> batch.amazonaws.com/compute-environment-uuid=uuid
```

リソース: コンピューティング環境テンプレート

次の例では、空のコンピューティング環境テンプレートを示しています。このテンプレートを使ってコンピューティング環境を作成し、それをファイルに保存して AWS CLI `--cli-input-json` オプションで使用できます。これらのパラメータの詳細については、「CreateComputeEnvironment」(AWS Batch API リファレンス) を参照してください。

> **Note**
>
> コンピューティング環境テンプレートは、以下の AWS CLI コマンドで生成できます。
>
> ```
> $ aws batch create-compute-environment --generate-cli-skeleton
> ```

```
{
    "computeEnvironmentName": "",
    "type": "UNMANAGED",
    "state": "DISABLED",
    "unmanagedvCpus": 0,
    "computeResources": {
        "type": "EC2",
        "allocationStrategy": "BEST_FIT_PROGRESSIVE",
        "minvCpus": 0,
        "maxvCpus": 0,
        "desiredvCpus": 0,
        "instanceTypes": [
            ""
        ],
        "imageId": "",
        "subnets": [
            ""
```

```
        ],
        "securityGroupIds": [
            ""
        ],
        "ec2KeyPair": "",
        "instanceRole": "",
        "tags": {
            "KeyName": ""
        },
        "placementGroup": "",
        "bidPercentage": 0,
        "spotIamFleetRole": "",
        "launchTemplate": {
            "launchTemplateId": "",
            "launchTemplateName": "",
            "version": ""
        },
        "ec2Configuration": [
            {
                "imageType": "",
                "imageIdOverride": "",
                "imageKubernetesVersion": ""
            }
        ]
    },
    "serviceRole": "",
    "tags": {
        "KeyName": ""
    },
    "eksConfiguration": {
        "eksClusterArn": "",
        "kubernetesNamespace": ""
    }
}
```

インスタンスタイプのコンピューティングテーブル

次の表に、AWS リージョン、インスタンスファミリーキーワード、使用可能なインスタンスファミリーをリストしています。AWS Batch は最新のファミリーからインスタンスを割り当てようとしますが、インスタンスファミリーの可用性は AWS リージョン によって異なるため、前の世代のインスタンスファミリーを取得する可能性があります。

default_x86_64

リージョン	インスタンスファミリー
AWS Batch をサポートするすべての AWS リージョン	m6i、c6i、r6i c7i

default_arm64

リージョン	インスタンスファミリー
AWS Batch をサポートするすべての AWS リージョン	m6g、c6g、r6g c7g

最適

リージョン	インスタンスファミリー
• ap-northeast-1 • ap-northeast-2 • ap-south-1 • ap-southeast-1 • ap-southeast-2 • ca-central-1 • cn-north-1 • cn-northwest-1 • eu-central-1 • eu-west-1 • eu-west-2 • sa-east-1 • us-east-1 • us-east-2 • us-gov-west-1	m4、c4、r4

リージョン	インスタンスファミリー
• us-west-1 • us-west-2	
• af-south-1 • ap-east-1 • ap-northeast-3 • ap-south-2 • ap-southeast-3 • ap-southeast-4 • ca-west-1 • eu-central-2 • eu-north-1 • eu-south-1 • eu-south-2 • eu-west-3 • il-central-1 • me-central-1 • me-south-1 • us-gov-east-1 • us-isob-east-1 • us-iso-east-1 • us-isof-south-1 • us-isof-east-1 • eu-isoe-west-1 • us-northeast-1	m5、c5、r5
• ap-southeast-5 • ap-southeast-7 • ap-east-2 • mx-central-1	m6、c6、r6

AWS Batch でコンピューティング環境を更新する

AWS Batch は、コンピューティング環境を更新するための複数の戦略を提供します。各戦略は、特定の更新シナリオと要件に合わせて設計されています。これらのアプローチは、同じ基盤となる更新 API を使用しますが、更新を効果的に管理するためのさまざまな規範的手法を示しています。これらの更新は、AWS Batch コンソールまたは AWS CLI を使用して管理できます。これらの戦略を理解することで、ワークロードの中断を最小限に抑えつつ、ニーズに最適な方法を選択できます。

このトピックでは、利用可能な更新戦略の概要と、各アプローチを使用するタイミングに関するガイダンスについて説明します。詳細な手順については、各更新戦略の個々のセクションを参照してください。

> **Important**
>
> AWS Batch は、Amazon EC2 起動テンプレート、Amazon EC2 Auto Scaling グループ、Amazon EC2 スポットフリート、Amazon ECS クラスターなど、ユーザーに代わってアカウント内で複数の AWS リソースを作成および管理します。これらのマネージドリソースは、最適な AWS Batch オペレーションを確保するために特別に設定されています。これらの AWS Batch マネージドリソースを手動で変更すると、AWS Batch ドキュメントに明示的に記載されていない限り、`INVALID` コンピューティング環境、最適ではないインスタンススケーリング動作、ワークロード処理の遅延、予期しないコストなど、想定外の動作が発生する可能性があります。これらの手動変更は、AWS Batch サービスで決定的にサポートすることはできません。コンピューティング環境を管理するには、サポートされている AWS Batch API または AWS Batch コンソールを必ず使用してください。

トピック

コンピューティング環境の更新戦略

スケーリングまたはインフラストラクチャの更新を使用すると、コンピューティング環境が更新されます。ブルー/グリーン更新戦略では、新しいコンピューティング環境 (グリーン) を作成し、ワークロードを古いコンピューティング環境 (ブルー) から新しいコンピューティング環境 (グリーン) に移行します。

AWS Batch には、コンピューティング環境の更新に関する 3 つの異なる戦略があります。

スケーリングの更新

スケーリングの更新は、既存のインスタンスを置き換えることなくインスタンスを追加または削除することで、コンピューティング環境の容量を調整します。これは最速の更新シナリオであり、ダウンタイムは必要ありません。容量設定 (vCPUs) を変更する必要がある場合は、スケーリングの更新を使用します。これらの更新は通常、数分で完了します。

Fargate の更新は、スケーリングの更新と同じ手順を使用して実行されます。詳細については、スケーリングの更新を実行する を参照してください。

インフラストラクチャの更新

インフラストラクチャの更新は、コンピューティング環境のインスタンスを、設定が更新された新しいインスタンスに置き換えます。これらの更新には、特定のサービスロールと割り当て戦略の設定が必要ですが、ダウンタイムを最小限に抑えることができ、実行中のジョブが中断される可能性があります。インスタンスタイプ、AMI 設定、ネットワーク設定、サービスロール、環境の状態、またはその他のインフラストラクチャコンポーネントを変更する必要がある場合は、インフラストラクチャの更新を使用します。これらの更新は通常、ジョブの完了にもよりますが 10 ~ 30 分で完了します。

詳細については、インフラストラクチャの更新を実行する を参照してください。

ブルー/グリーン更新

ブルー/グリーン更新により、既存の環境とともに新しいコンピューティング環境が作成され、ダウンタイムなしでワークロードを段階的に移行できます。このアプローチは最も安全な更新パスを提供しますが、2 つの環境を一時的に実行する必要があります。ダウンタイムゼロが必要である場合、完全なデプロイ前に変更をテストする場合、クイックロールバック機能が必要な場合、またはインフラストラクチャの更新でサポートされていない設定を使用している場合は、ブルー/グリーン更新を使用します。完了までの時間はさまざまで、ユーザーが制御します。

詳細については、コンピューティング環境のブルー/グリーン更新を実行する を参照してください。

適切な更新戦略の選択

この決定ガイドを使用して、ニーズに最も適した更新戦略を選択します。

次の場合は、スケーリングの更新を選択します。

コンピューティング容量 (vCPUs) を調整するだけで済む場合は、スケーリング更新戦略を選択します。スケーリングの更新は、ダウンタイムなしの迅速な更新が必要で、インフラストラクチャ設定の変更が不要な場合に最適です。

詳細な手順については、スケーリングの更新を実行する を参照してください。

次の場合は、インフラストラクチャの更新を選択します。

インスタンスタイプ、AMI 設定、サービスロール、環境の状態、またはネットワーク設定を変更する必要がある場合は、インフラストラクチャの更新戦略を選択します。環境では、サービスにリンクされたロール [AWSServiceRoleForBatch] と、`BEST_FIT_PROGRESSIVE`、`SPOT_CAPACITY_OPTIMIZED`、`SPOT_PRICE_CAPACITY_OPTIMIZED` の割り当て戦略を使用する必要があります。インフラストラクチャの更新は、更新中にジョブの若干の中断が許容され、最新の Amazon ECS 最適化 AMI の自動更新が必要な場合に適しています。

詳細な手順については、インフラストラクチャの更新を実行する を参照してください。

次の場合は、ブルー/グリーン更新を選択します。

ワークロードのダウンタイムがゼロである必要がある場合、または本番ワークロードの移行前に変更をテストする必要がある場合は、ブルー/グリーン更新戦略を選択します。このアプローチは、クイックロールバック機能が重要である場合、環境が `BEST_FIT` 配分戦略を使用する場合、または環境においてサービスにリンクされたロール [AWSServiceRoleForBatch] を使用しない場合に不可欠です。ブルー/グリーン更新は、手動更新を必要とするカスタム AMI を使用する場合や、大規模な設定変更を行う必要がある場合にも最適です。

詳細な手順については、コンピューティング環境のブルー/グリーン更新を実行する を参照してください。

AMI 更新に関する考慮事項

AWS Batch は、これらの条件がすべて満たされた場合、インフラストラクチャの更新中に最新の Amazon ECS 最適化 AMI に更新できます。

> **Note**
>
> インフラストラクチャの更新が完了すると、updateToLatestImageVersion が false に設定されます。別の更新を開始するには、updateToLatestImageVersion を true に設定する必要があります。

- コンピューティング環境は、サービスにリンクされたロール [AWSServiceRoleForBatch] を使用します。
- 割り当て戦略を BEST_FIT_PROGRESSIVE、SPOT_CAPACITY_OPTIMIZED、SPOT_PRICE_CAPACITY_OPTIMIZED に設定します。
- imageId、imageIdOverride または起動テンプレートで AMI ID が明示的に指定されていない
- updateToLatestImageVersion は true に設定されています。

ブルー/グリーンデプロイを使用した AMI の更新

以下のシナリオでは、ブルー/グリーンデプロイを使用して AMI を更新する必要があります。

- Amazon ECS 最適化 AMI の特定のバージョンを使用する場合
- AMI ID が次のいずれかで指定されている場合:
  - 起動テンプレート (テンプレートを更新するか、削除する必要があります)
  - imageId パラメータ
  - EC2 設定の imageIdOverride パラメータ
- BEST_FIT 配分戦略を使用する場合 (インフラストラクチャの更新をサポートしていない)
- サービスにリンクされたロール [AWSServiceRoleForBatch] を使用していない場合

スケーリングの更新を実行する

スケーリングの更新では、インスタンスを追加または削除することで、コンピューティング環境の容量を調整します。これは最速の更新戦略であり、既存のインスタンスを置き換える必要はありません。スケーリングの更新は、任意のサービスロールタイプと割り当て戦略で機能するため、最も柔軟な更新オプションになります。

スケーリングの更新をトリガーする変更

次の設定のみを変更すると、AWS Batch はスケーリングの更新を実行します。これらの設定のいずれかを他のコンピューティング環境設定とともに変更すると、AWS Batch は代わりにインフラストラクチャの更新を実行します。

以下の設定は、排他的に変更された場合にスケーリングの更新をトリガーします。

- `desiredvCpus` – 環境の vCPU ターゲット数を設定します。
- `maxvCpus` – 起動できる vCPU の最大数を定義します。
- `minvCpus` – 維持する vCPU の最小数を指定します。

Fargate コンピューティング環境では、スケーリングの更新のためにこれらの設定を変更することもできます。

- `securityGroupIds` – コンピューティング環境のセキュリティグループ ID。
- `subnets` – コンピューティング環境のサブネット。

> **Note**
>
> AWS Batch は `desiredvCpus` を動的に調整するため、`desiredvCpus` を使用してスケーリングの更新を開始することはお勧めしません。代わりに、`minvCpus` を使用する必要があります。
> `desiredvCpus` を更新する場合、値は `minvCpus` ~ `maxvCpus` である必要があります。新たな値は、現在の `desiredvCpus` 値と同等かそれ以上である必要があります。詳細については、「the section called “desiredvCpus 設定を更新すると、エラーメッセージが表示されます”」を参照してください。

> **Important**
>
> これらのスケーリング設定のいずれかを他のコンピューティング環境設定 (インスタンスタイプ、AMI ID、起動テンプレートなど) と一緒に変更すると、AWS Batch はスケーリングの更新の代わりにインフラストラクチャの更新を実行します。インフラストラクチャの更新には時間がかかり、既存のインスタンスが置き換えられる可能性があります。

Performing scaling updates using the AWS マネジメントコンソール

1. AWS Batch コンソールを https://console.aws.amazon.com/batch/ で開きます。
2. ナビゲーションペインで、[環境]、[コンピューティング環境] タブの順に選択します。
3. 更新するコンピューティング環境を選択します。
4. [Actions (アクション)] を選択してから [Edit (編集)] を選択します。
5. スケーリングの更新をサポートする 1 つ以上の設定を変更します。例:
   - [最小 vCPU] に、vCPU の最小数を入力します。
   - [必要な vCPU] に、vCPU の要求数を入力します。
   - [最大 vCPU] に、vCPU の最大数を入力します。
6. [変更の保存] をクリックします。
7. コンピューティング環境のステータスをモニタリングします。更新に必要なのはスケーリング操作のみであるため、すぐに完了します。

Performing scaling updates using the AWS CLI

update-compute-environment コマンドを使用して、スケーリングの更新を実行します。次の 2 つの例では一般的なスケーリング操作について説明しています。スケーリングの更新をサポートする以下の設定を 1 つ以上変更できます。

- この例では、必要な vCPU と 最大 vCPU を更新します。

```
aws batch update-compute-environment \
    --compute-environment your-compute-environment-name \
    --compute-resources minvCpus=2,maxvCpus=8
```

スケーリングの更新のモニタリング

AWS Batch コンソールを使用してスケーリングの更新をモニタリングし、コンピューティング環境のステータスを表示し、インスタンス数と vCPU メトリクスを確認します。describe-compute-environments コマンドとともに AWS CLI を使用してステータスを確認し、インスタンス数と vCPU 値をモニタリングすることもできます。

インフラストラクチャの更新を実行する

インフラストラクチャの更新は、コンピューティング環境のインスタンスを、設定が更新された新しいインスタンスに置き換えます。この更新戦略は、スケーリング更新よりも時間がかかり、特定のサービスロールと割り当て戦略の設定が必要です。インフラストラクチャの更新は、サービスの可用性を維持しながら、基本的なコンピューティング環境設定を変更する方法を提供します。

> **Important**
>
> インフラストラクチャの更新には、サービスにリンクされたロール [AWSServiceRoleForBatch] と、BEST_FIT_PROGRESSIVE、SPOT_CAPACITY_OPTIMIZED、SPOT_PRICE_CAPACITY_OPTIMI の割り当て戦略が必要です。環境がこれらの要件を満たしていない場合は、代わりにブルー/グリーン更新を使用します。

インフラストラクチャの更新をトリガーする変更

次のいずれかの設定を変更すると、AWS Batch はインフラストラクチャの更新を実行します。インフラストラクチャの更新は、スケーリング更新設定とともにこれらの設定を変更する場合にも発生します。

以下の設定により、インフラストラクチャの更新がトリガーされます。

コンピューティング設定

- `allocationStrategy` – AWS Batch がインスタンスタイプを選択する方法を決定します。
- `instanceTypes` – 使用する EC2 インスタンスタイプを指定します。
- `bidPercentage` – スポットインスタンスのオンデマンド料金の最大パーセンテージ。
- `type` – コンピューティング環境タイプ (`EC2` または `SPOT`)。

AMI と起動設定

- `imageId` – インスタンスに使用する特定の AMI。
- `ec2Configuration` – `imageIdOverride` を含む EC2 設定。
- `launchTemplate` – EC2 起動テンプレートの設定。
- `ec2KeyPair` – インスタンスアクセス用の SSH キーペア。

- updateToLatestImageVersion – AMI の自動更新設定。

ネットワークとセキュリティ

- subnets – インスタンスが起動される VPC サブネット (EC2 コンピューティング環境の場合) 。
- securityGroupIds – インスタンスのセキュリティグループ (EC2 コンピューティング環境の場合) 。
- placementGroup – EC2 プレイスメントグループ設定。

その他の設定

- instanceRole – EC2 インスタンスの IAM ロール。
- tags – EC2 インスタンスに適用されるタグ。

> **Important**
>
> インフラストラクチャの更新設定をスケーリング更新設定 (desiredvCpus、maxvCpus、minvCpus など) とともに変更すると、AWS Batch はインフラストラクチャの更新を実行します。インフラストラクチャの更新には、スケーリングの更新よりもずっと時間がかかります。

インフラストラクチャの更新中の AMI の選択

インフラストラクチャの更新中に、これら 3 つの設定のいずれかで AMI が指定されているかどうかに応じて、コンピュート環境の AMI ID が変更される場合があります。AMI は imageId (in computeResources)、imageIdOverride (in ec2Configuration)、または launchTemplate で指定されている起動テンプレートで指定されます。これらの設定のいずれにも AMI ID が指定されておらず、updateToLatestImageVersion 設定が true であるとします。その場合、がサポートしている最新の Amazon ECS 最適化 AMI AWS Batch がインフラストラクチャの更新に使用されます。

これらの設定の少なくとも 1 つで AMI ID が指定されている場合、更新は、更新前に使用された AMI ID が提供された設定によって異なります。コンピュート環境を作成する場合、AMI ID を選択する際の優先順位は、最初に起動テンプレート、次に imageId の設定、最後に imageIdOverride の設定です。ただし、使用される AMI ID が起動テンプレートからのものである場合、imageId または

imageIdOverride の設定を更新しても AMI ID は更新されません。起動テンプレートから選択した AMI ID を更新する唯一の方法は、起動テンプレートを更新することです。起動テンプレートのバージョンパラメータが $Default または $Latest の場合、指定された起動テンプレートのデフォルトまたは最新バージョンが評価されます。デフォルトで別の AMI ID が選択されている場合、または起動テンプレートの最新バージョンが選択されている場合、その AMI ID が更新に使用されます。

起動テンプレートを使用して AMI ID を選択しなかった場合は、imageId または imageIdOverride のパラメータで指定されている AMI ID が使用されます。両方を指定すると、imageIdOverride パラメータで指定された AMI ID が使用されます。

コンピュート環境が imageId、imageIdOverride、または launchTemplate パラメータで指定された AMI ID を使用しており、AWS Batch でサポートされている最新の Amazon ECS 最適化 AMI を使用するとします。次に、更新により AMI ID を提供した設定を削除する必要があります。このため imageId、そのパラメータには空の文字列を指定する必要があります。このため imageIdOverride、ec2Configuration パラメータには空の文字列を指定する必要があります。

AMI ID が起動テンプレートから取得された場合は、AWS Batch次のいずれかの方法でサポートされている最新の Amazon ECS 最適化 AMI に変更できます。

- launchTemplateId または launchTemplateName パラメータに空の文字列を指定して、起動テンプレートを削除します。これにより、AMI ID だけではなく、起動テンプレート全体が削除されます。
- 更新バージョンの起動テンプレートで AMI ID が指定されていない場合は、updateToLatestImageVersion パラメータを true に設定する必要があります。

更新中のジョブ処理

更新ポリシーを使用してインフラストラクチャ更新時に実行されているジョブの処理方法を設定します。terminateJobsOnUpdate=true を設定すると、実行中のジョブはすぐに終了し、jobExecutionTimeoutMinutes 設定は無視され、インスタンスが置き換えられるとすぐに更新が続行されます。terminateJobsOnUpdate=false を設定すると、実行中のジョブは指定されたタイムアウト期間中は継続され (デフォルトのタイムアウトは 30 分)、タイムアウト期間を超えるとジョブは終了します。

> **Note**
>
> 更新中に終了したジョブを再試行するには、ジョブの再試行戦略を設定します。詳細については、「the section called “ジョブの再試行の自動化”」を参照してください。

Performing infrastructure updates using the AWS マネジメントコンソール

1. AWS Batch コンソールを https://console.aws.amazon.com/batch/ で開きます。
2. ナビゲーションペインで、[環境]、[コンピューティング環境] タブの順に選択します。
3. 更新するコンピューティング環境を選択します。
4. [アクション] を選択してから [編集] を選択します。
5. [更新動作] セクションで、実行中のジョブの処理方法を設定します。
   - [AMI を最新バージョンに更新] を選択すると、AMI は最新バージョンに更新されます。
   - [更新時にジョブをすぐに終了] を選択すると、更新プロセスの実行時にジョブを終了します。
   - [ジョブ実行タイムアウト] には、更新プロセスを開始する前に待機する分数を入力します。
6. インフラストラクチャの更新が必要な設定を 1 つ以上変更します。例:
   - インスタンスロール
   - EC2 スポットインスタンスを使用する
   - 許可されたインスタンスタイプ
   - 配置グループ
   - EC2 キーペア
   - EC2 設定
   - 起動テンプレート
   - サブネット
   - セキュリティグループ
7. [Save changes] (変更の保存) をクリックします。
8. コンピューティング環境のステータスをモニタリングします。更新プロセス中に環境が `UPDATING` を示します。

Performing infrastructure updates using the AWS CLI

update-compute-environment コマンドを使用して、インフラストラクチャの更新を必要とする 1 つ以上の設定 を変更します。次の 3 つの例は、一般的なインフラストラクチャオペレーションです。

- この例では、インスタンスタイプを更新し、更新ポリシーを設定します。

```
aws batch update-compute-environment \
    --compute-environment your-compute-environment-name \
    --compute-resources instanceTypes=default_x86_64 \
    --update-policy terminateJobsOnUpdate=false,jobExecutionTimeoutMinutes=30
```

- この例では、VPC サブネットとセキュリティグループを更新します。

```
aws batch update-compute-environment \
    --compute-environment your-compute-environment-name \
    --compute-resources subnets=subnet-abcd1234,subnet-efgh5678
 securityGroupIds=sg-abcd1234 \
    --update-policy terminateJobsOnUpdate=true
```

- この例では、最新の Amazon ECS 最適化 AMI の自動更新を有効にします。

```
aws batch update-compute-environment \
    --compute-environment your-compute-environment-name \
    --compute-resources updateToLatestImageVersion=true \
    --update-policy terminateJobsOnUpdate=false,jobExecutionTimeoutMinutes=60
```

インフラストラクチャの更新のモニタリング

AWS Batch コンソールを使用してインフラストラクチャの更新をモニタリングし、コンピューティング環境のステータスが UPDATING に変わることを確認し、インスタンスの置き換えの進行状況をモニタリングして、失敗した更新がないか確認します。コンピューティング環境の状態が VAILD になると、更新は成功します。CloudWatch を使用して、インスタンスの終了イベントを追跡し、更新中にジョブの状態をモニタリングすることもできます。AWS CLI で、 describe-compute-environments コマンドを使用してステータスを確認し、インスタンスのライフサイクルイベントをモニタリングします。

コンピューティング環境のブルー/グリーン更新を実行する

ブルー/グリーン更新は、既存のコンピューティング環境 (ブルー) に沿って新しいコンピューティング環境 (グリーン) を作成することでダウンタイムとリスクを軽減する更新戦略です。このアプローチにより、既存の環境を運用しながら、ワークロードを新しい環境に徐々に移行できます。ブルー/グリーン更新は最も安全な更新パスを提供し、任意のサービスロールタイプまたは割り当て戦略で機能します。

概要

ブルー/グリーン更新には、本番環境に最適ないくつかの利点があります。更新プロセス中にワークロードを継続的に実行することで、ダウンタイムをゼロにします。このアプローチによって簡単にロールバックできる機能を得られ、問題が発生したらすぐに元の環境に戻すことができます。本番ワークロードに完全に切り替える前に、新しい環境のパフォーマンスを検証して段階的な移行戦略を実行できます。また、この方法では、元の環境は変更されず、削除されるまで動作するため、リスク軽減の面でも優れています。

ブルー/グリーン更新が必要な場合

ブルー/グリーン更新は、以下の状況で使用する必要があります。

- コンピューティング環境において BEST_FIT 配分戦略を使用する場合 (インフラストラクチャの更新をサポートしていない)
- コンピューティング環境が AWSServiceRoleForBatch サービスにリンクされたロールを使用しない場合
- 異なるサービスロールタイプ間で移行する必要がある場合

ブルー/グリーン更新が推奨される場合

ブルー/グリーン更新は、ワークロードにとってダウンタイムゼロが重要な本番環境で特に推奨されます。このアプローチは、本番ワークロードを移行する前に新しい設定をテストし、変更がパフォーマンスと信頼性の要件に適合していることを確認する必要がある場合に適しています。クイックロールバック機能がオペレーションにとって重要な場合、特に大幅な変更でカスタム AMI を更新する場合は、ブルー/グリーン更新を選択します。この方法は、変更に完全にコミットする前にパフォーマンス特性と動作を検証し、更新プロセスの信頼性を提供する場合にも最適です。

前提条件

ブルー/グリーン更新を実行する前に、以下があることを確認してください。

- コンピューティング環境を作成および管理するための適切な IAM アクセス許可
- ジョブキュー設定を表示および変更するためのアクセス
- 移行中の潜在的な障害を処理するためにジョブ定義用に設定されたジョブ再試行戦略。詳細については、ジョブの再試行の自動化 を参照してください。
- 新しいコンピューティング環境の AMI ID。これは以下のいずれかとなります。
  - Amazon ECS 最適化 AMI の最近の承認済みバージョン (デフォルトで使用)

- Amazon ECS コンテナインスタンス AMI 仕様を満たすカスタム AMI。カスタム AMI を使用する場合は、次のいずれかの方法で指定できます。
  - EC2 設定でイメージ ID オーバーライドフィールドを使用する
  - 起動テンプレートでの指定

  カスタム AMI の作成の詳細については、「チュートリアル: コンピューティングリソース AMI を作成する」を参照してください。

新しい環境を作成する前に、既存のコンピューティング環境の設定を記録する必要があります。これを行うには、AWS マネジメントコンソール または AWS CLI を使用します。

> **Note**
>
> 次の手順では、AMI のみを変更するブルー/グリーン更新を実行する方法について詳しく説明します。新しい環境の他の設定を更新できます。

> **Important**
>
> 古い (ブルー) コンピューティング環境を削除すると、インスタンスが終了するため、それらのインスタンスで現在実行中のジョブはすべて失敗します。これらの障害を自動的に処理するように、ジョブ定義でジョブ再試行戦略を設定します。詳細については、ジョブの再試行の自動化 を参照してください。
> 新しい環境に確信を持てたら、以下を行います。
>
> 1. ジョブキューを編集して古いコンピューティング環境を削除します。
> 2. 古い環境で実行中のジョブが完了するまで待ちます。
> 3. 古いコンピューティング環境を削除します。

Performing blue/green updates using the AWS マネジメントコンソール

1. 現在のコンピューティング環境のクローンを作成する

   a. AWS Batch コンソールを https://console.aws.amazon.com/batch/ で開きます。
   b. 既存のコンピューティング環境を選択します。
   c. [アクション] を選択してから [クローン] を選択します。

d. [名前] で、新しいコンピューティング環境の一意の名前を入力します。

e. [次へ] を選択します。

f. [インスタンス設定] セクションで AMI の設定を更新します。

i. [追加設定] を展開します。

ii. [EC2 設定] では、[イメージタイプ] に新しい AMI タイプを指定し、[イメージ ID オーバーライド] フィールドに AMI ID を指定します。

g. [次へ] を選択します。

h. [ネットワーク設定] で [次へ] を選択します。

i. 既存の環境から自動的にコピーされる他の設定を確認します。

j. [コンピューティング環境の作成] を選択します。

k. 新しいコンピューティング環境のステータスが `VALID` になるまで待ちます。

2. ジョブキューの順序を変更する

a. ナビゲーションペインで [キュー] を選択します。

b. 既存のコンピューティング環境に関連付けられているジョブキューを選択します。

c. [編集] を選択します。

d. [接続されたコンピューティング環境] で、新しいコンピューティング環境を追加します。

- 新しいコンピューティング環境を既存の環境よりも高い順序で追加して、ワークロードを移行します。
- 新しい環境が正しく動作していることを確認したら、低い順序の番号を付けることで、それをプライマリ環境にすることができます。

e. [ジョブキューの更新] を選択します。

3. クリーンアップ

a. 新しい環境でジョブの実行をモニタリングし、すべてが期待どおりに動作していることを確認します。

b. 新しい環境に確信を持てたら、以下を行います。

1. ジョブキューを編集して古いコンピューティング環境を削除します。

2. 古い環境で実行中のジョブが完了するまで待ちます。

3. 古いコンピューティング環境を削除します。

Performing blue/green updates using the AWS CLI

1. AWS CLI を使用して設定を取得するには、以下のコマンドを使用します。

```
aws batch describe-compute-environments \
  --compute-environments your-compute-environment-name
```

   新しい環境を作成するときに、参照用に出力を保存します。

2. 既存の環境の設定を使用して新しいコンピューティング環境を作成しますが、新しい AMI を使用します。以下に、コマンド構造の例を示します。

   この例の値を前のステップの実際の設定に置き換えます。

```
cat <<EOF > ./blue-green-compute-environment.json
{
  "computeEnvironmentName": "your-new-compute-environment-name",
  "type": "MANAGED",
  "state": "ENABLED",
  "computeResources": {
    "instanceRole": "arn:aws:iam::012345678901:instance-profile/
ecsInstanceRole",
    "type": "EC2",
    "minvCpus": 2,
    "desiredvCpus": 2,
    "maxvCpus": 256,
    "instanceTypes": [
      "optimal"
    ],
    "allocationStrategy": "BEST_FIT_PROGRESSIVE",
    "ec2Configuration": [
      {
        "imageType": "ECS_AL2023",
        "imageIdOverride": "ami-0abcdef1234567890"
      }
    ],
    "subnets": [,
      "subnet-0abcdef1234567890"
    ],
    "securityGroupIds": [
      "sg-0abcdef1234567890"
    ]
  }
```

```
}
EOF
```

```
$ aws batch create-compute-environment --cli-input-json file://./blue-green-
compute-environment.json
```

3. 新しい環境が使用可能になるまで待ちます。

```
aws batch describe-compute-environments \
  --compute-environments your-new-compute-environment-name \
  --query 'computeEnvironments[].status'
```

4. 新しいコンピューティング環境をジョブキューに追加します。

```
aws batch update-job-queue \
  --job-queue your-job-queue \
  --compute-environment-order order=1,computeEnvironment=your-existing-
environment \
  order=2,computeEnvironment=your-new-compute-environment-name
```

5. 検証したら、再度更新して新しい環境をプライマリにします。

```
aws batch update-job-queue \
  --job-queue your-job-queue \
  --compute-environment-order order=1,computeEnvironment=your-new-compute-
environment-name
```

古い環境ですべてのジョブが完了したら、無効にしてから削除します。

```
aws batch update-compute-environment \
    --compute-environment your-existing-environment \
    --state DISABLED
```

```
aws batch delete-compute-environment \
  --compute-environment your-existing-environment
```

コンピューティングリソースの AMI

デフォルトでは、AWS Batch のマネージド型のコンピューティング環境では、承認されたバージョンの Amazon ECS に最適化された最新の AMI をコンピューティングリソースに使用します。ただし、マネージド型およびアンマネージド型のコンピューティング環境で使用する AMI を独自に作成することもできます。次のいずれかが必要な場合は、独自の AMI を作成することをお勧めします。

- AMIルートまたはデータボリュームのストレージサイズを増やす
- サポートされているAmazon EC2インスタンスタイプにインスタンスストレージボリュームを追加します。
- Amazon ECS コンテナエージェントをカスタマイズする
- Docker をカスタマイズする
- サポートされている Amazon EC2 インスタンスタイプで、コンテナから GPU ハードウェアにアクセスできるように GPU ワークロードの AMI を設定する

> **Note**
>
> コンピューティング環境の作成後、AWS Batch はコンピューティング環境内の AMI をアップグレードしません。また、Amazon ECS の最適化された AMI の新しいバージョンが使用可能な場合、AWS Batch はコンピューティング環境内の AMI を更新しません。ゲストオペレーティングシステムの管理はユーザーの責任です。これには、アップデートとセキュリティパッチが含まれます。また、コンピューティングリソースにインストールするその他のアプリケーションソフトウェアやユーティリティについても責任を負うものとします。AWS Batch ジョブに新しい AMI を使用するには、次の手順を実行します。
>
> 1. 新しい AMI を使用して新しいコンピューティング環境を作成します。
> 2. コンピューティング環境を既存のジョブキューに追加します。
> 3. 古いコンピューティング環境をジョブキューから削除します。
> 4. 以前のコンピューティング環境を削除します。
>
> 2022 年 4 月、AWS Batch によりコンピューティング環境の更新のためにサポートが強化されました。詳細については、AWS Batch でコンピューティング環境を更新するを参照してください。コンピューティング環境の拡張アップデートを使用して AMI を更新するには、次のルールに従います。

- サービスロール(serviceRole)パラメータを設定しないか、AWSServiceRoleForBatch サービス連動ロールに設定します。
- 割り当て戦略 (allocationStrategy) パラメータを BEST_FIT_PROGRESSIVE、SPOT_CAPACITY_OPTIMIZED、または SPOT_PRICE_CAPACITY_OPTIMIZED に設定します。
- 最新のイメージバージョンへの更新 (updateToLatestImageVersion) パラメータを true に設定します。
- imageId、imageIdOverride(ec2Configuration) または起動テンプレート (launchTemplate) には AMI ID を指定しないでください。AMI ID を指定しないと、インフラストラクチャの更新が開始された時点で、AWS Batch は AWS Batch がサポートしている最新の Amazon ECS 最適化 AMIを選択します。代わりに、imageId または imageIdOverride パラメータを使用してAMI IDを指定できます。あるいは、LaunchTemplate プロパティによって識別される起動テンプレートを指定できます。これらのプロパティのいずれかを変更すると、インフラストラクチャの更新が開始されます。AMI ID が起動テンプレートで指定されている場合、imageId または imageIdOverride パラメータで AMI ID を指定しても AMI ID を置き換えることはできません。AMI ID は、別の起動テンプレートを指定することでのみ置き換えることができます。起動テンプレートのバージョンが $Default または $Latest に設定されている場合、AMI ID は起動テンプレートの新しいデフォルトバージョンを設定 ($Default の場合) するか、起動テンプレートに新しいバージョンを追加 ($Latest の場合) することで置き換えることができます。

これらのルールに従うと、インフラストラクチャの更新を開始する更新により、AMI ID が再選択されます。起動テンプレート (launchTemplate) の version 設定が $Latest または $Default に設定されている場合、launchTemplate が更新されていなくても、起動テンプレートの最新バージョンまたはデフォルトバージョンがインフラストラクチャの更新時に評価されます。

トピック

- コンピューティングリソースの AMI 仕様
- チュートリアル: コンピューティングリソース AMI を作成する
- GPU ワークロードの AMI を使用する
- Amazon Linux の廃止

- Amazon EKS Amazon Linux 2 AMI の廃止
- Amazon ECS Amazon Linux 2 AMI の廃止

コンピューティングリソースの AMI 仕様

基本的な AWS Batch コンピューティングリソース AMI の仕様は、以下の項目で構成されます。

必須

- HVM 仮想化タイプの AMI で、バージョン 3.10 以上の Linux カーネルを実行している最新の Linux ディストリビューション。Windows コンテナはサポートされていません。

> **Important**
>
> マルチノード並列ジョブは、`ecs-init` パッケージがインストールされた Amazon Linux インスタンスで起動されたコンピューティングリソースでのみ実行できます。コンピューティング環境を作成するときに、デフォルトの Amazon ECS 最適化 AMI を使用することが推奨されます。これを行うには、カスタム AMI を指定しません。詳細については、マルチノード並列ジョブを参照してください。

- Amazon ECS コンテナエージェント。最新バージョンの使用をお勧めします。詳細については、Amazon Elastic Container Service デベロッパーガイドのAmazon ECS コンテナエージェントのインストールを参照してください。
- `awslogs` ログドライバーは、Amazon ECS コンテナエージェントが開始するときの `ECS_AVAILABLE_LOGGING_DRIVERS` 環境変数として利用可能なログドライバとして指定する必要があります。詳細については、Amazon Elastic Container Service デベロッパーガイドのAmazon ECS コンテナエージェントの構成を参照してください。
- バージョン 1.9 以上を実行する Docker デーモン、および Docker ランタイムの依存関係。詳細については、Docker ドキュメントのランタイムの依存関係を確認するを参照してください。

> **Note**
>
> 対応する Amazon ECS エージェントバージョンに同梱されており、そのバージョンでテストされた Docker バージョンをお勧めします。Amazon ECS は、GitHub で Amazon ECS を使用して最適化した AMI の Linux バリアントの変更ログを提供します。詳細については、「Changelog」を参照してください。

推奨

- Amazon ECS エージェントを実行およびモニタリングするための初期化およびナニープロセス。Amazon ECS に最適化した AMI は `ecs-init` 起動プロセスを使用し、その他のオペレーティングシステムは `systemd` を使用する場合があります。詳細および例については、「Amazon Elastic Container Service 開発者ガイド」の「コンテナインスタンスのユーザーデータ設定スクリプトの例」を参照してください。`ecs-init` についての詳細は、GitHub の`ecs-init` プロジェクトを参照してください。少なくとも、マネージド型のコンピューティング環境ではブート時に Amazon ECS エージェントをスタートする必要があります。コンピューティングリソースで実行されていない Amazon ECS エージェントでは、AWS Batch からジョブを受け入れることはできません。

Amazon ECS に最適化された AMI は、これらの要件および推奨事項に従って事前設定されています。Amazon ECS最適化AMIまたはコンピュートリソースにインストールされている`ecs-init`パッケージのAmazon Linux AMIを使用することをお勧めします。アプリケーションが特定のオペレーティングシステムや、これらのAMIでまだ利用できないDockerバージョンを必要とする場合は、別のAMIを選択してください。詳細については、Amazon Elastic Container Service デベロッパーガイドのAmazon ECS に最適化された AMIを参照してください。

チュートリアル: コンピューティングリソース AMI を作成する

コンピューティングリソースのカスタム AMI を独自に作成して、マネージド型およびアンマネージド型のコンピューティング環境で使用できます。手順については、コンピューティングリソースの AMI 仕様を参照してください。カスタム AMI を作成したら、その AMI を使用するコンピューティング環境を作成し、その環境をジョブキューに関連付けることができます。最後に、そのキューへのジョブの送信をスタートできます。

コンピューティングリソースのカスタム AMI を作成するには

1. 基本 AMI を選択してスタートします。基本 AMI は、HVM 仮想化を使用する必要があります。基本 AMI を Windows AMI にすることはできません。

> **Note**
>
> コンピューティング環境用に選択する AMI は、そのコンピューティング環境で使用するインスタンスタイプのアーキテクチャと一致している必要があります。例えば、コンピューティング環境で A1 インスタンスタイプを使用する場合、選択するコンピューティングリソース AMI で ARM インスタンスをサポートしている必要がありま

す。Amazon ECS は、Amazon ECS に最適化された Amazon Linux 2 AMI の、x86 と ARM の両バージョンを提供しています。詳細については、Amazon Elastic Container Service デベロッパーガイドの「Amazon ECS に最適化された Amazon Linux 2 AMI」を参照してください。

Amazon ECS に最適化された Amazon Linux 2 AMI は、マネージド型のコンピューティング環境のコンピューティングリソースのデフォルト AMI です。Amazon ECS 最適化 Amazon Linux 2 AMI は、 AWS エンジニア AWS Batch によって事前設定され、 でテストされています。これは最小限の AMI であり、 の使用を開始して、 で実行されているコンピューティングリソース AWS をすばやく取得できます。詳細については、Amazon Elastic Container Service デベロッパーガイドのAmazon ECS に最適化された AMIを参照してください。

または、別の Amazon Linux 2 バリアントを選択し、次のコマンドを使用して `ecs-init` パッケージをインストールできます。詳細については、Amazon Elastic Container Service デベロッパーガイドのAmazon Linux 2 EC2 インスタンスへの Amazon ECS コンテナエージェントのインストールを参照してください。

```
$ sudo amazon-linux-extras disable docker
$ sudo amazon-linux-extras install ecs-init
```

たとえば、 AWS Batch コンピューティングリソースで GPU ワークロードを実行する場合は、Amazon Linux Deep Learning AMI から開始できます。次に、 AWS Batch ジョブを実行するように AMI を設定します。詳細については、「GPU ワークロードの AMI を使用する」を参照してください。

> **Important**
>
> `ecs-init` パッケージをサポートしていない基本 AMI も選択できます。ただし、その場合はブート時に Amazon ECS エージェントを開始して実行を続けるように設定する必要があります。Amazon ECS コンテナエージェントを開始してモニタリングする、`systemd` を使用したユーザーデータ設定スクリプトの例をいくつか用意しています。詳細については、Amazon Elastic Container Service デベロッパーガイドのコンテナインスタンスのユーザーデータ設定スクリプトの例を参照してください。

2. AMI の適切なストレージオプションがある、選択された基本 AMI からインスタンスを起動します。アタッチされた Amazon EBS ボリュームまたはインスタンスストレージボリューム (選択

したインスタンスタイプでサポートされている場合) のサイズと数を設定できます。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスの起動」と「Amazon EC2 インスタンスストア」を参照してください。

3. SSH を使用してインスタンスに接続し、必要に応じて設定タスクを実行します。これには、以下のステップのいずれかまたはすべてが含まれる場合があります。

   - Amazon ECS コンテナエージェントをインストールする 詳細については、Amazon Elastic Container Service デベロッパーガイドのAmazon ECS コンテナエージェントのインストールを参照してください。
   - インスタンスストアボリュームをフォーマットするスクリプトを設定する。
   - インスタンスストアボリュームまたは Amazon EFS ファイルシステムを `/etc/fstab` ファイルに追加し、ブート時にマウントする。
   - Docker オプションを設定する (デバッグの有効化、基本イメージサイズの調整など)。
   - パッケージのインストールやファイルのコピー。

   詳細については、「Amazon EC2 ユーザーガイド」の「SSH クライアントを使用して Linux インスタンスに接続する」を参照してください。

4. Amazon ECS コンテナエージェントをインスタンスで開始する場合は、AMI を作成する前にインスタンスを停止して永続的なデータチェックポイントを削除する必要があります。この操作を行わないと、AMI から起動されたインスタンスでエージェントが起動しません。

   a. Amazon ECS コンテナエージェントを停止します。

      - Amazon ECS 対応 Amazon Linux 2 AMI:

        ```
        sudo systemctl stop ecs
        ```

      - Amazon ECS に最適化された Amazon Linux AMI:

        ```
        sudo stop ecs
        ```

   b. 永続的なデータチェックポイントファイルを削除します。デフォルトでは、このファイルは `/var/lib/ecs/data/` ディレクトリにあります。ファイルが存在する場合は、次のコマンドを使用してこれらのファイルを削除します。

      ```
      sudo rm -rf /var/lib/ecs/data/*
      ```

5. 実行中のインスタンスから新しい AMI を作成します。詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EBS-backed Linux AMI を作成する」を参照してください。

で新しい AMI を使用するには AWS Batch

1. 新しい AMI が作成されたら、新規 AMI でコンピューティング環境を作成します。これを行うには、イメージタイプを選択し、 AWS Batch コンピューティング環境を作成するときにイメージ ID オーバーライドボックスにカスタム AMI ID を入力します。詳細については、「the section called “チュートリアル: Amazon EC2 リソースを使用して、マネージド型のコンピューティング環境を作成する”」を参照してください。

   > **Note**
   >
   > コンピューティング環境用に選択する AMI は、そのコンピューティング環境で使用するインスタンスタイプのアーキテクチャと一致している必要があります。例えば、コンピューティング環境で A1 インスタンスタイプを使用する場合、選択するコンピューティングリソース AMI で ARM インスタンスをサポートしている必要があります。Amazon ECS は、Amazon ECS に最適化された Amazon Linux 2 AMI の、x86 と ARM の両バージョンを提供しています。詳細については、Amazon Elastic Container Service デベロッパーガイドの「Amazon ECS に最適化された Amazon Linux 2 AMI」を参照してください。

2. ジョブキューを作成し、新しいコンピューティング環境を関連付けます。詳細については、ジョブキューを作成するを参照してください。

   > **Note**
   >
   > ジョブキューに関連付けられているすべてのコンピューティング環境で、同じアーキテクチャを共有する必要があります。 AWS Batch では、単一のジョブキューでのコンピューティング環境アーキテクチャタイプの混在をサポートしていません。

3. (オプション) 新しいジョブキューにサンプルジョブを送信します。詳細についてはジョブ定義の例、シングルノードのジョブ定義を作成する、およびチュートリアル: ジョブを送信するを参照してください。

GPU ワークロードの AMI を使用する

ユーザーの AWS Batch コンピューティングリソースで GPU ワークロードを実行するには、GPU 対応の AMI を使用する必要があります。詳細については、Amazon Elastic Container Service デベロッパーガイドの Amazon ECS での GPU の使用および Amazon ECS に最適化された AMIを参照してください。

マネージドコンピューティング環境では、コンピューティング環境で p3、p4、p5、p6、g3、g3s、g4、g5、g6 のインスタンスタイプまたはインスタンスファミリーが指定されている場合、AWS Batch は Amazon ECS GPU 最適化 AMI を使用します。

アンマネージド型のコンピューティング環境では、 Amazon ECS GPU に最適化された AMI をお勧めします。AWS Command Line Interface または AWS Systems Manager パラメータストアの GetParameter、GetParameters、および GetParametersByPath オペレーションを使用して、推奨される Amazon ECS GPU 最適化 AMI のメタデータを取得できます。

> **Note**
>
> p5 のインスタンスファミリーは、Amazon ECS GPU に最適化された AMI に等しいバージョン、または Amazon ECS GPU に最適化された AMI の 20230912 よりも遅いバージョンでのみサポートされており、p2 および g2 のインスタンスタイプとは互換性がありません。p5 インスタンスを使用する必要がある場合は、コンピューティング環境に p2 または g2の インスタンスが含まれておらず、最新のデフォルトの Batch AMI を使用していることを確認してください。新しいコンピューティング環境を作成すると最新の AMI が使用されますが、ユーザーが p5 を追加するようにコンピューティング環境を更新する場合は、ComputeResource プロパティで updateToLatestImageVersion を true に設定することにより、最新の AMI を使用していることを確認できます。AMIのGPUインスタンスと互換性の詳細については、Amazon Elastic Container Service デベロッパーガイドのAmazon ECS での GPU との連動を参照してください。

次に、GetParameter コマンドの使用方法を示します。

AWS CLI

```
$ aws ssm get-parameter --name /aws/service/ecs/optimized-ami/amazon-linux-2/gpu/
recommended \
                        --region us-east-2 --output json
```

出力には、Value パラメータのAMI 情報が含まれています。

```
{
    "Parameter": {
        "Name": "/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended",
        "LastModifiedDate": 1555434128.664,
        "Value": "{\"schema_version\":1,\"image_name\":\"amzn2-ami-ecs-gpu-
hvm-2.0.20190402-x86_64-ebs\",\"image_id\":\"ami-083c800fe4211192f\",\"os\":\"Amazon
 Linux 2\",\"ecs_runtime_version\":\"Docker version 18.06.1-ce\",\"ecs_agent_version
\":\"1.27.0\"}",
        "Version": 9,
        "Type": "String",
        "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ecs/optimized-ami/
amazon-linux-2/gpu/recommended"
    }
}
```

Python

```
from __future__ import print_function

import json
import boto3

ssm = boto3.client('ssm', 'us-east-2')

response = ssm.get_parameter(Name='/aws/service/ecs/optimized-ami/amazon-linux-2/
gpu/recommended')
jsonVal = json.loads(response['Parameter']['Value'])
print("image_id   = " + jsonVal['image_id'])
print("image_name = " + jsonVal['image_name'])
```

出力には、AMI ID と AMI 名のみが含まれます。

```
image_id   = ami-083c800fe4211192f
image_name = amzn2-ami-ecs-gpu-hvm-2.0.20190402-x86_64-ebs
```

GetParameters の使用例を以下に示します。

AWS CLI

```
$ aws ssm get-parameters --names  /aws/service/ecs/optimized-ami/amazon-linux-2/gpu/
recommended/image_name \
                                  /aws/service/ecs/optimized-ami/amazon-linux-2/gpu/
recommended/image_id \
                          --region us-east-2 --output json
```

出力にはパラメータそれぞれの完全なメタデータが含まれています。

```
{
    "InvalidParameters": [],
    "Parameters": [
        {
            "Name": "/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/
image_id",
            "LastModifiedDate": 1555434128.749,
            "Value": "ami-083c800fe4211192f",
            "Version": 9,
            "Type": "String",
            "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ecs/optimized-ami/
amazon-linux-2/gpu/recommended/image_id"
        },
        {
            "Name": "/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/
image_name",
            "LastModifiedDate": 1555434128.712,
            "Value": "amzn2-ami-ecs-gpu-hvm-2.0.20190402-x86_64-ebs",
            "Version": 9,
            "Type": "String",
            "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ecs/optimized-ami/
amazon-linux-2/gpu/recommended/image_name"
        }
    ]
}
```

Python

```
from __future__ import print_function

import boto3

ssm = boto3.client('ssm', 'us-east-2')
```

```
response = ssm.get_parameters(
            Names=['/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/
image_name',
                   '/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/
image_id'])
for parameter in response['Parameters']:
    print(parameter['Name'] + " = " + parameter['Value'])
```

出力には、AMI ID とAMI 名が含まれており、フルパス名が使用されています。

```
/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/image_id =
 ami-083c800fe4211192f
/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/image_name = amzn2-
ami-ecs-gpu-hvm-2.0.20190402-x86_64-ebs
```

以下の例は、GetParametersByPathコマンドの使用方法を示しています。

AWS CLI

```
$ aws ssm get-parameters-by-path --path /aws/service/ecs/optimized-ami/amazon-
linux-2/gpu/recommended \
                                 --region us-east-2 --output json
```

出力には、指定されたパスに基づくすべてのパラメータの完全なメタデータが含まれています。

```
{
    "Parameters": [
        {
            "Name": "/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/
ecs_agent_version",
            "LastModifiedDate": 1555434128.801,
            "Value": "1.27.0",
            "Version": 8,
            "Type": "String",
            "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ecs/optimized-ami/
amazon-linux-2/gpu/recommended/ecs_agent_version"
        },
        {
            "Name": "/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/
ecs_runtime_version",
```

```
            "LastModifiedDate": 1548368308.213,
            "Value": "Docker version 18.06.1-ce",
            "Version": 1,
            "Type": "String",
            "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ecs/optimized-ami/
amazon-linux-2/gpu/recommended/ecs_runtime_version"
        },
        {
            "Name": "/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/
image_id",
            "LastModifiedDate": 1555434128.749,
            "Value": "ami-083c800fe4211192f",
            "Version": 9,
            "Type": "String",
            "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ecs/optimized-ami/
amazon-linux-2/gpu/recommended/image_id"
        },
        {
            "Name": "/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/
image_name",
            "LastModifiedDate": 1555434128.712,
            "Value": "amzn2-ami-ecs-gpu-hvm-2.0.20190402-x86_64-ebs",
            "Version": 9,
            "Type": "String",
            "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ecs/optimized-ami/
amazon-linux-2/gpu/recommended/image_name"
        },
        {
            "Name": "/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/
os",
            "LastModifiedDate": 1548368308.143,
            "Value": "Amazon Linux 2",
            "Version": 1,
            "Type": "String",
            "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ecs/optimized-ami/
amazon-linux-2/gpu/recommended/os"
        },
        {
            "Name": "/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/
schema_version",
            "LastModifiedDate": 1548368307.914,
            "Value": "1",
            "Version": 1,
            "Type": "String",
```

```
            "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ecs/optimized-ami/
amazon-linux-2/gpu/recommended/schema_version"
        }
    ]
}
```

Python

```
from __future__ import print_function

import boto3

ssm = boto3.client('ssm', 'us-east-2')

response = ssm.get_parameters_by_path(Path='/aws/service/ecs/optimized-ami/amazon-
linux-2/gpu/recommended')
for parameter in response['Parameters']:
    print(parameter['Name'] + " = " + parameter['Value'])
```

出力には、名前のフルパスを使用して、指定されたパスにあるすべてのパラメータ名の値が含まれています。

```
/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/ecs_agent_version =
 1.27.0
/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/ecs_runtime_version =
 Docker version 18.06.1-ce
/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/image_id =
 ami-083c800fe4211192f
/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/image_name = amzn2-
ami-ecs-gpu-hvm-2.0.20190402-x86_64-ebs
/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/os = Amazon Linux 2
/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/schema_version = 1
```

詳細については、Amazon Elastic Container Service デベロッパーガイドのAmazon ECS に最適化された AMI メタデータの取得を参照してください。

Amazon Linux の廃止

Amazon Linux AMI (Amazon Linux 1 とも呼ばれます) は 2023 年 12 月 31 日にサポートを終了しました。AWS Batch は 2024 年 1 月 1 日以降、Amazon Linux AMI のセキュリティ更新プログラムや

バグ修正を受け取れないため、サポートを終了しました。Amazon Linux のサポート終了の詳細については「Amazon Linux AMI に関するよくある質問」を参照してください。

予期しないワークロードの中断を避け、セキュリティ更新プログラムやその他の更新プログラムを引き続き受け取れるように、既存の Amazon Linux ベースのコンピューティング環境を Amazon Linux 2023 にアップデートすることをお勧めします。

Amazon Linux AMI を使用するコンピューティング環境は、2023 年 12 月 31 日のサポート終了日を過ぎても機能し続ける可能性があります。ただしこれらのコンピューティング環境は、AWS から新しいソフトウェア更新、セキュリティパッチ、バグ修正を受け取ることはできません。サポート終了後に Amazon Linux AMI 上のこうしたコンピューティング環境を維持することはお客様の責任となります。最適なパフォーマンスとセキュリティを維持するために、AWS Batch コンピューティング環境を Amazon Linux 2023 または Amazon Linux 2 に移行することをお勧めします。

AWS Batch を Amazon Linux AMI から Amazon Linux 2023 または Amazon Linux 2 に移行する際のサポートについては、「コンピューティング環境を更新します - AWS Batch」を参照してください。

Amazon EKS Amazon Linux 2 AMI の廃止

AWS は、2025 年 11 月 26 日より、Amazon EKS 最適化 Amazon Linux 2 AMI のサポートを終了します。最適なパフォーマンスとセキュリティを維持するために、2025 年 11 月 26 日よりも前に AWS Batch Amazon EKS コンピューティング環境を Amazon Linux 2023 に移行することをお勧めします。

2025 年 11 月 26 日のサポート終了日を過ぎても、Amazon EKS コンピューティング環境で Batch 提供の Amazon EKS 最適化 Amazon Linux 2 AMI を引き続き使用できますが、これらのコンピューティング環境は AWS から新しいソフトウェア更新、セキュリティパッチ、バグ修正を受け取れなくなります。サポート終了後に Amazon EKS 最適化 Amazon Linux 2 AMI 上のこうしたコンピューティング環境を維持することはお客様の責任となります。

Amazon EKS AL2 サポート終了の詳細については、「Amazon EKS ユーザーガイド」の「Amazon EKS AMI の廃止に関するよくある質問」を参照してください。

AWS Batch Amazon EKS コンピューティング環境を Amazon Linux 2 から Amazon Linux 2023 に移行するためのサポートについては、「EKS AL2 から EKS AL2023 へアップグレードする方法」を参照してください。

Amazon ECS Amazon Linux 2 AMI の廃止

AWS は Amazon Linux 2 のサポートを終了します。2026 年 1 月から、AWS Batch は新しい Amazon ECS コンピューティング環境のデフォルト AMI を Amazon Linux 2 から Amazon Linux 2023 に変更します。最適なパフォーマンスとセキュリティを維持するために、AWS Batch Amazon ECS コンピューティング環境を Amazon Linux 2023 に移行することをお勧めします。

Amazon Linux 2 のサービス終了の詳細については、「Amazon Linux 2 に関するよくある質問」を参照してください。

Amazon Linux 2 および Amazon Linux 2023 の相違点の詳細については、「Amazon Linux 2023 ユーザーガイド」の「Amazon Linux 2 と Amazon Linux 2023 の比較」を参照してください。

Amazon ECS 最適化 AMI における Amazon Linux 2023 の変更については、「Amazon ECS ユーザーガイド」の「Amazon Linux 2 から Amazon Linux 2023 Amazon ECS 最適化 AMI への移行」を参照してください。

AWS Batch Amazon ECS コンピューティング環境を Amazon Linux 2 から Amazon Linux 2023 に移行するためのサポートについては、「ECS AL2 から ECS AL2023 に移行する方法」を参照してください。

で Amazon EC2 起動テンプレートを使用する AWS Batch

AWS Batch はAmazon EC2 EC2 起動テンプレートの使用をサポートします。起動テンプレートを使用すると、カスタマイズされた AMIs を作成することなく、 AWS Batch コンピューティングリソースのデフォルト設定を変更できます。

> **Note**
>
> 起動テンプレートは AWS Fargate リソースではサポートされていません。

コンピューティング環境に関連付ける前に、起動テンプレートを作成する必要があります。起動テンプレートは、Amazon EC2 コンソールから作成することができます。または、 AWS CLI または AWS SDK を使用できます。たとえば、次の JSON ファイルは、デフォルトの AWS Batch コンピューティングリソース AMI の Docker データボリュームのサイズを変更し、暗号化するように設定する起動テンプレートを表します。

```
{
```

```
    "LaunchTemplateName": "increase-container-volume-encrypt",
    "LaunchTemplateData": {
        "BlockDeviceMappings": [
            {
                "DeviceName": "/dev/xvda",
                "Ebs": {
                    "Encrypted": true,
                    "VolumeSize": 100,
                    "VolumeType": "gp2"
                }
            }
        ]
    }
}
```

前の起動テンプレートを作成するには、JSON を という名前のファイルに保存`lt-data.json`し、次の AWS CLI コマンドを実行します。

```
aws ec2 --region <region> create-launch-template --cli-input-json file://lt-data.json
```

起動テンプレートの詳細については、「Amazon EC2 ユーザーガイド」の「起動テンプレートからのインスタンスの起動」を参照してください。

起動テンプレートを使用してコンピューティング環境を作成する場合、次の既存のコンピューティング環境パラメータを起動テンプレートに移動できます。

> **Note**
>
> 上記のパラメータ (Amazon EC2 タグ以外) が起動テンプレートおよびコンピューティング環境設定の両方で指定されているとします。その場合は、コンピューティング環境パラメータが優先されます。Amazon EC2 タグは、起動テンプレートとコンピューティング環境設定間でマージされます。タグキーの対立がある場合、コンピューティング環境設定の値が優先されます。

- Amazon EC2 のキーペア
- Amazon EC2 AMI ID
- セキュリティグループ ID
- Amazon EC2 タグ

次の起動テンプレートパラメータは によって無視されます AWS Batch。

- インスタンスタイプ (コンピューティング環境作成時にインスタンスタイプを指定)
- インスタンスロール (コンピューティング環境作成時にインスタンスロールを指定)
- ネットワークインターフェイスサブネット (コンピューティング環境の作成時に、サブネットを指定)
- インスタンス市場オプション (スポットインスタンス設定を制御するAWS Batch 必要があります)
- API の終了を無効にする (インスタンスのライフサイクルを制御するAWS Batch 必要があります)

AWS Batch は、インフラストラクチャの更新中にのみ、新しい起動テンプレートバージョンで起動テンプレートを更新します。詳細については、「AWS Batch でコンピューティング環境を更新する」を参照してください。

デフォルトおよびオーバーライド起動テンプレート

コンピューティング環境のデフォルトの起動テンプレートと、特定のインスタンスタイプとファミリーのオーバーライド起動テンプレートを定義できます。これは、コンピューティング環境のほとんどのインスタンスタイプにデフォルトのテンプレートを使用できるようにするのに便利です。

置換変数 `$Default` および `$Latest` は、特定のバージョンに名前を付ける代わりに使用できます。オーバーライド起動テンプレートを指定しない場合、デフォルトの起動テンプレートが自動的に適用されます。

`$Default` または `$Latest`変数を使用する場合、 AWS Batch はコンピューティング環境の作成時に現在の情報を適用します。デフォルトまたは最新バージョンが今後変更される場合は、UpdateComputeEnvironment または AWS マネジメントコンソール - を使用して情報を更新する必要があります AWS Batch。

柔軟性を高めるために、特定のコンピューティングインスタンスタイプまたはファミリーに適用されるオーバーライド起動テンプレートを定義できます。

> **Note**
>
> コンピューティング環境ごとに、最大 10 個のオーバーライド起動テンプレートを指定できます。

targetInstanceTypes パラメータを使用して、このオーバーライド起動テンプレートを使用するインスタンスタイプまたはファミリーを選択します。インスタンスタイプまたはファミリーは、まず instanceTypes パラメータで識別する必要があります。

起動テンプレートのオーバーライドを定義し、後で削除する場合は、空の配列を渡して UpdateComputeEnvironment API オペレーションで overrides パラメータの設定を解除できます。UpdateComputeEnvironment API オペレーションを送信するときに、overrides パラメータを含めないことも選択できます。詳細については、「」を参照してください。LaunchTemplateSpecification.overrides

詳細については、 AWS Batch API リファレンスガイドLaunchTemplateSpecificationOverride.targetInstanceTypesの「」を参照してください。

起動テンプレートの Amazon EC2 ユーザーデータ

起動テンプレート内の Amazon EC2 ユーザーデータを、cloud-init をインスタンス起動時に提供できます。ユーザーデータは、以下を含む一般的な設定シナリオを実行できます。ただし、これらに限定されるものではありません。

- 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
- パーティションおよびファイルシステムの作成

起動テンプレートの Amazon EC2 ユーザーデータは、MIME マルチパートアーカイブの形式であることが必要です。これは、ユーザーデータがコンピューティングリソースの設定に必要な他の AWS Batch ユーザーデータとマージされるためです。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、Amazon ECS コンテナエージェントの設定情報を書き込むユーザーデータシェルスクリプトを使用して、Docker デーモンを設定するクラウドブートフックを組み合わせることができます。

を使用している場合は AWS CloudFormation、AWS::CloudFormation::Init タイプを cfn-init ヘルパースクリプトとともに使用して、一般的な設定シナリオを実行できます。

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。

- コンテンツの種類およびパート境界の宣言: `Content-Type: multipart/mixed; boundary="==BOUNDARY=="`
- 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: `--==BOUNDARY==`。この境界の前の行は空白にしておく必要があります。
  - ブロックのコンテンツの種類の宣言: `Content-Type: text/cloud-config; charset="us-ascii"`。コンテンツの種類の詳細については、Cloud-Init のドキュメントを参照してください。コンテンツタイプ宣言の後の行は空白にしておく必要があります。
  - 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: `--==BOUNDARY==--`。この境界の前の行は空白にしておく必要があります。

> **Note**
>
> Amazon EC2 コンソールの起動テンプレートにユーザーデータを追加するには、ユーザーデータをプレーンテキストとして貼り付けます。または、ファイルからアップロードすることもできます。 AWS CLI または AWS SDK を使用する場合は、この base64 JSON ファイルに示すように、CreateLaunchTemplate を呼び出すときに、まずユーザーデータをエンコードし、その文字列を `UserData`パラメータの値として送信する必要があります。
>
> ```
> {
>     "LaunchTemplateName": "base64-user-data",
>     "LaunchTemplateData": {
>         "UserData":
>  "ewogICAgIkxhdW5jaFRlbXBsYXRlTmFtZSI6ICJpbmNyZWFzZS1jb250YWluZXItdm9sdW..."
>     }
> }
> ```

トピック

- リファレンス: Amazon EC2 起動テンプレートの例

リファレンス: Amazon EC2 起動テンプレートの例

以下は、独自のテンプレートを作成するのに使用できる MIME マルチパートファイルの例です。

例

- 例: 既存の Amazon EFS ファイルシステムのマウント
- 例: デフォルトの Amazon ECS コンテナエージェント設定の上書き
- 例: 既存の Amazon FSx for Lustre ファイルシステムのマウント

例: 既存の Amazon EFS ファイルシステムのマウント

Example

このサンプル MIME マルチパートファイルは、コンピューティングリソースを設定して amazon-efs-utils パッケージをインストールし、既存の Amazon EFS ファイルシステムを /mnt/efs にマウントします。

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/cloud-config; charset="us-ascii"

packages:
- amazon-efs-utils

runcmd:
- file_system_id_01=fs-abcdef123
- efs_directory=/mnt/efs

- mkdir -p ${efs_directory}
- echo "${file_system_id_01}:/ ${efs_directory} efs tls,_netdev" >> /etc/fstab
- mount -a -t efs defaults

--==MYBOUNDARY==--
```

例: デフォルトの Amazon ECS コンテナエージェント設定の上書き

Example

このサンプル MIME マルチパートファイルは、コンピューティングリソースのデフォルトの Docker イメージクリーンアップ設定を上書きします。

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="
```

```
--==MYBOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
echo ECS_IMAGE_CLEANUP_INTERVAL=60m >> /etc/ecs/ecs.config
echo ECS_IMAGE_MINIMUM_CLEANUP_AGE=60m >> /etc/ecs/ecs.config

--==MYBOUNDARY==--
```

例: 既存の Amazon FSx for Lustre ファイルシステムのマウント

Example

このサンプル MIME マルチパートファイルは、コンピューティングリソースを設定して Extras Library から `lustre2.10` パッケージをインストールし、`/scratch` にある既存の Fsx for Lustre ファイルシステムを `fsx` というマウント名でマウントします。この例は Amazon Linux 2 用です。他の Linux ディストリビューションのインストール手順については、「Amazon FSx for Lustre ユーザーガイド」の「Lustre Client のインストール」を参照してください。詳細については、「Amazon FSx for Lustre ユーザーガイド」の「Amazon FSx ファイルシステムの自動マウント」を参照してください。

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/cloud-config; charset="us-ascii"

runcmd:
- file_system_id_01=fs-0abcdef1234567890
- region=us-east-2
- fsx_directory=/scratch
- amazon-linux-extras install -y lustre2.10
- mkdir -p ${fsx_directory}
- mount -t lustre ${file_system_id_01}.fsx.${region}.amazonaws.com@tcp:fsx
 ${fsx_directory}

--==MYBOUNDARY==--
```

コンテナプロパティの volumes および mountPoints メンバーでは、マウントポイントをコンテナにマッピングする必要があります。

```
{
    "volumes": [
        {
            "host": {
                "sourcePath": "/scratch"
            },
            "name": "Scratch"
        }
    ],
    "mountPoints": [
        {
            "containerPath": "/scratch",
            "sourceVolume": "Scratch"
        }
    ],
}
```

インスタンスメタデータサービス (IMDS) の設定

インスタンスメタデータサービス (IMDS) は、EC2 インスタンスに関するメタデータを、それらのインスタンスで実行されているアプリケーションに提供します。すべての新しいワークロードに IMDSv2 を使用し、既存のワークロードを IMDSv1 から IMDSv2 に移行してセキュリティを強化します。IMDS と IMDS の設定の詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスメタデータを使用して EC2 インスタンスを管理する」および「新しいインスタンスのインスタンスメタデータオプションを設定する」を参照してください。

設定シナリオ

コンピューティング環境の設定に基づいて、適切な設定方法を選択する:

起動テンプレートのないデフォルトの AMI

デフォルトの AWS Batch AMI を使用し、起動テンプレートを指定しない場合は、次のいずれかのオプションを選択する:

1. Amazon Linux 2023 のデフォルト AMI を使用する – Amazon Linux 2023 では、デフォルトで IMDSv2 が必要です。コンピューティング環境を作成するときは、イメージタイプとして Amazon Linux 2023 を選択します。
2. アカウントレベルの IMDSv2 設定を指定 – すべての新しいインスタンスにIMDSv2 を要求するように AWS アカウントを設定します。この設定は、アカウントで起動するすべての新しいインス

タンスに影響します。手順については、「Amazon EC2 ユーザーガイド」の「アカウントのデフォルトとして IMDSv2 を設定する」を参照してください。

> **Note**
>
> アカウントレベルの IMDS 設定は、起動テンプレートまたは AMI 設定で上書きできます。起動テンプレートの設定は、アカウントレベルの設定よりも優先されます。

起動テンプレートのないカスタムの AMI

起動テンプレートなしでカスタム AMI を使用する場合は、次のいずれかのオプションを選択する:

1. Amazon Linux 2023 をベースとして使用する – Amazon Linux 2023 をベースイメージとして使用してカスタム AMI を構築します。カスタム AMI の作成について詳しくは、「チュートリアル: コンピューティングリソース AMI を作成する」を参照してください。
2. カスタム AMI で IMDSv2 を設定する – カスタム AMI を作成するときに、IMDSv2 を要求するように設定します。手順については、「Amazon EC2 ユーザーガイド」の「カスタム AMI のインスタンスメタデータのオプションを設定する」を参照してください。
3. アカウントレベルの IMDSv2 設定を指定 – すべての新しいインスタンスにIMDSv2 を要求するように AWS アカウントを設定します。この設定は、アカウントで起動するすべての新しいインスタンスに影響します。手順については、「Amazon EC2 ユーザーガイド」の「アカウントのデフォルトとして IMDSv2 を設定する」を参照してください。

> **Note**
>
> アカウントレベルの IMDS 設定は、起動テンプレートまたは AMI 設定で上書きできます。起動テンプレートの設定は、アカウントレベルの設定よりも優先されます。

起動テンプレートの使用

コンピューティング環境で起動テンプレートを使用する場合は、IMDSv2 を要求するメタデータオプションを起動テンプレートに追加します。Batch での起動テンプレートの使用の詳細については、「で Amazon EC2 起動テンプレートを使用する AWS Batch」を参照してください。

```
{
    "LaunchTemplateName": "batch-imdsv2-template",
```

```
    "VersionDescription": "IMDSv2 only template for Batch",
    "LaunchTemplateData": {
        "MetadataOptions": {
            "HttpTokens": "required"
        }
    }
}
```

AWS CLI を使用して起動テンプレートを作成する:

```
aws ec2 create-launch-template --cli-input-json file://imds-template.json
```

EC2 設定

AWS Batch は、EC2 および EC2 スポットコンピューティング環境に Amazon ECS に最適化された AMI を使用します。デフォルトは Amazon Linux 2 (ECS_AL2) です。2026 年 1 月以降、デフォルトは AL2023 (ECS_AL2023) に変更されます。

AWS は Amazon Linux 2 のサポートを終了します。最適なパフォーマンスとセキュリティを維持するために、AWS Batch Amazon ECS コンピューティング環境を Amazon Linux 2023 に移行することをお勧めします。詳細については、Amazon ECS Amazon Linux 2 AMI の廃止 を参照してください。

予期しないワークロードの中断を避け、セキュリティ更新プログラムやその他の更新プログラムを引き続き受け取れるように、既存の Amazon Linux ベースのコンピューティング環境を Amazon Linux 2023 にアップデートすることをお勧めします。

AWS Batch を Amazon Linux AMI から Amazon Linux 2023 に移行する際のサポートについては、「ECS AL2 から ECS AL2023 に移行する方法」を参照してください。

トピック

- ECS AL2 から ECS AL2023 に移行する方法

ECS AL2 から ECS AL2023 に移行する方法

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、Linux ベースのオペレーティングシステムです。AL2 および AL2023 間の相違点の詳細については、「Amazon Linux 2023 ユーザーガイド」の「Amazon Linux 2023 と Amazon Linux 2 の比較」を参照してください。

2026 年 1 月から、AWS Batch は新しい Amazon ECS コンピューティング環境のデフォルト AMI を Amazon Linux 2 から Amazon Linux 2023 に変更します。これは、AWS が Amazon Linux 2 のサポートを終了するためです。デフォルトの AMI は、新しいコンピューティング環境を作成するときに imageType.Ec2Configuration フィールドに値を指定しない場合に使用されます。最適なパフォーマンスとセキュリティを維持するために、AWS Batch Amazon ECS コンピューティング環境を Amazon Linux 2023 に移行することをお勧めします。

コンピューティング環境の設定方法に応じて、AL2 から AL2023 への以下のアップグレードパスのいずれかを使用できます。

Ec2Configuration.ImageType を使用したアップグレード

- 起動テンプレートまたは起動テンプレートの上書きを使用していない場合は、Ec2Configuration.ImageType を ECS_AL2023 (または GPU インスタンスを使用している場合は ECS_AL2023_NVIDIA) に変更し、UpdateComputeEnvironment を実行します。
- Ec2Configuration.ImageIdOverride を指定する場合、Ec2Configuration.ImageType は Ec2Configuration.ImageIdOverride で指定された AMI タイプと一致する必要があります。

  ImageIdOverride と ImageType が一致しない場合、コンピューティング環境が正しく機能しない可能性があります。

起動テンプレートを使用したアップグレード

- ECS_AL2023 に基づいて AMI を指定する起動テンプレートを使用する場合は、起動テンプレートが Amazon Linux 2023 と互換性があることを確認してください。Amazon ECS 最適化 AMI における Amazon Linux 2023 の変更については、「Amazon ECS ユーザーガイド」の「Amazon Linux 2 から Amazon Linux 2023 Amazon ECS 最適化 AMI への移行」を参照してください。
- AL2023 AMI の場合、カスタムユーザーデータまたは初期化スクリプトが AL2023 環境およびパッケージ管理システムと互換性があることを確認します。

CloudFormation を使用したアップグレード

- CloudFormation を使用してコンピューティング環境を管理する場合は、テンプレートを更新して、Ec2Configuration の ImageType プロパティを ECS_AL2 から ECS_AL2023 (または GPU インスタンスを使用する場合は ECS_AL2023_NVIDIA) に変更します。

```
ComputeEnvironment:
  Type: AWS::Batch::ComputeEnvironment
```

```
  Properties:
    ComputeResources:
      Ec2Configuration:
        - ImageType: ECS_AL2023
```

次に、CloudFormation スタックを更新して変更を適用します。

- CloudFormation テンプレートで `ImageIdOverride` を使用してカスタム AMI が指定されている場合は、AMI ID が AL2023 ベースの AMI に対応し、`ImageType` 設定と一致していることを確認します。

移行に関する考慮事項

Amazon Linux 2 から Amazon Linux 2023 に移行するときは、次の点を考慮してください。

- パッケージ管理 – Amazon Linux 2023 はパッケージ管理において `yum` の代わりに `dnf` を使用します。
- システムサービス – 一部のシステムサービスとその設定は、AL2 と AL2023 で異なる場合があります。
- コンテナランタイム – AL2 と AL2023 はどちらも Docker をサポートしていますが、AL2023 のデフォルト設定は異なる場合があります。
- セキュリティ – AL2023 には強化されたセキュリティ機能が含まれており、セキュリティ関連の設定の更新が必要になる場合があります。
- Instance Metadata Service Version 2 (IMDSv2) – IMDSv2 は、EC2 インスタンスメタデータにアクセスするためにトークンベースの認証を要求するセッション指向のサービスであり、セキュリティを強化します。IMDS の詳細については、「Amazon EC2 ユーザーガイド」の「Instance Metadata Service Version 2 の仕組み」を参照してください。

変更と移行に関する考慮事項の包括的なリストについては、「Amazon ECS ユーザーガイド」の「Amazon Linux 2 から Amazon Linux 2023 Amazon ECS 最適化 AMI への移行」を参照してください。

AWS Batch のインスタンスタイプの配分戦略

マネージド型のコンピューティング環境が作成されると、AWS Batch はジョブのニーズに最も適したインスタンスタイプを、指定された `instanceTypes` から選択します。配分戦略は、AWS Batch

が追加のキャパシティを必要とする場合の動作を定義します。このパラメータは、Fargate リソースで実行されているジョブには適用されません。このパラメータを指定しないようにしてください。

`BEST_FIT` (デフォルト)

AWS Batch は、最もコストの低いインスタンスタイプを優先して、ジョブのニーズに最も適したインスタンスタイプを選択します。選択したインスタンスタイプの追加インスタンスが利用できない場合、AWS Batch は追加インスタンスが利用可能になるまで待機します。十分なインスタンスが利用できない場合、またはユーザーが Amazon EC2 Service Quotasに達した場合、現在実行中のジョブが完了するまで追加のジョブは実行されません。この配分戦略により、コストは低くなりますが、スケーリングは制限される場合があります。`BEST_FIT` でスポットフリートを使用する場合は、スポットフリートの IAM ロールを指定する必要があります。`BEST_FIT` コンピューティング環境の更新時にはサポートされません。詳細については、AWS Batch でコンピューティング環境を更新する を参照してください。

> **Note**
>
> アカウントの AWS リソースは AWS Batch が管理します。BEST_FIT 配分戦略をとるコンピューティング環境には当初、デフォルトで起動設定が使用されていました。しかし、新しい AWS アカウントでの起動設定の使用は、いずれ制限されます。したがって、2024 年 4 月下旬以降、新規作成された BEST_FIT コンピューティング環境はデフォルトでテンプレートを起動します。サービスロールに起動テンプレートを管理するアクセス権限がない場合、AWS Batch は起動設定を引き続き使用できます。既存のコンピューティング環境では引き続き起動設定が使用されます。

`BEST_FIT_PROGRESSIVE`

AWS Batch は、キュー内のジョブの要件を満たすのに十分な大きさのインスタンスタイプを追加で選択します。vCPUあたりのコストが低いインスタンスタイプが優先されます。以前選択したインスタンスタイプの追加のインスタンスが利用できない場合、AWS Batch は新しいインスタンスタイプを選択します。

> **Note**
>
> マルチノード並列ジョブでは、AWS Batch は使用できる最適なインスタンスタイプを選択します。容量不足が原因でインスタンスタイプが使用できなくなった場合、ファミリー内の他のインスタンスタイプは起動されません。

SPOT_CAPACITY_OPTIMIZED

AWS Batch は、キュー内のジョブの要件を満たすのに十分な大きさの 1 つ以上のインスタンスタイプを選択します。中断されにくいインスタンスタイプが推奨されます。この配分戦略は、スポットインスタンスのコンピューティングリソースでのみ使用できます。

SPOT_PRICE_CAPACITY_OPTIMIZED

料金とキャパシティの最適化配分戦略では、料金とキャパシティの両方を考慮して、中断される可能性が最も低く、料金が最も低いスポットインスタンスプールを選択します。この配分戦略は、スポットインスタンスのコンピューティングリソースでのみ使用できます。

> **Note**
>
> ほとんどの場合、SPOT_CAPACITY_OPTIMIZED ではなく SPOT_PRICE_CAPACITY_OPTIMIZED を使用することをお勧めします。

BEST_FIT_PROGRESSIVE および BEST_FIT 戦略はオンデマンドまたはスポットインスタンスを使用し、SPOT_CAPACITY_OPTIMIZED および SPOT_PRICE_CAPACITY_OPTIMIZED 戦略はスポットインスタンスを使用します。ただし、AWS Batch はお客様の容量要件を満たすために maxvCpus を超える必要がある場合があります。この場合、AWS Batch は 1 つのインスタンスよりも maxvCpus を超えることはありません。

コンピューティングリソースメモリの管理

Amazon ECS コンテナエージェントが、コンテナインスタンスコンピュートリソースをクラスターコンピューティング環境に登録する場合、エージェントはタスクジョブの実行のために、コンテナインスタンスコンピューティングリソースが使用できるメモリ容量を決定する必要があります。プラットフォームのメモリ・オーバーヘッドとシステム・カーネルが占有するメモリのため、この数値はAmazon EC2インスタンスのインストール済みメモリ量とは異なります。例えば、m4.large インスタンスには 8 GiB のメモリがインストールされています。しかし、これはコンピューティングが登録されたときに、 ジョブに厳密に8192 MiBのメモリに常に変換されるとは限りません。

ジョブに8192 MiB を指定した場合、この要件を満たす使用可能なメモリが 8192 MiB以上のメモリがないとします。その場合、そのジョブをコンピューティング環境に置くことができなくなります。マネージド型コンピューティング環境を使用している場合、リクエストに対応するためには、AWS Batchはより大きなインスタンスタイプを起動する必要があります。

デフォルト AWS Batch コンピューティングリソースAMIも、Amazon ECS コンテナエージェントやその他の重要なシステムプロセス用に32 MiBのメモリを予約します。このメモリはジョブ割り当てには利用できない。詳細については、システムメモリを予約するを参照してください。

Amazon ECS コンテナエージェントは、Docker `ReadMemInfo()`関数を使用してオペレーティングシステムで使用可能な合計メモリのクエリを実行します。Linuxは、合計メモリを判断するためにコマンドラインユーティリティを提供します。

Example- Linux 合計メモリを決定

free コマンドは、オペレーティングシステムによって認識される合計メモリを復活します。

```
$ free -b
```

Amazon ECSに最適化されたAmazon Linux AMIを実行する `m4.large` インスタンスの出力例。

```
             total       used       free     shared    buffers     cached
Mem:    8373026816  348180480 8024846336      90112   25534464  205418496
-/+ buffers/cache:  117227520 8255799296
```

このインスタンスには 8373026816バイトの合計メモリがあります。つまり、7985 MiBがタスクに使用できます。

トピック

- システムメモリを予約する
- チュートリアル: コンピューティングリソースのメモリを表示する
- Amazon EKS の AWS Batch におけるメモリと vCPU に関する考慮事項

システムメモリを予約する

タスクジョブでコンピューティングリソース上の全メモリを占有している場合、タスクジョブがメモリを必要とする重要なシステムプロセスと競い、システム障害の引き金となる可能性があります。Amazon ECS コンテナエージェントは、`ECS_RESERVED_MEMORY` と呼ばれる設定変数を提供します。この変数を使用して、タスクジョブに割り当てられたプールから特定のMiBメモリを削減できます。これにより、重要なシステムプロセスのメモリを効果的に確保することができます。

デフォルト AWS Batch コンピュートリソース AMI は、Amazon ECS コンテナエージェントやその他の重要なシステムプロセス用に 32 MiB のメモリを予約します。Amazon ECS コンテナエージェントやその他の重要なシステムプロセスには、5% のメモリバッファを予約することをお勧めします。

チュートリアル: コンピューティングリソースのメモリを表示する

Amazon ECS コンソールまたは DescribeContainerInstances API操作で、コンテナインスタンスコンピューティングリソースが登録されているメモリ量を表示できます。特定のインスタンスタイプに対して、可能な限り大きなメモリをタスクジョブに割り当てて、リソース使用率を最大化しようとする場合は、そのコンピューティングリソースに使用可能なメモリを観察でき、その後にタスクジョブに可能な限り多くのメモリを割り当てることができます。

コンピューティングリソースのメモリを表示するために

1. コンソールをhttps://console.aws.amazon.com/ecs/v2で開きます。
2. クラスターを選択し、表示するコンピューティングリソースをホストするクラスターを選択します。

   コンピューティング環境のクラスター名は、クラスター環境名で始めます。
3. インフラストラクチャ を選択します。
4. コンテナインスタンスで、コンテナインスタンスを選択します。
5. リソースとネットワーキング セクションに、コンピューティングリソース用に登録され、使用できるメモリが表示されます。

   [総容量] は、Amazon ECS の初回起動時に登録されたコンピューティングリソースのメモリの値です。[使用可能] メモリの値は。まだジョブに割り当てられていないメモリの値です。

Amazon EKS の AWS Batch におけるメモリと vCPU に関する考慮事項

Amazon EKSのAWS Batchでは、コンテナで使用できるリソースを指定できます。例えば、vCPU とメモリリソースの `requests` 値、または `limits` 値を指定できます。

vCPU リソースを指定する際の制約は次のとおりです：

- 少なくとも 1つのvCPU`requests`、または`limits`値 のいずれか指定する必要があります。
- １つのvCPU ユニットは、１つの物理コアまたは仮想コアに相当します。
- vCPUの値は、整数または0.25単位で入力する必要があります。
- 有効な最小値は0.25です。
- 両方が特定される場合、`requests`の値は`limits`の値以下でなくてはなりません。このようにして、ソフトとハードのvCPU設定の両方を行うことができます。

- vCPU値をミリCPU形式で指定することはできません。例えば、100m は有効な値ではありません。
- AWS Batch は、この requests 値をスケーリングの決定に使用します。requests 値が指定されていない場合、limits 値は requests 値にコピーされます。

メモリリソースを指定する際の制約は、次のとおりです：

- 少なくとも1つのメモリrequests または limits 値を指定する必要があります。
- メモリ値は mebibytes (MiBs) 内である必要があります。
- 両方を指定する場合、requests 値は limits 値と等しくなければなりません。
- AWS Batch は、この requests 値をスケーリングの決定に使用します。requests 値が指定されていない場合、limits 値は requests 値にコピーされます。

GPUリソースを指定する際の制約は、次のとおりです：

- 両方を指定する場合、requests 値はlimits 値と等しくなければなりません。
- AWS Batchは、この requests 値をスケーリングの決定に使用します。requests 値が指定されていない場合、limits 値は requests 値にコピーされます。

例: ジョブ定義

Amazon EKS ジョブ定義の以下の AWS Batch では、ソフト vCPU 共有を設定します。これにより、Amazon EKS のAWS Batchは、インスタンスタイプの vCPU容量のすべて使用できるようになります。ただし、実行中の他のジョブがある場合、そのジョブには最大数の 2 vCPUs が割り当てられます。メモリは、2 GBに制限されています。

```
{
   "jobDefinitionName": "MyJobOnEks_Sleep",
   "type": "container",
   "eksProperties": {
       "podProperties": {
           "containers": [
               {
                   "image": "public.ecr.aws/amazonlinux/amazonlinux:2",
                   "command": ["sleep", "60"],
                   "resources": {
                       "requests": {
```

```
                            "cpu": "2",
                            "memory": "2048Mi"
                        }
                    }
                }
            ]
        }
    }
}
```

Amazon EKS AWS Batch の以下のジョブ定義のrequest 値は 1 で、最大の 4 vCPUs をジョブに割り当てます。

```
{
    "jobDefinitionName": "MyJobOnEks_Sleep",
    "type": "container",
    "eksProperties": {
        "podProperties": {
            "containers": [
                {
                    "image": "public.ecr.aws/amazonlinux/amazonlinux:2",
                    "command": ["sleep", "60"],
                    "resources": {
                        "requests": {
                            "cpu": "1"
                        },
                        "limits": {
                            "cpu": "4",
                            "memory": "2048Mi"
                        }
                    }
                }
            ]
        }
    }
}
```

以下のAmazon EKS のAWS Batchジョブ定義では 、vCPU limits 値をに 1を、メモリ limits 値を 1 GB に設定します。

```
{
    "jobDefinitionName": "MyJobOnEks_Sleep",
```

```
   "type": "container",
   "eksProperties": {
       "podProperties": {
           "containers": [
               {
                   "image": "public.ecr.aws/amazonlinux/amazonlinux:2",
                   "command": ["sleep", "60"],
                   "resources": {
                       "limits": {
                           "cpu": "1",
                           "memory": "1024Mi"
                       }
                   }
               }
           ]
       }
   }
}
```

AWS Batch Amazon EKS 上のジョブを Amazon EKS ポッドにAWS Batch 変換するときに、AWS Batch `limits` 値をその `requests` 値にコピーします。これは、`requests` 値が指定されていない場合です。前述のジョブ定義の例を送信すると、ポッドspec は次のようになります。

```
apiVersion: v1
kind: Pod
...
spec:
  ...
  containers:
    - command:
        - sleep
        - 60
      image: public.ecr.aws/amazonlinux/amazonlinux:2
      resources:
        limits:
          cpu: 1
          memory: 1024Mi
        requests:
          cpu: 1
          memory: 1024Mi
      ...
```

ノードCPUとメモリの予約

AWS Batch は、vCPU とメモリの予約に関してbootstrap.shファイルのデフォルトロジックに依存します。bootstrap.sh ファイルの詳細については、bootstrap.sh を参照してください。vCPUメモリリソースのサイズを決定する場合、以下の例を検討してください。

> **Note**
>
> 実行中のインスタンスがない場合、vCPUとメモリの予約が最初にAWS Batchスケーリングロジックと意思決定に影響を与える可能性があります。インスタンスが実行されたら、AWS Batch は初期割り当てを調整します。

例: ノード CPU 予約

CPU 予約値は、インスタンスで使用可能な vCPU の総数を使用してミリコア単位で計算されます。

vCPU番号	予約率
1	6%
2	1%
3 ~ 4	0.5%
4以上	0.25%

上記の値を使用すると、次のようになります:

- 仮想CPUが2つある c5.large インスタンスのvCPUs予約値は,70 m です。これは次の方法で計算されます：(1*60) + (1*10) = 70 m。
- 96個のvCPUsを搭載したc5.24xlarge インスタンスのCPU予約値は、 310 mです。これは次の方法で計算されます：(1*60) + (1*10) + (2*5) + (92*2.5) = 310 m。

この例では、c5.large インスタンスでジョブを実行するために使用できるミリコアvCPU ユニットは、1930 (計算は2000 - 70) です。ジョブに 2 (2*1000 m) の vCPU ユニットが必要で、そのジョブが 単一のc5.large インスタンスに収まらないとします。ただし、1.75 vCPU ユニットが必要なジョブには適しています。

例: ノードメモリ予約

メモリ予約値は、以下を使用してメビバイト単位で計算されます:

- インスタンスキャパシティーは MB 単位です。例えば、8 GB のインスタンスは 7,748 です MiB。
- kubeReserved 値。kubeReserved 値は、システムデーモン用に確保するメモリ量です。kubeReserved 値は次のように計算されます: ((11 * インスタンスタイプでサポートされる最大ポッド数) + 255)。各インスタンスタイプによりサポートされるポッドの最大数のリストは、GitHub のeni-max-pods.txtを参照してください。
- HardEvictionLimit 値。使用可能なメモリがHardEvictionLimit 値を下回ると、インスタンスはポッドを削除しようとします。

割り当て可能なメモリの計算式は次のとおりです : (*Instance_Capacity_IN_MIB*)-(11 * (*#####*#))-255-(*HardEvictionLimit ##*))。

1つの c5.large インスタンスは最大29個のポッドをサポートします。HardEvictionLimit 値が100 MiBの8 GB c5.large インスタンスの場合、割り当て可能なメモリは7074ですMiB。これは次の方法で計算されます: (7748-(11 * 29) -255 -100) = 7074 MiB。この例では、8,192件のMiB のタスクジョブは 、8 gibibyte (GiB) インスタンスであるにもかかわらず、このインスタンスには当てはまりません。

DaemonSets

DaemonSets を使用する場合は、以下を考慮してください :

- Amazon EKS インスタンスでAWS Batchがまったく実行されていない場合、最初はDaemonSets AWS Batch のスケーリングロジックと意思決定に影響する可能性があります。AWS Batchは最初に0.5 のvCPU ユニットと、500 MiB を想定されるDaemonSetsに割り当てます。インスタンス実行されたら、AWS Batchは初期割り当てを調整します。
- DaemonSetがvCPUまたはメモリの制限を定義している場合、Amazon EKSジョブのAWS Batchが少なくなります。AWS Batchジョブに割り当てられる DaemonSets の数は、できるだけ少なくすることをお勧めします。

Fargate のコンピューティング環境

Fargate は、Amazon EC2 インスタンスのサーバーやクラスターを管理することなくコンテナを実行する AWS Batch ために で使用できるテクノロジーです。Fargateを使用すると、コンテナを実行す

るために仮想マシンのクラスター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、クラスター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 リソースを使用してジョブを実行する場合、アプリケーションをコンテナにパッケージ化し、CPU とメモリ要件を指定して、ネットワークとIAM ポリシーを定義して、アプリケーションを起動します。各 Fargate ジョブは、独自の分離境界を持ち、基本となるカーネル、CPU リソース、メモリリソース、Elastic Network Interface を別のジョブと共有しません。

トピック

Fargateをいつ使うべきか

ほとんどのシナリオで Fargate を使用することをお勧めします。Fargate は、コンテナに指定したリソース要件に厳密に一致するようにコンピューティングを起動し、スケールします。Fargate を使用すると、追加のサーバーに対してオーバープロビジョニングまたは料金を支払う必要はありません。また、インスタンスタイプなど、インフラストラクチャ関連のパラメータの詳細について心配する必要はありません。コンピューティング環境をスケールアップする必要がある場合、Fargate リソースで実行されるジョブをより迅速に開始できます。通常、新しい Amazon EC2 インスタンスの作成に数分かかります。しかし、Fargate で実行されるジョブは約 30 秒でプロビジョニングできます。正確な所要時間は、コンテナイメージのサイズやジョブ数など、いくつかの要因によって異なります。

ただし、ジョブに次のいずれかが必要な場合は、Amazon EC2 を使用することをお勧めします。

- 16 個以上の vCPU
- 120 ギガバイト (GiB) 以上のメモリ
- 1 個のGPU
- 1 個のカスタム Amazon マシンイメージ (AMI)
- LinuxParameters パラメータのいずれか

ジョブの数が多い場合は、Amazon EC2 インフラストラクチャを使用することをお勧めします。例えば、同時実行ジョブの数が Fargate スロットリング制限を超える場合です。これは、EC2 で

は、Fargate リソースよりも高いレートで EC2 リソースにジョブをディスパッチできるためです。さらに、EC2 を使用すると、同時に実行できるジョブが増えます。詳細については、「Amazon Elastic Container Service デベロッパーガイド」の「Fargate サービスクォータ」を参照してください。

Fargate でのジョブ定義

AWS Batch の ジョブは、使用可能なすべてのジョブ定義パラメータをサポート AWS Fargate しているわけではありません。一部のパラメータはまったくサポートされていません。また、その他のパラメータは Fargate ジョブでは異なる動作をします。

次のリストでは、Fargate ジョブで有効でないか、または制限されていないジョブ定義パラメータについて説明します。

`platformCapabilities`

`FARGATE` と指定する必要があります。

```
"platformCapabilities": [ "FARGATE" ]
```

`type`

`container` と指定する必要があります。

```
"type": "container"
```

`containerProperties` のパラメータ

`executionRoleArn`

Fargate リソースで実行されているジョブには指定する必要があります。詳細については、Amazon Elastic Container Service デベロッパーガイドのタスク用の IAM ロールを参照してください。

```
"executionRoleArn": "arn:aws:iam::123456789012:role/ecsTaskExecutionRole"
```

`fargatePlatformConfiguration`

(オプション、Fargate ジョブ定義の場合のみ)。Fargate プラットフォームのバージョンを指定するか、最新のプラットフォームバージョンの場合は `LATEST` を指定します。`platformVersion` の可能な値はデフォルトで `1.3.0`、`1.4.0`、`LATEST` です。

```
"fargatePlatformConfiguration": { "platformVersion": "1.4.0" }
```

`instanceType`, `ulimits`

Fargate リソースで実行されているジョブには適用されません。

`memory`, `vcpus`

これらの設定は、`resourceRequirements` で指定する必要があります。

`privileged`

このパラメータを指定しないか、`false` を指定します。

```
"privileged": false
```

`resourceRequirements`

メモリと vCPU の要件は、サポートされている値を使用して指定する必要があります。GPU リソースは、Fargate リソースで実行されているジョブではサポートされていません。

GuardDuty Runtime Monitoring を使用する場合、GuardDuty セキュリティエージェントには多少のメモリオーバーヘッドがあります。したがって、メモリ制限には GuardDuty セキュリティエージェントのサイズを含める必要があります。GuardDuty セキュリティエージェントのメモリ制限については、「GuardDuty ユーザーガイド」の「CPU およびメモリ制限」を参照してください。ベストプラクティスの詳細については、「Amazon ECS 開発者ガイド」の「ランタイムモニタリングを有効にした後、Fargate タスクのメモリ不足エラーに対処するにはどうすればよいですか?」を参照してください。

```
"resourceRequirements": [
  {"type": "MEMORY", "value": "512"},
  {"type": "VCPU",   "value": "0.25"}
]
```

`linuxParameters` のパラメータ

`devices`, `maxSwap`, `sharedMemorySize`, `swappiness`, `tmpfs`

Fargate リソースで実行されているジョブには適用されません。

`logConfiguration` のパラメータ

`logDriver`

`awslogs` と `splunk` のみがサポートされています。詳細については、awslogs ログドライバーを使用するを参照してください。

`networkConfiguration` のメンバー

`assignPublicIp`

プライベートサブネットにインターネットへトラフィックを送るための NAT ゲートウェイが接続されていない場合、`assignPublicIp` はENABLEDでなければなりません。詳細については、AWS Batch IAM 実行ロールを参照してください。

Fargate のジョブキュー

AWS Batch のジョブキュー AWS Fargate は基本的に変更されません。`computeEnvironmentOrder` にリストされているコンピューティング環境は、すべて Fargate のコンピューティング環境 (`FARGATE` または `FARGATE_SPOT`) であることが唯一の制限事項です。EC2 と Fargate コンピューティング環境は混在できません。

Fargate のコンピューティング環境

AWS Batch の コンピューティング環境は、使用可能なすべてのコンピューティング環境パラメータをサポート AWS Fargate しているわけではありません。一部のパラメータはまったくサポートされていません。Fargate では特定の制限があるパラメータもあります。

次のリストでは、Fargate ジョブで有効でない、または制限されているコンピューティング環境パラメータについて説明します。

`type`

このパラメータを `MANAGED` に設定する必要があります。

```
"type": "MANAGED"
```

`computeResources` オブジェクトのパラメータ

`allocationStrategy`, `bidPercentage`, `desiredvCpus`, `imageId`, `instanceTypes`, `ec2Configuration`, `ec2KeyPair`, `instanceRole`, `launchTemplate`, `minvCpus`, `placementGroup`, `spotIamFleetRole`

これらは、Fargate コンピューティング環境には適用されないため、指定できません。

`subnets`

このパラメータにリストされているサブネットに NAT ゲートウェイが接続されていない場合は、ジョブ定義の `assignPublicIp` パラメータを `ENABLED` に設定する必要があります。

`tags`

これは、Fargate コンピューティング環境では適用されないため、指定できません。Fargate のコンピューティング環境用のタグを指定するには、`computeResources` オブジェクトにない `tags` パラメータを使用します。

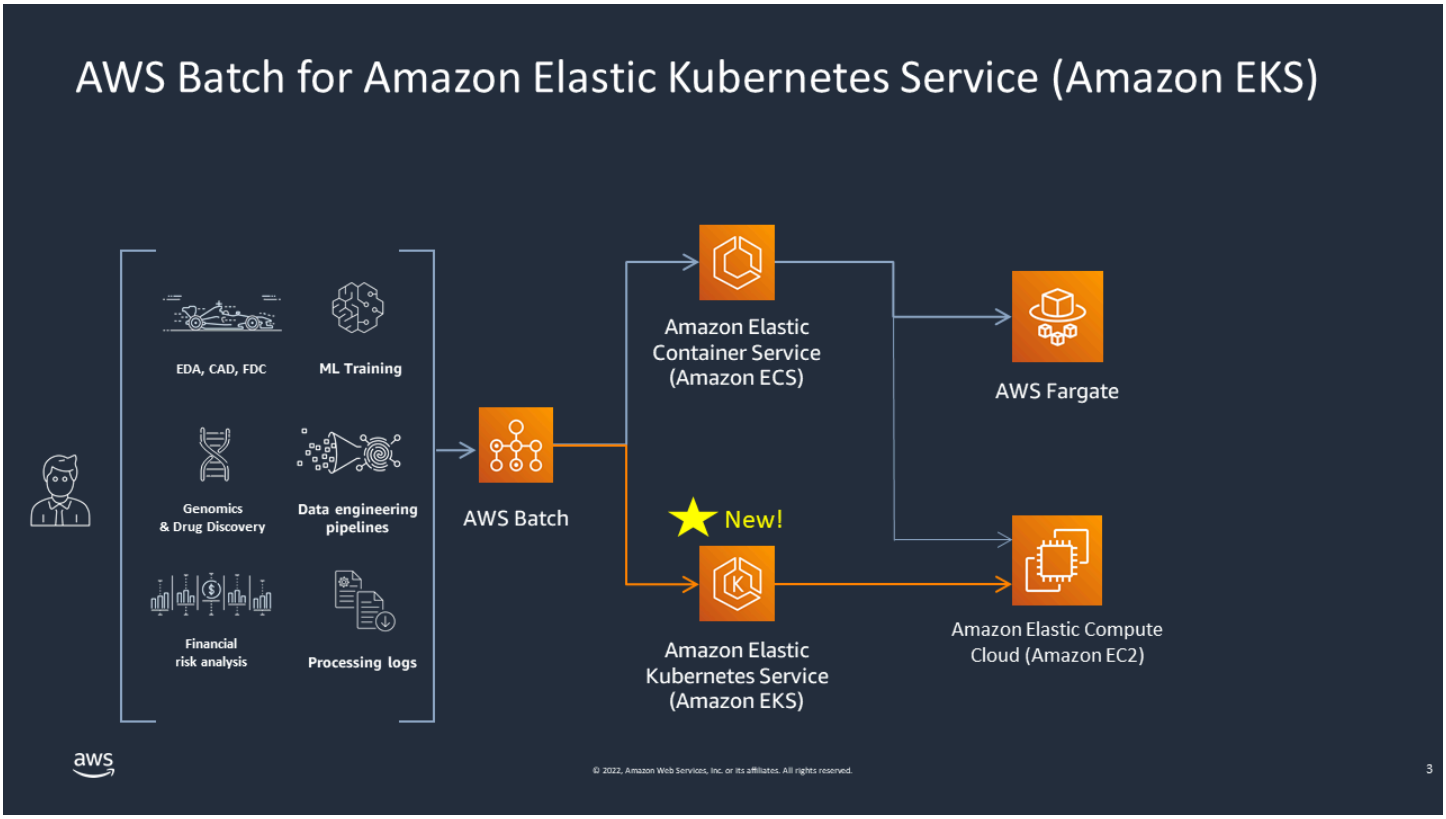
`type`

これは `FARGATE` または `FARGATE_SPOT` であることが必要です。

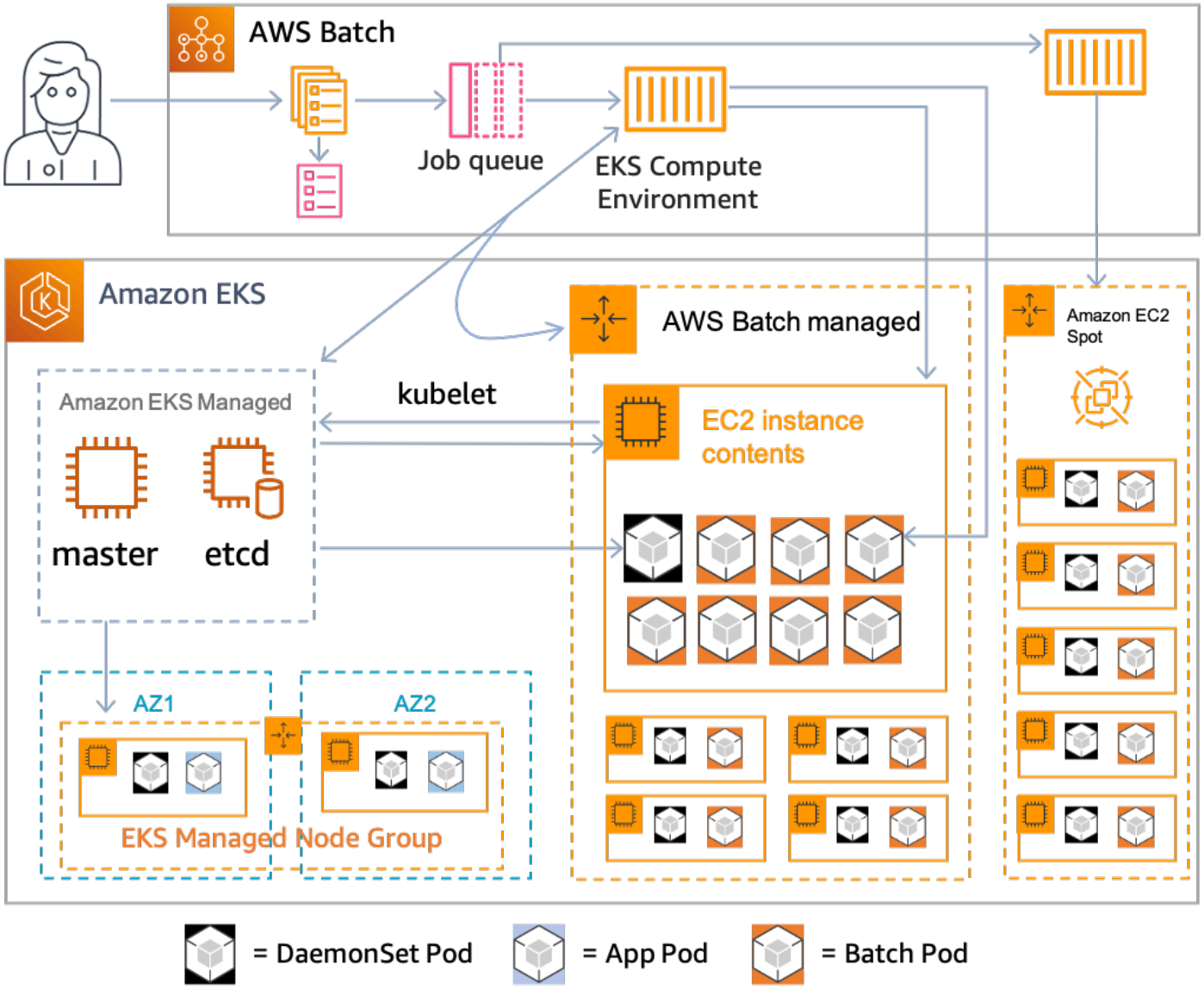
```
"type": "FARGATE_SPOT"
```

Amazon EKS コンピュート環境

Amazon EKS で AWS Batch のご利用開始にあたって EKS コンピュート環境を作成するための簡単なガイドを提供します。このセクションでは、Amazon EKS のコンピューティング環境についてより詳しく説明します。



AWS Batch は、マネージドバッチ機能を提供することで、Amazon EKS クラスターのバッチワークロードを簡素化します。これには、キューイング、依存関係の追跡、マネージドジョブの再試行と優先順位、ポッド管理、ノードスケーリングが含まれます。 は、複数のアベイラビリティーゾーンと複数の Amazon EC2 インスタンスタイプとサイズを処理 AWS Batch できます。 は、いくつかの Amazon EC2 スポットのベストプラクティス AWS Batch を統合して、耐障害性のある方法でワークロードを実行するため、中断が少なくなります。 AWS Batch を使用すると、わずかな夜間ジョブや数百万のミッションクリティカルなジョブを自信を持って実行できます。



AWS Batch は、Amazon Elastic Kubernetes Service (Amazon EKS) によって管理されるKubernetes クラスター内のバッチワークロードをオーケストレーションするマネージドサービスです。 は、「オーバーレイ」モデルを使用して、クラスターの外部でこのオーケストレーション AWS Batch を実行します。 AWS Batch はマネージドサービスであるため、クラスターにインストールまたは管理するKubernetesコンポーネント (オペレーターやカスタムリソースなど) はありません。 では、Kubernetes API サーバーとの AWS Batch 通信を に許可するロールベースのアクセスコントロール (RBAC) Kubernetes でクラスターを設定 AWS Batch する必要があります。 は Kubernetes APIs を AWS Batch 呼び出してポッドとノードを作成、モニタリング、削除します。

AWS Batch には、ジョブ容量の割り当てに関する最適化により、ジョブキューの負荷に基づいて Kubernetesノードをスケールするスケーリングロジックが組み込まれています。ジョブキューが空の場合、 はノードを設定した最小容量に AWS Batch スケールダウンします。デフォルトではゼロで

す。 はこれらのノードのライフサイクル全体 AWS Batch を管理し、ノードにラベルとテイントをデコレーションします。これにより、他のKubernetesワークロードは が管理するノードに配置されません AWS Batch。例外は です。これはDaemonSets、ジョブの適切な実行に必要なモニタリングやその他の機能を提供するために AWS Batch ノードをターゲットにできます。さらに、 AWS Batch は、管理していないクラスター内のノードでジョブ、特にポッドを実行しません。こうすることで、クラスター上の他のアプリケーションに個別のスケーリングロジックとサービスを使用できます。

ジョブを送信するには AWS Batch、 AWS Batch API と直接やり取りします。 はジョブを AWS Batch に変換しpodspecs、Amazon EKS クラスター AWS Batch の によって管理されるノードにポッドを配置するリクエストを作成します。kubectl などのツールを使用して、実行中のポッドやノードを表示できます。ポッドの実行が完了すると、 はKubernetesシステムへの負荷を軽減するために作成したポッド AWS Batch を削除します。

有効な Amazon EKS クラスターを に接続することで開始できます AWS Batch。次に、 AWS Batch ジョブキューをアタッチし、podspec同等の属性を使用して Amazon EKS ジョブ定義を登録します。最後に、ジョブ定義を参照する SubmitJob API オペレーションを使用してジョブを送信します。詳細については、「Amazon EKS で AWS Batch のご利用開始にあたって」を参照してください。

Amazon EKS

トピック

- Amazon EKS デフォルト AMI
- AMI が混在する環境
- サポートされる Kubernetes バージョン
- コンピューティング環境の Kubernetes バージョンを更新する
- Kubernetes ノードの責任分担
- AWS Batch マネージドノードDaemonSetで を実行する
- Amazon EKS 起動テンプレートをカスタマイズする
- EKS AL2 から EKS AL2023 へアップグレードする方法

Amazon EKS デフォルト AMI

Amazon EKS コンピューティング環境を作成するときは、Amazon マシンイメージ (AMI) を指定する必要はありません。 は、CreateComputeEnvironment リクエストで指定されたKubernetesバー

ジョンとインスタンスタイプに基づいて Amazon EKS 最適化 AMI AWS Batch を選択します。一般的に、デフォルトの AMI 選択を使用することをお勧めします。Amazon EKS 最適化 AMI の詳細については、「Amazon EKS ユーザーガイド」の「Amazon EKS 最適化 Amazon Linux AMI」を参照してください。

> **Important**
>
> Amazon Linux 2023 AMIs は、Amazon EKS AWS Batch の のデフォルトです。
> AWS は、 から Amazon EKS AL2-optimizedおよび AL2-accelerated AMIs のサポートを終了します11/26/25。サポート11/26/25end-of-supportを過ぎても、Amazon EKS コンピューティング環境で AWS Batch Amazon EKS 最適化 Amazon Linux 2 AMIs を引き続き使用できますが、これらのコンピューティング環境は から新しいソフトウェア更新、セキュリティパッチ、またはバグ修正を受け取りません AWS。AL2 から AL2023 へのアップグレードの詳細については、「AWS Batch ユーザーガイド」の「EKS AL2 から EKS AL2023 へアップグレードする方法」を参照してください。

次のコマンドを実行して、Amazon EKS コンピューティング環境に AWS Batch 選択した AMI タイプを確認します。次の例は GPU 以外のインスタンスタイプです。

```
# compute CE example: indicates Batch has chosen the AL2 x86 or ARM EKS 1.32 AMI,
 depending on instance types
    $ aws batch describe-compute-environments --compute-environments My-Eks-CE1 \
        | jq '.computeEnvironments[].computeResources.ec2Configuration'
    [
      {
        "imageType": "EKS_AL2",
        "imageKubernetesVersion": "1.32"
      }
    ]
```

次の例は GPU インスタンスタイプです。

```
# GPU CE example: indicates Batch has choosen the AL2 x86 EKS Accelerated 1.32 AMI
    $ aws batch describe-compute-environments --compute-environments My-Eks-GPU-CE \
        | jq '.computeEnvironments[].computeResources.ec2Configuration'
    [
      {
        "imageType": "EKS_AL2_NVIDIA",
        "imageKubernetesVersion": "1.32"
```

```
      }
    ]
```

AMI が混在する環境

起動テンプレートオーバーライドを使用して、Amazon Linux 2 (AL2) AMI と Amazon Linux 2023 (AL2023) AMI の両方を持つコンピューティング環境を作成できます。これは、アーキテクチャごとに異なる AMI を使用する場合や、AL2 から AL2023 への移行期間において役立ちます。

> **Note**
>
> AWS は、 から Amazon EKS AL2-optimizedおよび AL2-accelerated AMIs のサポートを終了します11/26/25。サポート11/26/25end-of-supportを過ぎても、Amazon EKS コンピューティング環境で AWS Batch Amazon EKS 最適化 Amazon Linux 2 AMIs を引き続き使用できますが、これらのコンピューティング環境は から新しいソフトウェア更新、セキュリティパッチ、またはバグ修正を受け取りません AWS。AMI が混在する環境は移行期間中に役立つため、既存の AL2 ベースのワークロードとの互換性を維持しながら、ワークロードを AL2023 に段階的に移行できます。

両方の AMI タイプを使用した設定例:

```
{
  "computeResources": {
    "launchTemplate": {
      "launchTemplateId": "TemplateId",
      "version": "1",
      "userdataType": "EKS_BOOTSTRAP_SH",
      "overrides": [
        {
          "instanceType": "c5.large",
          "imageId": "ami-al2-custom",
          "userdataType": "EKS_BOOTSTRAP_SH"
        },
        {
          "instanceType": "c6a.large",
          "imageId": "ami-al2023-custom",
          "userdataType": "EKS_NODEADM"
        }
      ]
```

```
    },
    "instanceTypes": ["c5.large", "c6a.large"]
  }
}
```

サポートされる Kubernetes バージョン

AWS Batch Amazon EKS の は現在、次のKubernetesバージョンをサポートしています。

- 1.34
- 1.33
- 1.32
- 1.31
- 1.30
- 1.29

CreateComputeEnvironment API オペレーションまたは API オペレーション UpdateComputeEnvironment を使用してコンピュート環境を作成または更新すると、次のようなエラーメッセージが表示されることがあります。この問題は、Kubernetes でサポートされていないバージョンを EC2Configuration で指定した場合に発生します。

```
At least one imageKubernetesVersion in EC2Configuration is not supported.
```

この問題を解決するには、コンピュート環境を削除し、サポートされている Kubernetes バージョンで再作成してください。

Amazon EKS クラスターでマイナーバージョンアップグレードを実行できます。例えば、イナーバージョンがサポートされていない場合でも、クラスターを 1.xx から 1.yy マにアップグレードできます。

ただし、INVALID メジャーバージョンを更新すると、コンピュート環境のステータスがに変わることがあります。例えば、1.xx から 2.yy へのメジャーバージョンアップグレードを実行する場合などです。メジャーバージョンが でサポートされていない場合は AWS Batch、次のようなエラーメッセージが表示されます。

```
reason=CLIENT_ERROR - ... EKS Cluster version [2.yy] is unsupported
```

コンピューティング環境の Kubernetes バージョンを更新する

を使用すると AWS Batch、Amazon EKS クラスターのアップグレードをサポートするようにコンピューティング環境Kubernetesのバージョンを更新できます。コンピューティング環境Kubernetesのバージョンは、 がジョブを実行するために AWS Batch 起動するKubernetesノードの Amazon EKS AMI バージョンです。Amazon EKS ノードの Kubernetes バージョンのアップグレードは、Amazon EKS クラスターのコントロールプレーンのバージョンを更新する前後どちらでも実行できます。コントロールプレーンをアップグレードした後に、ノードを更新することをお勧めします。詳細については、「Amazon EKS ユーザーガイド」の「Amazon EKS クラスターの Kubernetes バージョンを更新」を参照してください。

コンピュート環境の Kubernetes バージョンをアップグレードするには、UpdateComputeEnvironment API オペレーションを使用します。

```
$ aws batch update-compute-environment \
    --compute-environment <compute-environment-name> \
    --compute-resources \
      'ec2Configuration=[{imageType=EKS_AL2,imageKubernetesVersion=1.32}]'
```

Kubernetes ノードの責任分担

コンピュート環境のメンテナンスは共同責任です。

- AWS Batch ノード、ラベル、テイント、名前空間、起動テンプレート、または自動スケーリンググループを変更または削除しないでください。 AWS Batch マネージドノードにテイントを追加しないでください。これらの変更を行うと、コンピュート環境がサポートされなくなり、アイドル状態のインスタンスなどの障害が発生します。
- ポッドを AWS Batch マネージドノードにターゲットにしないでください。ポッドを管理対象ノードにすると、スケーリングが中断したり、ジョブキューが停止したりします。セルフマネージド型ノードまたはマネージド型ノードグループ AWS Batch で を使用しないワークロードを実行します。詳細については、「Amazon EKS ユーザーガイド」の「マネージドノードグループ」を参照してください。
- AWS Batch マネージドノードで実行する DaemonSet をターゲットにできます。詳細については、「AWS Batch マネージドノードDaemonSetで を実行する」を参照してください。

AWS Batch はコンピューティング環境 AMIs を自動的に更新しません。更新するのはあなたの責任です。モジュールを最新バージョンにアップグレードするには、次のコマンドを実行します。

```
$ aws batch update-compute-environment \
    --compute-environment <compute-environment-name> \
    --compute-resources 'updateToLatestImageVersion=true'
```

AWS Batch はKubernetesバージョンを自動的にアップグレードしません。次のコマンドを実行して、コンピュータ環境Kubernetesのバージョンを *1.32* に更新します。

```
$ aws batch update-compute-environment \
    --compute-environment <compute-environment-name> \
    --compute-resources \
      'ec2Configuration=[{imageType=EKS_AL2,imageKubernetesVersion=1.32}]'
```

Kubernetes より新しいバージョンの AMI に更新する場合、更新時にジョブを終了するかどうか (`terminateJobsOnUpdate`) と、実行中のジョブが終了しない場合にインスタンスが置き換えられるまでの待機時間 (`jobExecutionTimeoutMinutes`.) を指定できます。詳細については、AWS Batch でコンピューティング環境を更新する および UpdateComputeEnvironment API オペレーションで設定されているインフラストラクチャ更新ポリシー (UpdatePolicy) を参照してください。

AWS Batch マネージドノードDaemonSetで を実行する

AWS Batch AWS Batch はマネージドKubernetesノードにテイントを設定します。次の を使用して、 AWS Batch マネージドノードで を実行するDaemonSetように をターゲットにできますtolerations。

```
tolerations:
  - key: "batch.amazonaws.com/batch-node"
    operator: "Exists"
```

そのためのもう 1 つの方法は、次の `tolerations` のように実行することです。

```
tolerations:
  - key: "batch.amazonaws.com/batch-node"
    operator: "Exists"
    effect: "NoSchedule"
  - key: "batch.amazonaws.com/batch-node"
    operator: "Exists"
    effect: "NoExecute"
```

Amazon EKS 起動テンプレートをカスタマイズする

AWS Batch Amazon EKS の では、起動テンプレートがサポートされています。起動テンプレートでできることには制約があります。

Important

- EKS AL2 AMIs、 は AWS Batch を実行します`/etc/eks/bootstrap.sh`。`/etc/eks/bootstrap.sh` を起動テンプレートや cloud-init user-data スクリプトでは実行しないでください。bootstrap.sh には `--kubelet-extra-args` パラメータ以外にもパラメータを追加できます。これを行うには、`AWS_BATCH_KUBELET_EXTRA_ARGS` `/etc/aws-batch/batch.config` ファイルに変数を設定します。詳細は以下の例を参照してください。
- EKS AL2023 の場合、 は EKS の NodeConfigSpec AWS Batch を使用してインスタンスを EKS クラスターに参加させます。 AWS Batch は EKS クラスターの NodeConfigSpec に ClusterDetails を入力するため、指定する必要はありません。

Note

は値を上書きするため AWS Batch 、起動テンプレートで以下のNodeConfigSpec設定は設定しないことをお勧めします。詳細については、「Kubernetes ノードの責任分担」を参照してください。

- `Taints`
- `Cluster Name`
- `apiServerEndpoint`
- `certificatAuthority`
- `CIDR`
- プレフィックス `batch.amazonaws.com/` が付いたラベルを作成しないでください

> **Note**
>
> CreateComputeEnvironment を呼び出した後に起動テンプレートが変更された場合は、UpdateComputeEnvironment を呼び出して代替用の起動テンプレートのバージョンを評価する必要があります。

トピック

kubelet 追加因数を追加する

AWS Batch は、 kubelet コマンドへの引数の追加をサポートしています。サポートされるパラメータのリストについては、「Kubernetes ドキュメント」の「kubelet」 を参照してください。EKS AL2 AMI の次の例では、*--node-labels mylabel=helloworld* が kubelet コマンドラインに追加されています。

```
MIME-Version: 1.0
      Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

      --==MYBOUNDARY==
      Content-Type: text/x-shellscript; charset="us-ascii"

      #!/bin/bash
      mkdir -p /etc/aws-batch

      echo AWS_BATCH_KUBELET_EXTRA_ARGS=\"--node-labels mylabel=helloworld\" >> /etc/
aws-batch/batch.config

      --==MYBOUNDARY==--
```

EKS AL2023 AMI の場合、ファイル形式は YAML です。サポートされるパラメータのリストについては、「Kubernetes ドキュメント」の「NodeConfigSpec」 を参照してください。EKS AL2023 AMI の次の例では、*--node-labels mylabel=helloworld* が kubelet コマンドラインに追加されています。

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: application/node.eks.aws

apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  kubelet:
    flags:
    - --node-labels=mylabel=helloworld

--==MYBOUNDARY==--
```

コンテナランタイムを設定する

環境変数を使用して AWS Batch CONTAINER_RUNTIME、マネージドノードでコンテナランタイムを設定できます。次の例では、bootstrap.sh コンテナランタイムを実行時に containerd を設定しています。詳細については、「Kubernetes ドキュメント」の「 containerd」を参照してください。

最適化された EKS_AL2023 または EKS_AL2023_NVIDIA AMI を使用している場合は、containerd のみがサポートされているため、コンテナランタイムを指定する必要はありません。

> **Note**
>
> CONTAINER_RUNTIME 環境変数は bootstrap.sh の --container-runtime オプションと同等です。詳細については、「Kubernetes ドキュメント」の「 Options」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
      mkdir -p /etc/aws-batch
```

```
      echo CONTAINER_RUNTIME=containerd >> /etc/aws-batch/batch.config

      --==MYBOUNDARY==--
```

Amazon EFS ボリュームをマウントする

起動テンプレートを使用してボリュームをノードにマウントできます。次の例では、cloud-config、packages および runcmd の設定が使用されています。詳細については、「cloud-init ドキュメント」の「クラウド設定例」を参照してください。

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/cloud-config; charset="us-ascii"

packages:
- amazon-efs-utils

runcmd:
- file_system_id_01=fs-abcdef123
- efs_directory=/mnt/efs

- mkdir -p ${efs_directory}
- echo "${file_system_id_01}:/ ${efs_directory} efs _netdev,noresvport,tls,iam 0 0"
 >> /etc/fstab
- mount -t efs -o tls ${file_system_id_01}:/ ${efs_directory}

--==MYBOUNDARY==--
```

このボリュームをジョブで使用するには、RegisterJobDefinition の EKSProperties パラメーターにボリュームを追加する必要があります。次の例は、仕事定義の大部分です。

```
{
    "jobDefinitionName": "MyJobOnEks_EFS",
    "type": "container",
    "eksProperties": {
        "podProperties": {
            "containers": [
                {
                    "image": "public.ecr.aws/amazonlinux/amazonlinux:2",
                    "command": ["ls", "-la", "/efs"],
```

```
                    "resources": {
                        "limits": {
                            "cpu": "1",
                            "memory": "1024Mi"
                        }
                    },
                    "volumeMounts": [
                        {
                            "name": "efs-volume",
                            "mountPath": "/efs"
                        }
                    ]
                }
            ],
            "volumes": [
                {
                    "name": "efs-volume",
                    "hostPath": {
                        "path": "/mnt/efs"
                    }
                }
            ]
        }
    }
}
```

ノードでは、Amazon EFS ボリュームが /mnt/efs ディレクトリにマウントされます。Amazon EKS ジョブのコンテナでは、ボリュームは /efs ディレクトリにマウントされます。

IPv6 サポート

AWS Batch は、IPv6 アドレスを持つ Amazon EKS クラスターをサポートします。 AWS Batch サポートにカスタマイズは必要ありません。ただし、始める前に、「Amazon EKS ユーザーガイド」の「ポッドとサービスへの IPv6 アドレスの割り当て」で説明されている考慮事項と条件を確認することをお勧めします。

EKS AL2 から EKS AL2023 へアップグレードする方法

Amazon EKS 最適化 AMI は、Amazon Linux 2 (AL2) および Amazon Linux 2023 (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、Linux ベースのオペレーティングシス

テムです。AL2 および AL2023 間の相違点の詳細については、「Amazon EKS ユーザーガイド」の「Amazon Linux 2 から Amazon Linux 2023 にアップグレードする」を参照してください。

> **Important**
>
> AWS は、 から Amazon EKS AL2-optimizedおよび AL2-accelerated AMIs のサポートを終了します11/26/25。最適なパフォーマンスとセキュリティを維持するために、11/26/25 より前に AWS Batch Amazon EKS コンピューティング環境を Amazon Linux 2023 に移行することをお勧めします。11/26/25 end-of-supportを過ぎても、Amazon EKS コンピューティング環境で AWS Batchが提供する Amazon EKS 最適化 Amazon Linux 2 AMIs を引き続き使用できますが、これらのコンピューティング環境は から新しいソフトウェア更新、セキュリティパッチ、またはバグ修正を受け取りません AWS。サポート終了後に Amazon EKS 最適化 Amazon Linux 2 AMI 上のこうしたコンピューティング環境を維持することはお客様の責任となります。

コンピューティング環境の設定方法に応じて、AL2 から AL2023 への以下のアップグレードパスのいずれかを使用できます。

Ec2Configuration.ImageType を使用したアップグレード

- 起動テンプレートまたは起動テンプレートの上書きを使用していない場合は、Ec2Configuration.ImageType を `EKS_AL2023`または に変更し`EKS_AL2023_NVIDIA`、UpdateComputeEnvironment を実行します。
- Ec2Configuration.ImageIdOverride を指定する場合、Ec2Configuration.ImageType は Ec2Configuration.ImageIdOverride で指定された AMI タイプと一致する必要があります。

  `ImageIdOverride` と `ImageType` が一致しない場合、ノードはクラスターに参加しません。

起動テンプレートを使用したアップグレード

- 起動テンプレートまたは起動テンプレートの上書きでkubelet追加の引数が定義されている場合は、新しい`kubelet`追加の引数形式に更新する必要があります。

  `kubelet` の追加の引数の形式が一致しない場合、追加の引数は適用されません。
- AL2023 AMI の場合、サポートされているコンテナランタイムは containerd のみです。起動テンプレートの `EKS_AL2023` のコンテナランタイムを指定する必要はありません。

でカスタマイズされたコンテナランタイムを指定することはできませんEKS_AL2023。

- EKS_AL2023 に基づいて AMI を指定する起動テンプレートまたは起動テンプレートオーバーライドを使用する場合は、userdataType を EKS_NODEADM に設定する必要があります。

  userdataType と AMI が一致しない場合、ノードはクラスターに参加しません。

のサービス環境 AWS Batch

サービス環境では、 AWS Batch を SageMaker AI と統合できます。サービス環境には、 が SageMaker Training ジョブを送信および管理 AWS Batch するために必要な SageMaker AI 固有の設定パラメータが含まれています。同時に、 AWS Batchのキューイング、スケジューリング、優先度管理機能も提供します。

サービス環境では、データサイエンティストと ML エンジニアは、優先順位を付けて SageMaker トレーニングジョブをサービスジョブキューに送信できます。この統合により、ML ワークロードを手動で調整する必要がなくなり、偶発的な過剰支出を防ぎ、組織の機械学習ワークフロー全体でリソース使用率が向上します。

トピック

AWS Batchにおけるサービス環境とは

サービス環境は、SageMaker AI AWS Batch との統合に必要な設定パラメータを含む AWS Batch リソースです。サービス環境では AWS Batch 、 が SageMaker トレーニングジョブを送信および管理しながら、 AWS Batchのキューイング、スケジューリング、優先度管理機能を提供できます。

サービス環境は、データサイエンスチームが機械学習ワークロードを管理する際に直面する一般的な課題に対処します。多くの場合、組織はトレーニングモデルで使用できるインスタンスの数を制限して偶発的な過剰支出を防止し、予算の制約に対応し、リザーブドインスタンスでコストを削減し、ワークロードにおいて特定のインスタンスタイプを使用します。しかし、データサイエンティストは、割り当てられたインスタンスで実行できる数よりも多くのワークロードを同時に実行したい場合もあり、どのワークロードをいつ実行するかを手動で調整する必要があります。

この調整の課題は、データサイエンティストの少ないチームから大規模な運用に至るまで、あらゆる規模の組織に影響を与えます。組織が成長するにつれて、複雑さが増し、ワークロードの調整管理にさらに時間がかかるようになり、インフラストラクチャ管理者の関与が必要になることがよくありま

す。これらの手動作業により、時間が浪費され、インスタンスの効率が低下し、顧客には実際のコストが発生します。

サービス環境では、データサイエンティストと ML エンジニアは設定可能なキューに優先順位を付けて SageMaker トレーニングジョブを送信できるため、ワークロードはリソースが利用可能になったら介入せずともすぐ自動的に実行されます。この統合は、 AWS Batchの広範なキューイングおよびスケジューリング機能を活用して、お客様が組織の目標に合わせてキューイングおよびスケジューリングポリシーをカスタマイズできるようにします。

サービス環境と他の AWS Batch コンポーネントとの連携方法

サービス環境は他の AWS Batch コンポーネントと統合され、SageMaker Training ジョブのキューイングを有効にします。

- ジョブキュー - サービス環境はジョブキューに関連付けられ、キューが SageMaker トレーニングジョブのサービスジョブを処理できるようにします。
- サービスジョブ - サービス環境に関連付けられたキューにサービスジョブを送信すると、 は環境の設定 AWS Batch を使用して対応する SageMaker トレーニングジョブを送信します。
- スケジューリングポリシー - サービス環境は AWS Batch スケジューリングポリシーと連携して、SageMaker トレーニングジョブの実行順序を優先および管理します。

この統合により、SageMaker トレーニングジョブの完全な機能と柔軟性を維持しながら、 AWS Batch成熟したキューイング機能とスケジューリング機能を活用できます。

サービス環境のベストプラクティス

サービス環境は、SageMaker トレーニングジョブを大規模に管理するための機能を提供します。これらのベストプラクティスに従うことで、機械学習ワークフローに影響を与えかねない一般的な設定の問題を回避しながら、コスト、パフォーマンス、運用効率を最適化できます。

サービス環境の容量を計画するときは、SageMaker トレーニングジョブのキューイングに適用される特定のクォータと制限を考慮してください。各サービス環境には、インスタンス数で表される最大容量制限があり、これによって同時に実行できる SageMaker トレーニングジョブの数が直接制御されます。これらの制限を理解することで、リソースの競合を防ぎ、予測可能なジョブ実行時間を確保できます。

最適なサービス環境のパフォーマンスは、SageMaker トレーニングジョブスケジューリングならではの特性を理解することにかかっています。従来のコンテナ化されたジョブとは異なり、サービス

ジョブは SCHEDULED 状態に移行し、SageMaker AI は必要なトレーニングインスタンスを取得してプロビジョニングします。つまり、ジョブの開始時間は、インスタンスの可用性とリージョンの容量によって大きく異なる可能性があります。

> **Important**
>
> サービス環境には、SageMaker トレーニングワークロードのスケーリング能力に影響を与える可能性のある特定のクォータがあります。アカウントごとに最大 50 のサービス環境を作成できます。各ジョブキューは、関連付けられた 1 つのサービス環境のみをサポートします。さらに、個々のジョブのサービスリクエストペイロードは 10 KiB に制限され、SubmitServiceJob API はアカウントごとに 1 秒あたり 5 トランザクションに制限されます。キャパシティプランニング中にこれらの制限を理解することで、予期しないスケーリングの制約を回避できます。

サービス環境を効果的にモニタリングするには、 AWS Batch と SageMaker AI サービスメトリクスの両方に注意する必要があります。ジョブ状態遷移は、システムパフォーマンス、特にキャパシティの可用性パターンを示す SCHEDULED 状態で費やされた時間に関する貴重なインサイトを提供します。サービス環境は、コンピューティング環境と同様に独自のライフサイクル状態を維持し、運用上の健全性を監視する必要がある CREATING、VALID、INVALID、DELETING 状態に遷移します。通常、モニタリングプラクティスが成熟している組織は、キューの深さ、ジョブ完了率、インスタンス使用率パターンを追跡し、サービス環境の設定を時間をかけて最適化します。

のサービス環境の状態とライフサイクル AWS Batch

サービス環境は、現在の運用ステータスと SageMaker トレーニングジョブを処理する準備状況を示すライフサイクル状態を維持します。これらの状態を理解することで、サービス環境の健全性をモニタリングし、設定の問題をトラブルシューティングし、信頼性の高いジョブ処理を確実に行えます。状態管理システムは、SageMaker トレーニングジョブ統合ならではの要件に対応しつつ、コンピューティング環境からの確立されたパターンに従います。

サービス環境の状態は、設定の検証、リソースの可用性、運用上のヘルスチェック AWS Batch に基づいて、 によって自動的に管理されます。物理インフラストラクチャを管理するコンピューティング環境とは異なり、サービス環境は設定の検証と SageMaker AI サービスとの統合準備に重点を置いています。状態遷移は、サービス環境が SageMaker トレーニングジョブを正常に送信および管理できるかどうかを可視化します。

サービス環境の状態の定義

サービス環境は、現在の運用ステータスと SageMaker トレーニングジョブを処理する準備状況を示す 4 つの可能な状態のいずれかになります。個々の状態は、最初の作成から運用準備、最終的な削除まで、サービス環境ライフサイクルの具体的なフェーズを表します。各状態とその意味を次表に示します。

State	説明
CREATING	初期状態は、サービス環境を作成する状態です。この状態で、 は設定パラメータ AWS Batch を検証し、SageMaker AI サービスとの統合を確立します。サービス環境はジョブを処理できず、それに関連付けられたジョブキューはサービスジョブの送信を受け入れません。作成プロセスは通常、適切に設定されたサービス環境では数秒以内に完了します。
VALID	運用状態は、サービス環境がすべての設定検証チェックに合格し、SageMaker トレーニングジョブを処理する準備が整っていることを示す状態です。この状態は、サービス環境設定が正しく、必要なすべてのアクセス許可が設定されており、ユーザーに代わって SageMaker AI にジョブを正常に送信 AWS Batch できることを示します。サービス環境では、運用ライフサイクルのほとんどがこの状態になります。
INVALID	サービス環境において、SageMaker トレーニングジョブの処理を妨げる設定またはアクセス許可の問題が発生したことを示す状態。無効なサービス環境に関連付けられたジョブキューは、根本的な問題が解決されるまで、新しいサービスジョブの送信を処理できません。
DELETING	サービス環境の削除をリクエストしたときに発生する状態。この状態の間、 AWS Batch はアクティブな SageMaker トレーニングジョブが

State	説明
	環境に関連付けられていないことを確認し、必要なクリーンアップオペレーションを実行します。この状態のサービス環境は新しいジョブ送信を処理できず、関連するすべてのリソースが適切にクリーンアップされると削除プロセスが完了します。

サービス環境の状態の遷移

サービス環境の状態の遷移は、設定の変更、検証結果、運用状態のモニタリングに基づいて自動的に行われます。この AWS Batch サービスは、サービス環境のヘルスを継続的にモニタリングし、それに応じて状態を更新します。これらの遷移について理解することで、設定変更がいつ有効になるか、および無効な状態を引き起こす問題を解決する方法を予測できます。

作成と検証が成功すると、サービス環境は `CREATING` から `VALID` に遷移します。この遷移により、すべての設定パラメータが正しく、必要な IAM アクセス許可が正しく設定され、サービス環境が SageMaker AI サービスと正常に統合できることを確認します。`VALID` 状態になると、関連付けられたジョブキューはサービスジョブの送信の処理を開始できます。

サービス環境は、設定検証が失敗したとき、または依存関係が使用できなくなったときに、`VALID` から `INVALID` に遷移します。これは、IAM ロールの変更、クォータに違反する容量制限の変更、またはサービス環境の機能に影響する外部リソースの変更が原因で発生する可能性があります。[ステータス理由] フィールドには、無効な状態の原因に関する具体的な詳細が表示されます。

根本的な問題が解決されると、サービス環境は `INVALID` から `VALID` に戻る形で遷移することができます。これには、IAM アクセス許可の更新、キャパシティ設定の修正、必要な AWS リソースへのアクセスの復元が含まれる場合があります。遷移は通常、設定の問題が対処されたことを AWS Batch が検出すると自動的に行われます。

AWS Batchのサービス環境を作成する

で SageMaker トレーニングジョブを実行する前に AWS Batch、サービス環境を作成する必要があります。が SageMaker AI サービスと統合 AWS Batch するために必要な設定パラメータを含むサービス環境を作成し、ユーザーに代わって SageMaker トレーニングジョブを送信できます。

前提条件

サービス環境を作成する前に、以下があることを確認してください。

- IAM アクセス許可 – サービス環境を作成および管理するためのアクセス許可。詳細については、「AWS Batch IAM ポリシー、ロール、アクセス許可」を参照してください。

Create a service environment (AWS Console)

AWS Batch コンソールを使用して、ウェブインターフェイスを介してサービス環境を作成します。

サービス環境を作成するには

1. https://console.aws.amazon.com/batch/ で AWS Batch コンソールを開きます。
2. ナビゲーションペインで [Environments (環境)] を選択します。
3. [環境の作成] を選択し、[サービス環境] を選択します。
4. [サービス環境設定] で [SageMaker AI] を選択します。
5. [名前] に、サービス環境の一意の名前を入力します。有効な文字は、a-z、A-Z、0-9、ハイフン (-)、アンダースコア (_) です。
6. [最大インスタンス数] には、同時トレーニングインスタンスの最大数を入力します。
7. (オプション) [タグを追加] を選択し、キーと値のペアを入力してタグを追加します。
8. [次へ] を選択します。
9. 新しいサービス環境の詳細を確認し、[サービス環境の作成] を選択します。

Create a service environment (AWS CLI)

`create-service-environment` コマンドを使用して、 CLI AWS でサービス環境を作成します。

サービス環境を作成するには

1. 基本的な必須パラメータを使用してサービス環境を作成します。

```
aws batch create-service-environment \
    --service-environment-name my-sagemaker-service-env \
```

```
    --service-environment-type SAGEMAKER_TRAINING \
    --capacity-limits capacityUnit=NUM_INSTANCES,maxCapacity=10
```

2. (オプション) タグを使用してサービス環境を作成します。

```
aws batch create-service-environment \
    --service-environment-name my-sagemaker-service-env \
    --service-environment-type SAGEMAKER_TRAINING \
    --capacity-limits capacityUnit=NUM_INSTANCES,maxCapacity=10 \
    --tags team=data-science,project=ml-training
```

3. サービス環境が正常に作成されたことを確認します。

```
aws batch describe-service-environments \
    --service-environment my-sagemaker-service-env
```

サービス環境は、CREATING 状態の環境リストに表示されます。作成が正常に完了すると、状態が VALID に変わり、サービス環境がジョブの処理を開始できるようにサービスジョブキューを追加する準備が整います。

でサービス環境を更新する AWS Batch

サービス環境を更新して、容量制限の変更、運用状態の変更、リソースタグの更新を行うことができます。サービス環境の更新により、SageMaker トレーニングのワークロード要件の変化に応じて容量を調整したり、環境を再作成せずに運用設定を変更したりできます。サービス環境を更新する前に、どのパラメータを変更できるか、その変更が実行中のジョブにどのような影響を与えるかを理解してください。

サービス環境の容量制限、状態、またはタグを変更できます。

Update a service environment (AWS Console)

AWS Batch コンソールを使用して、ウェブインターフェイスを介してサービス環境を更新します。

サービス環境を更新するには

1. https://console.aws.amazon.com/batch/ で AWS Batch コンソールを開きます。
2. ナビゲーションペインで [Environments (環境)] を選択します。

3. [サービス環境] タブを選択します。
4. 更新するサービス環境を選択します。
5. [アクション] を選択した後、以下のいずれかを選択します。
   - [状態] - [有効化] または [無効化] を選択して状態を変更します。
   - [容量制限] - [インスタンスの最大数] を変更します。
6. 変更を保存するには、[変更を保存] を選択します。

サービス環境はすぐに更新されます。環境の詳細を確認して、変更が正常に適用されたことを確認します。サービス環境を無効にすると、関連するジョブキューは、再度有効にするまで新しいサービスジョブの送信の処理を停止します。

Update a service environment (AWS CLI)

`update-service-environment` コマンドを使用して、 CLI AWS でサービス環境を変更します。

サービス環境の容量制限を更新するには

1. サービス環境の容量制限を更新します。

```
aws batch update-service-environment \
    --service-environment my-sagemaker-service-env \
    --capacity-limits capacityUnit=NUM_INSTANCES,maxCapacity=20
```

2. 更新が正常に適用されたことを確認します。

```
aws batch describe-service-environments \
    --service-environments my-sagemaker-service-env
```

サービス環境の状態を更新するには

1. 新しいジョブの処理を停止するには、サービス環境を無効にします。

```
aws batch update-service-environment \
    --service-environment my-sagemaker-service-env \
    --state DISABLED
```

2. サービス環境を再度有効にして処理を再開します。

```
aws batch update-service-environment \
    --service-environment my-sagemaker-service-env \
    --state ENABLED
```

サービス環境の更新はすぐに有効になります。新しいジョブを送信する前に、サービス環境の状態をモニタリングして、更新が正常に完了していることを確認します。

AWS Batchでサービス環境を削除する

SageMaker トレーニングジョブで不要になったサービス環境を削除できます。サービス環境を削除すると、設定が削除され、それ以上ジョブが送信されなくなります。サービス環境を削除する前に、アクティブな SageMaker トレーニングジョブが依存していないことおよび、サービス環境にジョブキューが関連付けられていないことを確認してください。

Important

サービス環境の削除は元に戻せません。削除されると、サービス環境またはその設定を復元することはできません。今後同様の機能が必要な場合は、必須設定で新しいサービス環境を作成する必要があります。後で再アクティブ化する必要がある場合は、削除せずにサービス環境を無効にすることを検討してください。

Note

アカウント内のすべてのサービス環境を削除しても、 AWS Batch と SageMaker AI 統合用に作成されたサービスにリンクされたロールは自動的に削除されません。サービスにリンクされたロールは、今後のサービス環境の作成に引き続き使用できます。サービスにリンクされたロールを削除する場合は、アカウントにサービス環境が存在しないことを確認した後、IAM を使用して個別に削除する必要があります。

削除の前提条件

サービス環境を削除する前に、サービスジョブキューの関連付けをすべて解除してからサービス環境を無効にする必要があります。

サービス環境を削除する前に:

- アクティブなジョブの確認 - SageMaker トレーニングジョブがサービス環境で現在実行されていないことを確認します。
- ジョブキューの確認 - サービス環境に関連付けられたジョブキューを特定し、ジョブキューを別のサービス環境に関連付けるか、ジョブキューを無効にして削除します。

ジョブキュー管理: 削除されたサービス環境に関連付けられたジョブキューは引き続き存在できますが、サービスジョブを処理できません。未使用のジョブキューを削除するか、元のサービス環境を削除する前に別のサービス環境に関連付ける必要があります。

Delete a service environment (AWS Console)

AWS Batch コンソールを使用して、ウェブインターフェイスからサービス環境を削除します。

サービス環境を削除するには

1. https://console.aws.amazon.com/batch/ で AWS Batch コンソールを開きます。
2. ナビゲーションペインで [Environments (環境)] を選択します。
3. [サービス環境] タブを選択した後、[サービス環境] を選択します。
4. サービス環境が有効になっている場合は、[アクション] を選択してから [無効化] を選択します。
5. サービス環境が無効になったら、[アクション] を選択し、[削除] を選択します。
6. 確認ダイアログで、[確認] を選択します。

サービス環境は、削除中に `DELETING` 状態を表示します。削除が完了すると、サービス環境は環境リストから消えます。

Delete a service environment (AWS CLI)

`delete-service-environment` コマンドを使用して、 CLI AWS でサービス環境を削除します。

サービス環境を削除するには

1. サービス環境に関連付けられているジョブキューを確認します。

```
aws batch describe-job-queues
```

サービス環境に関連付けられているジョブキューがある場合は、サービス環境からジョブキューの関連付けを解除して別のサービス環境に関連付けるか、ジョブキューを削除することができます。

2. サービス環境を無効にします:

```
aws batch update-service-environment \
    --service-environment my-sagemaker-service-env \
    --state DISABLED
```

3. サービス環境を削除します:

```
aws batch delete-service-environment \
    --service-environment my-sagemaker-service-env
```

4. 削除プロセスをモニタリングします:

```
aws batch describe-service-environments \
    --service-environment my-sagemaker-service-env
```

サービス環境は、削除プロセス中に DELETING 状態に遷移します。削除が完了すると、サービス環境は describe (説明) オペレーションに表示されなくなります。関連付けられたジョブキューは残りますが、別のサービス環境に関連付けられるまでサービスジョブを処理できません。

ジョブキュー

ジョブは、コンピューティング環境で実行するようにスケジュールされるまで、ジョブキューに送信されます。 AWS アカウントには複数のジョブキューを含めることができます。例えば、優先度の高いジョブには Amazon EC2 オンデマンドインスタンスを使用するキューを作成し、優先度の低いジョブには Amazon EC2 スポットインスタンスを使用する別のキューを作成できます。ジョブキューには優先順位があり、この順位に基づいてスケジューラはどのキューのどのジョブを最初に実行するかを判断します。

トピック

- ジョブキューを作成する
- でジョブキューを表示する AWS Batch
- AWS Batch のジョブキューを削除する
- 公平配分スケジューリングポリシー
- リソース認識型スケジューリング

ジョブキューを作成する

AWS Batch でジョブを送信する前に、ジョブキューを作成する必要があります。ジョブキューの作成する場合、キューに 1 つ以上のコンピューティング環境を関連付け、そして各コンピューティング環境に優先順位を割り当てます。

また、ジョブキューに優先順位を設定し、それによりAWS Batch スケジューラーがジョブを配置する順序を決定します。つまり、コンピューティング環境が 1 つ以上のジョブキューに関連付けられている場合、優先度の高いジョブキューから先にスケジュールされます。

トピック

- Amazon EC2 ジョブキューを作成する
- Fargate ジョブキューを作成する
- Amazon EKS ジョブキューを作成する
- AWS Batch で SageMaker トレーニングジョブのキューを作成する
- ジョブキューテンプレート

Amazon EC2 ジョブキューを作成する

Amazon Elastic Compute Cloud (Amazon EC2) のジョブキューを作成するには、次の手順を実行します。

Amazon EC2 ジョブキューを作成するには

1. AWS Batch コンソールを https://console.aws.amazon.com/batch/ で開きます。
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. ナビゲーションペインで キューを選択します。
4. 作成 を選択します。
5. オーケストレーションタイプ には、Amazon Elastic Compute Cloud (Amazon EC2) を選択します。
6. キューの名前 に、一意のジョブキュー名を入力します。ジョブ名には、大文字、小文字の英文字、数字、ハイフン、アンダースコア(_)を含めることができ、最大 128 文字まで使用可能です。
7. 優先度 には、ジョブキューの優先度を示す整数値を入力します。同じコンピューティング環境環境に関連付けられているジョブキュー間では、優先度が高いジョブキューほど先に処理されます。優先度は降順に決定されます。例えば、優先度の値が 10 のジョブキューは、優先度の値が 1 のジョブキューより先にスケジュールされます。
8. (オプション) スケジューリングポリシーの Amazon リソースネーム (ARN) では、既存のスケジューリングポリシーを選択します。
9. 接続されているコンピューティング環境では、リストから 1 つ以上のコンピューティング環境を選択して、ジョブキューに関連付けます。ユーザーは、ジョブキューの配置を試行させたいと考える順序で、コンピューティング環境環境を選択します。ジョブスケジューラーでは、どのコンピューティング環境で特定のジョブを開始するかを、コンピューティング環境の順番で決めます。コンピューティング環境は、ジョブキューに関連付ける前に、`VALID`の状態になっていることが必要です。1 つのジョブキューには、最大 3 つのコンピューティング環境を関連付けることができます。既存のコンピューティング環境環境がない場合は、コンピュート環境の作成 を選択します。

> **Note**
>
> ジョブキューに関連付けられているすべてのコンピューティング環境で、同じプロビジョニングモデルを共有する必要があります。AWS Batch は、単一のジョブキューでのプロビジョニングモデルの混在をサポートしていません。

10. コンピューティング環境環境の順序 では、上矢印と下矢印を選択して希望する順序を設定します。
11. 作成 を選択して終了し、ジョブキューを作成します。

Fargate ジョブキューを作成する

AWS Fargate のジョブキューを作成するには、次のステップを実行します。

Fargate ジョブキューを作成するには

1. AWS Batch コンソールを https://console.aws.amazon.com/batch/ で開きます。
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. ナビゲーションペインで キューを選択します。
4. 作成] を選択します。
5. オーケストレーションタイプ] には Fargate を選択します。
6. キューの名前 に、一意のジョブキュー名を入力します。ジョブ名には、大文字、小文字の英文字、数字、ハイフン、アンダースコア(_)を含めることができ、最大 128 文字まで使用可能です。
7. 優先度 には、ジョブキューの優先度を示す整数値を入力します。同じコンピューティング環境環境に関連付けられているジョブキュー間では、優先度が高いジョブキューほど先に処理されます。優先度は降順に決定されます。例えば、優先度の値が 10 のジョブキューは、優先度の値が 1 のジョブキューより先にスケジュールされます。
8. (オプション) スケジューリングポリシーの Amazon リソースネーム (ARN) では、既存のスケジューリングポリシーを選択します。
9. 接続されているコンピューティング環境では、リストから 1 つ以上のコンピューティング環境を選択して、ジョブキューに関連付けます。ユーザーは、ジョブキューの配置を試行させたいと考える順序で、コンピューティング環境環境を選択します。ジョブスケジューラーでは、どのコンピューティング環境で特定のジョブを開始するかを、コンピューティング環境の順番で決めま

す。コンピューティング環境は、ジョブキューに関連付ける前に、VALIDの状態になっていることが必要です。1 つのジョブキューには、最大3つのコンピューティング環境を関連付けることができます。

> **Note**
>
> ジョブキューに関連付けられているすべてのコンピューティング環境で、同じプロビジョニングモデルを共有する必要があります。AWS Batch は、単一のジョブキューでのプロビジョニングモデルの混在をサポートしていません。

10. コンピューティング環境環境の順序 では、上矢印と下矢印を選択して希望する順序を設定します。
11. 作成 を選択して終了し、ジョブキューを作成します。

Amazon EKS ジョブキューを作成する

Amazon Elastic Kubernetes Service (Amazon EKS) のジョブキューを作成するには、次の手順を実行します。

Amazon EKS ジョブキューを作成するには

1. AWS Batch コンソールを https://console.aws.amazon.com/batch/ で開きます。
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. ナビゲーションペインで キューを選択します。
4. 作成 を選択します。
5. オーケストレーションタイプ では、Amazon Elastic Kubernetes Service (Amazon EKS)] を選択します。
6. キューの名前 に、一意のジョブキュー名を入力します。ジョブ名には、大文字、小文字の英文字、数字、ハイフン、アンダースコア(_)を含めることができ、最大 128 文字まで使用可能です。
7. 優先度 には、ジョブキューの優先度を示す整数値を入力します。同じコンピューティング環境環境に関連付けられているジョブキュー間では、優先度が高いジョブキューほど先に処理されます。優先度は降順に決定されます。例えば、優先度の値が 10 のジョブキューは、優先度の値が 1 のジョブキューより先にスケジュールされます。
8. (オプション) スケジューリングポリシーの Amazon リソースネーム (ARN) では、既存のスケジューリングポリシーを選択します。

9. 接続されているコンピューティング環境では、リストから 1 つ以上のコンピューティング環境を選択して、ジョブキューに関連付けます。ユーザーは、ジョブキューの配置を試行させたいと考える順序で、コンピューティング環境環境を選択します。ジョブスケジューラーでは、どのコンピューティング環境で特定のジョブを開始するかを、コンピューティング環境の順番で決めます。コンピューティング環境は、ジョブキューに関連付ける前に、VALIDの状態になっていることが必要です。1 つのジョブキューには、最大3つのコンピューティング環境を関連付けることができます。

   > **Note**
   >
   > ジョブキューに関連付けられているすべてのコンピューティング環境で、同じプロビジョニングモデルを共有する必要があります。AWS Batch は、単一のジョブキューでのプロビジョニングモデルの混在をサポートしていません。

   > **Note**
   >
   > ジョブキューに関連付けられているすべてのコンピューティング環境で、同じアーキテクチャを共有する必要があります。AWS Batch では、単一のジョブキューでのコンピューティング環境アーキテクチャタイプの混在をサポートしていません。

10. コンピューティング環境環境の順序 では、上矢印と下矢印を選択して希望する順序を設定します。
11. 作成 を選択して終了し、ジョブキューを作成します。

AWS Batch で SageMaker トレーニングジョブのキューを作成する

SageMaker トレーニングジョブのキューは、SageMaker AI サービスと直接統合され、基盤となるコンピューティングインフラストラクチャを管理することなく、サーバーレスジョブスケジューリングを提供します。

前提条件

SageMaker トレーニングジョブのキューを作成する前に、以下があることを確認してください。

- サービス環境 – 容量制限を定義するサービス環境。詳細については、AWS Batchのサービス環境を作成する を参照してください。

- IAM アクセス許可 – AWS Batch ジョブキューとサービス環境を作成および管理するためのアクセス許可。詳細については、AWS Batch IAM ポリシー、ロール、アクセス許可 を参照してください。

Create a SageMaker Training job queue (AWS Batch console)

1. AWS Batch コンソールを https://console.aws.amazon.com/batch/ で開きます。
2. ナビゲーションペインで [ジョブキュー]、[作成] を選択します。
3. [オーケストレーションタイプ] で、[SageMaker トレーニング] を選択します。
4. [ジョブキューの設定] で以下を行います。
   a. [名前] にジョブキューの名前を入力します。
   b. [優先度] には 0～1000 の値を入力します。サービス環境では、優先度の高いジョブキューが優先されます。
   c. (オプション) スケジューリングポリシーの Amazon リソースネーム (ARN) では、既存のスケジューリングポリシーを選択します。
   d. [接続されたサービス環境] では、リストからサービス環境を選択してジョブキューに関連付けます。
5. (オプション) [ジョブ状態の制限] の場合:
   a. [設定ミス] の場合、`SERVICE_ENVIRONMENT_MAX_RESOURCE` を選択し、[最大実行可能時間 (秒)] を入力します。
   b. [容量] の場合、`INSUFFICIENT_INSTANCE_CAPACITY` を選択し、[最大実行可能時間 (秒)] を入力します。
6. [ジョブキューの作成] を選択します。

Create a SageMaker Training job queue (AWS CLI)

`create-job-queue` コマンドを使用して SageMaker トレーニングジョブのキューを作成します。

次の例では、サービス環境を使用する基本的な SageMaker トレーニングジョブのキューを作成します。

```
aws batch create-job-queue \
  --job-queue-name my-sm-training-fifo-jq \
```

```
  --job-queue-type SAGEMAKER_TRAINING \
  --priority 1 \
  --service-environment-order order=1,serviceEnvironment=ExampleServiceEnvironment
```

ExampleServiceEnvironment をサービス環境の名前に置き換えます。

このコマンドにより、以下のような出力が返されます。

```
{
  "jobQueueName": "my-sm-training-fifo-jq",
  "jobQueueArn": "arn:aws:batch:region:account:job-queue/my-sm-training-fifo-jq"
}
```

ジョブキューを作成したら、ジョブキューが正常に作成され、有効な状態であることを確認します。

describe-job-queues コマンドを使用して、ジョブキューの詳細を表示します。

```
aws batch describe-job-queues --job-queues my-sm-training-fifo-jq
```

このコマンドにより、以下のような出力が返されます。

```
{
  "jobQueues": [
    {
      "jobQueueName": "my-sm-training-fifo-jq",
      "jobQueueArn": "arn:aws:batch:region:account:job-queue/my-sm-training-fifo-
jq",
      "state": "ENABLED",
      "status": "VALID",
      "statusReason": "JobQueue Healthy",
      "priority": 1,
      "computeEnvironmentOrder": [],
      "serviceEnvironmentOrder": [
        {
          "order": 1,
          "serviceEnvironment": "arn:aws:batch:region:account:service-
environment/ExampleServiceEnvironment"
        }
      ],
      "jobQueueType": "SAGEMAKER_TRAINING",
```

```
      "tags": {},
      "jobStateTimeLimitActions": []
    }
  ]
}
```

以下を確認してください。

- state は、ENABLED です。
- status は、VALID です。
- statusReason は、JobQueue Healthy です。
- jobQueueType は、SAGEMAKER_TRAINING です。
- serviceEnvironmentOrder がサービス環境を参照する

ジョブキューテンプレート

以下に示すのは、空のジョブ定義テンプレートです。このテンプレートを使用して、ジョブキューを作成できます。その後、このジョブキューをファイルに保存し、AWS CLI --cli-input-json オプションとともに使用できます。これらのパラメータの詳細については、CreateJobQueue (AWS BatchAPI リファレンス) を参照してください。

> **Note**
>
> ジョブキューテンプレートは、以下の AWS CLI コマンドで生成できます。
>
> ```
> $ aws batch create-job-queue --generate-cli-skeleton
> ```

```
{
   "computeEnvironmentOrder": [
      {
         "computeEnvironment": "",
         "order": 0
      }
   ],
   "jobQueueName": "",
   "jobStateTimeLimitActions": [
      {
```

```
         "state": "RUNNABLE",
         "action": "CANCEL",
   "maxTimeSeconds": 0,
         "reason": ""

      }
   ],
   "priority": 0,
   "schedulingPolicyArn": "",
   "state": "ENABLED",
   "tags": {
      "KeyName": ""
   }
}
```

でジョブキューを表示する AWS Batch

ジョブキューを作成してジョブを送信した後は、その進行状況をモニタリングできることが重要です。[ジョブの詳細] ページを使用すると、ジョブキューを確認、管理、モニタリングできます。

ジョブキュー情報を表示する

AWS Batch コンソールから、ナビゲーションペインでジョブキューを選択し、目的のジョブキューを選択して詳細を表示します。このページでは、ジョブキューを確認して管理したり、ジョブキューのスナップショット、ジョブの状態制限、環境の順序、タグ、ジョブキューの JSON コードなど、キューのオペレーションに関する追加情報を確認したりできます。

ジョブキューの詳細

このセクションでは、ジョブキューの概要とメンテナンスのオプションを確認できます。このセクションには Amazon リソースネーム (ARN) があります。

を通じてこの情報を検索するには AWS Command Line Interface、 `DescribeJobQueues`オペレーションをジョブキュー名または対応する ARN とともに使用します。

アクティブな共有

公平配分スケジューリングを使用するジョブキューの場合、 AWS Batch はさまざまな共有識別子が容量をどのように消費するかを可視化します。この情報は、リソースの分散を理解し、調整が必要な可能性のある共有を特定するのに役立ちます。

> **Note**
>
> アクティブ共有タブは、ジョブキューのスケジューリングアルゴリズムが公平共有である場合にのみ存在します。

上位 20 個のアクティブな共有セクションには、スケジュールされたジョブ、開始ジョブ、実行中のジョブを持つ共有識別子が表示されます。このビューには以下が含まれます。

- 共有識別子名 - 共有の一意の識別子。

  共有識別子は、公平配分スケジューリングのためにジョブをグループ化するラベルです。同じ共有識別子を持つジョブを送信すると、 はリソース割り当ての目的で同じワークロードの一部として AWS Batch ジョブを処理します。共有識別子は、さまざまなチーム、プロジェクト、ワークロードタイプ間でコンピューティング容量を公平に分散するのに役立ちます。詳細については、「配分識別子を使用してワークロードを識別する」を参照してください。
- キャパシティ使用率 - ジョブが使用するように設定されたリソースの量。これは`instances`、環境vCPUに応じて 、`cpu`、または で測定されます。
- ジョブの表示アクション - その共有のすべてのジョブを表示するリンク。

この情報は、リスト形式とグラフ形式の両方で表示できます。

- リストビュー - 正確な容量番号を含む表形式表示
- グラフビュー - 相対的な使用率を示すビジュアル棒グラフ

ジョブキュースナップショット

このセクションでは、キューにある最初の 100 個のRUNNABLEジョブの静的リストを示します。検索フィールドを使用して結果セクションの任意の列の情報を検索すると、リストを絞り込むことができます。スナップショットの結果エリアのジョブは、ジョブキューの実行戦略に従って並べ替えられます。先入れ先出し (FIFO) ジョブキューの場合、ジョブの順序は送信時間に従います。公平配分スケジューリングのジョブキューの場合、ジョブの順序はジョブの優先度と配分の使用状況に基づきます。キャパシティ必須フィールドには、ジョブが使用するように設定されたリソースの量が表示されます。

結果はジョブキューのスナップショットであり、結果リストが自動的に更新されることはありません。リストを更新するには、セクションの上部にある [更新] を選択します。ジョブ名のハイパーリ

ンクを選択して [ジョブの詳細] に移動すると、ジョブのステータスやその他の関連情報が表示されます。

を通じてこの情報を検索するには AWS CLI、 GetJobQueueSnapshotオペレーションをジョブキュー名または対応する ARN とともに使用します。

```
aws batch get-job-queue-snapshot --job-queue my-sm-training-fifo-jq
```

ジョブの状態制限

このタブを使用すると、ジョブがキャンセルされるまでに RUNNABLE 状態を維持できる時間に関する設定情報を確認できます。

を通じてこの情報を検索するには AWS CLI、 DescribeJobQueuesオペレーションをジョブキュー名または対応する ARN とともに使用します。

環境の順序

ジョブキューが複数の環境で実行されている場合、このタブにその順序と概要を確認できます。

を通じてこの情報を検索するには AWS CLI、 DescribeJobQueuesオペレーションをジョブキュー名または対応する ARN とともに使用します。

タグ

このタブを使用して、このジョブキューに関連付けられているタグを確認および管理できます。

JSON

このタブを使用して、このジョブキューに関連付けられている JSON コードをコピーできます。その後、JSON を AWS CloudFormation テンプレートと AWS CLI スクリプトに再利用できます。

AWS Batch のジョブキューを削除する

ジョブキューが不要になった場合は、ジョブキューを無効化および削除できます。

Delete a job queue (AWS Batch console)

1. AWS Batch コンソールを https://console.aws.amazon.com/batch/ で 開きます。
2. ナビゲーションペインで [ジョブキュー] を選択し、ジョブキューを選択します。
3. [アクション]、[無効化] の順に選択します。

4. ジョブキューの状態が [無効] になったら [アクション]、[削除] の順に選択します。
5. モーダルウィンドウで、[ジョブキューの削除] を選択します。

Delete a job queue (AWS CLI)

1. 新しいジョブの送信を防ぐために、ジョブキューを無効にします。

```
aws batch update-job-queue \
  --job-queue my-sm-training-fifo-jq \
  --state DISABLED
```

2. 実行中のジョブが完了するまで待ってからジョブキューを削除します。

```
aws batch delete-job-queue \
  --job-queue my-sm-training-fifo-jq
```

公平配分スケジューリングポリシー

AWS Batch スケジューラは、ジョブキューに送信されたジョブを実行するタイミング、場所、方法を判断します。ジョブキューを作成するときにスケジューリングポリシーを指定しない場合、AWS Batch ジョブスケジューラはデフォルトで FIFO 戦略を使用します。FIFO 戦略では、重要なジョブが以前に送信されたジョブの後ろでスタックする可能性があります。別のスケジューリングポリシーを指定することで、特定のニーズに応じてコンピューティングリソースを割り当てることができます。

> **Note**
>
> ジョブの実行順序を保証する必要がある場合は、SubmitJob の dependsOn パラメータを使用し、各ジョブの依存関係を指定します。

スケジューリングポリシーを作成してジョブキューに適用すると、公平分配スケジューリングが有効になります。ジョブキューにスケジューリングポリシーがある場合、スケジューリングポリシーによってジョブの実行順序が決まります。詳細については、「公平配分スケジューリングポリシーを使用して配分識別子を割り当てる」を参照してください。

トピック

配分識別子を使用してワークロードを識別する

配分識別子を使用してジョブにタグを付け、ユーザーとワークロードを区別できます。AWS Batch スケジューラは、*(T * weightFactor)* の式を使用して各公平配分識別子の使用状況を追跡します。ここで、*T* は時間の経過に伴う vCPU の使用率です。スケジューラは、配分識別子から使用率が最も低いジョブを選択します。配分識別子は、オーバーライドせずに使用できます。

> **Note**
>
> 配分識別子はジョブキュー内で一意であり、ジョブキュー全体で集計されることはありません。

公平配分のスケジューリングの優先順位を設定して、配分識別子でのジョブの実行順序を設定できます。スケジューリング優先度の高いジョブが最初にスケジュールされます。公平配分のスケジューリングポリシーを指定しない場合、ジョブキューに送信されるすべてのジョブは FIFO 順にスケジュールされます。ジョブの送信時には、配分識別子や公平配分スケジューリング優先度を指定することはできません。

> **Note**
>
> アタッチされたコンピューティングリソースは、明示的にオーバーライドされない限り、すべての配分識別子に均等に割り当てられます。

公平配分スケジューリングポリシーを使用して配分識別子を割り当てる

スケジューリングポリシーを使用して、ジョブキュー内のコンピューティングリソースをユーザーやワークロード間でどのように配分するかを設定できます。公平配分スケジューリングポリシーを使用すると、ワークロードやユーザーに異なる配分識別子を割り当てることができます。AWS Batch は、各配分識別子に、一定期間に利用可能な総リソースに対する割合を割り当てます。

公平配分比率は、shareDecaySeconds および shareDistribution の値を使用して計算されます。ポリシーに減衰時間を割り当てることで、公平配分分析にかける時間を増やすことができます。減衰時間が長いほど、時間により多くの重みが与えられ、定義された重みよりも少なくなります。コンピューティング予約を指定することで、アクティブでない配分識別子に対してコンピューティングリソースを確保できます。詳細については、「SchedulingPolicyDetail」を参照してください。

公平配分スケジューリングを使用してジョブをスケジュールする

公平配分スケジューリングは、ジョブのスケジュール設定に役立つ一連の制御を提供します。

> **Note**
>
> スケジューリングポリシーのパラメータの詳細については、「SchedulingPolicyDetail」を参照してください。

- 配分の減衰秒数 - AWS Batch スケジューラが各配分識別子の公平配分の割合を計算するために使用する時間 (秒単位)。値がゼロ (0) の場合、現在の使用量のみが測定されることを示します。減衰時間が長いほど、時間にさらに多くの重みが与えられます。

  > **Note**
  >
  > 減衰時間は次のように計算されます。*shareDecaySeconds + OrderMinutes*ここで、*OrderMinutes* は分単位の時間です。

- コンピューティング予約 — 1 つの配分識別子に含まれるジョブが、ジョブキューにアタッチされているすべてのリソースを使い切ることを防ぎます。予約比率は *(computeReservation/100)^ActiveFairShares* であり、ActiveFairShares は、アクティブな配分識別子の数です。

  > **Note**
  >
  > 配分識別子に SUBMITTED、PENDING、RUNNABLE、STARTING、または RUNNING の状態のジョブがある場合、そのジョブはアクティブな配分識別子と見なされます。減衰時間が切れると、配分識別子は非アクティブと見なされます。

- 重み係数 – 配分識別子の重み係数。デフォルト値は 1 です。値を小さくすると、配分識別子からジョブが実行されるか、配分識別子のランタイムが長くなります。例えば、重み係数 0.125 (1/8)

の配分識別子を使用するジョブには、重み係数 1 の配分識別子を使用するジョブの 8 倍のコンピューティングリソースが割り当てられます。

> **Note**
>
> この属性は、デフォルトの重み係数である 1 を更新する必要がある場合のみ設定します。

ジョブキューがアクティブでジョブを処理しているときに、ジョブキュースナップショットを使用して最初の 100 個の RUNNABLE ジョブリストを確認できます。詳細については、「でジョブキューを表示する AWS Batch」を参照してください。

チュートリアル: スケジューリングポリシーを作成する

スケジューリングポリシーを使用してジョブキューを作成する前に、スケジューリングポリシーを作成する必要があります。公平配分スケジューリングポリシーを作成するときは、キューの重みに 1 つ以上の配分識別子または配分識別子のプレフィックスを関連付けて、オプションで減衰期間とコンピューティング予約をポリシーに割り当てます。

スケジューリングポリシーを作成するには

1. AWS Batch コンソールを https://console.aws.amazon.com/batch/ で開きます。
2. ナビゲーションバーから、使用するリージョンを選択します。
3. ナビゲーションペインで、[Scheduling policies] (ポリシー)、[Create] (ポリシーの作成) の順に選択します。
4. [Name] (名前) に、スケジューリングポリシーの一意の名前を入力します。最大 128 文字の英字 (大文字と小文字)、数字、ハイフン、アンダースコアを使用できます。
5. (オプション) [割合の減衰秒] に、公平配分スケジューリングポリシーの共有減衰時間の整数値を入力します。共有減衰時間が長いほど、ジョブをスケジュールするときに長時間にわたってコンピューティングリソースの使用量がかなり大きくなります。これにより、配分識別子を使用するジョブは、その配分識別子が最近コンピューティングリソースを使用していなかった場合に、その配分識別子の重みよりも多くのコンピューティングリソースを一時的に使用できるようになります。
6. (オプション) [コンピューティング予約] に、公平配分スケジューリングポリシーのコンピューティング予約を示す整数値を入力します。コンピューティング予約は、現在アクティブでない配分識別子に使用するために、一部の vCPUs を予備に保持します。

予約比率は (*computeReservation*/100)^*ActiveFairShares* であり、ActiveFairShare は、アクティブな配分識別子の数です。

例えば、computeReservation の値が 50 の場合、AWS Batch は、配分識別子が 1 つしかない場合は、使用可能な最大 VCPU の 50% を確保し、2 つの配分識別子がある場合は 25%、配分識別子が 3 つある場合は 12.5% を予約します。ある computeReservation の値が 25 の場合、AWS Batch は、配分識別子が 1 つしかない場合は、使用可能な最大の VCPU の 25%、配分識別子が 2 つある場合は 6.25%、配分識別子が 3 つある場合は 1.56% を予約します。

7. [属性の共有] セクションでは、公平配分スケジューリングポリシーに関連付ける各配分識別子の配分識別子と重みを指定できます。

   a. [Add share identifier] (配分識別子を追加) を選択します。

   b. [配分識別子] に配分識別子を指定します。文字列が「**」で終わる場合、これはジョブの配分識別子の照合に使用される配分識別子のプレフィックスになります。スケジューリングポリシー内のすべての配分識別子と配分識別子のプレフィックスは一意でなければならず、重複することはできません。例えば、同じ公平配分スケジューリングポリシーに配分識別子のプレフィックス「UserA**」と配分識別子「UserA1」を含めることはできません。

   c. [重み係数] には、配分識別子の相対的な重みを指定します。デフォルト値は 1.0 です。値が小さいほど、コンピューティングリソースの優先順位が高くなります。配分識別子のプレフィックスが使用されている場合、プレフィックスで始まる配分識別子を持つジョブは重み係数を共有します。これにより、これらのジョブの重み係数が効果的に増加し、個々の優先度は下がりますが、配分識別子のプレフィックスには同じ重み係数が維持されます。

8. (オプション) タグセクションで、スケジューリングポリシーに関連付ける各タグのキーと値を指定します。詳細については、AWS Batch リソースのタグ付け を参照してください。

9. [Submit] (送信) を選択して終了し、スケジューリングポリシーを作成します。

リファレンス: スケジューリングポリシーテンプレート

以下に示すのは、空のスケジューリングポリシーテンプレートです。このテンプレートを使ってスケジューリングポリシーを作成し、それをファイルに保存して AWS CLI --cli-input-json オプションで使用できます。これらのパラメータの詳細については、「CreateComputeEnvironment」(AWS Batch API リファレンス内) を参照してください。

> **Note**
>
> ジョブキューテンプレートは、以下の AWS CLI コマンドで生成できます。

```
$ aws batch create-scheduling-policy --generate-cli-skeleton
```

```
{
    "name": "",
    "fairsharePolicy": {
        "shareDecaySeconds": 0,
        "computeReservation": 0,
        "shareDistribution": [
            {
                "shareIdentifier": "",
                "weightFactor": 0.0
            }
        ]
    },
    "tags": {
        "KeyName": ""
    }
}
```

リソース認識型スケジューリング

AWS Batch は、ジョブキュー (JQ) に関連付けられたコンピューティング環境 (CE) の vCPU、GPU、メモリの可用性に基づいてジョブをスケジュールします。ただし、これらの CE リソースの可用性だけではジョブが正常に実行されるとは限りません。なぜなら、ジョブが他の必要なリソースに依存し、これらのジョブがキャンセルまたは終了される可能性があるからです。これにより、コンピューティングリソースが非効率的に使用されます。この問題を解決するために、リソース対応スケジューリングでは、CE で実行されるジョブをスケジュールする前に、CE 以外の依存リソースの可用性を確認できます。

AWS Batch リソース対応スケジューリングを使用すると、サードパーティーのライセンストークン、データベースアクセス帯域幅、サードパーティー API への呼び出しを調整する必要性など、ジョブの実行に必要な消費型リソースに基づいてジョブをスケジュールできます。ジョブの実行に必要な消費型リソースを指定すると、Batch はジョブをスケジュールするときにこれらのリソースの依存関係を考慮します。手動での介入を回避して、消費型リソースの不足によるジョブの失敗や長時間の待機をなくすことができます。必要なすべてのリソースが使用可能なジョブのみを割り当てることで、コンピューティングリソースの使用率の低下を抑制できます。

リソース対応スケジューリングは、FIFO スケジューリングポリシーと公平配分スケジューリングポリシーの両方で使用でき、EKS、ECS、Fargate など、Batch でサポートされているすべてのコンピューティングプラットフォームで使用できます。配列ジョブ、マルチノード並列 (MNP) ジョブ、および通常の Batch ジョブで使用できます。

リソース対応スケジューリングを設定するには、まずジョブの実行に必要なすべての消費型リソースと、各リソースで使用可能な合計数を指定します。次に、消費型リソースを必要とするジョブごとに、必要な各リソースの名前と必要な数量を指定します。バッチは、ジョブキュー内のジョブで使用可能な消費型リソースの数を追跡し、ジョブを正常に実行する上で必要なすべての消費型リソースが使用可能である場合にのみ、ジョブの実行がスケジュールされるようにします。

トピック

消費型リソースを作成する

まず、ジョブの実行時に消費され、限られた数量でのみ利用できる非 CE リソースを表す消費型リソースを作成する必要があります。各消費型リソースには、次のものがあります。

- リソース名 (`consumableResourceName`)。アカウントレベルで一意である必要があります。
- (オプション) リソースタイプ (`resourceType`)。ジョブの完了後にリソースを再利用できるかどうかを示します。これは次のいずれかになります。
  - `REPLENISHABLE` (デフォルト)
  - `NON_REPLENISHABLE`
- 合計数量 (`totalQuantity`)。使用可能な消費型リソースの合計数量を指定します。

アカウントあたりの消費型リソースの最大数は 50,000 です。

コンソール:

1. [AWS Batch コンソール] の左側のナビゲーションパネルで、[消費型リソース] を選択します。

2. [消費型リソースを作成] を選択します。
3. 一意の [リソース名]、[合計リソース数] を入力し、[リソースのタイプ] が [補充可能] か [補充不可] かを選択します。
4. [消費型リソースを作成] を選択します。

API:

`CreateConsumableResource` API を使用して、必要なリソースを定義します。

ジョブの実行に必要なリソースを指定する

ジョブを登録するときに、作成した 1 つ以上のリソースの名前 (`consumableResource`) と、ジョブの各インスタンスに必要なそのリソースの数量 (`quantity`) を指定できます。

Batch は、各リソースの使用可能な単位を常時追跡します。ジョブキュー内のジョブごとに、Batch スケジューラは、指定されたリソース依存関係が使用可能な場合にのみジョブが実行されるようにします。

ジョブがキューの先頭に到達したときにジョブの消費型リソースが利用できない場合、ジョブは必要なすべてのリソースが使用可能になるか、ジョブの状態の時間制限に達するまで `RUNNABLE` 状態で待機します (「でジョブキューを表示する AWS Batch」を参照)。Batch がすべてのリソースが使用可能であることを確認すると、ジョブは `STARTING` 状態に遷移し、次に `RUNNING` に遷移します。リソースは、ジョブが `STARTING` に遷移するとロックされ、ジョブが `SUCCEEDED` または `FAILED` に遷移するとロックが解除されます。

ジョブを送信するときに、特定のジョブに必要なリソースの数量を更新することもできます。

コンソール:

ジョブを定義するときにリソースとその必要数量を指定するには:

1. [AWS Batch コンソール] からジョブ定義ウィザードを使用してジョブを定義します ([ジョブ定義] -> [作成])。
2. ウィザードのステップ 4: [コンテナの設定] の [消費型リソース] で、リストから必要なリソースの [名前] を選択します。[リクエスト値] フィールドに、このジョブのインスタンスに必要なこのリソースの数量を入力し、[消費型リソースの追加] を選択します。
3. ジョブに必要なすべての消費型リソースについて、前のステップを繰り返します。定義するジョブごとに最大 5 つのリソースを指定できます。

4. ジョブ定義ウィザードを完了してから [ジョブ定義の作成] を選択するまでの間、作成した消費型リソースのリストを確認できます。

ジョブの送信時に必要なリソースの量を更新するには:

1. [AWS Batch コンソール] の左側のナビゲーションペインで、[ジョブ]、[新しいジョブを送信] の順に選択します。
2. ウィザードのステップ 2: [オーバーライドを設定] の [消費型リソースのオーバーライド] で、ジョブに必要な数量をオーバーライドする消費型リソースの新しい [リクエスト値] を入力します。
3. このジョブに対して行うすべてのオーバーライドが完了したら [次へ] を選択し、[確認して送信] に進みます。

API:

RegisterJobDefinition API でジョブを登録するときは、リクエストの `consumableResourceProperties` 部分で `consumableResourceList` を使用し、ジョブのインスタンスを実行する上で必要な消費型リソースとそれぞれの数量を指定します。

SubmitJob API を使用してジョブを送信すると、リクエストの `consumableResourcePropertiesOverride` 部分を使用して、消費型リソースのリストとそれぞれの数量を上書きできます。これは、使用可能な合計数量ではなく、ジョブの各インスタンスに必要なリソースの数量のみを上書きすることに注意してください。

使用中かつ使用可能なリソースの数を確認する

Batch では、特定の時点で使用可能なリソースの数 (`availableQuantity`)、使用中のリソースの数 (`inUseQuantity`)、およびリソースの合計数 (`totalQuantity`) をクエリできます。

ジョブが `STARTING` 状態になると、消費されたリソースはそのリソースの使用可能な数量から減算されます。リソースが `REPLENISHABLE` の場合、ジョブが `SUCCEEDED` または `FAILED` 状態に遷移するとすぐに、消費されたリソースの数は使用可能な数量に再び追加され、合計数量は同じままになります。リソースが `NON_REPLENISHABLE` の場合、消費されたリソースの数は合計数量と使用可能な数量の両方から減算され、ジョブが `SUCCEEDED` または `FAILED` 状態に遷移しても追加されません。

> **Note**
>
> この情報は最大 30 秒遅れることがあります。

コンソール:

1. [AWS Batch コンソール] の左側のナビゲーションパネルで、[消費型リソース] を選択します。
2. [補充可能] タブまたは [補充不可] タブのいずれかを選択して、作成したそのタイプのリソースを表示します。
3. [補充可能] リソースごとに、コンソールには [名前]、リソースの [合計] 数量、現在 [使用中] の数量、まだ [使用可能] な数量、[使用率] の計算値 (使用中のリソースの数量をそのリソースの合計数量で割ったもの) が表示されます。

   [補充不可] のリソースごとに、コンソールには、[名前]、現在 [使用中] の数、まだ [使用可能] な数が表示されます。

コンソールのジョブ詳細ページから、消費型リソースに関する最新情報を表示することもできます。

1. [AWS Batch コンソール] の左側のナビゲーションパネルで [ジョブ] を選択し、ジョブの名前を選択してそのジョブの詳細ページを開きます。
2. [補充可能なリソース] と [補充不可能なリソース] の両方に関する情報により、ジョブでそれらが必要かどうかを確認できます。どちらのタイプについても、コンソールにはそのリソースの [名前]、そのジョブにおいて [リクエスト済み] の数量、まだ [使用可能] な数量、現在 [使用中] の数量、リソースの [合計] 数量、[現在の使用率] の計算値 (使用中のリソースの数量をそのリソースの合計数量で割ったもの) が表示されます。

API:

DescribeConsumableResource API を使用すると次のような情報が返されます。

```
{
   "availableQuantity": number,
   "consumableResourceArn": "string",
   "consumableResourceName": "string",
   "createdAt": number,
   "inUseQuantity": number,
   "resourceType": "string",
```

```
   "tags": {
      "string" : "string"
   },
   "totalQuantity": number
}
```

ListConsumableResources API は、使用中のリソースの数量 (inUseQuantity) と現在使用可能なリソースの合計数量 (totalQuantity) も、アカウントで作成したすべての消費型リソースのリストの一部としてレポートします。この API では、消費型リソース名に基づいて、消費型リソースリストクエリをフィルタリングすることもできます。

ジョブで使用中のリソースの数量を更新する

リソースの合計数量を新しい値にリセットしたり、合計数量に加算したり、そこから減算したりできます。

指定した新しい合計数量が以前より大きい場合、Batch はそれに応じてより多くのジョブをスケジュールします。新しい合計数量が以前より少なく、このリソースのユニットが使用されていない場合、Batch は合計 (または使用可能な) 数量を減らします。使用中のユニットがある場合、Batch は使用可能な数量をすぐに減らし、ジョブが終了すると、Batch は合計 (使用可能な) 数量を減らして、最終的に新しい数に到達します。

コンソール:

1. [AWS Batch コンソール] の左側のナビゲーションパネルで、[消費型リソース] を選択します。
2. [補充可能] タブまたは [補充不可] タブのいずれかを選択して、作成したそのタイプのリソースを表示します。
3. [補充不可] のリソース:

    1. 更新するリソースを選択して [アクション] を選択し、[リソースの設定]、[リソースの追加]、[リソースの削除] を選択します。
    2. ポップアップウィンドウが表示され、前のステップで選択したアクションに応じて、[合計値の設定]、[リソースの追加]、[リソースの削除]を行えます。新しい合計値として設定する数量、合計数量に追加する数量、または合計数量から減算する数量を入力し、[OK] を選択します。

    [補充不可] のリソース:

1. 更新するリソースを選択して [アクション] を選択し、[リソースの設定]、[リソースの追加]、[リソースの削除] を選択します。
2. ポップアップウィンドウが表示され、前のステップで選択したアクションに応じて、[使用可能な値の設定]、[リソースの追加]、[リソースの削除]を行えます。使用可能な新しい値として設定する数量、使用可能な数量に追加する数量、または使用可能な数量から減算する数量を入力し、[OK] を選択します。

API:

`UpdateConsumableResource` API を使用して、リソースの新しい合計数量を設定したり、合計数量を増減したりできます。

特定の消費型リソースが必要なジョブを検索する

Batch では、特定の消費型リソースが必要なジョブのリストを取得できます。

コンソール:

1. [AWS Batch コンソール] の左側のナビゲーションパネルで、[消費型リソース] を選択します。
2. リストで、消費型リソースの名前を選択します。そのリソースの詳細ページが開きます。
3. [ジョブの検索] で、ジョブのリストに適用するフィルターを入力します。ジョブ名 (「等しい」、「次で始まる:」)、日付範囲 (ジョブが作成された日時)、その他の条件 (「ジョブキュー」、「ジョブ定義」、「共有ジョブ識別子」) でフィルタリングできます。適用するフィルターのタイプごとに、ドロップダウンリストから使用可能なオプションを選択し、要求された追加情報を入力します。

   [検索] を選択します。
4. 消費型リソースを必要とするジョブのフィルタリングされたリストが表示されます。このリストには、ジョブの名前、ステータス、リクエストされた消費型リソースのユニット数、その他の必要な消費型リソースなどが含まれます。このリストを使用して、[キャンセル] または [終了] するジョブを 1 つ以上選択できます。ジョブの名前を選択して、そのジョブの詳細ページを開くこともできます。
5. [ジョブを検索] で [結果を更新] または [検索をクリア] して最初からやり直すことができるようになりました。

API:

ListJobsByConsumableResource API を使用して、特定の消費型リソースを使用するジョブのリストを取得できます。この API では、ジョブステータスまたはジョブ名を使用してジョブリストクエリをフィルタリングすることもできます。

消費型リソースを削除する

消費型リソースは、リソースを必要とするジョブがまだ実行中であっても、いつでも削除できます。消費型リソースが削除されると、delete コマンドが受信されてからジョブスケジューラが削除を優先するまでにギャップが生じる可能性があるため、リソースを消費するジョブは、削除呼び出しの直後にスケジュールされる可能性があります。削除された消費型リソースにリソースタイプ (resourceType) REPLENISHABLE がある場合、それはジョブが完了すると無視されます。消費型リソースを削除して同じ名前で再作成する場合、そのリソースは同じリソースと見なされ、RUNNABLE ジョブで使用できます。

コンソール:

1. [AWS Batch コンソール] の左側のナビゲーションパネルで、[消費型リソース] を選択します。
2. [補充可能] タブまたは [補充不可] タブのいずれかを選択して、作成したそのタイプのリソースを表示します。
3. 削除する個々のリソースを選択し、[削除] を選択します。ポップアップウィンドウ [消費型リソースの削除] が表示されます。削除を確認するには、[削除] を選択します。

コンソールの詳細ページから消費型リソースを削除することもできます。

1. [AWS Batch コンソール] の左側のナビゲーションパネルで、[消費型リソース] を選択します。
2. [補充可能] タブまたは [補充不可] タブのいずれかを選択して、作成したそのタイプのリソースを表示します。
3. 削除するリソースの名前を選択します。消費型リソースの詳細ページが表示されます。[削除] を選択します。ポップアップウィンドウ [消費型リソースの削除] が表示されます。削除を確認するには、[削除] を選択します。

API:

DeleteConsumableResource API を使用して消費型リソースを削除します。

ジョブ定義

AWS Batch ジョブ定義は、ジョブの実行方法を指定します。各ジョブはジョブ定義を参照する必要がありますが、ジョブ定義に指定されているパラメータの多くはランタイムに上書きできます。

ジョブ定義には以下のような属性が指定されています。

- ジョブのコンテナで使用する Docker イメージ。
- コンテナで使用する vCPU の数とメモリの量。
- コンテナの開始時に実行するコマンド。
- コンテナの開始時に渡す環境変数 (ある場合)。
- コンテナで使用するデータボリューム。
- ジョブが AWS アクセス許可に使用する IAM ロール (ある場合) 。

内容

シングルノードのジョブ定義を作成する

AWS Batch でジョブを実行する前に、ジョブ定義を作成する必要があります。このプロセスは、シングルノードとマルチノード並列ジョブでは若干異なります。このトピックでは特に、マルチノード並列ジョブ (ギャングスケジューリングとも呼ばれます) 以外の AWS Batch ジョブのジョブ定義の作成方法を取り上げます。

Amazon Elastic コンテナサービスのリソースで、マルチノード並列ジョブ定義を作成できます。詳細については、the section called “マルチノード並列ジョブ定義を作成する”を参照してください。

トピック

- Amazon EC2 リソースにシングルノードのジョブ定義を作成する
- Fargate リソースでシングルノードのジョブ定義を作成する
- Amazon EKS リソースにシングルノードのジョブ定義を作成する
- Amazon EC2 リソースに複数のコンテナを持つシングルノードのジョブ定義を作成する

Amazon EC2 リソースにシングルノードのジョブ定義を作成する

Amazon Elastic Compute Cloud (Amazon EC2) リソースにシングルノードのジョブ定義を作成するには、次の手順を実行します。

Amazon EC2 リソースに新しいジョブ定義を作成するには:

1. AWS Batch コンソールを https://console.aws.amazon.com/batch/ で開きます。
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. 左側のナビゲーションペインで ジョブ定義 を選択します。
4. [作成] を選択します。
5. オーケストレーションタイプ には、Amazon Elastic Compute Cloud (Amazon EC2) を選択します。
6. EC2 プラットフォームの設定で、マルチノード並列を有効にする をオフにします。
7. 名前] に、一意のジョブ定義名を入力します。名前の最大長は 128 文字です。大文字、小文字、数字、ハイフン(-)、アンダースコア(_) を含めることができます。
8. (オプション) 実行タイムアウト で、タイムアウト値 (秒単位) を入力します。実行タイムアウトは、未完了のジョブが終了するまでの時間です。試行時間がこの秒数を超過すると、試行は停止し、FAILED ステータスに移行します。詳細については、ジョブのタイムアウトを参照してください。最小値は 60 秒です。
9. (オプション) スケジューリング優先度 をオンにします。0 ~ 100 の値でスケジューリング優先度を入力します。値が大きいほど優先度が高くなります。
10. (オプション) ジョブの試行回数 には、AWS Batch ジョブを RUNNABLE ステータスに移行する最大回数を入力します。1 ~ 10 の整数を入力します。
11. (オプション) 再試行戦略の条件 では、終了時に評価を追加 を選択します。少なくとも 1 つのパラメータ値を入力し、アクション を選択します。条件セットごとに、アクション を 再試行 または 終了 に設定する必要があります。これらのアクションは、以下のことを意味します。

- 再試行 — AWS Batch は、指定したジョブ試行回数に達するまで再試行します。
- 終了 — AWS Batch は、ジョブの再試行を停止します。

> **Important**
>
> 終了時に評価を追加 を選択した場合は、少なくとも 1 つのパラメータを設定して アクション] を選択するか、終了時に評価を削除] を選択します。

12. (オプション) タグ を展開し、タグを追加 を選択してリソースにタグを追加します。キーとオプショナル値を入力し、新しいタグを追加を選択します。
13. (オプション) タグを伝播 をオンにして、タグをジョブとジョブ定義から Amazon ECS タスクに伝達することができます。
14. 次のページ を選択します。
15. コンテナ設定 セクションで次の操作を行います。

    a. イメージ で、ジョブに使用する Docker イメージを選択します。デフォルトでは、Docker Hub レジストリのイメージを使用できます。*repository-url*/*image*:*tag* で他のリポジトリを指定することもできます。名前の最大長は 225 文字です。大文字、小文字、数字、ハイフン (-)、アンダースコア (_)、コロン (:)、スラッシュ (/)、およびシャープ (#) を含めることができます。このパラメータは、Docker Remote API の コンテナの作成 セクションにある `Image` および docker run の `IMAGE` パラメータにマッピングされます。

    > **Note**
    >
    > Docker イメージのアーキテクチャは、スケジュールされているコンピューティングリソースのプロセッサアーキテクチャと一致している必要があります。例えば、ARM ベースの Docker イメージは、ARM ベースのコンピューティングリソースでのみ実行することができます。

    - Amazon ECR Public リポジトリ内のイメージには、完全な `registry/repository[:tag]` または `registry/repository[@digest]` 命名規則が使用されます (例えば、`public.ecr.aws/`*registry_alias*`/`*my-web-app*`:`*latest*)。
    - Amazon ECR リポジトリ内のイメージには、完全な `registry/repository[:tag]` 命名規則が使用されます (例え

ば、*aws_account_id*.dkr.ecr.*region*.amazonaws.com/*my-web-app*:*latest*)。

- Docker Hub の公式リポジトリのイメージでは、1 つの名前 (例: ubuntu または mongo) を使用します。
- Docker Hub の他のリポジトリのイメージは、組織名で修飾されます (例: amazon/amazon-ecs-agent)。
- 他のオンラインリポジトリのイメージは、さらにドメイン名で修飾されます (例: quay.io/assemblyline/ubuntu)。

b. [コマンド] では、コマンドを JSON 文字列配列に相当するものとしてフィールドに入力します。

このパラメータは、Docker Remote API の コンテナの作成 セクションにある Cmd にマッピングされ、COMMAND パラメータは docker run にマッピングされます。DockerCMD パラメータの詳細については、https://docs.docker.com/engine/reference/builder/#cmdを参照してください。

> **Note**
>
> コマンドには、パラメータ置換のデフォルト値とプレースホルダーを使用できます。詳細については、パラメータを参照してください。

c. (オプション) 実行ロール で、ユーザーに代わって AWS API コールを実行するためのアクセス許可を Amazon ECS コンテナエージェントに付与する IAM ロールを指定できます。この機能では、タスク用の Amazon ECS IAM ロールを使用します。詳細については、「Amazon Elastic Container Service デベロッパーガイド」の「Amazon ECS タスク実行 IAM ロール」を参照してください。

d. ジョブロール設定 で、AWS API へのアクセス権限を持つ IAM ロールを選択します。この機能では、タスク用の Amazon ECS IAM ロールを使用します。詳細については、Amazon Elastic Container Service デベロッパーガイドのタスク用の IAM ロールを参照してください。

> **Note**
>
> ここには、Amazon Elastic Container Service Task Role] 信頼関係があるロールのみが表示されます。AWS Batch ジョブの IAM ロールの作成に関する詳細について

は、Amazon Elastic Container Service デベロッパーガイドのタスク用の IAM ロールとポリシーの作成を参照してください。

16. [パラメータ] で [パラメータを追加] を選択し、パラメータ代替プレースホルダーを [キー] とオプションの [値] のペアとして追加します。
17. 環境設定 セクションで:
    a. vCPU では、コンテナ用に予約する vCPU の数を指定します。このパラメータは、Docker Remote API の コンテナの作成 セクションにある `CpuShares` にマッピングされ、`--cpu-shares` オプションは docker run にマッピングされます。各 vCPU は 1,024 個の CPU 配分に相当します。少なくとも 1 つの vCPU を指定する必要があります。
    b. メモリ には、コンテナで使用できるメモリー制限を入力します。コンテナは、ここで指定したメモリを超えようとすると、強制終了されます。このパラメータは、Docker Remote API のコンテナの作成セクションの `Memory` にマップされ、`--memory` オプションは docker run にマップされます。ジョブに対して少なくとも 4 MiB のメモリを指定する必要があります。

    > **Note**
    > 特定のインスタンスタイプのジョブにメモリの優先順位を付けることで、リソース使用率を最大化できます。詳細については、コンピューティングリソースメモリの管理を参照してください。

    c. GPU の数 では、コンテナ用に予約する GPU の数を指定します。
    d. (オプション) 環境変数 で 環境変数を追加 を選択し、環境変数を名前と値のペアとして追加します。これらの変数は、コンテナに渡されます。
    e. (オプション) シークレット で、シークレットを追加 を選択して、シークレットを名前と値のペアとして追加します。これらのシークレットはコンテナに公開されます。詳細については、「LogConfiguration:secretOptions」を参照してください。
18. 次のページ を選択します。
19. Linux 設定 セクションで :
    a. ユーザー] には、コンテナ内で使用するユーザー名を入力します。このパラメータは、Docker Remote API のコンテナの作成セクションの `User` にマップされ、`--user` オプションは docker run にマップされます。

b. (オプション) 特権付与 スライダーを右にドラッグすることで、ホストインスタンスに対する昇格されたアクセス許可 (root ユーザーと同様) をジョブコンテナに付与することができます。このパラメータは、Docker Remote API のコンテナの作成セクションの Privileged にマップされ、--privileged オプションは docker run にマップされます。

c. (オプション)init プロセスを有効にする をオンにすると、コンテナ内で init プロセスを実行できます。このプロセスは信号を転送し、プロセスを利用します。

20. (オプション) ファイルシステム設定 セクションで：

a. 読み取り専用ファイルシステムを有効にする をオンにして、ボリュームへの書き込みアクセスを削除します。

b. 共有メモリサイズ では、/dev/shm ボリュームのサイズ (MiB) を入力します。

c. 最大スワップサイズ では、コンテナが使用できるスワップメモリの合計容量 (MiB 単位) を入力します。

d. スワップ動作 では、コンテナのスワップ動作を示す 0 ~ 100 の値を入力します。値を指定せず、スワップが有効になっている場合、この値はデフォルトで 60 に設定されます。詳細については、「LinuxParameters:swappiness」を参照してください。

e. (オプション) [追加設定] を展開します。

f. (オプション) Tmpfs では、tmpfs を追加 を選択して tmpfs マウントを追加します。

g. (オプション) デバイス では、デバイスを追加 を選択してデバイスを追加します。

i. コンテナパス] では、コンテナインスタンスでのデバイスのパスを指定します。このパスは、ホストインスタンスにマッピングされたデバイスを公開するために使用されます。空白のままにすると、ホストパスがコンテナで使用されます。

ii. ホストパス] では、ホストインスタンスでのデバイスのパスを指定します。

iii. アクセス許可] では、デバイスに適用する 1 つ以上のアクセス許可を選択します。使用できる権限は、読み取り、書き込み、MKNOD です。

h. (オプション) ボリューム設定 で、ボリュームを追加 を選択して、コンテナに渡すボリュームのリストを作成します。ボリュームの [名前] と [ソースパス] を入力し、[ボリュームを追加] を選択します。EFS を有効にする オプションを選択することもできます。

i. (オプション) マウントポイント で、マウントポイント構成を追加 を選択し、データボリュームにマウントポイントを追加します。ソースボリュームとコンテナパスを指定する必要があります。これらのマウントポイントはコンテナインスタンス上の Docker daemon に渡されます。ボリュームを 読み取り専用 にすることもできます。

j. (オプション) [Ulimit 設定] では、[ulimit を追加] を選択して、コンテナの `ulimits` 値を追加します。名前、ソフトリミット、ハードリミット の値を入力し、ulimit を追加 を選択します。

21. [タスクプロパティ] セクションで、次の操作を行います。

a. [実行ロール - 条件付き] で、Amazon ECS エージェントがユーザーに代わって AWS API コールを実行できるようにするロールを選択します。実行ロールの作成について詳しくは、「チュートリアル: IAM 実行ロールを作成する」を参照してください。

b. [ECS 実行コマンドを有効化] を選択し、Amazon ECS コンテナシェルへの直接アクセスを有効化し、ホスト OS をバイパスします。[タスクロール] を選択する必要があります。

> **Important**
>
> [ECS 実行] コマンドでは、ファイルシステムが書き込み可能である必要があります。

c. [タスクロール] で Amazon ECS Identity and Access Management (IAM) ロールを選択し、コンテナがユーザーに代わって AWS API コールを実行できるようにします。詳細については、「Amazon Elastic Container Service デベロッパーガイド」の「Amazon ECS のタスク IAM ロール」を参照してください。

22. (オプション) [ロギング設定] セクション:

a. [ログドライバー] で、使用するログドライバーを選択します。使用できるログドライバーの詳細については、「LogConfiguration:logDriver」を参照してください。

> **Note**
>
> デフォルトでは、`awslogs` ログドライバーが使用されます。

b. [オプション] では、[オプションを追加] を選択してオプションを追加します。名前と値のペアを入力し、[オプションを追加] を選択します。

c. [シークレット] で、[シークレットを追加] を選択します。名前と値のペアを入力し、[シークレットを追加] を選択してシークレットを追加します。

> **Tip**
>
> 詳細については、「LogConfiguration:secretOptions」を参照してください。

23. 次のページ を選択します。
24. ジョブ定義のレビューについては、設定手順を確認してください。変更する必要がある場合は、編集] を選択します。完了したら、ジョブ定義の作成 を選択します。

Fargate リソースでシングルノードのジョブ定義を作成する

AWS Fargate リソースにシングルノードのジョブ定義を作成するには、次の手順を実行します。

Fargate リソースに新しいジョブ定義を作成するには:

1. AWS Batch コンソールを https://console.aws.amazon.com/batch/ で開きます。
2. 上部の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. 左側のナビゲーションペインで ジョブ定義 を選択します。
4. 作成] を選択します。
5. オーケストレーションタイプ] には Fargate を選択します。詳細については、Fargate のコンピューティング環境を参照してください。
6. 名前] には、一意のジョブ定義名を入力します。名前の最大長は 128 文字です。大文字、小文字、数字、ハイフン(-)、アンダースコア(_) を含めることができます。
7. (オプション) 実行タイムアウト で、タイムアウト値 (秒単位) を入力します。実行タイムアウトは、未完了のジョブが終了するまでの時間です。試行時間がこの秒数を超過すると、試行は停止し、FAILED ステータスに移行します。詳細については、ジョブのタイムアウトを参照してください。最小値は 60 秒です。
8. (オプション) スケジューリング優先度 をオンにします。0 ~ 100 の値でスケジューリング優先度を入力します。値が大きいほど、低い値よりも優先されます。
9. (オプション) タグ を展開し、タグを追加 を選択してリソースにタグを追加します。タグを伝達をオンにして、タグをジョブとジョブ定義から伝達することができます。
10. Fargate プラットフォーム設定 セクションで:
    a. ランタイムプラットフォーム では、コンピューティング環境アーキテクチャを選択します。
    b. オペレーティングシステムファミリ では、コンピューティング環境のオペレーティングシステムを選択します。
    c. CPU アーキテクチャ で、vCPU アーキテクチャを選択します。
    d. Fargate プラットフォームバージョン では、LATEST または特定のランタイム環境バージョンを入力します。

e. (オプション) パブリック IP アドレスの割り当て をオンにして、Fargate ジョブネットワークネットワークインターフェイスにパブリック IP アドレスを割り当てます。プライベートサブネットで動作しているジョブがインターネットにアウトバウンドトラフィックを送信するためには、プライベートサブネットに NAT ゲートウェイをインターネットへのルートリクエストに接続する必要があります。コンテナイメージをプルできるように、この操作を行う必要がある場合があります。詳細については、Amazon Elastic Container Service デベロッパーガイドのAmazon ECS のタスクネットワーキングを参照してください。

f. (オプション) エフェメラルストレージ で、タスクに割り当てるエフェメラルストレージの容量を入力します。エフェメラルストレージの容量は 21 GiB から 200 GiB の間に設定する必要があります。値を指定しない場合には、デフォルトで 20 GiB の一時ストレージが割り当てられます。

> **Note**
>
> エフェメラルストレージには、Fargate プラットフォームバージョン 1.4 以降が必要です。

g. 実行ロール では、Amazon ECS コンテナおよび Fargate エージェントがユーザーに代わって AWS API コールを呼び出せるようにする IAM ロールを選択します。この機能では、タスク用の Amazon ECS IAM ロールを使用します。設定の前提条件を含む詳細については、Amazon Elastic Container Service 開発者ガイドのAmazon ECS タスク実行 IAM ロールを参照してください。

h. ジョブの試行回数 には、AWS Batch ジョブを `RUNNABLE` ステータスに移行する最大回数を入力し 1 ~ 10 の整数を入力します。

i. (オプション) 再試行戦略の条件 では、終了時に評価を追加 を選択します。少なくとも 1 つのパラメータ値を入力し、アクション を選択します。条件セットごとに、アクション を 再試行 または 終了 に設定する必要があります。これらのアクションは、以下のことを意味します。

- 再試行 — AWS Batch は、指定したジョブ試行回数に達するまで再試行します。
- 終了 — AWS Batch は、ジョブの再試行を停止します。

> **Important**
>
> 終了時に評価を追加 を選択した場合は、少なくとも 1 つのパラメータを設定してアクションを選択するか、終了時に評価を削除 を選択する必要があります。

11. 次のページ を選択します。
12. コンテナ設定 セクションで次の操作を行います。

    a. イメージ] で、ジョブに使用する Docker イメージを選択します。デフォルトでは、Docker Hub レジストリのイメージを使用できます。*repository-url*/*image*:*tag* で他のリポジトリを指定することもできます。名前の最大長は 225 文字です。大文字と小文字の英文字、数字、ハイフン (-)、アンダースコア (_)、コロン (:)、ピリオド (.)、スラッシュ (/)、および数字 (#)を含めることができます。このパラメータは、Docker Remote API の コンテナの作成 セクションにある `Image` および docker run の `IMAGE` パラメータにマッピングされます。

    > **Note**
    >
    > Docker イメージのアーキテクチャは、スケジュールされているコンピューティングリソースのプロセッサアーキテクチャと一致している必要があります。例えば、ARM ベースの Docker イメージは、ARM ベースのコンピューティングリソースでのみ実行することができます。

    - Amazon ECR Public リポジトリ内のイメージには、完全な `registry/repository[:tag]` または `registry/repository[@digest]` 命名規則が使用されます (例えば、`public.ecr.aws/`*registry_alias*`/`*my-web-app*`:`*latest*)。
    - Amazon ECR リポジトリ内のイメージには、完全な `registry/repository[:tag]` 命名規則が使用されます (例えば、*aws_account_id*`.dkr.ecr.`*region*`.amazonaws.com /`*my-web-app*`:`*latest*)。
    - Docker Hub の公式リポジトリのイメージでは、1 つの名前 (例: `ubuntu` または `mongo`) を使用します。
    - Docker Hub の他のリポジトリのイメージは、組織名で修飾されます (例: `amazon/amazon-ecs-agent`)。

- 他のオンラインリポジトリのイメージは、さらにドメイン名で修飾されます (例: `quay.io/assemblyline/ubuntu`)。

b. [コマンド] では、コマンドを JSON 文字列配列に相当するものとしてフィールドに入力します。

このパラメータは、Docker Remote API の コンテナの作成 セクションにある `Cmd` にマッピングされ、`COMMAND` パラメータは docker run にマッピングされます。Docker`CMD` パラメータの詳細については、https://docs.docker.com/engine/reference/builder/#cmdを参照してください。

> **Note**
> コマンドには、パラメータ置換のデフォルト値とプレースホルダーを使用できます。詳細については、パラメータを参照してください。

c. (オプション) パラメータを名前と値のペアとしてジョブ定義に追加することで、デフォルトのジョブ定義を上書きすることができます。パラメータを追加するには:

- パラメータ で パラメータの追加 を選択し、名前と値のペアを入力して パラメータの追加 を選択します。

> **Important**
> パラメータを追加 を選択した場合は、少なくとも1つのパラメータを設定するか、パラメータの削除 を選択する必要があります。

d. 環境設定 セクションで:

i. ジョブロール設定 で、AWS API へのアクセス権限を持つ IAM ロールを選択します。この機能では、タスク用の Amazon ECS IAM ロールを使用します。詳細については、Amazon Elastic Container Service デベロッパーガイドのタスク用の IAM ロールを参照してください。

> **Note**
> ここには、Amazon Elastic Container Service Task Role] 信頼関係があるロールのみが表示されます。AWS Batch ジョブの IAM ロールの作成方法に関する

詳細については、「Amazon Elastic Container Service デベロッパーガイド」の「タスク用の IAM ロールとポリシーの作成」を参照してください。

ii. vCPU では、コンテナ用に予約する vCPU の数を指定します。このパラメータは、Docker Remote API の コンテナの作成 セクションにある `CpuShares` にマッピングされ、`--cpu-shares` オプションは docker run にマッピングされます。各 vCPU は 1,024 個の CPU 配分に相当します。少なくとも 1 つの vCPU を指定する必要があります。

iii. メモリ で、コンテナで使用できるメモリ制限を入力します。コンテナは、ここで指定したメモリを超えようとすると、停止されます。このパラメータは、Docker Remote API のコンテナの作成セクションの `Memory` にマップされ、`--memory` オプションは docker run にマップされます。ジョブに対して少なくとも 4 MiB のメモリを指定する必要があります。

GuardDuty Runtime Monitoring を使用する場合、GuardDuty セキュリティエージェントには多少のメモリオーバーヘッドがあります。したがって、メモリ制限には GuardDuty セキュリティエージェントのサイズを含める必要があります。GuardDuty セキュリティエージェントのメモリ制限については、「GuardDuty ユーザーガイド」の「CPU およびメモリ制限」を参照してください。ベストプラクティスの詳細については、「Amazon ECS 開発者ガイド」の「ランタイムモニタリングを有効にした後、Fargate タスクのメモリ不足エラーに対処するにはどうすればよいですか?」を参照してください。

> **Note**
>
> 特定のインスタンスタイプのジョブにメモリの優先順位を付けることで、リソース使用率を最大化できます。詳細については、コンピューティングリソースメモリの管理を参照してください。

e. (オプション) 環境変数 で 環境変数を追加 を選択し、環境変数を名前と値のペアとして追加します。これらの変数は、コンテナに渡されます。

f. (オプション) シークレット で、シークレットを追加 を選択して、シークレットを名前と値のペアとして追加します。これらのシークレットはコンテナに公開されます。詳細については、「LogConfiguration:secretOptions」を参照してください。

g. 次のページ を選択します。

13. (オプション) Linux 設定 セクションで:

a. ユーザー では、コンテナ内で使用するユーザー名を入力します。

b. init プロセスを有効にする をオンにすると、コンテナ内で init プロセスを実行できます。このプロセスは信号を転送し、プロセスを利用します。

c. 読み取り専用ファイルシステムを有効にする をオンにして、ボリュームへの書き込みアクセスを削除します。

d. (オプション) 追加設定] を展開します。

e. マウントポイント設定 では、マウントポイント設定の追加 を選択し、データボリュームにマウントポイントを追加します。ソースボリュームとコンテナパスを指定する必要があります。これらのマウントポイントはコンテナインスタンス上の Docker daemon に渡されます。

f. ボリューム設定 で、ボリュームを追加 を選択して、コンテナに渡すボリュームのリストを作成します。ボリュームの 名前 と ソースパス を入力し、ボリュームを追加 を選択します。

g. [タスクプロパティ] セクションで、次の操作を行います。

   i. [実行ロール - 条件付き] で、Amazon ECS エージェントがユーザーに代わって AWS API コールを実行できるようにするロールを選択します。実行ロールの作成について詳しくは、「チュートリアル: IAM 実行ロールを作成する」を参照してください。

   ii. [ECS 実行コマンドを有効化] を選択し、Amazon ECS コンテナシェルへの直接アクセスを有効化し、ホスト OS をバイパスします。[タスクロール] を選択する必要があります。

   > **Important**
   >
   > [ECS 実行] コマンドでは、ファイルシステムが書き込み可能である必要があります。

   iii. [タスクロール] で Amazon ECS Identity and Access Management (IAM) ロールを選択し、コンテナがユーザーに代わって AWS API コールを実行できるようにします。詳細については、「Amazon Elastic Container Service デベロッパーガイド」の「Amazon ECS のタスク IAM ロール」を参照してください。

h. ロギング設定 セクションで:

i. (オプション) ログドライバー で、使用するログドライバーを選択します。使用できるログドライバーの詳細については、「LogConfiguration:logDriver」を参照してください。

> **Note**
>
> デフォルトでは、`awslogs` ログドライバーが使用されます。

ii. オプション では、オプションを追加 を選択してオプションを追加します。名前と値のペアを入力し、オプションを追加 を選択します。

iii. (オプション) シークレット で、シークレットを追加 を選択してシークレットを追加します。名前と値のペアを入力し、シークレットを追加 を選択します。

> **Tip**
>
> 詳細については、「LogConfiguration:secretOptions」を参照してください。

14. 次のページ を選択します。
15. ジョブ定義のレビューについては、設定手順を確認してください。変更する必要がある場合は、編集] を選択します。完了したら、ジョブ定義の作成 を選択します。

Amazon EKS リソースにシングルノードのジョブ定義を作成する

Amazon Elastic Kubernetes Service (Amazon EKS) でシングルノードのジョブ定義を作成するには、次の手順を実行します。

Amazon EKS リソースに新しいジョブ定義を作成するには:

1. AWS Batch コンソールを https://console.aws.amazon.com/batch/ で開きます。
2. 上部の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. 左側のナビゲーションペインで ジョブ定義 を選択します。
4. 作成] を選択します。
5. オーケストレーションタイプ には、Elastic Kubernetes Service (EKS) を選択します。
6. 名前] に、一意のジョブ定義名を入力します。名前の最大長は 128 文字です。大文字、小文字、数字、ハイフン(-)、アンダースコア(_) を含めることができます。

7. (オプション) 実行タイムアウト で、タイムアウト値 (秒単位) を入力します。実行タイムアウトは、未完了のジョブが終了するまでの時間です。試行時間がこの秒数を超過すると、試行は停止し、FAILED ステータスに移行します。詳細については、ジョブのタイムアウトを参照してください。最小値は 60 秒です。
8. (オプション) スケジューリング優先度 をオンにします。0 ~ 100 の値でスケジューリング優先度を入力します。値が大きいほど、低い値よりも優先されます。
9. (オプション) タグ を展開し、次にタグを追加] を選択してリソースにタグを追加します。
10. 次のページ を選択します。
11. EKS pod プロパティ セクションで:

    a. サービスアカウント名には、podで実行されるプロセスのIDを提供するアカウントを入力します。

    b. ホストネットワーク を有効にしてKubernetespod ネットワークモデルを使用し、受信接続用のリスニングポートを開きます。この設定は送信通信でのみオフにしてください。

    c. (オプション) DNSポリシー で、次のいずれかを選択します：

    - 値なし (null化) — pod は 、Kubernetes 環境からの DNS 設定を無視します。
    - デフォルト — podは、実行されているノードからDNS名前解決を継承します。

    > **Note**
    >
    > DNSポリシーが指定されていない場合、Default はデフォルトDNSポリシーではありません。代わりに、ClusterFirst が使用されます。

    - ClusterFirst — 設定されたクラスタードメインサフィックスと一致しないDNS クエリは、ノードから継承されたアップストリームのネームサーバーに転送されます。
    - ClusterFirstWithHostNet — [ホストネットワーク] がオンになっている場合に使用します。

    d. (オプション) [ボリューム] で、[ボリュームの追加] を選択し、次の操作を行います。

        i. ボリュームの [名前] を追加します。

        ii. (オプション) ホスト上のディレクトリの [ホストパス] を追加します。

        iii. (オプション) [中間] と [サイズ制限] を追加して、[Kubernetes emptyDir] を設定します。

        iv. (オプション) ポッドの [シークレット名] と、シークレットが [オプション] であるかどうかを指定します。

v. (オプション) Kubernetes [永続ボリュームクレーム] をポッドにアタッチする [クレーム名] と、それが [読み取り専用] かどうかを定義します。

e. (オプション) ポッドラベル には、[ポッドラベルを追加] を選択し、名前と値のペアを入力します。

> **Important**
>
> ポッドラベルのプレフィックスには`kubernetes.io/`、`k8s.io/`、または `batch.amazonaws.com/` を含めることはできません。

f. (オプション) [ポッドアノテーション] で [アノテーションの追加] を選択し、名前と値のペアを入力します。

> **Important**
>
> ポッドアノテーションのプレフィックスには `kubernetes.io/`、`k8s.io/`、`batch.amazonaws.com/` を含めることはできません。

g. 次のページ を選択します。

h. コンテナの設定 セクションで次の操作を行います。

i. 名前 に、コンテナの一意の名前を入力します。名前は文字または数字で始まる必要があり、最長63文字まで入力できます。小文字と大文字の英文字、数字、およびハイフン (-) を含めることができます。

ii. Image (イメージ) で、ジョブに使用するDocker イメージを選択します。デフォルトでは、Docker Hubレジストリのイメージを使用できます。*`repository-url/image:tag`* で他のリポジトリを指定することもできます。名前の長さは最大255文字です。大文字と小文字の英文字、数字、ハイフン (-)、アンダースコア (_)、コロン (:)、ピリオド (.)、スラッシュ (/)、および数字 (#)を含めることができます。このパラメータは、Docker Remote API の「Create a container」セクションにある `Image` および docker run の `IMAGE` パラメータにマッピングされます。

> **Note**
>
> Docker イメージのアーキテクチャは、スケジュールされているコンピューティングリソースのプロセッサアーキテクチャと一致している必要があります。例

えば、ARM ベースの Docker イメージは、ARM ベースのコンピューティングリソースでのみ実行することができます。

- Amazon ECR Public リポジトリ内のイメージには、完全な `registry/repository[:tag]` または `registry/repository[@digest]` 命名規則が使用されます (例えば、`public.ecr.aws/`*`registry_alias`*`/`*`my-web-app`*`:`*`latest`*)。
- Amazon ECR リポジトリ内のイメージには、完全な `registry/repository[:tag]` 命名規則が使用されます (例えば、*`aws_account_id`*`.dkr.ecr.`*`region`*`.amazonaws.com /`*`my-web-app`*`:`*`latest`*)。
- Docker Hub の公式リポジトリのイメージでは、1 つの名前 (例: `ubuntu` または `mongo`) を使用します。
- Docker Hub の他のリポジトリのイメージは、組織名で修飾されます (例: `amazon/amazon-ecs-agent`)。
- 他のオンラインリポジトリのイメージは、さらにドメイン名で修飾されます (例: `quay.io/assemblyline/ubuntu`)。

iii. (オプション) イメージプルポリシー では、イメージをプルするタイミングを選択します。

iv. (オプション) [コマンド] には、コンテナに渡す JSON コマンドを入力します。

v. (オプション) 引数、コンテナに渡す引数を入力します。引数が指定されていない場合は、コンテナイメージコマンドが使用されます。

i. (オプション) 名前と値のマッピングとしてジョブ定義にパラメータを追加して、ジョブ定義のデフォルトを上書きできます。パラメータを追加するには :

- パラメータ] には、名前と値のペアを入力し、次にパラメータの追加 を選択します。

> **Important**
> パラメータを追加 を選択した場合は、少なくとも 1つのパラメータを設定するか、パラメータの削除 を選択する必要があります。

j. 環境設定 セクションで:

i. vCPU では、コンテナ用に予約する vCPU の数を指定します。このパラメータは、Docker Remote API の コンテナの作成 セクションにある `CpuShares` にマッピングされ、`--cpu-shares` オプションは docker run にマッピングされます。各 vCPU は 1,024 個の CPU 配分に相当します。少なくとも 1 つの vCPU を指定する必要があります。

ii. メモリ には、コンテナで使用できるメモリー制限を入力します。コンテナは、ここで指定したメモリを超えようとすると、停止されます。このパラメータは、Docker Remote API のコンテナの作成セクションの `Memory` にマップされ、`--memory` オプションは docker run にマップされます。ジョブに対して少なくとも 4 MiB のメモリを指定する必要があります。

> **Note**
>
> 特定のインスタンスタイプのジョブにメモリの優先順位を付けることで、リソース使用率を最大化できます。詳細については、コンピューティングリソースメモリの管理を参照してください。

k. (オプション) 環境変数 で 環境変数を追加 を選択し、環境変数を名前と値のペアとして追加します。これらの変数は、コンテナに渡されます。

l. (オプション) ボリュームマウント では:

i. ボリュームマウントを追加 を選択します。

ii. 名前 を入力し、そして次にボリュームがマウントされているコンテナに [マウントパス] を入力します。[SubPath] を入力して、ルートではなく、参照されるボリューム内のサブパスを指定します。

iii. 読み取り専用 を選択すると、ボリュームへの書き込み権限が削除されます。

iv. ボリュームマウントを追加 を選択します。

m. (オプション)ユーザーとして実行 には、ユーザー ID を入力してコンテナプロセスを実行します。

> **Note**
>
> コンテナを実行するには、ユーザー ID がイメージに存在している必要があります。

n. (オプション)グループとして実行 には、コンテナプロセスのランタイムを実行するグループ ID を入力します。

> **Note**
>
> コンテナを実行するには、グループ ID がイメージに存在している必要があります。

o. (オプション) ホストインスタンスに対する昇格されたアクセス権限 (root ユーザーと同様に) をジョブのコンテナに付与するには、特権付与 を選択します。このパラメータは、Docker Remote API のコンテナの作成セクションの Privileged にマップされ、--privileged オプションは docker run にマップされます。

p. (オプション) 読み取り専用ルートファイルシステム を有効にして、ルートファイルシステムへの書き込みアクセスを削除します。

q. (オプション) 非ルートユーザーとして pod でコンテナを実行するために、[非ルートとして実行] をオンにします。

> **Note**
>
> 非ルートユーザーとして実行する がオンになっている場合、kubelet は実行時にイメージを検証して、イメージが UID 0 として実行されていないことを確認します。

r. 次のページ を選択します。

12. ジョブ定義のレビューについては、設定手順を確認してください。変更する必要がある場合は、編集] を選択します。完了したら、ジョブ定義の作成 を選択します。

Amazon EC2 リソースに複数のコンテナを持つシングルノードのジョブ定義を作成する

Amazon Elastic Compute Cloud (Amazon EC2) リソースに複数のコンテナを持つシングルノードのジョブ定義を作成するには、次の手順を実行します。

Amazon EC2 リソースに新しいジョブ定義を作成するには:

1. AWS Batch コンソールを https://console.aws.amazon.com/batch/ で開きます。
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. 左側のナビゲーションペインで ジョブ定義 を選択します。
4. [作成] を選択します。
5. [オーケストレーションタイプ] には、[Amazon Elastic Compute Cloud (Amazon EC2)] を選択します。
6. [ジョブ定義構造] で、[レガシー containerProperties 構造の使用] の処理をオフにします。
7. EC2 プラットフォームの設定で、マルチノード並列を有効にする をオフにします。
8. [次へ] を選択します。
9. [全般設定] セクションで、次のように入力します。

    a. 名前 に、一意のジョブ定義名を入力します。名前の最大長は 128 文字です。大文字、小文字、数字、ハイフン (-)、アンダースコア (_) を含めることができます。

    b. [実行タイムアウト - オプション] で、タイムアウト値 (秒単位) を入力します。実行タイムアウトは、未完了のジョブが終了するまでの時間です。試行時間がこの秒数を超過すると、試行は停止し、`FAILED` ステータスに移行します。詳細については、ジョブのタイムアウトを参照してください。最小値は 60 秒です。

    c. [スケジューリング優先度 - オプション] をオンにします。0 ~ 100 の値でスケジューリング優先度を入力します。値が大きいほど優先度が高くなります。

    d. [タグ - オプション] を展開し、[タグを追加] を選択してリソースにタグを追加します。キーとオプショナル値を入力し、新しいタグを追加を選択します。

    e. [タグを伝播] をオンにして、タグをジョブとジョブ定義から Amazon ECS タスクに伝達することができます。

10. [再試行戦略 - オプション] セクションで、次のように入力します。

    a. [ジョブの試行回数] には、AWS Batch がジョブを `RUNNABLE` ステータスに移行させるまでの試行回数を入力します。1 ~ 10 の整数を入力します。

    b. [再試行戦略の条件] では、[終了時に評価を追加] を選択します。少なくとも 1 つのパラメータ値を入力し、アクション を選択します。条件セットごとに、アクション を 再試行 または 終了 に設定する必要があります。これらのアクションは、以下のことを意味します。

        - 再試行 — AWS Batch は、指定したジョブ試行回数に達するまで再試行します。
        - 終了 — AWS Batch は、ジョブの再試行を停止します。

> **Important**
>
> 終了時に評価を追加 を選択した場合は、少なくとも 1 つのパラメータを設定して [アクション] を選択するか、[終了時に評価を削除] を選択します。

11. [タスクプロパティ] セクションに、次のように入力します。

    a. [実行ロール - 条件付き] で、Amazon ECS エージェントがユーザーに代わって AWS API コールを実行できるようにするロールを選択します。実行ロールの作成について詳しくは、「チュートリアル: IAM 実行ロールを作成する」を参照してください。

    b. [ECS 実行コマンドを有効化] を選択し、Amazon ECS コンテナシェルへの直接アクセスを有効化し、ホスト OS をバイパスします。[タスクロール] を選択する必要があります。

    > **Important**
    >
    > [ECS 実行] コマンドでは、ファイルシステムが書き込み可能である必要があります。

    c. [タスクロール] で Amazon ECS Identity and Access Management (IAM) ロールを選択し、コンテナがユーザーに代わって AWS API コールを実行できるようにします。詳細については、「Amazon Elastic Container Service デベロッパーガイド」の「Amazon ECS のタスク IAM ロール」を参照してください。

    d. [IPC モード] の場合は `host`、`task`、`none` を選択します。`host` が指定されている場合、同一のコンテナインスタンス上にある (ホスト IPC モードを指定した) タスク内のすべてのコンテナは、ホスト Amazon EC2 インスタンスと同じ IPC リソースを共有します。タスクが指定されている場合、指定されたタスク内のすべてのコンテナは同じ IPC リソースを共有します。none (なし) が指定されている場合、タスクのコンテナ内の IPC リソースはプライベートです。タスク内またはコンテナインスタンスの他のコンテナと共有されることはありません。値を指定しない場合、IPC リソース名前空間の共有はコンテナインスタンスの Docker デーモンの設定によって異なります。

    e. [PID モード] の場合は、`host` または `task` を選択します。例えば、サイドカーのモニタリングでは、`pidMode` が同じタスクで実行されている他のコンテナに関する情報にアクセスする必要がある場合があります。`host` が指定されている場合、同じコンテナインスタンスでホスト PID モードを指定したタスク内のすべてのコンテナは、ホスト Amazon EC2 インスタンスと同じプロセス名前空間を共有します。`task` が指定されている場合、指定したタ

スク内のすべてのコンテナは同じプロセス名前空間を共有します。値が指定されていない場合、デフォルトは各コンテナのプライベート名前空間です。

12. [消費型リソース] セクションに、次のように入力します。

    a. 一意の [名前] と [リクエストされた値] を入力します。

    b. [消費型リソースの追加] を選択すると、消費型リソースを追加できます。

13. [ストレージ] セクションで、以下の操作を行います。

    a. ボリュームの [名前] と [ソースパス] を入力し、[ボリュームを追加] を選択します。[EFS を有効にする] オプションを選択することもできます。

    b. [ボリュームの追加] を選択すると、ボリュームを追加できます。

14. [パラメータ] で [パラメータを追加] を選択し、パラメータ代替プレースホルダーを [キー] とオプションの [値] のペアとして追加します。

15. 次のページ を選択します。

16. コンテナの設定 セクションで次の操作を行います。

    a. [Name] (名前) に、コンテナの名前を入力します。

    b. [必須コンテナ] では、コンテナが必須であれば有効にします。

    c. イメージ で、ジョブに使用する Docker イメージを選択します。デフォルトでは、Docker Hub レジストリのイメージを使用できます。*repository-url*/*image*:*tag* で他のリポジトリを指定することもできます。名前の最大長は 225 文字です。大文字、小文字、数字、ハイフン (-)、アンダースコア (_)、コロン (:)、スラッシュ (/)、およびシャープ (#) を含めることができます。このパラメータは、Docker Remote API の コンテナの作成 セクションにある `Image` および docker run の `IMAGE` パラメータにマッピングされます。

    > **Note**
    >
    > Docker イメージのアーキテクチャは、スケジュールされているコンピューティングリソースのプロセッサアーキテクチャと一致している必要があります。例えば、ARM ベースの Docker イメージは、ARM ベースのコンピューティングリソースでのみ実行することができます。

    - Amazon ECR Public リポジトリ内のイメージには、完全な `registry/repository[:tag]` または `registry/repository[@digest]` 命名規則が使用されます (例えば、`public.ecr.aws/`*`registry_alias`*`/`*`my-web-app`*`:`*`latest`*)。

- Amazon ECR リポジトリ内のイメージには、完全な `registry/repository[:tag]` 命名規則が使用されます (例えば、*aws_account_id*`.dkr.ecr.`*region*`.amazonaws.com/`*my-web-app*`:`*latest*)。
- Docker Hub の公式リポジトリのイメージでは、1 つの名前 (例: `ubuntu` または `mongo`) を使用します。
- Docker Hub の他のリポジトリのイメージは、組織名で修飾されます (例: `amazon/amazon-ecs-agent`)。
- 他のオンラインリポジトリのイメージは、さらにドメイン名で修飾されます (例: `quay.io/assemblyline/ubuntu`)。

d. [リソース要件] には、以下をそれぞれ設定します。

i. [vCPUs] では、コンテナの CPU 数を選択します。

ii. [メモリ] では、コンテナのメモリ量を選択します。

iii. [GPU - オプション] では、コンテナの GPU の数を選択します。

e. [ユーザー] には、コンテナ内で使用するユーザー名を入力します。

f. 読み取り専用ファイルシステムを有効にする をオンにして、ボリュームへの書き込みアクセスを削除します。

g. [特権] をオンにして、ルートユーザーと同様に、ホストインスタンスに対する昇格されたアクセス許可をジョブコンテナに付与します。

h. [コマンド] では、コマンドを JSON 文字列配列に相当するものとしてフィールドに入力します。

このパラメータは、Docker Remote API の コンテナの作成 セクションにある `Cmd` にマッピングされ、`COMMAND` パラメータは docker run にマッピングされます。Docker`CMD` パラメータの詳細については、https://docs.docker.com/engine/reference/builder/#cmdを参照してください。

> **Note**
>
> コマンドには、パラメータ置換のデフォルト値とプレースホルダーを使用できます。詳細については、パラメータ を参照してください。

i. [リポジトリ認証情報 - オプション] では、認証情報を含むシークレットの ARN を入力します。

j. [環境変数 - オプション] では、[環境変数の追加] を選択し、コンテナに渡す環境変数を追加します。

k. [Linux パラメータ - オプション] セクションでは、以下を行います:

i. init プロセスを有効にする をオンにすると、コンテナ内で init プロセスを実行できます。

ii. [共有メモリサイズ] では、/dev/shm ボリュームのサイズ (MiB) を入力します。

iii. 最大スワップサイズ では、コンテナが使用できるスワップメモリの合計容量 (MiB 単位) を入力します。

iv. スワップ動作 では、コンテナのスワップ動作を示す 0 ~ 100 の値を入力します。値を指定せず、スワップが有効になっている場合、この値はデフォルトで 60 に設定されます。

v. デバイス で、デバイスを追加 を選択してデバイスを追加します。

A. コンテナパス] では、コンテナインスタンスでのデバイスのパスを指定します。このパスは、ホストインスタンスにマッピングされたデバイスを公開するために使用されます。空白のままにすると、ホストパスがコンテナで使用されます。

B. ホストパス] では、ホストインスタンスでのデバイスのパスを指定します。

C. アクセス許可] では、デバイスに適用する 1 つ以上のアクセス許可を選択します。使用できる権限は、読み取り、書き込み、MKNOD です。

vi. Tmpfs では、tmpfs を追加 を選択して `tmpfs` マウントを追加します。

l.

> **Note**
>
> Firelens のログ記録は、専用コンテナで行う必要があります。Firelens ログ記録を設定するには:
>
> - 専用 Firelens コンテナを除くすべてのコンテナで、[ロギングドライバー] を `awsfirelens` に設定します。
> - Firelens コンテナで、[Firelens 設定 - オプション] および ログ記録先に対する [ログ記録設定 - オプション] を設定します。

[Firelens 設定 - オプション] セクションで以下を行います。

> **Important**
>
> AWS Batch は、非 MNP、非 FARGATE Amazon ECS ジョブに host ネットワークモードを適用します。Amazon ECS Firelens には ルートユーザーが必要です。host ネットワークモードを使用するタスクを実行する場合、セキュリティを強化するために ルートユーザー (UID 0) を使用してコンテナを実行しないでください。したがって、Firelens ログ記録を使用する非 MNP および非 FARGATE ECS ジョブはすべて、セキュリティのベストプラクティスを満たしません。

i. [タイプ] で、fluentd または fluentbit を選択します。

ii. [オプション] に、オプションの名前と値のペアを入力します。[追加オプション] を使用して [オプション] を追加できます。

m. [ロギング設定 - オプション] で、以下を行います。

i. [ログドライバー] で、使用するログドライバーを選択します。使用できるログドライバーの詳細については、「LogConfiguration:logDriver」を参照してください。

> **Note**
>
> デフォルトでは、awslogs ログドライバーが使用されます。

ii. [オプション] では、[オプションを追加] を選択してオプションを追加します。名前と値のペアを入力し、[オプションを追加] を選択します。

iii. [シークレット] で、[シークレットを追加] を選択します。名前と値のペアを入力し、[シークレットを追加] を選択してシークレットを追加します。

> **Tip**
>
> 詳細については、「LogConfiguration:secretOptions」を参照してください。

n. [マウントポイント - オプション] で [マウントポイントを追加] を選択し、データボリュームにマウントポイントを追加します。ソースボリュームとコンテナパスを指定する必要があります。

o. [シークレット - オプション] で [シークレットを追加] を選択し、シークレットを追加します。名前と値のペアを入力し、シークレットを追加 を選択します。

> **Tip**
>
> 詳細については、「LogConfiguration:secretOptions」を参照してください。

p. [Ulimits - オプション] では、[ulimit を追加] を選択して、コンテナの `ulimits` 値を追加します。名前、ソフトリミット、ハードリミット の値を入力し、ulimit を追加 を選択します。

q. [依存関係 - オプション]で、[コンテナ依存関係の追加] を選択します。コンテナの名前と状態を選択して、このコンテナがいつ起動するかを決定します。

17. コンテナが 1 つしか設定されていない場合は、[コンテナの追加] を選択し、新しいコンテナの設定を完了する必要があります。それ以外の場合は、[次へ] を選択して確認します。

マルチノード並列ジョブ定義を作成する

AWS Batch でジョブを実行する前に、ジョブ定義を作成する必要があります。このプロセスは、シングルノードとマルチノード並列ジョブでは若干異なります。このトピックでは、特に AWS Batch のマルチノード並列ジョブ (ギャングスケジューリングとも呼ばれます) のジョブ定義を作成する方法について説明します。詳細については、「マルチノード並列ジョブ」を参照してください。

> **Note**
>
> AWS Fargate は、マルチノード並列ジョブをサポートしていません。

内容

チュートリアル: Amazon EC2 リソースにマルチノード並列ジョブ定義を作成する

Amazon Elastic Compute Cloud (Amazon EC2) リソースにマルチノード並列ジョブ定義を作成する方法を説明します。

> **Note**
>
> 単一ノードのジョブ定義を作成するには、「Amazon EC2 リソースにシングルノードのジョブ定義を作成する」を参照してください。

Amazon EC2 リソースにマルチノード並列ジョブ定義を作成するには:

1. AWS Batch コンソールを https://console.aws.amazon.com/batch/ で開きます。
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. ナビゲーションペインで、ジョブ定義 を選択します。
4. 作成] を選択します。
5. [オーケストレーションタイプ] には、[Amazon Elastic Compute Cloud (Amazon EC2)] を選択します。
6. [マルチノード並列を有効にする] では、マルチノード並列をオンにします。
7. 名前 には、一意のジョブ定義名を入力します。名前には、アルファベット (大文字、小文字)、数字、ハイフン (-)、アンダースコア (_) を含めることができ、最大 128 文字まで使用可能です。
8. (オプション) 実行のタイムアウト で、ジョブの試行で実行する最大秒数を指定します。試行時間がこの秒数を超過すると、試行は停止し、FAILED ステータスに移行します。詳細については、ジョブのタイムアウトを参照してください。
9. (オプション) [スケジューリング優先度] をオンにします。0 ~ 100 の値でスケジューリング優先度を入力します。値が大きいほど、低い値よりも優先されます。
10. (オプション) ジョブの試行回数 には、AWS Batch ジョブを RUNNABLE ステータスに移行する最大回数を入力します。1 ~ 10 の整数を入力します。
11. (オプション) 再試行戦略の条件 では、終了時に評価を追加 を選択します。少なくとも 1 つのパラメータ値を入力し、アクション を選択します。条件セットごとに、アクション を 再試行 または 終了 に設定する必要があります。これらのアクションは、以下のことを意味します。

    - 再試行 — AWS Batch は、指定したジョブ試行回数に達するまで再試行します。
    - 終了 — AWS Batch は、ジョブの再試行を停止します。

> **Important**
>
> 終了時に評価を追加 を選択した場合は、少なくとも 1 つのパラメータを設定して アクション] を選択するか、終了時に評価を削除] を選択します。

12. (オプション) タグ を展開し、タグを追加 を選択してリソースにタグを追加します。タグの追加 を選択し、キーとオプションの値を入力します。タグを伝播 をオンにして、タグをジョブとジョブ定義から Amazon ECS タスクに伝達することもできます。
13. 次のページ を選択します。
14. ノード数] にジョブで使用するノードの合計数を入力します。
15. 主要なノード] で、主要なノードに使用するノードインデックスを入力します。デフォルトの主要なノードインデックスは、`0` です。
16. インスタンスタイプ でインスタンスのタイプを選択します。

> **Note**
>
> 選択したインスタンスタイプはすべてのノードに適用されます。

17. [パラメータ] で [パラメータを追加] を選択し、パラメータ代替プレースホルダーを [キー] とオプションの [値] のペアとして追加します。
18. [ノード範囲] セクション :

    a. ノード範囲の追加] を選択します。これにより、ノード範囲 セクションが作成されます。

    b. ターゲットノード] で、*range_start*:*range_end* 表記を使用してノードグループの範囲を指定します。

    ジョブに指定したノードに対して 5 つまでのノード範囲を作成できます。ノード範囲はノードに対してインデックス値を使用し、ノードインデックスは 0 から開始します。最終ノードグループの範囲終了インデックス値が、指定したノード数より 1 つ少ないことを確認してください。例えば、10 個のノードを指定し、1 つのノードグループを使用するとします。その場合、終了範囲は 9 にする必要があります。

    c. イメージ で、ジョブに使用する Docker イメージを選択します。デフォルトでは、Docker Hub レジストリのイメージを使用できます。*repository-url*/*image*:*tag* で他のリポジトリを指定することもできます。名前の長さは最大 225 文字です。大文字、小文字、数字、ハイフン (-)、アンダースコア (_)、コロン (:)、スラッシュ (/)、およびシャープ (#) を含

めることができます。このパラメータは、Docker Remote API の コンテナの作成 セクションにある Image および IMAGE の docker run パラメータにマッピングされます。

> **Note**
>
> Docker イメージのアーキテクチャは、スケジュールされているコンピューティングリソースのプロセッサアーキテクチャと一致している必要があります。例えば、ARM ベースの Docker イメージは、ARM ベースのコンピューティングリソースでのみ実行することができます。

- Amazon ECR Public リポジトリ内のイメージには、完全な `registry/repository[:tag]` または `registry/repository[@digest]` 命名規則が使用されます (例えば、`public.ecr.aws/`*`registry_alias`*`/`*`my-web-app`*`:`*`latest`*)。
- Amazon ECR リポジトリ内のイメージには、完全な `registry/repository[:tag]` 命名規則が使用されます。例えば、*`aws_account_id`*`.dkr.ecr.`*`region`*`.amazonaws.com/`*`my-web-app`*`:`*`latest`*
- Docker Hub の公式リポジトリのイメージでは、1 つの名前 (例: ubuntu または mongo) を使用します。
- Docker Hub の他のリポジトリのイメージは、組織名で修飾されます (例: `amazon/amazon-ecs-agent`)。
- 他のオンラインリポジトリのイメージは、さらにドメイン名で修飾されます (例: `quay.io/assemblyline/ubuntu`)。

d. [コマンド] では、コマンドを JSON 文字列配列に相当するものとしてフィールドに入力します。

このパラメータは、Docker Remote API の コンテナの作成 セクションにある Cmd にマッピングされ、COMMAND パラメータは docker run にマッピングされます。DockerCMD パラメータの詳細については、https://docs.docker.com/engine/reference/builder/#cmdを参照してください。

> **Note**
>
> コマンドには、パラメータ置換のデフォルト値とプレースホルダーを使用できます。詳細については、パラメータを参照してください。

e. [vCPU] で、コンテナ用に予約する vCPU の数を指定します。このパラメータは、Docker Remote API のコンテナの作成セクションの `CpuShares` にマップされ、`--cpu-shares` オプションは docker run にマップされます。各 vCPU は 1,024 個の CPU 配分に相当します。少なくとも 1 つの vCPU を指定する必要があります。

f. メモリ] で、ジョブのコンテナに適用されるメモリのハード制限 (MiB 単位) を指定します。コンテナは、ここで指定したメモリを超えようとすると、停止されます。このパラメータは、Docker Remote API のコンテナの作成セクションの `Memory` にマップされ、`--memory` オプションは docker run にマップされます。ジョブに対して少なくとも 4 MiB のメモリを指定する必要があります。

> **Note**
>
> リソースの使用率を最大限に高めるには、特定のインスタンスタイプに対してできるだけ多くのメモリを提供します。詳細については、コンピューティングリソースメモリの管理を参照してください。

g. (オプション) GPU 数 に、ジョブで使用される GPU の数を指定します。ジョブは、指定された数の GPU が固定されているコンテナで実行されます。

h. (オプション) ジョブロール] では、AWS API を使用する権限をジョブのコンテナに付与する IAM ロールを指定できます。この機能では、タスク用の Amazon ECS IAM ロールを使用します。設定の前提条件を含む詳細については、「Amazon Elastic Container Service デベロッパーガイド」の「タスク用の IAM ロール」を参照してください。

> **Note**
>
> Fargate リソースで実行されているジョブには、ジョブロールが必要です。

> **Note**
>
> ここには、Amazon Elastic Container Service Task Role] 信頼関係があるロールのみが表示されます。AWS Batch ジョブの IAM ロールの作成に関する詳細については、Amazon Elastic Container Service デベロッパーガイドのタスク用の IAM ロールとポリシーの作成を参照してください。

i. (オプション) 実行ロール で、ユーザーに代わって AWS API コールを実行するためのアクセス許可を Amazon ECS コンテナエージェントに付与する IAM ロールを指定できま

す。この機能では、タスク用の Amazon ECS IAM ロールを使用します。詳細については、「Amazon Elastic Container Service デベロッパーガイド」の「Amazon ECS タスク実行 IAM ロール」を参照してください。

19. (オプション) 追加設定 を展開します。

    a. 環境変数 では、環境変数を追加 を選択し、環境変数を名前と値のペアとして追加します。これらの変数は、コンテナに渡されます。

    b. (オプション) ジョブロール設定 で、AWS API を使用する権限をジョブのコンテナに付与する IAM ロールを指定できます。この機能では、タスク用の Amazon ECS IAM ロールを使用します。設定の前提条件を含む詳細については、「Amazon Elastic Container Service デベロッパーガイド」の「タスク用の IAM ロール」を参照してください。

    > **Note**
    >
    > Fargate リソースで実行されているジョブには、ジョブロールが必要です。

    > **Note**
    >
    > ここには、Amazon Elastic Container Service Task Role] 信頼関係があるロールのみが表示されます。AWS Batch ジョブの IAM ロールの作成方法に関する詳細については、「Amazon Elastic Container Service デベロッパーガイド」の「タスク用の IAM ロールとポリシーの作成」を参照してください。

    c. 実行ロール で、ユーザーに代わって AWS API コールを実行するためのアクセス許可を Amazon ECS コンテナエージェントに付与する IAM ロールを指定できます。この機能では、タスク用の Amazon ECS IAM ロールを使用します。詳細については、「Amazon Elastic Container Service デベロッパーガイド」の「Amazon ECS タスク実行 IAM ロール」を参照してください。

20. [セキュリティ設定] セクション :

    a. (オプション) 特権 を有効にすると、ホストインスタンスに対する昇格されたアクセス権限 (`root` ユーザーと同様) をジョブのコンテナに付与することができます。このパラメータは、Docker Remote API のコンテナの作成セクションの `Privileged` にマップされ、`--privileged` オプションは docker run にマップされます。

b. (オプション) ユーザー には、コンテナ内で使用するユーザー名を入力します。このパラメータは、Docker Remote API のコンテナの作成セクションの `User` にマップされ、`--user` オプションは docker run にマップされます。

c. (オプション) [シークレット] で、[シークレットを追加] を選択して、シークレットを名前と値のペアとして追加します。これらのシークレットはコンテナに公開されます。詳細については、「LogConfiguration:secretOptions」を参照してください。

21. [Linux 設定] セクション:

a. [読み取り専用ファイルシステムを有効にする] をオンにして、ボリュームへの書き込みアクセスを削除します。

b. (オプション) [init プロセスを有効にする] をオンにして、コンテナ内で `init` プロセスを実行します。このプロセスは信号を転送し、プロセスを利用します。

c. [共有メモリサイズ] では、`/dev/shm` ボリュームのサイズ (MiB) を入力します。

d. 最大スワップサイズ では、コンテナが使用できるスワップメモリの合計容量 (MiB 単位) を入力します。

e. スワップ動作 では、コンテナのスワップ動作を示す 0 ~ 100 の値を入力します。値を指定せず、スワップが有効になっている場合、デフォルト値は 60 です。詳細については、「LinuxParameters:swappiness」を参照してください。

f. (オプション) デバイス では、デバイスを追加 を選択してデバイスを追加します。

i. コンテナパス] では、コンテナインスタンスでのデバイスのパスを指定します。このパスは、ホストインスタンスにマッピングされたデバイスを公開するために使用されます。空白のままにすると、ホストパスがコンテナで使用されます。

ii. ホストパス] では、ホストインスタンスでのデバイスのパスを指定します。

iii. アクセス許可] では、デバイスに適用する 1 つ以上のアクセス許可を選択します。使用できる権限は、[読み取り]、[書き込み]、[MKNOD] です。

22. (オプション) [マウントポイント] で、[マウントポイント構成を追加] を選択し、データボリュームにマウントポイントを追加します。ソースボリュームとコンテナパスを指定する必要があります。これらのマウントポイントは、コンテナインスタンスの Docker デーモンに渡されます。ボリュームを [読み取り専用] にすることもできます。

23. (オプション) [Ulimit 設定] では、[ulimit を追加] を選択して、コンテナの `ulimits` 値を追加します。[名前]、[ソフトリミット]、[ハードリミット] の値を入力し、[ulimit を追加] を選択します。

24. (オプション) [ボリューム設定] で、[ボリュームを追加] を選択して、コンテナに渡すボリュームのリストを作成します。ボリュームの [名前] と [ソースパス] を入力し、[ボリュームを追加] を選択します。[EFS を有効にする] オプションを選択することもできます。
25. (オプション) [Tmpfs] では、[tmpfs を追加] を選択して `tmpfs` マウントを追加します。
26. [タスクプロパティ] セクションで、次の操作を行います。

    a. [実行ロール - 条件付き] で、Amazon ECS エージェントがユーザーに代わって AWS API コールを実行できるようにするロールを選択します。実行ロールの作成について詳しくは、「チュートリアル: IAM 実行ロールを作成する」を参照してください。

    b.

    > **Important**
    >
    > ECS 実行コマンドを使用するには、コンピューティング環境がマルチノード並列ジョブのコンピューティング環境に関する考慮事項を満たす必要があります。

    [ECS 実行コマンドを有効化] を選択し、Amazon ECS コンテナシェルへの直接アクセスを有効化し、ホスト OS をバイパスします。[タスクロール] を選択する必要があります。

    > **Important**
    >
    > [ECS 実行] コマンドでは、ファイルシステムが書き込み可能である必要があります。

    c. [タスクロール] で Amazon ECS Identity and Access Management (IAM) ロールを選択し、コンテナがユーザーに代わって AWS API コールを実行できるようにします。詳細については、「Amazon Elastic Container Service デベロッパーガイド」の「Amazon ECS のタスク IAM ロール」を参照してください。

27. (オプション) [ロギング設定] セクション:

    a. [ログドライバー] で、使用するログドライバーを選択します。使用できるログドライバーの詳細については、「LogConfiguration:logDriver」を参照してください。

    > **Note**
    >
    > デフォルトでは、`awslogs` ログドライバーが使用されます。

b. [オプション] では、[オプションを追加] を選択してオプションを追加します。名前と値のペアを入力し、[オプションを追加] を選択します。

c. [シークレット] で、[シークレットを追加] を選択します。名前と値のペアを入力し、[シークレットを追加] を選択してシークレットを追加します。

> **Tip**
>
> 詳細については、「LogConfiguration:secretOptions」を参照してください。

28. [次のページ] を選択します。
29. ジョブ定義のレビューについては、設定手順を確認してください。変更する必要がある場合は、編集] を選択します。完了したら、ジョブ定義の作成 を選択します。

ContainerProperties を使用するジョブ定義のテンプレート

以下は、1 つのコンテナを含む空のジョブ定義のテンプレートです。このテンプレートを使ってジョブ定義を作成し、それをファイルに保存して AWS CLI `--cli-input-json` オプションで使用できます。これらのパラメータの詳細については、「JobDefinition」を参照してください。

> **Note**
>
> シングルコンテナジョブ定義のテンプレートは、以下の AWS CLI コマンドで生成できます。
>
> ```
> $ aws batch register-job-definition --generate-cli-skeleton
> ```

```
{
    "jobDefinitionName": "",
    "type": "container",
    "parameters": {
        "KeyName": ""
    },
    "schedulingPriority": 0,
    "containerProperties": {
        "image": "",
        "vcpus": 0,
        "memory": 0,
```

```
        "command": [
            ""
        ],
        "jobRoleArn": "",
        "executionRoleArn": "",
        "volumes": [
            {
                "host": {
                    "sourcePath": ""
                },
                "name": "",
                "efsVolumeConfiguration": {
                    "fileSystemId": "",
                    "rootDirectory": "",
                    "transitEncryption": "ENABLED",
                    "transitEncryptionPort": 0,
                    "authorizationConfig": {
                        "accessPointId": "",
                        "iam": "DISABLED"
                    }
                }
            }
        ],
        "environment": [
            {
                "name": "",
                "value": ""
            }
        ],
        "mountPoints": [
            {
                "containerPath": "",
                "readOnly": true,
                "sourceVolume": ""
            }
        ],
        "readonlyRootFilesystem": true,
        "privileged": true,
        "ulimits": [
            {
                "hardLimit": 0,
                "name": "",
                "softLimit": 0
            }
```

```
        ],
        "user": "",
        "instanceType": "",
        "resourceRequirements": [
            {
                "value": "",
                "type": "MEMORY"
            }
        ],
        "linuxParameters": {
            "devices": [
                {
                    "hostPath": "",
                    "containerPath": "",
                    "permissions": [
                        "WRITE"
                    ]
                }
            ],
            "initProcessEnabled": true,
            "sharedMemorySize": 0,
            "tmpfs": [
                {
                    "containerPath": "",
                    "size": 0,
                    "mountOptions": [
                        ""
                    ]
                }
            ],
            "maxSwap": 0,
            "swappiness": 0
        },
        "logConfiguration": {
            "logDriver": "syslog",
            "options": {
                "KeyName": ""
            },
            "secretOptions": [
                {
                    "name": "",
                    "valueFrom": ""
                }
            ]
```

```
        },
        "secrets": [
            {
                "name": "",
                "valueFrom": ""
            }
        ],
        "networkConfiguration": {
            "assignPublicIp": "DISABLED"
        },
        "fargatePlatformConfiguration": {
            "platformVersion": ""
        }
    },
    "nodeProperties": {
        "numNodes": 0,
        "mainNode": 0,
        "nodeRangeProperties": [
            {
                "targetNodes": "",
                "container": {
                    "image": "",
                    "vcpus": 0,
                    "memory": 0,
                    "command": [
                        ""
                    ],
                    "jobRoleArn": "",
                    "executionRoleArn": "",
                    "volumes": [
                        {
                            "host": {
                                "sourcePath": ""
                            },
                            "name": "",
                            "efsVolumeConfiguration": {
                                "fileSystemId": "",
                                "rootDirectory": "",
                                "transitEncryption": "DISABLED",
                                "transitEncryptionPort": 0,
                                "authorizationConfig": {
                                    "accessPointId": "",
                                    "iam": "ENABLED"
                                }
```

```
                        }
                    }
                ],
                "environment": [
                    {
                        "name": "",
                        "value": ""
                    }
                ],
                "mountPoints": [
                    {
                        "containerPath": "",
                        "readOnly": true,
                        "sourceVolume": ""
                    }
                ],
                "readonlyRootFilesystem": true,
                "privileged": true,
                "ulimits": [
                    {
                        "hardLimit": 0,
                        "name": "",
                        "softLimit": 0
                    }
                ],
                "user": "",
                "instanceType": "",
                "resourceRequirements": [
                    {
                        "value": "",
                        "type": "MEMORY"
                    }
                ],
                "linuxParameters": {
                    "devices": [
                        {
                            "hostPath": "",
                            "containerPath": "",
                            "permissions": [
                                "WRITE"
                            ]
                        }
                    ],
                    "initProcessEnabled": true,
```

```
                    "sharedMemorySize": 0,
                    "tmpfs": [
                        {
                            "containerPath": "",
                            "size": 0,
                            "mountOptions": [
                                ""
                            ]
                        }
                    ],
                    "maxSwap": 0,
                    "swappiness": 0
                },
                "logConfiguration": {
                    "logDriver": "awslogs",
                    "options": {
                        "KeyName": ""
                    },
                    "secretOptions": [
                        {
                            "name": "",
                            "valueFrom": ""
                        }
                    ]
                },
                "secrets": [
                    {
                        "name": "",
                        "valueFrom": ""
                    }
                ],
                "networkConfiguration": {
                    "assignPublicIp": "DISABLED"
                },
                "fargatePlatformConfiguration": {
                    "platformVersion": ""
                }
            }
        }
    ]
},
"retryStrategy": {
    "attempts": 0,
    "evaluateOnExit": [
```

```
            {
                "onStatusReason": "",
                "onReason": "",
                "onExitCode": "",
                "action": "RETRY"
            }
        ]
    },
    "propagateTags": true,
    "timeout": {
        "attemptDurationSeconds": 0
    },
    "tags": {
        "KeyName": ""
    },
    "platformCapabilities": [
        "EC2"
    ],
    "eksProperties": {
        "podProperties": {
            "serviceAccountName": "",
            "hostNetwork": true,
            "dnsPolicy": "",
            "containers": [
                {
                    "name": "",
                    "image": "",
                    "imagePullPolicy": "",
                    "command": [
                        ""
                    ],
                    "args": [
                        ""
                    ],
                    "env": [
                        {
                            "name": "",
                            "value": ""
                        }
                    ],
                    "resources": {
                        "limits": {
                            "KeyName": ""
                        },
```

```
                        "requests": {
                            "KeyName": ""
                        }
                    },
                    "volumeMounts": [
                        {
                            "name": "",
                            "mountPath": "",
                            "readOnly": true
                        }
                    ],
                    "securityContext": {
                        "runAsUser": 0,
                        "runAsGroup": 0,
                        "privileged": true,
                        "readOnlyRootFilesystem": true,
                        "runAsNonRoot": true
                    }
                }
            ],
            "volumes": [
                {
                    "name": "",
                    "hostPath": {
                        "path": ""
                    },
                    "emptyDir": {
                        "medium": "",
                        "sizeLimit": ""
                    },
                    "secret": {
                        "secretName": "",
                        "optional": true
                    }
                }
            ]
        }
    }
}
```

[ContainerProperties] のジョブ定義のパラメータ

ContainerProperties を使用するジョブ定義はいくつかの部分に分かれています。

- ジョブ定義名
- ジョブ定義のタイプ
- パラメータ置換プレースホルダーのデフォルト
- ジョブのコンテナプロパティ
- Amazon EKS リソースで実行されるジョブに必要なジョブ定義の Amazon EKS プロパティ
- マルチノード並列ジョブに必要なノードプロパティ
- Fargate リソースで実行されるジョブに必要なプラットフォーム機能
- ジョブ定義のデフォルトタグ伝達の詳細
- ジョブ定義のデフォルト再試行戦略
- ジョブ定義のデフォルトのスケジューリング優先度
- ジョブ定義のデフォルトタグ
- ジョブ定義のデフォルトタイムアウト

目次

ジョブ定義名

`jobDefinitionName`

ジョブ定義の登録時に名前を指定します。名前の最大長は 128 文字です。大文字および小文字の ASCII 文字、数字、ハイフン(-)、アンダースコア(_)を含めることができます。その名前で最初に登録するジョブ定義のリビジョン番号は 1 です。その名前で登録する後続のジョブ定義には、増分のリビジョン番号が付けられます。

タイプ: 文字列

必須: はい

タイプ

`type`

ジョブ定義の登録時にジョブのタイプを指定します。ジョブが Fargate リソースで実行されている場合、`multinode` はサポートされません。マルチノード並列ジョブの詳細については、マルチノード並列ジョブ定義を作成する を参照してください。

タイプ: 文字列

有効な値: `container` | `multinode`

必須: はい

パラメータ

`parameters`

ジョブを送信時に、プレースホルダに代わるパラメータを指定したり、デフォルトのジョブ定義パラメータを上書きすることができます。ジョブ送信リクエストのパラメータは、ジョブ定義のデフォルトよりも優先されます。これにより、同じ形式を使用する複数のジョブに同じジョブ定義を使用できます。送信時にコマンドの値をプログラムで変更することもできます。

タイプ: 文字列間のマッピング

必須: いいえ

ジョブ定義の登録時に、ジョブのコンテナプロパティの command フィールドでパラメータ置換プレースホルダーを使用できます。構文は次のとおりです。

```
"command": [
    "ffmpeg",
    "-i",
    "Ref::inputfile",
    "-c",
    "Ref::codec",
    "-o",
    "Ref::outputfile"
]
```

上の例では、パラメータ置換プレースホルダーとして *Ref::inputfile*、*Ref::codec*、*Ref::outputfile* がコマンドで使用されています。ジョブ定義の parameters オブジェクトで、これらのプレースホルダーのデフォルト値を設定できます。例えば、*Ref::codec* プレースホルダーのデフォルトを設定するには、ジョブ定義で次のように指定します。

```
"parameters" : {"codec" : "mp4"}
```

このジョブ定義を送信すると、実行時にコンテナのコマンドの *Ref::codec* 引数がデフォルト値の mp4 に置き換えられます。

コンテナプロパティ

ジョブ定義を登録する時に、ジョブの配置時にコンテナインスタンス上のDockerデーモンに渡されるコンテナプロパティのリストを指定します。ジョブ定義では、以下のコンテナプロパティを使用できます。単一のノードジョブでは、上記のプロパティはジョブ定義レベルに設定されます。マルチノード並列ジョブでは、コンテナプロパティは各ノードグループごとに ノードプロパティ レベルで設定されます。

command

コンテナに渡されるコマンド。このパラメータは Docker Remote API の コンテナの作成 セクションの Cmd にマッピングされ、COMMAND パラメータは docker run にマッピングされます。Docker CMD パラメータの詳細については、https://docs.docker.com/engine/reference/builder/#cmd を参照してください。

```
"command": ["string", ...]
```

タイプ: 文字列配列

必須: いいえ

`environment`

コンテナに渡す環境変数。このパラメータは、Docker Remote API のコンテナの作成セクションにEnvにマップされ、`--env`オプションはdocker runにマップされます。

> **Important**
>
> 認証情報データなどの機密情報にプレーンテキストの環境変数を使用することはお勧めしません。

> **Note**
>
> 環境変数は `AWS_BATCH` で始まることはできません。この命名規則は、AWS Batch サービスで設定される変数のために予約されています。

タイプ: キーと値のペアの配列

必須: いいえ

`name`

環境変数の名前。

タイプ: 文字列

必須: はい (`environment` を使用する場合)。

`value`

環境変数の値。

タイプ: 文字列

必須: はい (`environment` を使用する場合)。

```
"environment" : [
    { "name" : "envName1", "value" : "envValue1" },
    { "name" : "envName2", "value" : "envValue2" }
]
```

`executionRoleArn`

ジョブ定義の登録時に IAM ロールを指定できます。Amazon ECS コンテナエージェントは、このロールから付与されるアクセス権限を使用して、関連するポリシーに指定されている API アクションをユーザーに代わって呼び出します。Fargateリソース上で実行されるジョブは、実行ロールを提供する必要があります。詳細については、AWS Batch IAM 実行ロールを参照してください。

型: 文字列

必須: いいえ

`fargatePlatformConfiguration`

Fargateリソース上で実行されるジョブのプラットフォーム構成。EC2 リソースで実行されているジョブでは、このパラメータを指定しないでください。

タイプ: FargatePlatformConfiguration オブジェクト

必須: いいえ

`platformVersion`

ジョブに使用される AWS Fargate プラットフォームバージョン、または最近承認されたAWS Fargate プラットフォーム のバージョンを使用するための LATEST。

タイプ: 文字列

デフォルト: LATEST

必須: いいえ

`image`

ジョブの開始に使用するイメージ。この文字列は Docker デーモンに直接渡されます。Docker Hub レジストリのイメージはデフォルトで使用できます。*repository-url*/*image*:*tag* で他のリポジトリを指定することもできます。最大 255 文字の英字 (大文字と小文字)、数字、ハイフ

ン、アンダースコア、コロン、ピリオド、スラッシュ、シャープ記号を使用できます。このパラメータは、Docker Remote APIのコンテナの作成セクションと docker runの Image の IMAGE パラメータにマップされます。

> **Note**
>
> Docker イメージのアーキテクチャは、スケジュールされているコンピューティングリソースのプロセッサアーキテクチャと一致している必要があります。例えば、ARM ベースの Docker イメージは、ARM ベースのコンピューティングリソースでのみ実行することができます。

- Amazon ECR Public リポジトリ内のイメージには、完全な `registry/repository[:tag]` または `registry/repository[@digest]` 命名規則が使用されます (例えば、`public.ecr.aws/`*`registry_alias`*`/`*`my-web-app`*`:`*`latest`*)。
- Amazon ECR リポジトリ内のイメージには、完全な `registry/repository:[tag]` 命名規則が使用されます。例えば、*`aws_account_id`*`.dkr.ecr.`*`region`*`.amazonaws.com/`*`my-web-app`*`:`*`latest`*。
- Docker ハブの公式リポジトリのイメージでは、1 つの名前 (例: ubuntu、mongo) を使用します。
- Docker ハブの他のリポジトリのイメージは、組織名で修飾されます (例: `amazon/amazon-ecs-agent`)。
- 他のオンラインリポジトリのイメージは、さらにドメイン名で修飾されます (例: `quay.io/assemblyline/ubuntu`)。

タイプ: 文字列

必須: はい

`instanceType`

マルチノード並列ジョブに使用するインスタンスタイプ。マルチノード並列ジョブのすべてのノードグループは、同じインスタンスタイプを使用する必要があります。このパラメータは、シングルノード・コンテナ・ジョブやFargateリソース上で実行されるジョブには無効です。

タイプ: 文字列

必須: いいえ

`jobRoleArn`

ジョブ定義の登録時に IAM ロールを指定できます。ジョブコンテナは、このロールから付与されるアクセス権限を使用して、関連するポリシーに指定されている API アクションをユーザーに代わって呼び出します。詳細については、Amazon Elastic Container Service デベロッパーガイドのタスク用の IAM ロールを参照してください。

タイプ: 文字列

必須: いいえ

`linuxParameters`

コンテナに適用される Linux 固有の変更 (デバイスマッピングの詳細など)。

```
"linuxParameters": {
    "devices": [
        {
            "hostPath": "string",
            "containerPath": "string",
            "permissions": [
                "READ", "WRITE", "MKNOD"
            ]
        }
    ],
    "initProcessEnabled": true|false,
    "sharedMemorySize": 0,
    "tmpfs": [
        {
            "containerPath": "string",
            "size": integer,
            "mountOptions": [
                "string"
            ]
        }
    ],
    "maxSwap": integer,
    "swappiness": integer
}
```

タイプ: LinuxParameters オブジェクト

必須: いいえ

`devices`

コンテナにマッピングされたデバイスのリスト。このパラメータは、Docker Remote API のコンテナの作成セクションの `Devices` にマップされ、`--device` オプションは docker run] にマップされます。

> **Note**
> このパラメータは、Fargate リソースで実行されているジョブには適用されません。

タイプ: デバイスオブジェクト配列

必須: いいえ

`hostPath`

ホストコンテナインスタンスで使用可能なデバイスがあるパス。

タイプ: 文字列

必須: はい

`containerPath`

コンテナでデバイスが公開されるパス。指定しない場合、デバイスはホストパスと同じパスで公開されます。

タイプ: 文字列

必須: いいえ

`permissions`

コンテナでのデバイスのアクセス許可。指定しない場合、アクセス許可は `READ`、`WRITE`、`MKNOD` に設定されます。

タイプ: 文字列の配列

必須: いいえ

有効な値: `READ` | `WRITE` | `MKNOD`

`initProcessEnabled`

true の場合、信号を転送し、プロセスを利用するコンテナ内で `init` プロセスを実行します。このパラメータは、docker run] の `--init` オプションにマッピングされます。このパラメータは、コンテナインスタンスで Docker Remote API バージョン 1.25 以上を使用する必要があります。コンテナインスタンスの Docker Remote API のバージョンを確認するには、コンテナインスタンスにログインし、`sudo docker version | grep "Server API version"` コマンドを実行します。

タイプ: ブール値

必須: いいえ

`maxSwap`

ジョブで使用可能なスワップメモリの合計容量 (単位: MiB) を指定します。このパラメータは、docker run の `--memory-swap` オプションに変換されます。値はコンテナメモリの合計に `maxSwap` 値を加えた値です。詳細については、Docker ドキュメントの`--memory-swap` 詳細を参照してください。

`maxSwap` の `0` 値を指定した場合、コンテナはスワップを使用しません。許容値は、`0` または任意の正の整数です。`maxSwap` パラメータが省略された場合、コンテナは実行するコンテナインスタンスのスワップ設定を使用します。`maxSwap` パラメータを使用するには、`swappiness` 値を設定する必要があります。

> **Note**
> このパラメータは、Fargate リソースで実行されているジョブには適用されません。

タイプ: 整数

必須: いいえ

`sharedMemorySize`

`/dev/shm` ボリュームのサイズ値 (MiB) です。このパラメータは、docker run の `--shm-size` オプションにマッピングされます。

> **Note**
> このパラメータは、Fargate リソースで実行されているジョブには適用されません。

タイプ: 整数

必須: いいえ

`swappiness`

これにより、コンテナのメモリスワップ動作を調整できます。`swappiness` 値が `0` の場合、絶対に必要な場合を除きスワップは行われません。`swappiness` の値が `100` の場合は、ページが積極的にスワップされます。使用できる値は、`0` と `100` の間の整数です。`swappiness` パラメータを指定しない場合、デフォルト値の `60` が使用されます。`maxSwap` の値が指定されていない場合、このパラメータは無視されます。`maxSwap` が 0 に設定されている場合、コンテナはスワップを使用しません。このパラメータは、docker run の `--memory-swappiness` オプションにマッピングされます。

コンテナごとのスワップ構成を使用する場合は、次の点を考慮してください。

- スワップ領域を有効にし、コンテナが使用するコンテナインスタンスで割り当てる必要があります。

> **Note**
>
> Amazon ECS 最適化 AMI では、スワップはデフォルトで有効になっていません。この機能を使用するには、インスタンスでスワップを有効に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「インスタンスストアスワップボリューム」または「スワップファイルを使用して、Amazon EC2 インスタンスのスワップ領域として機能するようにメモリを割り当てる方法を教えてください。」を参照してください。

- スワップ領域のパラメータは、EC2 リソースを使用するジョブ定義でのみサポートされます。
- `maxSwap` および `swappiness` パラメータがジョブ定義から省略されている場合、各コンテナの `swappiness` には、デフォルト値の 60 が設定されます。スワップ使用量の合計は、コンテナのメモリ予約の2倍までに制限されます。

> **Note**
>
> このパラメータは、Fargate リソースで実行されているジョブには適用されません。

タイプ: 整数

必須: いいえ

`tmpfs`

tmpfs マウントのコンテナパス、マウントオプション、およびサイズ。

タイプ: Tmpfs オブジェクト配列

> **Note**
> このパラメータは、Fargate リソースで実行されているジョブには適用されません。

必須: いいえ

`containerPath`

tmpfs ボリュームをマウントするコンテナ内のファイルの絶対パス。

タイプ: 文字列

必須: はい

`mountOptions`

tmpfs ボリュームのマウントオプションのリストです。

有効な値:`defaults|ro|rw|suid|nosuid|dev|nodev|exec|noexec|sync|async|dirsync|remount|ma`

タイプ: 文字列の配列

必須: いいえ

`size`

tmpfs ボリュームのサイズ (MiB) です。

タイプ: 整数

必須: はい

`logConfiguration`

ジョブの口グ設定の仕様。

このパラメータは、Docker Remote API のコンテナサービスを作成 セクションの `LogConfig` にマップされ、`--log-driver` オプションは docker run にマップされます。デフォルトでは、コ

ンテナは Docker デーモンで使用されるのと同じロギングドライバーを使用します。ただし、コンテナで Docker デーモンとは異なるログドライバーを使用するには、コンテナの定義内でこのパラメータを使用してログドライバーを指定します。コンテナに異なるロギングドライバーを使用するには、コンテナインスタンスまたはリモートログ記録オプションの別のログサーバーでログシステムを適切に設定する必要があります。サポートされているさまざまなログドライバーのオプションの詳細については、Docker ドキュメントのログドライバーの設定を参照してください。

> **Note**
>
> AWS Batch では現在、Docker デーモンの使用可能なログドライバーがいくつかサポートされています (LogConfigurationデータ型を参照)。

このパラメータは、コンテナインスタンスで Docker Remote API バージョン 1.18 以上を使用する必要があります。コンテナインスタンスの Docker Remote API のバージョンを確認するには、コンテナインスタンスにログインし、`sudo docker version | grep "Server API version"` コマンドを実行します。

```
"logConfiguration": {
    "devices": [
        {
            "logDriver": "string",
            "options": {
                "optionName1" : "optionValue1",
                "optionName2" : "optionValue2"
            }
            "secretOptions": [
              {
                  "name" : "secretOptionName1",
                  "valueFrom" : "secretOptionArn1"
              },
              {
                  "name" : "secretOptionName2",
                  "valueFrom" : "secretOptionArn2"
              }
            ]
        }
    ]
}
```

タイプ: LogConfiguration オブジェクト

必須: いいえ

`logDriver`

ジョブに使用するログドライバー。デフォルトでは、AWS Batch が `awslogs` ログドライバーを有効にします。このパラメータの有効な値は、Amazon ECS コンテナエージェントがデフォルトで通信できるログドライバーです。

このパラメータは、Docker Remote API の コンテナサービスを作成セクションの `LogConfig` にマップされ、`--log-driver` オプションは docker run にマップされます。デフォルトでは、ジョブは Docker デーモンで使用されるのと同じロギングドライバーを使用します。ただし、ジョブで Docker デーモンとは異なるログドライバーを使用するには、コンテナの定義内でこのパラメータを使用してログドライバーを指定します。ジョブに別のログドライバーを指定する場合は、コンピュート環境のコンテナインスタンスでログシステムを設定する必要があります。または、別のログサーバーに設定して、リモートロギングオプションを提供する。サポートされているさまざまなログドライバーのオプションの詳細については、Docker ドキュメントのログドライバーの設定を参照してください。

> **Note**
>
> AWS Batch では現在、Docker デーモンに使用可能なログドライバーがいくつかサポートされています。Amazon ECS コンテナエージェントの今後のリリースで他のログドライバーが追加される可能性があります。

サポートされているログドライバーは `awslogs`、`fluentd`、`gelf`、`json-file`、`journald`、`logentries`、`syslog`、`splunk` です。

> **Note**
>
> Fargateリソース上で実行されるジョブは、`awslogs` および `splunk` のログドライバーに制限されます。

このパラメータは、コンテナインスタンスで Docker Remote API バージョン 1.18 以上を使用する必要があります。コンテナインスタンスの Docker Remote API のバージョンを確認するには、コンテナインスタンスにログインし、`sudo docker version | grep "Server API version"` コマンドを実行します。

> **Note**
>
> Amazon ECS コンテナエージェントは、そのインスタンス上で利用可能なロギングドライバーを環境変数 ECS_AVAILABLE_LOGGING_DRIVERS に登録する必要があります。そうしないと、そのインスタンスに配置されたコンテナはこれらのログ設定オプションを使用できません。詳細については、Amazon Elastic Container Service デベロッパーガイドの「Amazon ECS コンテナエージェントの設定」を参照してください。

`awslogs`

Amazon CloudWatch Logs ロギングドライバーを指定します。詳細については、awslogs ログドライバーを使用する、およびDocker ドキュメントの Amazon CloudWatch Logs ロギングドライバーを参照してください。

`fluentd`

Fluentd ロギングドライバーを指定します。使用方法やオプションなどの詳細については、Docker ドキュメントのFluentd logging driver を参照してください。

`gelf`

Graylog 拡張形式 (GELF) ログ記録ドライバーを指定します。使用方法やオプションなどの詳細については、Docker ドキュメントのGraylog Extended Format logging driver を参照してください。

`journald`

Journald ログ記録ドライバーを指定します。使用方法やオプションなどの詳細については、Docker ドキュメントのJournald logging driver(Journald logging driver) を参照してください。

`json-file`

JSON ファイルログ記録ドライバーを指定します。使用方法やオプションなどの詳細については、Docker ドキュメントのJSON File logging driver(JSON ファイルlogging driver) を参照してください。

`splunk`

Splunk ログ記録ドライバーを指定します。使用方法やオプションなどの詳細については、Docker ドキュメントのSplunk logging driver を参照してください。

`syslog`

syslog ログ記録ドライバーを指定します。使用方法やオプションなどの詳細については、Docker ドキュメントのSyslog logging driver(Syslog logging driver) を参照してください。

タイプ: 文字列

必須: はい

有効な値: `awslogs` | `fluentd` | `gelf` | `journald` | `json-file` | `splunk` | `syslog`

> **Note**
>
> 前述以外のカスタムドライバーを Amazon ECS コンテナエージェントで使用する場合は、GitHub で入手できる Amazon ECS コンテナエージェントプロジェクトをフォークし、そのドライバーを使用できるようにカスタマイズできます。含めたい変更については、プルリクエストを送信することをお勧めします。ただし、現在、Amazon Web Services では、このソフトウェアの変更されたコピーの実行をサポートしていません。

`options`

ジョブのログドライバーに送信するログ設定オプション。

このパラメータは、コンテナインスタンスで Docker Remote API バージョン 1.19 以上を使用する必要があります。

タイプ: 文字列間のマッピング

必須: いいえ

`secretOptions`

ログ設定に渡すシークレットを示すオブジェクト。詳細については、機密データを指定するを参照してください。

タイプ: オブジェクト配列

必須: いいえ

`name`

ジョブで設定するログドライバーオプションの名前。

タイプ: 文字列

必須: はい

`valueFrom`

コンテナのログ設定に公開するシークレットのAmazon Resource Name (ARN)。サポートされている値は、Secrets Manager シークレットの完全な ARN または SSN パラメータストア内のパラメータの完全な ARN のいずれかです。

> **Note**
>
> 起動するタスクと同じリージョンに SSM パラメータストアのパラメータが存在する場合は、パラメータの完全な ARN または名前のどちらも使用できます。パラメータが別のリージョンに存在する場合は、完全な ARN を指定する必要があります。

タイプ: 文字列

必須: はい

`memory`

このパラメータは廃止されました。代わりに *resourceRequirements* を使用してください。

ジョブ用に予約されたメモリの MiB 数。

resourceRequirements の使い方の一例として、ジョブ定義に次のような行が含まれている場合は、次のようになります。

```
"containerProperties": {
  "memory": 512
}
```

resourceRequirementsを使用した同等のラインは以下の通りです。

```
"containerProperties": {
```

```
  "resourceRequirements": [
    {
      "type": "MEMORY",
      "value": "512"
    }
  ]
}
```

タイプ: 整数

必須: はい

mountPoints

コンテナでのデータボリュームのマウントポイント。このパラメータは、Docker Remote APIのコンテナの作成セクションにVolumesにマップされ、--volumeオプションはdocker runにマップされます。

```
"mountPoints": [
                {
                  "sourceVolume": "string",
                  "containerPath": "string",
                  "readOnly": true|false
                }
              ]
```

タイプ: オブジェクト配列

必須: いいえ

sourceVolume

マウントするボリュームの名前。

タイプ: 文字列

必須: はい (mountPoints を使用する場合)。

containerPath

ホストボリュームをマウントするコンテナ上のパス。

タイプ: 文字列

必須: はい (mountPoints を使用する場合)。

`readOnly`

この値が `true` の場合、コンテナはボリュームへの読み取り専用アクセスを許可されます。この値が `false` の場合、コンテナはボリュームに書き込むことができます。

タイプ: ブール値

必須: いいえ

デフォルト: False

`networkConfiguration`

Fargate リソースで実行されているジョブのネットワーク構成。EC2 リソースで実行されているジョブでは、このパラメータを指定しないでください。

```
"networkConfiguration": {
   "assignPublicIp": "string"
}
```

タイプ: オブジェクト配列

必須: いいえ

`assignPublicIp`

ジョブにパブリック IP アドレスがあるかどうかを示します。これは、ジョブがアウトバウンドネットワークアクセスが必要な場合に必要です。

タイプ: 文字列

有効な値: `ENABLED` | `DISABLED`

必須: いいえ

デフォルト: `DISABLED`

`privileged`

このパラメータが true のとき、コンテナには、ホストコンテナインスタンスに対する昇格されたアクセス許可 (`root` ユーザーと同様) が付与されます。このパラメータは、Docker Remote API のコンテナの作成セクションの `Privileged` にマップされ、`--privileged` オプションは docker run にマップされます。このパラメータは、Fargate リソースで実行されているジョブには適用されません。これを指定したり、false と指定したりしないでください。

```
"privileged": true|false
```

タイプ: ブール値

必須: いいえ

`readonlyRootFilesystem`

このパラメータが true のとき、コンテナはそのルートファイルシステムへの読み取り専用アクセスを許可されます。このパラメータは、Docker Remote API の コンテナの作成セクションの `ReadonlyRootfs` にマップされ、`--read-only` オプションは docker run にマップされます。

```
"readonlyRootFilesystem": true|false
```

タイプ: ブール値

必須: いいえ

`resourceRequirements`

コンテナに割り当てるリソースのタイプと量。サポートされているリソースには `GPU`、`MEMORY` および `VCPU` などがあります。

```
"resourceRequirements" : [
  {
    "type": "GPU",
    "value": "number"
  }
]
```

タイプ: オブジェクト配列

必須: いいえ

`type`

コンテナに割り当てるリソースのタイプ。サポートされているリソースには `GPU`、`MEMORY` および `VCPU` などがあります。

タイプ: 文字列

必須: はい (`resourceRequirements` を使用する場合)。

`value`

コンテナ用に予約する指定されたリソースの量。値は、指定された `type` によって異なります。

タイプ =GPU

コンテナ用に予約する物理 GPU の数。ジョブ内のすべてのコンテナ用に予約されているGPUの数は、ジョブが起動されたコンピューティングリソースで使用できる GPU の数以下である必要があります。

タイプ =MEMORY

コンテナに適用されるメモリのハード制限 (MiB 単位)。コンテナは、ここで指定したメモリを超えようとすると、強制終了されます。このパラメータは、Docker Remote API のコンテナサービスを作成 セクションの `Memory` にマップされ、`--memory` オプションは docker run にマップされます。ジョブに対して少なくとも 4 MiB のメモリを指定する必要があります。これは必須ですが、マルチノード並列 (MNP) ジョブでは複数の場所で指定できます。各ノードに少なくとも 1 回指定する必要があります。このパラメータは、Docker Remote API のコンテナサービスを作成 セクションの `Memory` にマップされ、`--memory` オプションは docker run にマップされます。

> **Note**
>
> 特定のインスタンスタイプに対してできるだけ多くのメモリを提供してリソースの使用率を最大限に高めるには、コンピューティングリソースメモリの管理を参照してください。

Fargate リソースで実行されているジョブの場合、`value` はサポートされている値の 1 つと一致する必要があります。さらに、`VCPU` の値は、そのメモリ値でサポートされている値の 1 つである必要があります。

VCPU	MEMORY
0.25 vCPU	512、1024、2048 MiB
0.5 vCPU	1024-4096 MiB、1024 MiB 単位
1 vCPU	2048-8192 MiB、1024 MiB 単位

VCPU	MEMORY
2 vCPU	4096-16384 MiB、1024 MiB 単位
4 vCPU	8192-30720 MiB、1024 MiB 単位
8 vCPU	16384-61440 MiB、4096 MiB 単位
16 vCPU	32768-122880 MiB、8192 MiB 単位

タイプ = vCPU

ジョブ用に予約された vCPU の数。このパラメータは、Docker Remote API のコンテナサービスを作成 セクションの CpuShares にマップされ、--cpu-shares オプションは docker run にマップされます。各 vCPU は 1,024 個の CPU 配分に相当します。EC2 リソースを実行しているジョブの場合、少なくとも 1 つの vCPU を指定する必要があります。これは必須ですが、複数の場所で指定できます。各ノードに少なくとも 1 回指定する必要があります。

Fargate リソースで実行されているジョブの場合、value はサポートされている値のいずれかに一致し、MEMORY 値は VCPU 値でサポートされている値のいずれかである必要があります。サポートされている値は 0.25、0.5、1、2、4、8、および 16 です。

デフォルトでは、Fargate オンデマンド vCPU リソース数のクォータは 6 vCPUs です。Fargate Quotas の詳細については、「Amazon Web Services 全般のリファレンス」の「AWS Fargate quotas」を参照してください。

タイプ: 文字列

必須: はい (resourceRequirements を使用する場合)。

secrets

環境変数として公開されるジョブのシークレット。詳細については、機密データを指定するを参照してください。

```
"secrets": [
    {
      "name": "secretName1",
      "valueFrom": "secretArn1"
```

```
    },
    {
      "name": "secretName2",
      "valueFrom": "secretArn2"
    }
    ...
]
```

タイプ: オブジェクト配列

必須: いいえ

`name`

シークレットを含む環境変数の名前。

タイプ: 文字列

必須: はい (`secrets` を使用する場合)。

`valueFrom`

コンテナに公開するシークレット。サポートされている値は、Secrets Managerシークレットの完全なAmazon リソースネーム (ARN)、または SSMパラメータストアのパラメータの完全なARNです。

> **Note**
>
> 起動するジョブと同じリージョンに SSM Parameter Store の パラメータが存在する場合は、パラメータの完全な ARN または名前のどちらも使用できます。パラメータが別のリージョンに存在する場合は、完全な ARN を指定する必要があります。

タイプ: 文字列

必須: はい (`secrets` を使用する場合)。

`ulimits`

コンテナで設定する `ulimits` 値のリスト。このパラメータは、Docker Remote API の コンテナの作成セクションの `Ulimits` にマップされ、`--ulimit` オプションは docker run にマップされます。

```
"ulimits": [
  {
    "name": string,
    "softLimit": integer,
    "hardLimit": integer
  }
  ...
]
```

タイプ: オブジェクト配列

必須: いいえ

name

ulimit の type。

タイプ: 文字列

必須: はい (ulimits を使用する場合)。

hardLimit

ulimit タイプのハード制限。

タイプ: 整数

必須: はい (ulimits を使用する場合)。

softLimit

ulimit タイプのソフト制限。

タイプ: 整数

必須: はい (ulimits を使用する場合)。

user

コンテナ内で使用するユーザー名。このパラメータは、Docker Remote API の コンテナの作成セクションの User にマップされ、--user オプションは docker run にマップされます。

```
"user": "string"
```

タイプ: 文字列

必須: いいえ

vcpus

このパラメータは廃止されました。代わりに *resourceRequirements* を使用してください。

コンテナ用に予約された vCPU の数。

resourceRequirements の使い方の一例として、ジョブ定義に次のような行が含まれている場合は、次のようになります。

```
"containerProperties": {
  "vcpus": 2
}
```

resourceRequirements を使用した同等のラインは以下の通りです。

```
"containerProperties": {
  "resourceRequirements": [
    {
      "type": "VCPU",
      "value": "2"
    }
  ]
}
```

タイプ: 整数

必須: はい

volumes

ジョブ定義の登録時に、コンテナインスタンスの Docker デーモンに渡すボリュームのリストを指定できます。コンテナプロパティでは、以下のパラメータを使用できます。

```
"volumes": [
  {
    "name": "string",
```

```
    "host": {
      "sourcePath": "string"
    },
    "efsVolumeConfiguration": {
      "authorizationConfig": {
        "accessPointId": "string",
        "iam": "string"
      },
      "fileSystemId": "string",
      "rootDirectory": "string",
      "transitEncryption": "string",
      "transitEncryptionPort": number
    }
  }
]
```

name

ボリュームの名前。最大 255 文字の英字 (大文字と小文字)、数字、ハイフン、アンダースコアを使用できます。この名前は、コンテナ定義 sourceVolume の mountPoints パラメータで参照されます。

タイプ: 文字列

必須: いいえ

host

host パラメータの内容により、データボリュームがホストコンテナインスタンスで保持されるかどうか、保持される場合はその場所が決まります。host パラメータが空の場合は、Docker デーモンによってデータボリュームのホストパスが割り当てられます。ただし、関連付けられたコンテナが実行を停止した後も、データが保持されるという保証はありません。

> **Note**
>
> このパラメータは、Fargate リソースで実行されているジョブには適用されません。

タイプ: オブジェクト

必須: いいえ

`sourcePath`

コンテナに渡されるホストコンテナインスタンス上のパス。このパラメータが空の場合は、Docker デーモンによってホストパスが割り当てられます。

host パラメータに `sourcePath` の場所が含まれている場合、データボリュームは手動で削除するまでホストコンテナインスタンスの指定された場所に保持されます。`sourcePath` の値がホストコンテナインスタンスに存在しない場は、Docker デーモンによって作成されます。その場所が存在する場合は、ソースパスフォルダの内容がエクスポートされます。

タイプ: 文字列

必須: いいえ

`efsVolumeConfiguration`

このパラメータは、タスクストレージに Amazon Elastic File System を使用している場合に指定します。詳細については、Amazon EFS ボリュームを参照してください。

タイプ: オブジェクト

必須: いいえ

`authorizationConfig`

Amazon EFS ファイルシステムに対する認可構成の詳細。

型: 文字列

必須: いいえ

`accessPointId`

使用する Amazon EFS アクセスポイントの ID。アクセスポイントを指定した場合、`EFSVolumeConfiguration` で指定されているルートディレクトリの値を省略するか、`/` に設定する必要があります。これにより、EFS アクセスポイントに設定されたパスが強制されます。アクセスポイントを使用する場合は、`EFSVolumeConfiguration` で転送中の暗号化を有効にする必要があります。詳細については、Amazon Elastic ファイルシステムユーザーガイドの Amazon EFS アクセスポイントの使用を参照してください。

タイプ: 文字列

必須: いいえ

`iam`

Amazon EFS ファイルシステムのマウント時にジョブ定義で定義した AWS Batch ジョブの IAM ロールを使用するかどうかを指定します。使用する場合は、`EFSVolumeConfiguration` で転送中の暗号化を有効にする必要があります。このパラメータを省略すると、`DISABLED` のデフォルト値が使用されます。詳細については、Amazon EFS アクセスポイントの使用を参照してください。

タイプ: 文字列

有効な値: `ENABLED` | `DISABLED`

必須: いいえ

`fileSystemId`

使用する Amazon EFS ファイルシステムの ID。

型: 文字列

必須: いいえ

`rootDirectory`

ホスト内にルートディレクトリとしてマウントする Amazon EFS ファイルシステム内のディレクトリ。このパラメータを省略すると、Amazon EFS ボリュームのルートが使用されます。`/` を指定すると、このパラメータを省略した場合と同じ結果になります。最大長は 4,096 文字です。

> **Important**
>
> `authorizationConfig` に EFS アクセスポイントを指定する場合は、ルートディレクトリパラメータを省略するか、または `/` に設定する必要があります。これにより、Amazon EFS アクセスポイントに設定されているパスが強制されます。

タイプ: 文字列

必須: いいえ

`transitEncryption`

Amazon ECS ホストと Amazon EFS サーバー間で Amazon EFS データの転送中の暗号化を有効にするかどうかを指定します。Amazon EFS IAM 認可を使用する場合は、転送中の暗号化を有効にする必要があります。このパラメータを省略すると、`DISABLED` のデフォルト値が使用されます。詳細については、Amazon Elastic File System ユーザーガイドのEncrypting data in transitを参照してください。

タイプ: 文字列

有効な値: `ENABLED` | `DISABLED`

必須: いいえ

`transitEncryptionPort`

Amazon ECS ホストと Amazon EFS サーバーとの間で、暗号化されたデータを送信するときに使用するポート。転送中の暗号化ポートを指定しないと、Amazon EFS マウントヘルパーが使用するポート選択方式が使用されます。この値は 0 ~ 65,535 の範囲の値にする必要があります。詳細については、Amazon Elastic File System User Guide] (Amazon Elastic File System ユーザーガイド) のEFS Mount Helper] (EFS マウントヘルパー) を参照してください。

タイプ: 整数

必須: いいえ

Amazon EKS プロパティ

Amazon EKS ベースのジョブに固有のさまざまなプロパティが用意されたオブジェクト。これは、Amazon ECSベースのジョブ定義には指定しないでください。

`podProperties`

ジョブのKubernetesポッドリソースのプロパティ。

タイプ:EKSPodProperties オブジェクト

必須: いいえ

`containers`

Amazon EKS ポッドで使用されるコンテナのプロパティ。

タイプ:EKS コンテナオブジェクト

必須: いいえ

`args`

エントリポイントへの引数の配列。これを指定しない場合、コンテナイメージの CMD が使用されます。これは Kubernetes のポッドのエントリポイント部分の `args` メンバーに対応します。環境変数の参照は、コンテナの環境を使用して展開されます。

参照先の環境変数が存在しない場合、コマンド内の参照は変更されません。例えば、参照先が$(NAME1)で、NAME1 環境変数が存在しない場合、コマンド文字列は$(NAME1)のままになります。$$ が $ に置き換えられ、結果の文字列は展開されません。例えば、$$(VAR_NAME) は VAR_NAME 環境変数が存在するかどうかに関係なく、$(VAR_NAME) として渡されます。詳細については、「Dockerfile リファレンス」の「CMD」と「Kubernetes ドキュメンテーション」の「ポッドのコマンドと引数を定義する」を参照してください。

タイプ: 文字列の配列

必須: いいえ

`command`

コンテナのエントリポイント。これはシェル内では実行されません。これを指定しない場合、コンテナイメージの ENTRYPOINT が使用されます。環境変数の参照は、コンテナの環境を使用して展開されます。

参照先の環境変数が存在しない場合、コマンド内の参照は変更されません。例えば、参照先が$(NAME1)で、NAME1 環境変数が存在しない場合、コマンド文字列は$(NAME1)のままになります。$$ が $ に置き換えられ、結果の文字列は展開されません。例えば、$$(VAR_NAME) は VAR_NAME 環境変数が存在するかどうかに関係なく、$(VAR_NAME) として渡されます。エントリポイントは更新できません。詳細については、「Dockerfile リファレンス」の「エントリーポイント」、「コンテナのコマンドと引数を定義する」、「Kubernetes ドキュメント」の「エントリーポイント」を参照してください。

タイプ: 文字列の配列

必須: いいえ

`env`

コンテナに渡す環境変数。

> **Note**
>
> 環境変数はAWS_BATCHで始まることはできません。この命名規則は、AWS Batch で設定される変数のために予約されています。

タイプ:EKS コンテナ環境変数オブジェクトの配列

必須: いいえ

`name`

環境変数の名前。

タイプ: 文字列

必須: はい

`value`

環境変数の値。

タイプ: 文字列

必須: いいえ

`image`

コンテナの開始に使用する Docker イメージ。

タイプ: 文字列

必須: はい

`imagePullPolicy`

コンテナのイメージプルポリシー。サポートされている値は `Always`、`IfNotPresent`、`Never` です。このパラメータのデフォルトは `IfNotPresent` です。ただし、`:latest` タグが指定されている場合、デフォルトは `Always` です。詳細については、「Kubernetes ドキュメント」の「イメージの更新」を参照してください。

タイプ: 文字列

必須: いいえ

`name`

コンテナの名前。名前が指定されなかった場合、デフォルト名`Default`が使用されます。ポッド内の各コンテナには一意の名前が必要です。

タイプ: 文字列

必須: いいえ

`resources`

コンテナに割り当てるリソースのタイプと量。サポートされているリソースには `memory`、`cpu` および `nvidia.com/gpu` などがあります。詳細については、「Kubernetes ドキュメント」の「ポッドとコンテナのリソース管理」を参照してください。

タイプ:EKS コンテナリソース要件 オブジェクト

必須: いいえ

`limits`

コンテナ用に予約する指定されたリソースのタイプと量。値は、指定された `name` によって異なります。リソースは、`limits` または `requests` オブジェクトを使用してリクエストできます。

メモリ

コンテナ用のメモリのハード制限 (MiB)。整数とMiサフィックスを使用します。コンテナが指定されたメモリを超えようとすると、強制終了されます。ジョブに対して少なくとも 4 MiB のメモリを指定する必要があります。`limits`、`requests`、または両方で `memory` を指定できます。`memory` が両方の場所で指定されている場合、`limits` で指定される値は、`requests` で指定されている値と等しくなければなりません。

> **Note**
>
> リソースの使用率を最大限に高めるには、使用している特定のインスタンスタイプに対して可能な限り多くのメモリをジョブに割り当てます。この方法の詳細は、コンピューティングリソースメモリの管理を参照してください。

cpu

コンテナ用に予約された CPU の数。値は 0.25 の偶数の乗数でなければなりません。limits、requests、または両方で cpu を指定できます。cpu が両方の場所で指定されている場合、limits で指定される値は、requests で指定されている値以上でなければなりません。

nvidia.com/gpu

コンテナ用に予約された GPU の数。値は整数でなければなりません。memory は limits、requests、または両方で指定できます。memory が両方の場所で指定されている場合、limits で指定される値は、requests で指定されている値と等しくなければなりません。

タイプ: 文字列間のマッピング

値の長さの制限: 最小長は 1。最大長は 256 です。

必須: いいえ

requests

コンテナのリクエストに対するリソースのタイプと量。値は、指定された name によって異なります。リソースは、limits または requests オブジェクトを使用してリクエストできます。

メモリ

コンテナ用のメモリのハード制限 (MiB)。整数とMiサフィックスを使用します。コンテナが指定されたメモリを超えようとすると、強制終了されます。ジョブに対して少なくとも 4 MiB のメモリを指定する必要があります。limits、requests、または両方で memory を指定できます。memory が両方で指定されている場合、requests で指定される値は、limits で指定されている値と等しくなければなりません。

> **Note**
>
> 特定のインスタンスタイプに対してできるだけ多くのメモリを提供してリソースの使用率を最大限に高めるには、コンピューティングリソースメモリの管理を参照してください。

cpu

コンテナ用に予約された CPU の数。値は 0.25 の偶数の乗数でなければなりません。limits、requests、または両方で cpu を指定できます。cpu が両方で指定されている場合、limits で指定される値は、requests で指定されている値以上でなければなりません。

nvidia.com/gpu

コンテナ用に予約された GPU の数。値は整数でなければなりません。nvidia.com/gpu は limits、requests、または両方で指定できます。nvidia.com/gpu が両方で指定されている場合、requests で指定される値は、limits で指定されている値と等しくなければなりません。

タイプ: 文字列間のマッピング

値の長さの制限: 最小長は 1。最大長は 256 です。

必須: いいえ

securityContext

ジョブのセキュリティコンテキスト。詳細については、「Kubernetes ドキュメント」の「ポッドまたはコンテナのセキュリティコンテキストを設定する」を参照してください。

タイプ:EKS コンテナセキュリティコンテキスト オブジェクト

必須: いいえ

privileged

このパラメータが true のとき、コンテナには、ホストコンテナインスタンスに対する昇格されたアクセス許可が付与されます。パーミッションのレベルはrootユーザーのパーミッションと同じです。デフォルト値は false です。このパラメータは、「Kubernetes ドキュメント」の「特権ポッドのセキュリティポリシー」の privileged ポリシーに対応しています。

タイプ: ブール値

必須: いいえ

readOnlyRootFilesystem

このパラメータが true のとき、コンテナはそのルートファイルシステムへの読み取り専用アクセスを許可されます。デフォルト値は false です。このパラメータは、

「Kubernetes ドキュメント」の「ボリュームとファイルシステムのポッドセキュリティポリシー」の ReadOnlyRootFilesystem ポリシーに対応しています。

タイプ: ブール値

必須: いいえ

runAsGroup

このパラメータを指定すると、コンテナは指定されたグループ ID (gid) で実行されます。このパラメータを指定しない場合、デフォルトはイメージメタデータに指定されているグループです。このパラメータは、「Kubernetes ドキュメント」の「ユーザーとグループのポッドセキュリティポリシー」の RunAsGroup および MustRunAs ポリシーに対応しています。

タイプ: Long

必須: いいえ

runAsNonRoot

このパラメータを指定すると、コンテナは 0 以外の uid のユーザーとして実行されます。このパラメータを指定しない場合、そのようなルールが適用されます。このパラメータは、「Kubernetes ドキュメント」の「ユーザーとグループのポッドセキュリティポリシー」の RunAsUser および MustRunAsNonRoot ポリシーに対応しています。

タイプ: Long

必須: いいえ

runAsUser

このパラメータを指定すると、コンテナは指定されたユーザー ID (uid) で実行されます。このパラメータを指定しない場合、デフォルトはイメージメタデータに指定されているユーザーです。このパラメータは、「Kubernetes ドキュメント」の「ユーザーとグループのポッドセキュリティポリシー」の RunAsUser および MustRanAs ポリシーに対応しています。

タイプ: Long

必須: いいえ

`volumeMounts`

Amazon EKS ジョブのコンテナのボリュームマウント。Kubernetes のボリュームとボリュームマウントの詳細については、「Kubernetes ドキュメント」の「ボリューム」を参照してください。

タイプ:EKS コンテナボリュームマウント オブジェクトの配列

必須: いいえ

`mountPath`

ボリュームをマウントするコンテナ上のパス。

タイプ: 文字列

必須: いいえ

`name`

ボリュームマウントの名前。これは、ポッド内のいずれかのボリュームの名前と一致する必要があります。

タイプ: 文字列

必須: いいえ

`readOnly`

この値が `true` の場合、コンテナはボリュームへの読み取り専用アクセスを許可されます。それ以外の場合、コンテナはボリュームに書き込むことができます。デフォルト値は `false` です。

タイプ: ブール値

必須: いいえ

`dnsPolicy`

ポッドの DNS ポリシー。デフォルト値は `ClusterFirst` です。`hostNetwork` パラメータが指定されていない場合、デフォルトは `ClusterFirstWithHostNet` です。`ClusterFirst` は、設定されたクラスタドメインサフィックスと一致しない DNS クエリが、ノードから継承されたアップストリームのネームサーバーに転送されることを示します。[RegisterJobDefinition] API 操作で `dnsPolicy` に値が指定されなかった場合、[DescribeJobDefinitions] または [DescribeJobs] API 操作のいずれによっても

dnsPolicy に値は返されません。ポッド仕様設定には、hostNetwork パラメータの値に応じて、ClusterFirst または ClusterFirstWithHostNet のいずれかが含まれます。詳細については、「Kubernetes ドキュメント」の「ポッドの DNS ポリシー」を参照してください。

有効な値: Default | ClusterFirst | ClusterFirstWithHostNet

タイプ: 文字列

必須: いいえ

hostNetwork

ポッドがホストのネットワーク IP アドレスを使用するかどうかを示します。デフォルト値は true です。これを に設定すると、Kubernetes ポッドネットワークモデルが有効になります。ほとんどの AWS Batch ワークロードは下り専用で、受信接続の各ポッドに IP を割り当てる際のオーバーヘッドは必要ありません。詳細については、「Kubernetes ドキュメント」の「ホストの名前空間」と「ポッドネットワーキング」を参照してください。

タイプ: ブール値

必須: いいえ

serviceAccountName

ポッドを実行するために使用されるサービスアカウントの名前。詳細については、「Amazon EKS ユーザーガイド」の「Kubernetes サービスアカウント」と「IAM ロールを引き継ぐように Kubernetes サービスアカウントを設定する」、「Kubernetes ドキュメント」の「ポッドでのサービスアカウントを設定する」を参照してください。

タイプ: 文字列

必須: いいえ

volumes

Amazon EKS リソースを使用するジョブ定義のボリュームを指定します。

タイプ:EKS ボリュームオブジェクト の配列

必須: いいえ

EmptyDir

Kubernetes emptyDirボリュームの設定を指定します。emptyDir ボリュームは、ポッドがノードに割り当てられるときに最初に作成されます。そのポッドがノードで実行され

ている限り存在します。emptyDir ボリュームは最初は空です。ポッド内のすべてのコンテナは、emptyDir ボリューム内のファイルを読み書きできます。ただし、emptyDir ボリュームは各コンテナの同じパスまたは異なるパスにマウントできます。何らかの理由でポッドがノードから削除されると、emptyDir のデータは完全に削除されます。詳細については、「Kubernetes ドキュメント」の「emptyDir」を参照してください。

タイプ:ekSemptyDir オブジェクト

必須: いいえ

medium

ボリュームを保存するメディア。デフォルト値は、ノードのストレージを使用する空の文字列です。

""

(デフォルト) ノードのディスクストレージを使用します。

メモリ

ノードの RAM にバックアップされている tmpfs ボリュームを使用します。ノードが再起動するとボリュームの内容は失われ、ボリューム上のストレージはコンテナのメモリ制限に対して計算されます。

タイプ: 文字列

必須: いいえ

サイズ制限

ボリュームの最大サイズ。デフォルトでは、最大サイズは定義されていません。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

必須: いいえ

hostPath

Kubernetes hostPathボリュームの設定を指定します。hostPath ボリュームは、ホストノードのファイルシステムから既存のファイルまたはディレクトリをポッドにマウントします。詳細については、「Kubernetes ドキュメント」の「hostPath」を参照してください。

タイプ:EKShostPath オブジェクト

必須: いいえ

パス

ポッドのコンテナにマウントするホスト上のファイルまたはディレクトリのパス。

タイプ: 文字列

必須: いいえ

名前

ボリュームの名前。名前は DNS サブドメイン名として許可さ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「DNS サブドメイン名」を参照してください。

タイプ: 文字列

必須: はい

シークレット

Kubernetes `secret`ボリュームの設定を指定します。詳細については、「Kubernetes ドキュメント」の「シークレット」を参照してください。

タイプ:EKS シークレット オブジェクト

必須: いいえ

オプション

シークレットとシークレットのキーのどちらを定義する必要があるかを指定します。

タイプ: ブール値

必須: いいえ

secretName

シークレットの名前。名前は DNS サブドメイン名として許可さ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「DNS サブドメイン名」を参照してください。

タイプ: 文字列

必須: はい

プラットフォーム機能

`platformCapabilities`

ジョブ定義に必要なプラットフォーム機能。値が指定されていない場合、デフォルトは EC2 になります。Fargate リソースで実行されるジョブの場合、FARGATE が指定されている。

> **Note**
> ジョブを Amazon EKS リソースで実行する場合は、`platformCapabilities` を指定してはならず、

タイプ: 文字列

有効な値: EC2 | FARGATE

必須: いいえ

タグの伝播

`propagateTags`

タグをジョブまたはジョブ定義から対応する Amazon ECS タスクに伝播するかどうかを指定します。値を指定しない場合、タグは伝播されません。タグは、タスクの作成時にのみタスクに伝播できます。同じ名前のタグの場合、ジョブタグはジョブ定義タグよりも優先されます。ジョブとジョブ定義から合わせたタグの合計数が 50 を超える場合、ジョブは FAILED 状態に移行します。

> **Note**
> ジョブを Amazon EKS リソースで実行する場合は、`propagateTags` を指定してはならず、

タイプ: ブール値

必須: いいえ

ノードプロパティ

`nodeProperties`

マルチノード並列ジョブ定義を登録する場合、ノードプロパティの一覧を指定する必要があります。これらのノード・プロパティは、ジョブで使用するノード数、メイン・ノード・インデックス、および使用する異なるノード範囲を定義します。ジョブを Fargate リソースで実行する場合は、`nodeProperties` を指定することはできません。代わりに `containerProperties` を使用してください。ジョブ定義では、以下のノードプロパティを使用できます。詳細については、マルチノード並列ジョブを参照してください。

> **Note**
>
> ジョブを Amazon EKS リソースで実行する場合は、`nodeProperties` を指定してはならず、

タイプ: NodeProperties オブジェクト

必須: いいえ

`mainNode`

マルチノード並列ジョブの主要なノードにノードインデックスを指定します。このノードインデックス値は、ノード数未満である必要があります。

タイプ: 整数

必須: はい

`numNodes`

マルチノード並列ジョブに関連付けられたノードの数。

タイプ: 整数

必須: はい

`nodeRangeProperties`

マルチノード並列ジョブに関連付けられたノード範囲とそのプロパティの一覧。

> **Note**
>
> ノード・グループとは、すべて同じコンテナ・プロパティを共有するジョブ・ノードの同一グループです。AWS Batchを使用すると、各ジョブに最大5つの異なるノード・グループを指定できます。

タイプ: NodeRangeProperty オブジェクトの配列

必須: はい

`targetNodes`

ノードのインデックス値を使用したノードの範囲。`0:3` の範囲は、インデックス値が `0` から `3` のノードを示しています。開始範囲値が省略されている場合 (`:n`)、範囲の開始値にが使用されます。終了範囲値が省略されている場合 (`n:`)、範囲の終了値にはできるだけ高いノードインデックスが使用されます。累積ノード範囲は、すべてのノード (`0:n`) を考慮する必要があります。例えば、ノード範囲をネストできます。例: `0:10` および `4:5`。この場合は、`4:5` の範囲プロパティは、`0:10` プロパティを上書きします。

タイプ: 文字列

必須: いいえ

`container`

ノード範囲のコンテナの詳細。詳細については、コンテナプロパティを参照してください。

タイプ: ContainerProperties オブジェクト

必須: いいえ

再試行戦略

`retryStrategy`

ジョブ定義の登録時に、オプションとして、このジョブ定義で送信したジョブが失敗したときの再試行戦略を指定できます。SubmitJob] オペレーション中に指定されたリトライ戦略は、ここで定義されたリトライ戦略を上書きします。デフォルトでは、各ジョブは 1 回試行されます。複数の試行を指定すると、ジョブが失敗した場合、ジョブが再試行されます。失敗の例としては、

ジョブがゼロ以外の終了コードが返した場合やコンテナインスタンスが終了した場合が含まれます。詳細については、ジョブの再試行の自動化を参照してください。

タイプ: RetryStrategy オブジェクト

必須: いいえ

`attempts`

ジョブを RUNNABLE ステータスに移行する回数。1〜10 回の試行を指定できます。attempts の回数が 1 より大きい場合、ジョブは RUNNABLE に移行するまでにその回数内で再試行されます。

```
"attempts": integer
```

タイプ: 整数

必須: いいえ

`evaluateOnExit`

ジョブの再試行または失敗の条件を指定する最大5つのオブジェクトの配列。このパラメータを指定する場合は、attempts パラメータも指定する必要があります。evaluateOnExitを指定しても一致するエントリがない場合、ジョブは再試行されます。

```
"evaluateOnExit": [
   {
      "action": "string",
      "onExitCode": "string",
      "onReason": "string",
      "onStatusReason": "string"
   }
]
```

タイプ: EvaluateOnExit オブジェクトの配列

必須: いいえ

`action`

指定された条件 (onStatusReason、onReason および onExitCode) がすべて満たされた場合に実行するアクションを指定します。値は大文字と小文字が区別されません。

タイプ: 文字列

必須: はい

有効な値: RETRY | EXIT

onExitCode

ジョブに対して返された ExitCode の10進表現と照合する glob パターンが含まれています。パターンの最大長は 512 文字です。数字のみを含めることができます。文字や特殊文字を含めることはできません。必要に応じて末尾をアスタリスク (*) にでき、文字列の先頭だけが完全に一致する必要があります。

タイプ: 文字列

必須: いいえ

onReason

ジョブに対して返された Reason と照合する glob パターンが含まれます。パターンの最大長は 512 文字です。文字、数字、ピリオド (.)、コロン (:)、および空白 (スペースまたはタブを含む) を含めることができます。必要に応じて末尾をアスタリスク (*) にでき、文字列の先頭だけが完全に一致する必要があります。

タイプ: 文字列

必須: いいえ

onStatusReason

ジョブに対して返された StatusReason と照合する glob パターンが含まれます。パターンの最大長は 512 文字です。文字、数字、ピリオド (.)、コロン (:)、および空白 (スペースまたはタブを含む) を含めることができます。必要に応じて末尾をアスタリスク (*) にでき、文字列の先頭だけが完全に一致する必要があります。

タイプ: 文字列

必須: いいえ

スケジューリング優先順位

schedulingPriority

このジョブ定義で投入されるジョブのスケジューリング優先度。これは、公平配分ポリシーが適用されたジョブキュー内のジョブにのみ影響します。スケジューリング優先度が高いジョブは、スケジューリング優先度の低いジョブの前にスケジュールされます。

サポートされている最小値は 0 で、サポートされている最大値は 9999 です。

タイプ: 整数

必須: いいえ

タグ

`tags`

ジョブ定義に関連付けるキーバリューペアのタグ。詳細については、AWS Batch リソースのタグ付けを参照してください。

型: 文字列間のマッピング

必須: いいえ

タイムアウト

`timeout`

この期間を超えてジョブが実行されると AWS Batch でジョブが終了するように、ジョブのタイムアウト期間を設定できます。詳細については、ジョブのタイムアウトを参照してください。タイムアウトによりジョブが終了した場合、再試行は行われない。SubmitJob オペレーション時にタイムアウト設定を指定した場合は、ここで定義されているタイムアウト設定が上書きされます。詳細については、ジョブのタイムアウトを参照してください。

タイプ: JobTimeout オブジェクト

必須: いいえ

`attemptDurationSeconds`

`startedAt` によって未終了のジョブが終了されるまでの時間 (秒) (ジョブ試行の AWS Batch タイムスタンプから計測)。タイムアウトの最小値は 60 秒です。

配列ジョブの場合、タイムアウトは親配列ジョブではなく子ジョブに適用されます。

マルチノード並列 (MNP) ジョブの場合、タイムアウトは、ジョブ全体に適用され、個々のノードには適用されません。

タイプ: 整数

必須: いいえ

EcsProperties を使用してジョブ定義を作成する

EcsProperties を使用して AWS Batch のジョブ定義を行うと、ハードウェア、センサー、3D 環境、その他のシミュレーションを個別のコンテナでモデル化できます。この機能を使用することでワークロードコンポーネントを論理的に整理し、これらをメインアプリケーションから分離できます。この機能は、Amazon Elastic Container Service (Amazon ECS)、Amazon Elastic Kubernetes Service (Amazon EKS)、AWS Fargate の AWS Batch で使用できます。

ContainerProperties と EcsProperties のジョブ定義

ユースケースに応じて、ContainerProperties または EcsProperties のジョブ定義の使用を選択できます。大まかに言って、EcsProperties を使用した AWS Batch ジョブの実行は ContainerProperties を使用したジョブの実行と同様です。

ContainerProperties を使用する従来のジョブ定義構造は引き続きサポートされます。現在この構造を使用するワークフローがある場合は引き続き実行できます。

主な違いは、EcsProperties ベースの定義に対応するため、ジョブ定義に新しいオブジェクトが追加されていることです。

例えば、Amazon ECS と Fargate で ContainerProperties を使用するジョブ定義は次のような構造になります。

```
{
   "containerProperties": {
     ...
     "image": "my_ecr_image1",
     ...
  },
...
}
```

Amazon ECS と Fargate で EcsProperties を使用するジョブ定義は次のような構造になります。

```
{
```

```
  "ecsProperties": {
    "taskProperties": [{
      "containers": [
        {
          ...
          "image": "my_ecr_image1",
          ...
        },
        {
          ...
          "image": "my_ecr_image2",
          ...
        },
```

AWS Batch API への変更の概要

ContainerProperties と EcsProperties API データ型を使用する際の主な違いは以下のとおりです。

- ContainerProperties で使用されるパラメータの多くは、TaskContainerProperties でも使用されます。例えば、command、image、privileged、secrets、users です。これらはすべて「TaskContainerProperties」で説明しています。
- 一部の TaskContainerProperties パラメータには従来の構造に機能的に同等なものがありません。例えば、dependsOn、essential、name、ipcMode、pidMode です。詳細は「EcsTaskDetails」と「TaskContainerProperties」を参照してください。

  また、一部の ContainerProperties パラメータには EcsProperties 構造に機能的に同等なものやアプリケーションがありません。taskProperties では、新しいオブジェクトが最大 10 個の要素を受け入れることができるように、container が containers に置き換えられました。詳細は「RegisterJobDefinition:containerProperties」および「EcsTaskProperties:containers」を参照してください。
- taskRoleArn は機能的に jobRoleArn と同等です。詳細は「EcsTaskProperties:taskRoleArn」および「ContainerProperties:jobRoleArn」を参照してください。
- EcsProperties 構造には 1 ~ 10 個のコンテナを含めることができます。詳細は「EcsTaskProperties:containers」を参照してください。
- taskProperties および instanceTypes オブジェクトは配列ですが、現在受け入れることができるのは 1 つの要素のみです。例えば、EcsProperties:taskProperties と NodeRangeProperty:instanceTypes です。

Amazon ECS のマルチコンテナのジョブ定義

Amazon ECS のマルチコンテナ構造に対応するため、API データ型が異なることがあります。例えば、

- `ecsProperties` は、単一コンテナ定義では `containerProperties` と同じレベルです。詳細については、「AWS Batch API リファレンスガイド」の「EcsProperties」を参照してください。
- `taskProperties` には、Amazon ECS タスク用に定義されたプロパティが含まれます。詳細については、「AWS Batch API リファレンスガイド」の「EcsProperties」を参照してください。
- `containers` には、単一コンテナ定義での `containerProperties` と同様の情報が含まれます。主な違いは、`containers` では最大 10 個のコンテナを定義できる点です。詳細については、「AWS Batch API リファレンスガイド」の「ECSTaskProperties:containers」を参照してください。
- `essential` パラメータは、そのコンテナがジョブにどのように影響するかを示します。ジョブを進行させるには、すべての essential コンテナが正常に完了 (0 で終了) する必要があります。essential としてマークされたコンテナが失敗した場合 (0 以外で終了)、ジョブは失敗します。

  デフォルト値は `true` で、少なくとも 1 つのコンテナを `essential` としてマークする必要があります。詳細については、`essential` API リファレンスガイドの「AWS Batch」を参照してください。
- `dependsOn` パラメータを使用すると、コンテナの依存関係のリストを定義できます。詳細については、`dependsOn` API リファレンスガイドの「AWS Batch」を参照してください。

> **Note**
>
> `dependsOn` リストの複雑さと、関連するコンテナランタイムはジョブの開始時間に影響する可能性があります。依存関係の実行に時間がかかる場合、ジョブは完了するまで `STARTING` 状態のままになります。

`ecsProperties` と構造の詳細については、「RegisterJobDefinition」リクエスト構文で「ecsProperties」を参照してください。

Amazon EKS のマルチコンテナのジョブ定義

Amazon EKS のマルチコンテナ構造に対応するため、API データ型が異なることがあります。例えば、

- name は、コンテナに対する一意の識別子です。このオブジェクトは単一のコンテナには必要ありませんが、ポッド内の複数のコンテナを定義するときは必要です。name が単一のコンテナに定義されていない場合は、デフォルト名の default が適用されます。
- initContainers は eksPodProperties データ型内で定義されます。これらはアプリケーションコンテナの前に、常に完了するまで実行されます。また、次のコンテナが起動する前に正常に完了する必要があります。

  これらのコンテナは Amazon EKS Connector エージェントを使用して登録され、登録情報は Amazon Elastic Kubernetes Service のバックエンドデータストアに保持されます。initContainers オブジェクトは最大 10 個の要素を受け入れることができます。詳細については、Kubernetes ドキュメントの「Init Containers」を参照してください。

  > **Note**
  >
  > initContainers オブジェクトは、ジョブの開始時刻に影響を与える可能性があります。initContainers の実行に時間がかかる場合、ジョブは完了するまで STARTING 状態のままになります。

- shareProcessNamespace は、ポッド内のコンテナが同じプロセス名前空間を共有できるかどうかを示します。デフォルト値は false です。これを true に設定すると、コンテナは同じポッドにある他のコンテナのプロセスを認識してシグナルを送信できるようになります。
- どのコンテナにも重要性があり、ジョブが成功するにはすべてのコンテナが正常に完了 (0 で終了) する必要があります。1 つのコンテナが失敗すると (0 以外で終了)、ジョブは失敗します。

eksProperties および構造の詳細については、「RegisterJobDefinition」リクエスト構文で「eksProperties」を参照してください。

リファレンス: EcsProperties を使用した AWS Batch ジョブのシナリオ

EcsProperties を使用する AWS Batch ジョブ定義をニーズに基づいて構造化する方法を説明するために、このトピックでは次の RegisterJobDefinition ペイロードを使用します。これらの例をファイルにコピーし、必要に応じてカスタマイズしてから AWS Command Line Interface (AWS CLI) を使用して RegisterJobDefinition を呼び出してください。

Amazon EC2 での Amazon ECS の AWS Batch ジョブ

以下は、Amazon Elastic Compute Cloud での Amazon Elastic Container Service の AWS Batch ジョブの例です。

```
{
    "jobDefinitionName": "multicontainer-ecs-ec2",
    "type": "container",
    "ecsProperties": {
        "taskProperties": [
          {
            "containers": [
              {
                "name": "c1",
                "essential": false,
                "command": [
                  "echo",
                  "hello world"
                ],
                "image": "public.ecr.aws/amazonlinux/amazonlinux:latest",
                "resourceRequirements": [
                  {
                    "type": "VCPU",
                    "value": "2"
                  },
                  {
                    "type": "MEMORY",
                    "value": "4096"
                  }
                ]
              },
              {
                "name": "c2",
                "essential": false,
                "command": [
                  "echo",
                  "hello world"
                ],
                "image": "public.ecr.aws/amazonlinux/amazonlinux:latest",
                "resourceRequirements": [
                  {
                    "type": "VCPU",
                    "value": "2"
                  },
                  {
                    "type": "MEMORY",
                    "value": "4096"
                  }
```

```
                ]
              },
              {
                "name": "c3",
                "essential": true,
                "command": [
                  "echo",
                  "hello world"
                ],
                "image": "public.ecr.aws/amazonlinux/amazonlinux:latest",
                "firelensConfiguration": {
                  "type": "fluentbit",
                  "options": {
                    "enable-ecs-log-metadata": "true"
                  }
                 },
                "resourceRequirements": [
                  {
                    "type": "VCPU",
                    "value": "6"
                  },
                  {
                    "type": "MEMORY",
                    "value": "12288"
                  }
                ]
              }
            ]
          }
        ]
  }
}
```

Fargate での Amazon ECS の AWS Batch ジョブ

以下は、AWS Fargate での Amazon Elastic Container Service の AWS Batch ジョブの例です。

```
{
    "jobDefinitionName": "multicontainer-ecs-fargate",
    "type": "container",
    "platformCapabilities": [
        "FARGATE"
    ],
```

```
    "ecsProperties": {
        "taskProperties": [
          {
            "containers": [
              {
                "name": "c1",
                "command": [
                  "echo",
                  "hello world"
                ],
                "image": "public.ecr.aws/amazonlinux/amazonlinux:latest",
                "resourceRequirements": [
                  {
                    "type": "VCPU",
                    "value": "2"
                  },
                  {
                    "type": "MEMORY",
                    "value": "4096"
                  }
                ]
              },
              {
                "name": "c2",
                "essential": true,
                "command": [
                  "echo",
                  "hello world"
                ],
                "image": "public.ecr.aws/amazonlinux/amazonlinux:latest",
                "resourceRequirements": [
                  {
                    "type": "VCPU",
                    "value": "6"
                  },
                  {
                    "type": "MEMORY",
                    "value": "12288"
                  }
                ]
              }
            ],
            "executionRoleArn": "arn:aws:iam::1112223333:role/ecsTaskExecutionRole"
          }
```

```
        ]
  }
}
```

Amazon EKS の AWS Batch ジョブ

以下は、Amazon Elastic Kubernetes Service の AWS Batch ジョブの例です。

```
{
  "jobDefinitionName": "multicontainer-eks",
  "type": "container",
  "eksProperties": {
    "podProperties": {
      "shareProcessNamespace": true,
      "initContainers": [
        {
          "name": "init-container",
          "image": "public.ecr.aws/amazonlinux/amazonlinux:2",
          "command": [
            "echo"
          ],
          "args": [
            "hello world"
          ],
          "resources": {
            "requests": {
              "cpu": "1",
              "memory": "512Mi"
            }
          }
        },
        {
          "name": "init-container-2",
          "image": "public.ecr.aws/amazonlinux/amazonlinux:2",
          "command": [
            "echo",
            "my second init container"
          ],
          "resources": {
            "requests": {
              "cpu": "1",
              "memory": "512Mi"
            }
```

```
          }
        }
      ],
      "containers": [
        {
          "name": "c1",
          "image": "public.ecr.aws/amazonlinux/amazonlinux:2",
          "command": [
            "echo world"
         ],
          "resources": {
            "requests": {
              "cpu": "1",
              "memory": "512Mi"
            }
          }
        },
        {
          "name": "sleep-container",
          "image": "public.ecr.aws/amazonlinux/amazonlinux:2",
          "command": [
            "sleep",
            "20"
          ],
          "resources": {
            "requests": {
              "cpu": "1",
              "memory": "512Mi"
            }
          }
        }
      ]
    }
  }
}
```

ノードごとに複数のコンテナを持つ MNP AWS Batch ジョブ

以下は、ノードごとに複数のコンテナを持つマルチノード並列 (MNP) AWS Batch ジョブの例です。

```
{
  "jobDefinitionName": "multicontainer-mnp",
  "type": "multinode",
```

```
  "nodeProperties": {
    "numNodes": 6,
    "mainNode": 0,
    "nodeRangeProperties": [
      {
        "targetNodes": "0:5",
        "ecsProperties": {
          "taskProperties": [
            {
              "containers": [
                {
                  "name": "range05-c1",
                  "command": [
                    "echo",
                    "hello world"
                  ],
                  "image": "public.ecr.aws/amazonlinux/amazonlinux:latest",
                  "resourceRequirements": [
                    {
                      "type": "VCPU",
                      "value": "2"
                    },
                    {
                      "type": "MEMORY",
                      "value": "4096"
                    }
                  ]
                },
                {
                  "name": "range05-c2",
                  "command": [
                    "echo",
                    "hello world"
                  ],
                  "image": "public.ecr.aws/amazonlinux/amazonlinux:latest",
                  "resourceRequirements": [
                    {
                      "type": "VCPU",
                      "value": "2"
                    },
                    {
                      "type": "MEMORY",
                      "value": "4096"
                    }
```

```
                    ]
                }
            ]
        }
    ]
}
}
}
}
```

awslogs ログドライバーを使用する

デフォルトでは、AWS Batch は awslogs ログドライバが CloudWatch Logs にログ情報を送信することを可能にします。この機能により、コンテナからの異なるログを 1 か所で便利に表示できます。また、コンテナログがコンテナインスタンスのディスク容量を占めることも防止できます。このトピックでは、ジョブ定義で awslogs ログドライバーを設定する方法について説明します。

> **Note**
>
> AWS Batch コンソールでは、ジョブ定義を作成するときに ロギング設定 セクションで awslogs ログドライバーを設定できます。

> **Note**
>
> ジョブのコンテナによってログ記録される情報のタイプは、ENTRYPOINT コマンドによって大きく異なります。デフォルトでは、キャプチャされるログは、コンテナをローカルに実行した場合にインタラクティブターミナルに表示されるコマンド出力 (STDOUT および STDERR I/O ストリーム) を示します。awslogs ログドライバーは、これらのログを Docker から CloudWatch Logs に渡します。Docker ログの処理方法 (ファイルデータやストリームをキャプチャする別の方法) の詳細については、Docker ドキュメントのコンテナまたはサービスのログを表示するを参照してください。

コンテナインスタンスから CloudWatch Logs にシステムログを送信するには、での CloudWatch Logs の使用 AWS Batchを参照してください。CloudWatch Logs の詳細については、Amazon CloudWatch Logs ユーザーガイドのログファイルのモニタリングおよび CloudWatch Logs クォータを参照してください。

AWS Batch JobDefiniton データ型の awslogs ログドライバーのオプション

awslogs ログドライバーは、AWS Batch ジョブ定義で以下のオプションをサポートします。詳細については、Docker ドキュメントのCloudWatch Logs ロギングドライバーの設定を参照してください。

awslogs-region

必須: いいえ

awslogs ログドライバーが Docker ログを送信するリージョンを指定します。デフォルトでは、使用されるリージョンはジョブのリージョンと同じです。CloudWatch Logs では、異なるリージョンのジョブからすべてのログを 1 つのリージョンに送信するように選択できます。これにより、それらを 1 つの場所からすべて表示できます。または、より詳細なアプローチのために、リージョンごとにそれらを分離することもできます。ただし、このオプションを選択する場合は、指定したロググループが、指定したリージョンに存在することを確認してください。

awslogs-group

必須：オプション

この awslogs-group オプションを選択すると、awslogs ログドライバーがログストリームを送信するロググループを指定できます。これを指定しない場合、aws/batch/job が使用されます。

awslogs-stream-prefix

必須：オプション

awslogs-stream-prefix オプションを使用して、指定したプレフィックス、コンテナ名、コンテナの所属先における AWS Batch ジョブの Amazon ECS タスクの ID に、ログストリーミングを関連付けることができます。このオプションでプレフィックスを指定した場合、ログストリームの形式は以下のようになります。

```
prefix-name/default/ecs-task-id
```

awslogs-datetime-format

必須: いいえ

このオプションは、Python strftime 形式で複数行起動パターンを定義します。ログメッセージは、パターンに一致する 1 行と、それに続くパターンに一致しない行で構成されます。したがって、一致した行はログメッセージ間の区切り文字です。

この形式を使用する場合のユースケースの例としては、スタックダンプなどの解析された出力があり、これを使用しなければ、複数のエントリに記録されることになります。適切なパターンにより、単一のエントリにキャプチャさせます。

詳細については、awslogs-datetime-formatを参照してください。

`awslogs-datetime-format` と `awslogs-multiline-pattern` の両方が設定されている場合、このオプションは常に優先されます。

> **Note**
>
> 複数行のログ記録は、すべてのログメッセージの正規表現の解析とマッチングを実行します。これによりログ記録のパフォーマンスに悪影響が及ぶ可能性があります。

`awslogs-multiline-pattern`

必須: いいえ

このオプションでは、正規表現を使用して複数行起動パターンを定義します。ログメッセージは、パターンに一致する 1 行と、それに続くパターンに一致しない行で構成されます。したがって、一致した行はログメッセージ間の区切り文字です。

詳細については、Docker ドキュメントのawslogs-multiline-patternを参照してください。

`awslogs-datetime-format` も設定されている場合は、このオプションは無視されます。

> **Note**
>
> 複数行のログ記録は、すべてのログメッセージの正規表現の解析とマッチングを実行します。これによりログ記録のパフォーマンスに悪影響が及ぶ可能性があります。

`awslogs-create-group`

必須: いいえ

自動的に作成されたロググループが必要かどうかを指定します。このオプションを指定しない場合、デフォルトは `false` です。

> **Warning**
>
> このオプションは推奨されません。各ジョブがロググループの作成を試みるため、ジョブが失敗する可能性が高くなるため、CloudWatch Logs CreateLogGroup API アクションを使用して、事前にロググループを作成することをお勧めします。

> **Note**
>
> `logs:CreateLogGroup` を使用しようとする前に、実行ロールの IAM ポリシーには `awslogs-create-group` アクセス権限が含まれている必要があります。

ジョブ定義でログ設定を指定する

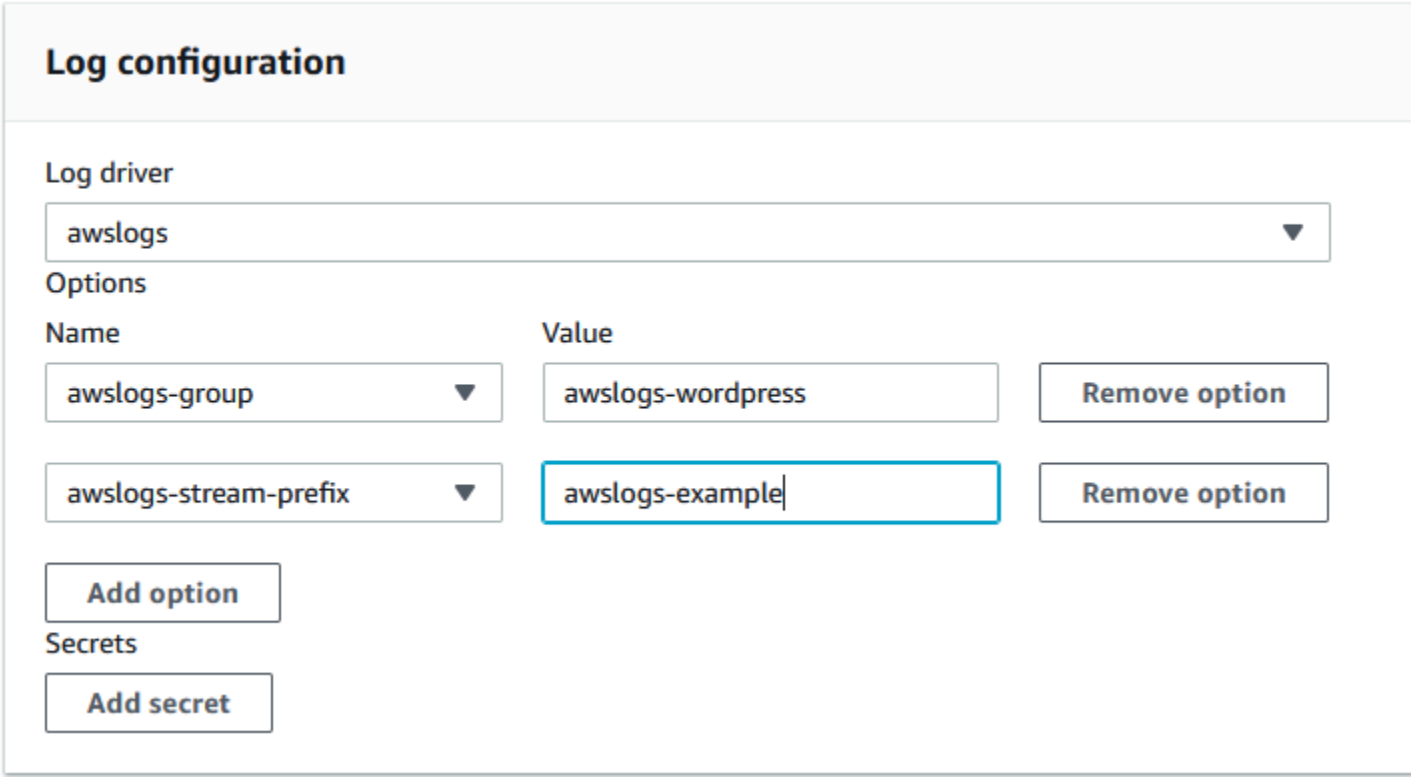
デフォルトでは、AWS Batch が `awslogs` ログドライバーを有効にします。ここでは、ジョブの `awslogs` ログ設定をカスタマイズする方法について説明します。詳細については、シングルノードのジョブ定義を作成するを参照してください。

次のログ設定 JSON スニペットは、ジョブごとに `logConfiguration` オブジェクトが指定されています。1 つは `awslogs-wordpress` というロググループにログを送る WordPress のジョブで、もう 1 つは `awslogs-mysql` というロググループにログを送る MySQL コンテナのものです。どちらのコンテナも `awslogs-example` ログストリームプレフィックスを使用します。

```
"logConfiguration": {
    "logDriver": "awslogs",
    "options": {
        "awslogs-group": "awslogs-wordpress",
        "awslogs-stream-prefix": "awslogs-example"
    }
}
```

```
"logConfiguration": {
    "logDriver": "awslogs",
    "options": {
        "awslogs-group": "awslogs-mysql",
        "awslogs-stream-prefix": "awslogs-example"
    }
}
```

AWS Batch コンソールで、`wordpress` ジョブ定義のログ設定は次の図のように指定されています。



`awslogs` ログドライバーを使用するタスク定義をジョブ定義ログ設定に登録すると、ジョブ定義を使用してジョブを送信し、CloudWatch Logs へのログの送信をスタートできます。詳細については、チュートリアル: ジョブを送信するを参照してください。

機密データを指定する

を使用すると AWS Batch、機密データをシークレットまたは Parameter Store AWS Systems Manager パラメータに保存し、ジョブ定義で参照することで、ジョブに AWS Secrets Manager 機密データを挿入できます。

以下の方法でシークレットをジョブに公開できます。

- 機密データを環境変数としてコンテナに挿入するには、`secrets` ジョブ定義パラメータを使用します。
- ジョブのログ設定内の機密情報を参照するには、`secretOptions` ジョブ定義パラメータを使用します。

トピック

- Secrets Manager を使用して機密データを指定する
- Systems Manager Parameter Store を使用して機密データを指定する

Secrets Manager を使用して機密データを指定する

を使用すると AWS Batch、機密データを AWS Secrets Manager シークレットに保存し、ジョブ定義で参照することで、機密データをジョブに挿入できます。Secrets Manager シークレットに保存された機密データは、環境変数として、またはログ設定の一部としてジョブに公開できます。

シークレットを環境変数として挿入する場合は、挿入するシークレットの JSON キーまたはバージョンを指定できます。このプロセスは、ジョブに公開される機密データの制御に役立ちます。シークレットのバージョニングの詳細については、AWS Secrets Manager ユーザーガイドの「AWS Secrets Managerの主な用語と概念」を参照してください。

Secrets Manager を使用して機密データを指定する際の考慮事項

Secrets Manager を使用してジョブの機密データを指定する場合は、以下を考慮する必要があります。

- シークレットの特定の JSON キーやバージョンを使用してシークレットを挿入するには、コンピューティング環境内のコンテナインスタンスに、 Amazon ECS コンテナエージェントのバージョン 1.37.0 以降が必要です。ただし、最新のコンテナエージェントバージョンを使用することをお勧めします。エージェントのバージョンの確認と最新バージョンへの更新の詳細については、Amazon Elastic Container Service デベロッパーガイドのAmazon ECS コンテナエージェントの更新を参照してください。

  シークレットの内容全体を環境変数として挿入したり、ログ設定にシークレットを挿入したりするには、コンテナインスタンスにバージョン 1.23.0 以降のコンテナエージェントが必要です。
- CreateSecret API の `SecretString` パラメータで作成されたシークレットであるテキストデータを格納するシークレットのみがサポートされます。CreateSecret API の `SecretBinary` パラメータで作成されたシークレットであるバイナリデータを格納するシークレットはサポートされていません。
- Secrets Manager シークレットを参照するジョブ定義を使用してジョブの機密データを取得する場合、インターフェイス VPC エンドポイントも使用している場合は、Secrets Manager のインターフェイス VPC エンドポイントを作成する必要があります 。詳細については、AWS Secrets Manager ユーザーガイドの「VPC EndpointでSecrets Managerを使用する」を参照してください。

- 重要なデータは、ジョブが最初に開始されたときにジョブに挿入されます。シークレットを後で更新またはローテーションすると、ジョブには更新された値が自動的に送信されなくなります。更新されたシークレット値で新しいジョブを起動するようサービスに強制するには、新しいジョブの起動が必要です。

AWS Batch シークレットに必要な IAM アクセス許可

この機能を使用するには、実行ロールを持っていて、ジョブ定義でそのロールを参照する必要があります。これにより、コンテナエージェントは必要な Secrets Manager リソースをプルすることを許可されます。詳細については、「AWS Batch IAM 実行ロール」を参照してください。

作成した Secrets Manager シークレットへのアクセスを許可するには、以下のアクセス許可をインラインポリシーとして実行ロールに手動で追加します。詳細については、IAM ユーザーガイドの「IAM ポリシーの追加と削除 」を参照してください。

- `secretsmanager:GetSecretValue` – Secrets Manager シークレットを参照する場合に必須です。
- `kms:Decrypt` – シークレットでカスタムの KMS キーを使用し、デフォルトのキーを使用しない場合にのみ必須です。そのカスタムキーの ARN はリソースとして追加されている必要があります。

次の例のインラインポリシーでは必須の アクセス許可を追加しています。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "kms:Decrypt"
            ],
            "Resource": [
                "arn:aws:secretsmanager:us-
east-2:777777777777:secret:<secret_name>",
                "arn:aws:kms:us-east-2:777777777777:key/<key_id>"
            ]
```

```
        }
    ]
}
```

環境変数としての機密データの挿入

以下の項目はジョブ定義内で指定できます。

- ジョブに設定する環境変数の名前が含まれている secrets オブジェクト
- Secrets Manager シークレットの Amazon リソースネーム (ARN)。
- ジョブに渡す機密データが含まれている追加のパラメータ

次の例は、Secrets Manager シークレットに指定する必要がある完全な構文を示しています。

```
arn:aws:secretsmanager:region:aws_account_id:secret:secret-name:json-key:version-stage:version-id
```

次のセクションでは、追加のパラメータについて説明します。これらのパラメータは省略可能です。ただし、それらを使用しない場合は、デフォルト値を使用するためにコロン : が含まれている必要があります。以下の例でより詳細なコンテキストを示します。

`json-key`

キーと値のペアのキーの名前を指定します。値は設定する環境変数の値です。JSON 形式の値のみがサポートされます。JSON キーを指定しないと、シークレットの内容全体が使用されます。

`version-stage`

使用するシークレットのバージョンのステージングラベルを指定します。バージョンのステージングラベルを指定した場合、バージョン ID は指定できません。バージョンのステージを指定しないと、デフォルトの動作として、AWSCURRENT ステージングラベルのシークレットが取得されます。

ステージングラベルは、シークレットが更新またはローテーションされたときに、シークレットのさまざまなバージョンを追跡するために使用します。シークレットの各バージョンには、1 つ以上のステージングラベルと 1 つの ID があります。詳細については、「 ユーザーガイド」の AWS 「Secrets Manager の主要な用語と概念」を参照してください。 AWS Secrets Manager

`version-id`

使用するシークレットのバージョンの固有 ID を指定します。バージョン ID を指定した場合、バージョンのステージングラベルは指定できません。バージョン ID を指定しないと、デフォルトの動作として、AWSCURRENT ステージングラベルのシークレットが取得されます。

バージョン ID は、シークレットが更新またはローテーションされたときに、シークレットのさまざまなバージョンを追跡するために使用します。シークレットの各バージョンには ID があります。詳細については、「 ユーザーガイド」の AWS 「Secrets Manager の主要な用語と概念」を参照してください。 AWS Secrets Manager

コンテナの定義の例

以下の例では、コンテナの定義で Secrets Manager シークレットを参照する方法を示します。

Exampleシークレット全体を参照する

次に示すのは、Secrets Manager シークレットのテキスト全体を参照するときの形式を示すタスク定義のス二ペットです。

```
{
  "containerProperties": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name-
AbCdEf"
    }]
  }]
}
```

Exampleシークレット内の特定のキーを参照する

次に示すのは、シークレットの内容と、シークレットに関連付けられているバージョンのステージングラベルおよびバージョン ID を表示する >get-secret-value コマンドの出力例です。

```
{
    "ARN": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-AbCdEf",
    "Name": "appauthexample",
    "VersionId": "871d9eca-18aa-46a9-8785-981dd39ab30c",
    "SecretString": "{\"username1\":\"password1\",\"username2\":\"password2\",
\"username3\":\"password3\"}",
    "VersionStages": [
```

```
        "AWSCURRENT"
    ],
    "CreatedDate": 1581968848.921
}
```

前のコンテナの定義の出力で特定のキーを参照するには、ARN の最後にキー名を指定します。

```
{
  "containerProperties": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-
AbCdEf:username1::"
    }]
  }]
}
```

Example特定のシークレットバージョンを参照する

次に示すのは、シークレットの暗号化されていない内容と、シークレットのすべてのバージョンのメタデータを表示する >describe-secret コマンドの出力例です。

```
{
    "ARN": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-AbCdEf",
    "Name": "appauthexample",
    "Description": "Example of a secret containing application authorization data.",
    "RotationEnabled": false,
    "LastChangedDate": 1581968848.926,
    "LastAccessedDate": 1581897600.0,
    "Tags": [],
    "VersionIdsToStages": {
        "871d9eca-18aa-46a9-8785-981dd39ab30c": [
            "AWSCURRENT"
        ],
        "9d4cb84b-ad69-40c0-a0ab-cead36b967e8": [
            "AWSPREVIOUS"
        ]
    }
}
```

前のコンテナの定義の出力で特定のバージョンのステージングラベルを参照するには、ARN の最後にキー名を指定します。

```
{
  "containerProperties": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-
AbCdEf::AWSPREVIOUS:"
    }]
  }]
}
```

前のコンテナの定義の出力で特定のバージョン ID を参照するには、ARN の最後にキー名を指定します。

```
{
  "containerProperties": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-
AbCdEf::9d4cb84b-ad69-40c0-a0ab-cead36b967e8"
    }]
  }]
}
```

Exampleシークレットの特定のキーおよびバージョンのステージングラベルを参照する

シークレット内の特定のキーと特定のバージョンのステージングラベルの両方を参照する方法は次のとおりです。

```
{
  "containerProperties": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-
AbCdEf:username1:AWSPREVIOUS:"
    }]
  }]
}
```

特定のキーとバージョン ID を指定するには、次の構文を使用します。

```
{
  "containerProperties": [{
```

```
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-
AbCdEf:username1::9d4cb84b-ad69-40c0-a0ab-cead36b967e8"
    }]
  }]
}
```

ログ設定に機密データを挿入する

ジョブの定義内で `logConfiguration` を指定するときに、コンテナに設定するログドライバーオプションの名前と、コンテナに提示する機密データが含まれている Secrets Manager シークレットの完全な ARN を使用して `secretOptions` を指定できます。

以下に示すのは、Secrets Manager シークレットを参照するときの形式を示すジョブ定義のスニペットです。

```
{
  "containerProperties": [{
    "logConfiguration": [{
      "logDriver": "splunk",
      "options": {
        "splunk-url": "https://cloud.splunk.com:8080"
      },
      "secretOptions": [{
        "name": "splunk-token",
        "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name-
AbCdEf"
      }]
    }]
  }]
}
```

AWS Secrets Manager シークレットを作成する

Secrets Manager コンソールを使用して、機密データ用のシークレットを作成できます。詳細については、AWS Secrets Manager ユーザーガイドの 「ベーシックシークレットの作成 を参照してください。

基本的なシークレットを作成するには

機密データのシークレットを作成するには、Secrets Manager を使用します。

1. https://console.aws.amazon.com/secretsmanager/ から Secrets Manager コンソールを開きます。
2. [Store a new secret] (新しいシークレットの保存) を選択します。
3. [Select secret type] (シークレットタイプの選択) で、[Other type of secrets] (他の種類のシークレット) を選択します。
4. カスタムシークレットの詳細を [Key] (キー) と [Value] (値) のペアの形式で指定します。例えば、`UserName` のキーを指定し、その値として適切なユーザー名を指定できます。`Password` の名前とパスワードのテキストを値として持つ 2 番目のキーを追加します。データベース名、サーバーアドレス、TCP ポートなどのエントリを追加することもできます。必要な情報を格納するのに必要な数だけペアを追加できます。

   または、[Plaintext] (プレーンテキスト) タブを選択して、任意の方法でシークレット値を入力することもできます。
5. シークレット内の保護されたテキストの AWS KMS 暗号化に使用する暗号化キーを選択します。選択しない場合、Secrets Manager は、アカウントのデフォルトキーが存在するかどうかを調べ、存在する場合はそれを使用します。デフォルトのキーが存在しない場合は、Secrets Manager が自動的に作成します。また、[Add new key(新しいキーの追加)] を選択して、このシークレット専用のカスタム KMS キーを作成します。独自の KMS キーを作成するには、アカウントに KMS キーを作成するアクセス権限が必要です。
6. [Next] (次へ) を選択します。
7. [Secret name] (シークレット名) に、オプションのパスと名前 (例: **`production/MyAwesomeAppSecret`** または **`development/TestSecret`**) を入力し、[Next] (次) を選択します。オプションで説明を追加することもできます。後でこのシークレットを思い出すのに役立ちます。

   シークレット名に使用できるのは、ASCII 文字、数字、または次の記号のみです: /_+=.@-
8. (オプション) この時点で、シークレットのローテーションを設定することができます。この手順では、[Disable automatic rotation] (自動ローテーションを無効化) は無効のままにし、[次] を選択します。

   新規または既存のシークレットでローテーションを設定する方法については、「シークAWS Secrets Manager レットのローテーション」を参照してください。
9. 設定を確認し、次に [Store secret] (シークレットの保存) を選択して入力した内容すべてを Secrets Manager の新しいシークレットとして保存します。

Systems Manager Parameter Store を使用して機密データを指定する

を使用すると AWS Batch、機密データを Parameter Store パラメータに保存し、コンテナ定義で参照することで、コンテナに機密データ AWS Systems Manager を挿入できます。

トピック

Systems Manager Parameter Store を使用した機密データの指定に関する考慮事項

Systems Manager パラメータストアのパラメータを使用してコンテナの機密データを指定する場合は、以下を考慮する必要があります。

- この機能では、コンテナインスタンスに、バージョン 1.23.0 以降のコンテナエージェントが必要です。ただし、最新のコンテナエージェントバージョンを使用することをお勧めします。エージェントのバージョンの確認と最新バージョンへの更新の詳細については、Amazon Elastic Container Service デベロッパーガイドのAmazon ECS コンテナエージェントの更新を参照してください。
- 機密データは、コンテナが最初に開始されたときにジョブのコンテナに挿入されます。シークレットまたは Parameter Store パラメータが後で更新またはローテーションされたときに、コンテナは更新された値を自動的に受け取りません。更新されたシークレットを使用して新しいジョブを強制的に起動するには、新しいジョブを起動する必要があります。

AWS Batch シークレットに必要な IAM アクセス許可

この機能を使用するには、実行ロールを持っていて、ジョブ定義でそのロールを参照する必要があります。これにより、Amazon ECS コンテナエージェントは必要な AWS Systems Manager リソースをプルできます。詳細については、「AWS Batch IAM 実行ロール」を参照してください。

作成した Parameter Store AWS Systems Manager パラメータへのアクセスを提供するには、以下のアクセス許可をインラインポリシーとして実行ロールに手動で追加します。詳細については、IAM ユーザーガイドのIAM ポリシーの追加と削除を参照してください。

- `ssm:GetParameters`— Systems Managerパラメータストアのパラメータをタスク定義で参照する場合は必須です。
- `secretsmanager:GetSecretValue` — Secrets Manager シークレットを直接参照するか、Systems Manager パラメータストアのパラメータがタスク定義で Secrets Manager シークレットを参照している場合は必須です。
- `kms:Decrypt` – シークレットでカスタムの KMS キーを使用し、デフォルトのキーを使用しない場合にのみ必須です。そのカスタムキーの ARN はリソースとして追加されている必要があります。

次の例のインラインポリシーでは必須許可を追加しています。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameters",
                "secretsmanager:GetSecretValue",
                "kms:Decrypt"
            ],
            "Resource": [
                "arn:aws:ssm:us-east-2:999999999999:parameter/<parameter_name>",
                "arn:aws:secretsmanager:us-
east-2:999999999999:secret:<secret_name>",
                "arn:aws:kms:us-east-2:999999999999:key/<key_id>"
            ]
        }
    ]
}
```

環境変数として機密データを挿入する

コンテナの定義内で、コンテナに設定する環境変数の名前と、コンテナに渡す機密データが含まれている Systems Manager パラメータストアのパラメータの ARN 全体を使用して `secrets` を指定できます。

以下に示すのは、Systems Manager パラメータストアのパラメータを参照するときの形式を示すタスク定義のスニペットです。起動するタスクと同じリージョンに、Systems Manager パラメータストアのパラメータが存在する場合は、パラメータの完全な ARN または名前のどちらも使用できます。パラメータが別のリージョンに存在する場合は、完全な ARN を指定する必要があります。

```
{
  "containerProperties": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:ssm:region:aws_account_id:parameter/parameter_name"
    }]
  }]
}
```

ログ設定に機密データを挿入する

コンテナの定義内で `logConfiguration` を指定するときに、コンテナに設定するログドライバーオプションの名前と、コンテナに渡す機密データが含まれている Systems Manager パラメータストアのパラメータの ARN 全体を使用して `secretOptions` を指定できます。

> **Important**
>
> 起動するタスクと同じリージョンに、Systems Manager パラメータストアのパラメータが存在する場合は、パラメータの完全な ARN または名前のどちらも使用できます。パラメータが別のリージョンに存在する場合は、完全な ARN を指定する必要があります。

以下に示すのは、Systems Manager パラメータストアのパラメータを参照するときの形式を示すタスク定義のスニペットです。

```
{
  "containerProperties": [{
    "logConfiguration": [{
      "logDriver": "fluentd",
      "options": {
        "tag": "fluentd demo"
      },
      "secretOptions": [{
        "name": "fluentd-address",
        "valueFrom": "arn:aws:ssm:region:aws_account_id:parameter/parameter_name"
```

```
      }]
    }]
  }]
}
```

Parameter Store AWS Systems Manager パラメータを作成する

AWS Systems Manager コンソールを使用して、機密データの Systems Manager パラメータストアパラメータを作成できます。詳細については、AWS Systems Manager ユーザーガイドのコマンドでパラメータを作成して使用する (コンソール)を参照してください。

パラメータストアのパラメータを作成するには

1. https://console.aws.amazon.com/systems-manager/ で AWS Systems Manager コンソールを開きます。
2. ナビゲーションペインで パラメータストア、パラメータの作成 の順に選択します。
3. 名前 に、階層とパラメータ名を入力します。例えば、`test/database_password` と入力します。
4. Description] (説明) に説明を入力します (省略可能)。
5. Type] (タイプ) として、String]、StringList]、SecureString] のいずれかを選択します。

> **Note**
>
> - SecureString] を選択すると、KMS Key ID] フィールドが表示されます。KMS キー ID、KMS キー ARN、エイリアス名、またはエイリアス ARN が指定されていない場合、システムは `alias/aws/ssm` を使用します。これは、Systems Manager 用のデフォルトの KMS キーです。このキーを使用しない場合は、カスタムキーを選択します。詳細については、AWS Systems Manager ユーザーガイドのSecure String パラメータを使用するを参照してください。
> - コンソールで `key-id` パラメータにカスタム KMS キーエイリアス名またはエイリアス ARN のいずれかを指定して Secure String パラメータを作成する場合は、そのエイリアスの前にプレフィックス `alias/` を指定する必要があります。ARN の例を次に示します。
>
>   ```
>   arn:aws:kms:us-east-2:123456789012:alias/MyAliasName
>   ```
>
>   エイリアス名の例を次に示します。

```
alias/MyAliasName
```

6. Value] (値) に値を入力します。例えば、`MyFirstParameter` です。SecureString] を選択している場合、入力した値は必ずマスクされます。
7. パラメータの作成 を選択します。

ジョブのプライベートレジストリの認証

を使用するジョブのプライベートレジストリ認証 AWS Secrets Manager を使用すると、認証情報を安全に保存し、ジョブ定義で参照できます。これにより、ジョブ定義で認証 AWS を必要とする 外のプライベートレジストリに存在するコンテナイメージを参照する方法が提供されます。この機能は、Amazon EC2 インスタンスと Fargate でホストされるジョブに対応しています。

> **Important**
>
> ジョブ定義で Amazon ECR に保存されたイメージを参照している場合、このトピックは適用されません。詳細については、Amazon Elastic Container Registry ユーザーガイドの「Amazon ECS で Amazon ECR イメージを使用する」を参照してください。

Amazon EC2 インスタンスでホストされるジョブの場合は、この機能のためにコンテナエージェントのバージョン `1.19.0` 以降が必要です。ただし、最新のコンテナエージェントバージョンを使用することをお勧めします。エージェントのバージョンの確認と最新バージョンへの更新の方法の詳細については、「Amazon Elastic Container Service 開発者ガイド」の「Amazon ECS コンテナエージェントをアップデートする」を参照してください。

Fargate でホストされているジョブの場合、この機能にはプラットフォームバージョン `1.2.0` 以降が必要です。詳細については、「Amazon Elastic Container Service 開発者ガイド」の「AWS Amazon ECS 向け Fargate プラットフォームバージョン」を参照してください。

コンテナの定義内で、作成したシークレットの詳細 で `repositoryCredentials` オブジェクトを指定します。参照するシークレットは、それを使用するジョブとは異なる AWS リージョン アカウントから取得することも、別のアカウントから取得することもできます。

> **Note**
>
> AWS Batch API、 AWS CLIまたは AWS SDK を使用する場合、シークレットが起動するジョブ AWS リージョン と同じ に存在する場合は、シークレットの完全な ARN または名前を使用できます。シークレットが別のアカウントに存在する場合は、シークレットの完全な ARN を指定する必要があります。を使用する場合 AWS マネジメントコンソール、シークレットの完全な ARN を常に指定する必要があります。

必要なパラメータが含まれるジョブ定義の スニペットを以下に示します。

```
"containerProperties": [
  {
    "image": "private-repo/private-image",
    "repositoryCredentials": {
      "credentialsParameter":
 "arn:aws:secretsmanager:region:123456789012:secret:secret_name"
    }
  }
]
```

プライベートレジストリの認証で必須の IAM アクセス許可

この機能を使用するには、ジョブの実行ロールが必要です。このロールを使用して、コンテナエージェントでコンテナイメージをプルできます。詳細については、「AWS Batch IAM 実行ロール」を参照してください。

作成したシークレットにアクセスできるようにするには、以下のアクセス許可を、インラインポリシーとしてジョブの実行ロールに追加します。詳細については、「IAM ポリシーの追加と削除」を参照してください。

- `secretsmanager:GetSecretValue`
- `kms:Decrypt` - カスタムの KMS キーを使用するが、デフォルトのキーは使用しない場合にのみ必須。カスタムキーの Amazon リソースネーム (ARN) は、リソースとして追加する必要があります。

次の例では、インラインポリシーによりアクセス許可を追加しています。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "secretsmanager:GetSecretValue"
            ],
            "Resource": [
                "arn:aws:secretsmanager:us-
east-1:123456789012:secret:secret_name",
                "arn:aws:kms:us-east-1:123456789012:key/key_id"
            ]
        }
    ]
}
```

チュートリアル: プライベートレジストリ認証のシークレットを作成する

を使用してプライベートレジストリ認証情報のシークレットを作成するには、次の手順を実行します AWS Secrets Manager。

基本的なシークレットを作成する

1. https://console.aws.amazon.com/secretsmanager/ で AWS Secrets Manager コンソールを開きます。
2. [Store a new secret] (新しいシークレットの保存) を選択します。
3. [Select secret type] (シークレットタイプの選択) で、[Other type of secrets] (他の種類のシークレット) を選択します。
4. [プレーンテキスト] を選択し、次の形式でプライベートレジストリ認証情報を入力します。

```
{
  "username" : "privateRegistryUsername",
  "password" : "privateRegistryPassword"
}
```

5. [次へ] を選択します。
6. [Secret name] (シークレット名) に、オプションのパスと名前 (例: **`production/MyAwesomeAppSecret`** または **`development/TestSecret`**) を入力し、[Next] (次へ) を選択します。オプションで説明を追加することもできます。後でこのシークレットを思い出すのに役立ちます。

   シークレット名に使用できるのは、ASCII 文字、数字、または以下の記号のみです: `/_+=.@-`。
7. (オプション) この時点で、シークレットのローテーションを設定することができます。この手順では、[Disable automatic rotation] (自動ローテーションを無効化) は無効のままにし、[次] を選択します。

   新規または既存のシークレットでローテーションを設定する方法については、「シークAWS Secrets Manager レットのローテーション」を参照してください。
8. 設定を確認した上で、[Store secret] (シークレットの保存) を選択し、入力した全内容を Secrets Manager の新しいシークレットとして保存します。

ジョブ定義を登録し、[プライベートレジストリ] で [プライベートレジストリ認証] をオンにします。その後、[Secrets Manager ARN または名前] で、シークレットの Amazon リソースネーム (ARN) を入力します。詳細については、「プライベートレジストリの認証で必須の IAM アクセス許可」を参照してください。

Amazon EFS ボリューム

Amazon Elastic File System (Amazon EFS) では、AWS Batch ジョブで使用するためのシンプルでスケーラブルなファイルストレージを提供します。Amazon EFSでは、ストレージ容量は伸縮性があります。ファイルの追加や削除時に、自動的にスケールされます。アプリケーションでは、必要なときに必要なストレージを確保できます。

Amazon EFS ファイルシステムを AWS Batch で使用して、コンテナインスタンスのフリート全体のファイルシステムデータをエクスポートできます。これにより、ジョブは、同じ永続的ストレージにアクセスできます。ただし、Docker デーモンが起動する前に、Amazon EFS ファイルシステムをマウントするように、コンテナインスタンス AMI を設定する必要があります。また、ファイルシステムを使用するには、ジョブ定義でコンテナインスタンスのボリュームマウントを参照する必要があります。以下のセクションは、AWS Batch でAmazon EFSの使用を開始するのに役立ちます。

Amazon EFS ボリュームに関する考慮事項

Amazon EFS ボリュームを使用する際には、以下の点を考慮する必要があります:

- EC2 リソースを使用するジョブの場合、Amazon ECS に最適化された AMI バージョン 20191212、コンテナエージェントバージョン 1.35.0 で、Amazon EFS ファイルシステムのサポートがパブリックプレビューとして追加されました。ただし、Amazon EFS ファイルシステムのサポートは、コンテナエージェントバージョンが 1.38.0 の Amazon ECS 最適化 AMI バージョン 20200319 (Amazon EFS アクセスポイントと IAM 認可機能が含まれるもの) で一般提供されています。これらの機能を利用するには、Amazon ECS に最適化された AMI バージョン 20200319 以降を使用することをお勧めします。詳細については、Amazon Elastic Container Service デベロッパーガイドの「Amazon ECS に最適化された AMI バージョン」を参照してください。

  > **Note**
  >
  > 独自の AMI を作成する場合、コンテナエージェント 1.38.0 以降、`ecs-init` バージョン 1.38.0-1 以降を使用し、Amazon EC2 インスタンスで以下のコマンドを実行する必要があります。これが Amazon ECS ボリュームプラグインを有効にするためのものです。コマンドは、ベースイメージとして Amazon Linux 2 と Amazon Linux のどちらを使用しているかによって異なります。
  >
  > Amazon Linux 2
  >
  > ```
  > $ yum install amazon-efs-utils
  > systemctl enable --now amazon-ecs-volume-plugin
  > ```
  >
  > Amazon Linux
  >
  > ```
  > $ yum install amazon-efs-utils
  > sudo shutdown -r now
  > ```

- Fargate リソースを使用するジョブの場合、プラットフォームバージョン 1.4.0 以降で Amazon EFS ファイルシステムのサポートが追加されました。詳細については、Amazon Elastic Container Service デベロッパーガイドの「AWS Fargate プラットフォームのバージョン」を参照してください。
- Fargate リソースで使用されるジョブに Amazon EFS ボリュームを指定する場合、Fargate は Amazon EFS ボリュームの管理を担当するスーパーバイザーコンテナを作成します。スーパーバイザーコンテナは、ジョブのメモリを少しだけ使用します。スーパーバイザーコンテナは、タスク

メタデータバージョン 4 エンドポイントにクエリを実行するときに表示されます。詳細については、Amazon Elastic Container Service ユーザーガイドAWS Fargate用のタスクメタデータエンドポイントバージョン 4 を参照してください。

Amazon EFS アクセスポイントの使用

Amazon EFS アクセスポイントは、EFS ファイルシステムへのアプリケーション固有のエントリポイントです。これにより、共有データセットへのアプリケーションアクセスが管理しやすくなります。Amazon EFS アクセスポイントの詳細およびアクセス制御方法については、Amazon Elastic File System ユーザーガイドの「Amazon EFS アクセスポイントの使用」を参照してください。

アクセスポイントを使用すると、アクセスポイントを介したすべてのファイルシステム要求に対してユーザーアイデンティティ (ユーザーの POSIX グループなど) を適用できます。また、ファイルシステムに対して別のルートディレクトリを適用し、このディレクトリまたはそのサブディレクトリ内のデータに対してのみ、クライアントにアクセスを許可することもできます。

> **Note**
>
> EFS アクセスポイントを作成するときは、ルートディレクトリとして機能するファイルシステム上のパスを指定します。AWS Batch ジョブ定義でアクセスポイント ID を持つ EFS ファイルシステムを参照する場合、ルートディレクトリを省略するか、EFS アクセスポイントに設定されたパスを強制する / に設定する必要があります。

AWS Batch ジョブの IAM ロールを使用して、特定のアプリケーションで使用するアクセスポイントを限定できます。IAM ポリシーとアクセスポイントを組み合わせると、アプリケーションから特定のデータセットへのアクセスを簡単に保護できます。この機能では、タスク用の Amazon ECS IAM ロールを使用します。詳細については、Amazon Elastic Container Service デベロッパーガイドのタスク用の IAM ロールを参照してください。

ジョブ定義内で Amazon EFS ファイルシステムを指定する

コンテナに Amazon EFS ファイルシステムボリュームを使用するには、ジョブ定義でボリュームとマウントポイントの設定を指定する必要があります。次のジョブ定義の JSON スニペットは、コンテナの [volumes] と [mountPoints] オブジェクトの構文を示します。

```
{
    "containerProperties": [
```

```
        {
            "image": "amazonlinux:2",
            "command": [
                "ls",
                "-la",
                "/mount/efs"
            ],
            "mountPoints": [
                {
                    "sourceVolume": "myEfsVolume",
                    "containerPath": "/mount/efs",
                    "readOnly": true
                }
            ],
            "volumes": [
                {
                    "name": "myEfsVolume",
                    "efsVolumeConfiguration": {
                        "fileSystemId": "fs-12345678",
                        "rootDirectory": "/path/to/my/data",
                        "transitEncryption": "ENABLED",
                        "transitEncryptionPort": integer,
                        "authorizationConfig": {
                            "accessPointId": "fsap-1234567890abcdef1",
                            "iam": "ENABLED"
                        }
                    }
                }
            ]
        }
    ]
}
```

`efsVolumeConfiguration`

タイプ: オブジェクト

必須: いいえ

このパラメータは、Amazon EFS ボリュームを使用する場合に指定します。

`fileSystemId`

型: 文字列

必須: はい

使用する Amazon EFS ファイルシステムの ID。

`rootDirectory`

型: 文字列

必須: いいえ

ホスト内にルートディレクトリとしてマウントする Amazon EFS ファイルシステム内のディレクトリ。このパラメータを省略すると、Amazon EFS ボリュームのルートが使用されます。/ を指定すると、このパラメータを省略した場合と同じ結果になります。最大 4,096 文字を使用できます。

> **Important**
>
> [authorizationConfig] に EFS アクセスポイントを指定する場合は、ルートディレクトリパラメータを省略するか、または [/] に設定する必要があります。これにより、EFS アクセスポイントに設定されたパスが強制されます。

`transitEncryption`

タイプ: 文字列

有効な値: `ENABLED` | `DISABLED`

必須: いいえ

AWS Batch ホストと Amazon EFS サーバー間で Amazon EFS データの転送中の暗号化を有効にするかどうかをします。Amazon EFS IAM 認可を使用する場合は、転送中の暗号化を有効にする必要があります。このパラメータを省略すると、`DISABLED` のデフォルト値が使用されます。詳細については、Amazon Elastic File System ユーザーガイドの「トランジット中のデータの暗号化」を参照してください。

`transitEncryptionPort`

タイプ: 整数

必須: いいえ

AWS Batch ホストと Amazon EFS サーバーの間で暗号化されたデータを送信するときに使用するポート。転送中の暗号化ポートを指定しないと、Amazon EFS マウントヘルパーが使

用するポート選択方式が使用されます。この値は 0～65,535 の範囲の値にする必要があります。詳細については、[Amazon Elastic File System User Guide] (Amazon Elastic File System ユーザーガイド) の[EFS Mount Helper] (EFS マウントヘルパー) を参照してください。

`authorizationConfig`

タイプ: オブジェクト

必須: いいえ

Amazon EFS ファイルシステムに対する認可構成の詳細。

`accessPointId`

型: 文字列

必須: いいえ

使用するアクセスポイント ID。アクセスポイントを指定する場合は、`[efsVolumeConfiguration]` のルートディレクトリ値を省略するか、これを `[/]` に設定する必要があります。これにより、EFS アクセスポイントに設定されたパスが強制されます。アクセスポイントを使用する場合は、`EFSVolumeConfiguration` で転送中の暗号化を有効にする必要があります。詳細については、Amazon Elastic ファイルシステムユーザーガイドの Amazon EFS アクセスポイントの使用を参照してください。

`iam`

型: 文字列

有効な値: `ENABLED | DISABLED`

必須: いいえ

Amazon EFS ファイルシステムのマウント時にジョブ定義で定義した AWS Batch ジョブの IAM ロールを使用するかどうかを決定します。使用する場合は、`EFSVolumeConfiguration` で転送中の暗号化を有効にする必要があります。このパラメータを省略すると、`DISABLED` のデフォルト値が使用されます。IAM 実行ロールの詳細については、「AWS Batch IAM 実行ロール」を参照してください。

ジョブ定義の例

次のトピックのジョブ定義の例では、環境変数、パラメータ置換やボリュームのマウントなどの一般的なパターンを使用する方法を示しています。

内容

環境変数

次のジョブ定義の例では、環境変数を使用してファイルタイプと Amazon S3 URL を指定します。この用例は、「簡単な "Fetch & Run" AWS Batch ジョブを作成する」コンピュータブログポストから引用しています。このブログポストで説明される `fetch_and_run.sh` スクリプトでは、S3から `myjob.sh` スクリプトをダウンロードし、そのファイルタイプを指定するための環境変数を使用します。

この例では、コマンドと環境変数はジョブ定義でハードコード化されていますが、さらに柔軟性のあるジョブ定義を作成するためにコマンドと環境変数を指定することもできます。

```
{
    "jobDefinitionName": "fetch_and_run",
    "type": "container",
    "containerProperties": {
        "image": "123456789012.dkr.ecr.us-east-1.amazonaws.com/fetch_and_run",
        "resourceRequirements": [
            {
                "type": "MEMORY",
                "value": "2000"
            },
            {
                "type": "VCPU",
                "value": "2"
            }
        ],
        "command": [
            "myjob.sh",
            "60"
        ],
        "jobRoleArn": "arn:aws:iam::123456789012:role/AWSBatchS3ReadOnly",
        "environment": [
            {
```

```
                "name": "BATCH_FILE_S3_URL",
                "value": "s3://amzn-s3-demo-source-bucket/myjob.sh"
            },
            {
                "name": "BATCH_FILE_TYPE",
                "value": "script"
            }
        ],
        "user": "nobody"
    }
}
```

パラメータ置換

次の例では、パラメータ置換とデフォルト値を設定するためのジョブ定義を説明しています。

Ref:: セクションの command 宣言は、パラメータ置換のためにプレースホルダーを設定するときに使用します。このジョブ定義でジョブを送信する場合、inputfile や outputfile のような値に上書きしてパラメータを指定します。parameters セクションは codec のためのデフォルト設定ですが、必要に応じてパラメータを上書きできます。

詳細については、「パラメータ」を参照してください。

```
{
    "jobDefinitionName": "ffmpeg_parameters",
    "type": "container",
    "parameters": {"codec": "mp4"},
    "containerProperties": {
        "image": "my_repo/ffmpeg",
        "resourceRequirements": [
            {
                "type": "MEMORY",
                "value": "2000"
            },
            {
                "type": "VCPU",
                "value": "2"
            }
        ],
        "command": [
            "ffmpeg",
            "-i",
```

```
            "Ref::inputfile",
            "-c",
            "Ref::codec",
            "-o",
            "Ref::outputfile"
        ],
        "jobRoleArn": "arn:aws:iam::123456789012:role/ECSTask-S3FullAccess",
        "user": "nobody"
    }
}
```

GPU 機能のテスト

次のジョブ定義の例では、GPU ワークロードの AMI を使用する で説明されている GPU ワークロード AMI が適切に設定されているかどうかをテストします。このジョブ定義の例では、GitHub から TensorFlow ディープ MNIST 分類子の例を実行します。

```
{
    "containerProperties": {
        "image": "tensorflow/tensorflow:1.8.0-devel-gpu",
        "resourceRequirements": [
            {
                "type": "MEMORY",
                "value": "32000"
            },
            {
                "type": "VCPU",
                "value": "8"
            }
        ],
        "command": [
            "sh",
            "-c",
            "cd /tensorflow/tensorflow/examples/tutorials/mnist; python mnist_deep.py"
        ]
    },
    "type": "container",
    "jobDefinitionName": "tensorflow_mnist_deep"
}
```

上の JSON テキストで [tensorflow_mnist_deep.json] という名前のファイルを作成し、次のコマンドを使用して AWS Batch ジョブ定義を登録できます。

```
aws batch register-job-definition --cli-input-json file://tensorflow_mnist_deep.json
```

マルチノード並列ジョブ

以下のジョブ定義の例では、マルチノード並列ジョブを示しています。詳細については、AWS Compute ブログの「AWS Batch のマルチノード並列ジョブによる緊密に結合した分子動力学ワークフローの構築」を参照してください。

```
{
  "jobDefinitionName": "gromacs-jobdef",
  "jobDefinitionArn": "arn:aws:batch:us-east-2:123456789012:job-definition/gromacs-
jobdef:1",
  "revision": 6,
  "status": "ACTIVE",
  "type": "multinode",
  "parameters": {},
  "nodeProperties": {
    "numNodes": 2,
    "mainNode": 0,
    "nodeRangeProperties": [
      {
        "targetNodes": "0:1",
        "container": {
          "image": "123456789012.dkr.ecr.us-east-2.amazonaws.com/gromacs_mpi:latest",
          "resourceRequirements": [
              {
                  "type": "MEMORY",
                  "value": "24000"
              },
              {
                  "type": "VCPU",
                  "value": "8"
              }
          ],
          "command": [],
          "jobRoleArn": "arn:aws:iam::123456789012:role/ecsTaskExecutionRole",
          "ulimits": [],
          "instanceType": "p3.2xlarge"
        }
      }
    ]
  }
```

```
}
```

ジョブ

ジョブは、 によって開始される作業の単位です AWS Batch。ジョブは、ECSクラスター内の Amazon ECSコンテナインスタンス上で実行されるコンテナ化されたアプリケーションとして呼び出すことができます。

コンテナ化されたジョブは、コンテナイメージ、コマンド、およびパラメータを参照できます。詳細については、「JobDefinition」を参照してください。

多数の独立したシンプルなジョブを送信できます。

トピック

- チュートリアル: ジョブを送信する
- のサービスジョブ AWS Batch
- ジョブの状態
- AWS Batch ジョブ環境変数
- ジョブの再試行の自動化
- ジョブの依存関係
- ジョブのタイムアウト
- Amazon EKS ジョブ
- マルチノード並列ジョブ
- Amazon EKS のマルチノード並列ジョブ
- 配列ジョブ
- GPU ジョブを実行する
- AWS Batch ジョブキューでジョブを表示する
- ジョブキュー内のジョブ AWS Batch を検索する
- AWS Batch ジョブのネットワークモード
- CloudWatch Logs で AWS Batch ジョブログを表示する
- AWS Batch ジョブ情報の確認

チュートリアル: ジョブを送信する

ジョブ定義を登録したら、 AWS Batch ジョブとして ジョブキューに送信できます。ジョブ定義で指定されたパラメーターの多くは、実行時にオーバーライドできます。

ジョブを送信する方法

1. https://console.aws.amazon.com/batch/ で AWS Batch コンソールを開きます。
2. ナビゲーションバーから、 AWS リージョン 使用する を選択します。
3. ナビゲーションペインで ジョブを選択します。
4. ジョブの送信を選択します。
5. 名前 に、一意のジョブ定義名を入力します。名前の最大長は 128 文字です。大文字、小文字、数字、ハイフン (-)、アンダースコア (_) を含めることができます。
6. ジョブ定義で、作成済みのジョブ定義を選択します。詳細については、シングルノードのジョブ定義を作成するを参照してください。
7. ジョブキューで、既存のジョブキューを選択します。詳細については、ジョブキューを作成するを参照してください。
8. [ジョブ依存関係] で [ジョブ依存関係を追加] を選択します。
   - ジョブ IDには、すべての依存関係のジョブ ID を入力します。次に ジョブの依存関係を追加 を選択します。ジョブは最大 20 個の依存関係を持つことができます。詳細については、ジョブの依存関係を参照してください。
9. (配列ジョブのみ) [Array size] (配列サイズ) で、配列サイズを 2 から 10,000 の間で指定します。
10. (オプション)タグを展開し、タグを追加 を選択してリソースにタグを追加します。キーとオプショナル値を入力し、新しいタグを追加を選択します。
11. 次のページ を選択します。
12. ジョブオーバーライドセクションで:
    a. (オプショナル) スケジュールの優先度 には、0 から 100 までのスケジューリング優先度の値を入力します。値が大きいほど優先度が高くなります。
    b. (オプショナル) Job 試行回数 には、 AWS Batch ジョブをステータスに移行しようとするRUNNABLE 最大回数を入力します。1から10までの数字を入力できます。詳細については、ジョブの再試行の自動化を参照してください。

c. (オプション) 実行タイムアウト で、タイムアウト値 (秒単位) を入力します。実行タイムアウトは、未完了のジョブが終了するまでの時間です。試行がタイムアウト時間を超えた場合、試行は中止され、FAILEDのステータスに移行します。詳細については、ジョブのタイムアウトを参照してください。最小値は 60 秒です。

> **Important**
>
> Fargateリソースで実行されるジョブが14日以上実行されることを当てにしないでください。14日後、ファーゲートのリソースは使用できなくなり、仕事が打ち切られる可能性があります。

d. (オプション) タグを伝播 をオンにして、タグをジョブとジョブ定義から Amazon ECS タスクに伝達することができます。

13. [Additional configuration] (追加設定) を展開します。
14. (オプション) 再試行戦略の条件 では、終了時に評価を追加 を選択します。少なくとも 1 つのパラメータ値を入力し、アクション を選択します。条件セットごとに、アクション を 再試行 または 終了 に設定する必要があります。これらのアクションは、以下のことを意味します。

- 再試行 — 指定したジョブ試行回数に達するまで AWS Batch 再試行します。
- 終了 – ジョブの再試行を AWS Batch 停止します。

> **Important**
>
> 終了時に評価を追加 を選択した場合は、少なくとも 1 つのパラメータを設定してアクション を選択するか、終了時に評価を削除 を選択します。

15. パラメーター で パラメーターの追加 を選択し、パラメーター置換プレースホルダーを追加します。キーを入力し、オプションで 値を入力します。
16. コンテナオーバーライドセクションで:

a. [コマンド] では、コマンドを JSON 文字列配列に相当するものとしてフィールドに入力します。

このパラメータは、Docker Remote API の コンテナの作成 セクションにある Cmd にマッピングされ、COMMAND パラメータは docker run にマッピングされます。Docker CMD パラメータの詳細については、https://docs.docker.com/engine/reference/builder/#cmd を参照してください。

> **Note**
>
> このパラメータには空の文字列を含めることはできません。

b. vCPU で、コンテナ用に予約する vCPU の数を指定します。このパラメータは、Docker Remote API の コンテナの作成 セクションにある `CpuShares` にマッピングされ、`--cpu-shares` オプションは docker run にマッピングされます。各 vCPU は 1,024 個の CPU 配分に相当します。少なくとも 1 つの vCPU を指定する必要があります。

c. メモリ で、コンテナで使用できるメモリ制限を入力します。コンテナは、ここで指定したメモリを超えようとすると、停止されます。このパラメータは、Docker Remote API のコンテナの作成セクションの `Memory` にマップされ、`--memory` オプションは docker run にマップされます。ジョブに対して少なくとも 4 MiB のメモリを指定する必要があります。

> **Note**
>
> 特定のインスタンスタイプのジョブにメモリの優先順位を付けることで、リソース使用率を最大化できます。詳細については、コンピューティングリソースメモリの管理を参照してください。

d. (オプショナル) GPU の数 には、コンテナ用に予約する GPU の数を選択します。

e. (オプション) 環境変数 で 環境変数を追加 を選択し、環境変数を名前と値のペアとして追加します。これらの変数は、コンテナに渡されます。

f. 次のページ を選択します。

g. ジョブレビューについては、設定手順を確認してください。変更する必要がある場合は、[Edit] (編集) を選択します。完了したら、ジョブ定義の作成 を選択します。

のサービスジョブ AWS Batch

AWS Batch サービスジョブを使用すると、 AWS Batch ジョブキューを介して AWS サービスにリクエストを送信できます。現在、 は SageMaker トレーニングジョブをサービスジョブとして AWS Batch サポートしています。が基盤となるコンテナ実行 AWS Batch を管理するコンテナ化されたジョブとは異なり、サービスジョブでは AWS Batch がジョブのスケジュールとキューイング機能を提供できますが、ターゲット AWS サービス (SageMaker AI など) は実際のジョブ実行を処理します。

AWS Batch SageMaker トレーニングジョブの場合、データサイエンティストは優先度の高いトレーニングジョブを設定可能なキューに送信できるため、リソースが利用可能になるとすぐにワークロードが介入することなく実行できます。この機能は、リソースの調整、偶発的な過剰支出の防止、予算の制約への対応、リザーブドインスタンスによるコストの最適化、チームメンバー間の手動調整の必要性の排除といった一般的な課題に対処します。

サービスジョブは、いくつかの重要な点でコンテナ化されたジョブとは異なります。

- ジョブの送信: サービスジョブは SubmitServiceJob API を使用して送信する必要があります。サービスジョブは AWS Batch コンソールから送信できません。
- ジョブ実行: サービスジョブを AWS Batch スケジュールしてキューに入れますが、ターゲット AWS サービスは実際のジョブワークロードを実行します。
- リソース識別子: サービスジョブは、「job」ではなく「service-job」を含む ARN を使用して、コンテナ化されたジョブとの区別を行います。

SageMaker Training AWS Batch のサービスジョブの使用を開始するには、「」を参照してくださいthe section called “SageMaker AI AWS Batch での の開始方法”。

トピック

- のサービスジョブペイロード AWS Batch
- でサービスジョブを送信する AWS Batch
- AWS Batch サービスジョブのステータスを SageMaker AI ステータスにマッピングする
- のサービスジョブの再試行戦略 AWS Batch
- AWS Batch キュー内のサービスジョブをモニタリングする
- サービスジョブを終了する

のサービスジョブペイロード AWS Batch

SubmitServiceJob を使用してサービスジョブを送信するときは、ジョブを定義する 2 つの主要なパラメータとして `serviceJobType`、`serviceRequestPayload` を指定します。

- は、ジョブを実行する AWS サービス`serviceJobType`を指定します。SageMaker トレーニングジョブの場合、この値は `SAGEMAKER_TRAINING` です。
- `serviceRequestPayload` は、通常はターゲットサービスに直接送信される完全なリクエストを含む JSON エンコードされた文字列です。SageMaker トレーニングジョブの場合、このペイロードには SageMaker AI CreateTrainingJob API で使用するのと同じパラメータが含まれています。

使用可能なすべてのパラメータの完全なリストとその説明については、SageMaker AI CreateTrainingJob API リファレンスを参照してください。CreateTrainingJob でサポートされているすべてのパラメータをサービスジョブのペイロードに含めることができます。

その他のトレーニングジョブの設定例については、「SageMaker AI デベロッパーガイド」の「API、CLI、SDK」を参照してください。

PySDK にはヘルパークラスとユーティリティがあるため、サービスジョブの作成には PySDK を使用することをお勧めします。PySDK の使用例については、GitHub の「SageMaker AI の例」を参照してください。

サービスジョブのペイロードの例

次の例は、「hello world」トレーニングスクリプトを実行する SageMaker トレーニングジョブのシンプルなサービスジョブのペイロードを示しています。

このペイロードは、SubmitServiceJob を呼び出すときに serviceRequestPayload パラメータに JSON 文字列として渡されます。

```
{
  "TrainingJobName": "my-simple-training-job",
  "RoleArn": "arn:aws:iam::123456789012:role/SageMakerExecutionRole",
  "AlgorithmSpecification": {
    "TrainingInputMode": "File",
    "TrainingImage": "763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-
training:2.0.0-cpu-py310",
    "ContainerEntrypoint": [
      "echo",
      "hello world"
    ]
  },
  "ResourceConfig": {
    "InstanceType": "ml.c5.xlarge",
    "InstanceCount": 1,
    "VolumeSizeInGB": 1
  },
  "OutputDataConfig": {
    "S3OutputPath": "s3://your-output-bucket/output"
  },
  "StoppingCondition": {
    "MaxRuntimeInSeconds": 30
  }
```

```
}
```

でサービスジョブを送信する AWS Batch

サービスジョブを送信するには AWS Batch、SubmitServiceJob API を使用します。ジョブは、AWS CLI または SDK を使用して送信できます。

実行ロールがまだない場合は、サービスジョブを送信する前に作成する必要があります。SageMaker AI 実行ロールを作成するには、「SageMaker AI デベロッパーガイド」の「SageMaker AI 実行ロールの使用方法」を参照してください。

サービスジョブの送信ワークフロー

サービスジョブを送信すると、 は次のワークフロー AWS Batch を実行します。

1. AWS Batch はSubmitServiceJobリクエストを受け取り、 AWS Batch特定のパラメータを検証します。serviceRequestPayload は検証なしで渡されます。
2. ジョブは SUBMITTED 状態になり、指定されたジョブキューに配置されます。
3. AWS Batch は、キューの前面にあるRUNNABLEジョブに対してサービス環境に使用可能な容量があるかどうかを評価します。
4. 容量が利用可能な場合、ジョブは SCHEDULED に移動し、ジョブは SageMaker AI に渡されます。
5. 容量が取得され、SageMaker AI がサービスジョブデータをダウンロードすると、サービスジョブは初期化を開始し、ジョブは STARTING に変更されます。
6. SageMaker AI がジョブの実行を開始すると、ステータスは RUNNING に変更されます。
7. SageMaker AI がジョブを実行する間、 は進行状況 AWS Batch を監視し、サービスの状態を AWS Batch ジョブの状態にマッピングします。サービスジョブの状態のマッピング方法の詳細については、「???」を参照してください。
8. サービスジョブが完了すると、SUCCEEDED に移動し、出力をダウンロードする準備が整います。

前提条件

サービスジョブを送信する前に、以下があることを確認してください。

- サービス環境 – 容量制限を定義するサービス環境。詳細については、「AWS Batchのサービス環境を作成する」を参照してください。

- SageMaker ジョブキュー – ジョブスケジューリングを提供する SageMaker ジョブキュー。詳細については、「AWS Batch で SageMaker トレーニングジョブのキューを作成する」を参照してください。
- IAM アクセス許可 – AWS Batch ジョブキューとサービス環境を作成および管理するためのアクセス許可。詳細については、「AWS Batch IAM ポリシー、ロール、アクセス許可」を参照してください。

CLI AWS を使用してサービスジョブを送信する

AWS CLI を使用してサービスジョブを送信する方法を以下に示します。

```
aws batch submit-service-job \
    --job-name "my-sagemaker-training-job" \
    --job-queue "my-sagemaker-job-queue" \
    --service-job-type "SAGEMAKER_TRAINING" \
    --service-request-payload '{\"TrainingJobName\": \"sagemaker-training-job-
example\", \"AlgorithmSpecification\": {\"TrainingImage\": \"123456789012.dkr.ecr.us-
east-1.amazonaws.com/pytorch-inference:1.8.0-cpu-py3\", \"TrainingInputMode
\": \"File\", \"ContainerEntrypoint\":  [\"sleep\", \"1\"]}, \"RoleArn\":
\"arn:aws:iam::123456789012:role/SageMakerExecutionRole\", \"OutputDataConfig\":
 {\"S3OutputPath\": \"s3://example-bucket/model-output/\"}, \"ResourceConfig\":
 {\"InstanceType\": \"ml.m5.large\", \"InstanceCount\": 1, \"VolumeSizeInGB\": 1}}'
    --client-token "unique-token-12345"
```

`serviceRequestPayload` パラメータの詳細については、「the section called “サービスジョブのペイロード”」を参照してください。

AWS Batch サービスジョブのステータスを SageMaker AI ステータスにマッピングする

SubmitServiceJob を使用して SageMaker ジョブキューにジョブを送信すると、 はジョブのライフサイクル AWS Batch を管理し、 AWS Batch ジョブの状態を同等の SageMaker トレーニングジョブの状態にマッピングします。SageMaker トレーニングジョブなどのサービスジョブは、従来のコンテナジョブとは異なる状態ライフサイクルに従います。サービスジョブはほとんどの状態をコンテナジョブと共有しますが、`SCHEDULED` 状態を導入し、特にターゲットサービスからの容量不足エラーを処理するために、さまざまな再試行動作を示します。

次の表は、 AWS Batch ジョブの状態と対応する SageMaker Status/SecondaryStatus を示しています。

バッチステータス	SageMaker AI プライマリステータス	SageMaker AI セカンダリステータス	説明
SUBMITTED	該当なし	該当なし	ジョブがキューに送信され、スケジューラの評価を待機しています。
RUNNABLE	該当なし	該当なし	ジョブはキューに入れられ、スケジューリングの準備が整います。この状態のジョブは、サービス環境で十分なリソースが使用可能になるとすぐに開始されます。十分なリソースが使用不可の場合、この状態が無限に続くことがあります。
SCHEDULED	InProgress	Pending	サービスジョブが SageMaker AI に正常に送信されました
STARTING	InProgress	Downloading	データとイメージをダウンロード中の SageMaker トレーニングジョブ。トレーニングジョブの容量が取得され、ジョブの初期化が開始されました。
RUNNING	InProgress	Training	SageMaker トレーニングジョブの実行アルゴリズム
RUNNING	InProgress	Uploading	トレーニング完了後に出力アーティファクトをアップロードする SageMaker トレーニングジョブ
SUCCEEDED	Completed	Completed	SageMaker トレーニングジョブが正常に完了しました。出力アーティファクトのアップロードが完了しました。
FAILED	Failed	Failed	SageMaker トレーニングジョブで回復不可能なエラーが発生しました。

バッチステータス	SageMaker AI プライマリステータス	SageMaker AI セカンダリステータス	説明
FAILED	Stopped	Stopped	SageMaker トレーニングジョブが StopTrainingJob を使用して手動で停止されました。

のサービスジョブの再試行戦略 AWS Batch

サービスジョブの再試行戦略により AWS Batch 、 は特定の条件下で失敗したサービスジョブを自動的に再試行できます。

サービスジョブには、いくつかの理由で複数回の試行が必要になる場合があります。

- 一時的なサービスの問題: 内部サービスエラー、スロットリング、または一時的な停止により、送信中または実行中にジョブが失敗する可能性があります。
- トレーニング初期化の失敗: イメージのプルの問題や初期化エラーなど、ジョブの起動中の問題は再試行時に解決される可能性があります。

適切な再試行戦略を設定することで、ジョブの成功率を向上させ、特に長時間実行されるトレーニングのワークロードにおける手動介入の必要性を減らすことができます。

> **Note**
>
> サービスジョブは、設定された再試行を消費することなく、容量不足エラーなど、特定のタイプの失敗を自動的に再試行します。再試行戦略は、アルゴリズムエラーやサービスの問題などの他のタイプの失敗を主に処理します。

再試行戦略の設定

サービスジョブの再試行戦略は、シンプルな再試行回数と条件付き再試行ロジックの両方をサポートする ServiceJobRetryStrategy を使用して設定されます。

再試行設定

最も簡単な再試行戦略では、サービスジョブが失敗した場合に実行する再試行回数を指定します。

```
{
  "retryStrategy": {
    "attempts": 3
  }
}
```

この設定では、サービスジョブが失敗した場合に最大 3 回再試行できます。

> **Important**
>
> attempts 値は、最初の試行を含め、ジョブを RUNNABLE 状態に配置できる合計回数を表します。3 の値は、ジョブが最初に 1 回試行され、失敗するとさらに 2 回再試行されることを意味します。

evaluateOnExit で設定を再試行する

evaluateOnExit パラメータを使用して、ジョブを再試行するか、失敗を許可する条件を指定できます。これは、さまざまなタイプの失敗で異なる処理が必要な場合に便利です。

evaluateOnExit 配列には最大 5 つの再試行戦略を含めることができ、それぞれがステータス理由に基づいてアクション (RETRY または EXIT) と条件を指定します。

```
{
  "retryStrategy": {
    "attempts": 5,
    "evaluateOnExit": [
      {
        "action": "RETRY",
        "onStatusReason": "Received status from SageMaker: InternalServerError*"
      },
      {
        "action": "EXIT",
        "onStatusReason": "Received status from SageMaker: ValidationException*"
      },
      {
        "action": "EXIT",
        "onStatusReason": "*"
      }
    ]
  }
```

```
}
```

この設定では、次のようになります。

- SageMaker AI の内部サーバーエラーが原因で失敗したジョブを再試行する
- 検証例外 (再試行によって解決されないクライアントエラー) が発生したジョブをすぐに失敗させる
- 他の失敗タイプに対して終了するキャッチオールルールを含める

ステータス理由のパターンマッチング

onStatusReason パラメータは、最大 512 文字のパターンマッチングをサポートします。パターンはワイルドカード (*) を使用し、SageMaker AI によって返されるステータス理由と照合できます。

サービスジョブの場合、SageMaker AI からのステータスメッセージには、SageMaker からのステータスの受信: AWS Batch」というプレフィックスが付けられ、生成されたメッセージと区別されます。一般的なパターンは次のとおりです。

- `Received status from SageMaker: InternalServerError*` - 内部サービスエラーの一致
- `Received status from SageMaker: ValidationException*` - クライアント検証エラーの一致
- `Received status from SageMaker: ResourceLimitExceeded*` - リソース制限エラーの一致
- `*CapacityError*` - 容量関連の失敗の一致

> **Tip**
>
> 特定のパターンマッチングを使用して、さまざまなエラータイプを適切に処理します。例えば、内部サーバーエラーを再試行しますが、ジョブパラメータの問題を示す検証エラーですぐに失敗します。

AWS Batch キュー内のサービスジョブをモニタリングする

`list-service-jobs` および `get-job-queue-snapshot` を使用して、SageMaker トレーニングジョブキュー内のジョブのステータスをモニタリングできます。

キューで実行中のジョブを表示します:

```
aws batch list-service-jobs \
  --job-queue my-sm-training-fifo-jq \
  --job-status RUNNING
```

キューで待機しているジョブを表示する:

```
aws batch list-service-jobs \
  --job-queue my-sm-training-fifo-jq \
  --job-status RUNNABLE
```

SageMaker に送信されたものの、まだ実行されていないジョブを表示します:

```
aws batch list-service-jobs \
  --job-queue my-sm-training-fifo-jq \
  --job-status SCHEDULED
```

キューの前面でジョブのスナップショットを取得します:

```
aws batch get-job-queue-snapshot --job-queue my-sm-training-fifo-jq
```

このコマンドは、キュー内の今後のサービスジョブの順序を示します。

詳細なサービスジョブ情報を取得する

DescribeServiceJob オペレーションを使用して、現在のステータス、サービスリソース識別子、詳細な試行情報など、特定のサービスジョブに関する包括的な情報を取得します。

特定のジョブに関する詳細情報を表示する:

```
aws batch describe-service-job \
  --job-id a4d6c728-8ee8-4c65-8e2a-9a5e8f4b7c3d
```

このコマンドは、次のようなジョブに関する包括的な情報を返します。

- ジョブ ARN と現在のステータス
- サービスリソース識別子 (SageMaker トレーニングジョブ ARN など)

- 優先度と再試行設定のスケジューリング
- 元のサービスパラメータを含むサービスリクエストペイロード
- 開始時刻と停止時刻を含む詳細な試行情報
- ターゲットサービスからのステータスメッセージ

SageMaker トレーニングジョブをモニタリングする

を使用して SageMaker トレーニングジョブをモニタリングする場合 AWS Batch、 AWS Batch ジョブ情報と基盤となる SageMaker トレーニングジョブの詳細の両方にアクセスできます。

ジョブ詳細のサービスリソース識別子には、SageMaker トレーニングジョブ ARN が含まれます。

```
{
  "latestAttempt": {
    "serviceResourceId": {
      "name": "TrainingJobArn",
      "value": "arn:aws:sagemaker:us-east-1:123456789012:training-job/my-training-job"
    }
  }
}
```

この ARN を使用して、SageMaker から直接追加の詳細を取得できます。

```
aws sagemaker describe-training-job \
  --training-job-name my-training-job
```

AWS Batch ステータスと SageMaker Training ジョブのステータスの両方を確認して、ジョブの進行状況をモニタリングします。 AWS Batch ジョブのステータスはジョブのライフサイクル全体を示し、SageMaker トレーニングジョブのステータスはトレーニングプロセスに関するサービス固有の詳細を提供します。

サービスジョブを終了する

TerminateServiceJob オペレーションを使用して、実行中のサービスジョブを停止します。

特定のサービスジョブを終了する:

```
aws batch terminate-service-job \
```

```
  --job-id a4d6c728-8ee8-4c65-8e2a-9a5e8f4b7c3d \
  --reason "Job terminated by user request"
```

サービスジョブを終了すると、 はジョブ AWS Batch を停止し、ターゲットサービスに通知します。SageMaker トレーニングジョブの場合、これにより SageMaker AI のトレーニングジョブも停止します。

ジョブの状態

ジョブキューに AWS Batch ジョブを送信すると、ジョブは SUBMITTED状態になります。その後、以下の状態を経由して完了 (コード 0 で終了) または失敗 (0 以外のコードで終了) します。 AWS Batch ジョブの各状態は以下のとおりです。

SUBMITTED

キューに送信され、まだスケジューラによって評価されていないジョブです。スケジューラは、ジョブを評価し、他のジョブの正常な完了に依存するかどうか (未処理の依存関係があるかどうか) を判断します。依存関係がある場合、ジョブの状態は PENDING に移行します。依存関係がない場合、ジョブの状態は RUNNABLE に移行します。

PENDING

ジョブはキュー内にあり、別のジョブやリソースへの依存関係があるため、まだ実行できません。依存関係が満たされると、ジョブの状態は RUNNABLE に移行します。

> **Note**
>
> 配列ジョブの親は、子ジョブが に更新PENDINGされると に更新RUNNABLEされ、子ジョブの実行中に PENDINGステータスのままになります。これらのジョブを表示するには、すべての子ジョブが終了状態になるまでPENDINGステータスでフィルタリングします。

RUNNABLE

ジョブはキュー内にあり、未処理の依存関係はありません。したがって、ホストにスケジューリングされる準備ができています。この状態のジョブは、ジョブのキューにマッピングされたコンピューティング環境のいずれかで十分なリソースが使用可能になり次第、開始されます。ただし、十分なリソースが使用不可の場合、この状態が無限に続くことがあります。

> **Note**
>
> ジョブが STARTING に進行しない場合は、トラブルーシューティングセクションのRUNNABLE 状態でジョブが止まるを参照してください。

STARTING

これらのジョブはホストにスケジュールされており、関連するコンテナ初期化操作が進行中です。コンテナイメージがプルされてコンテナが稼働状態になると、ジョブの状態は RUNNING に移行します。

イメージのプル期間、Amazon EKS initContainer の完了期間、Amazon ECS containerDependency 解決期間は STARTING 状態で発生します。ジョブのイメージをプルするのにかかる時間は、ジョブが STARTING 状態にある時間と同じです。

例えば、ジョブのイメージをプルするのに 3 分かかる場合、ジョブは 3 分間 STARTING 状態になります。initContainers の完了に合計 10 分かかる場合、Amazon EKS ジョブは 10 分間 STARTING 状態になります。Amazon ECS ジョブに複数の Amazon ECS containerDependencies がある場合、すべてのコンテナ依存関係 (それらのランタイム) が解決されるまで、ジョブは STARTING 状態になります。STARTING はタイムアウトに含まれないため、期間は RUNNING から始まります。詳細については、「ジョブの状態」を参照してください。

RUNNING

ジョブは、コンピューティング環境内の Amazon ECS コンテナインスタンスでコンテナジョブとして実行中です。ジョブのコンテナが終了すると、プロセス終了コードでジョブの正否が判定されます。終了コードの 0 は成功を示し、ゼロ以外の終了コードは失敗を示します。試行が失敗したジョブの再試行戦略に設定されている再試行回数が残っている場合は、ジョブの状態が RUNNABLE に移行します。詳細については、ジョブの再試行の自動化を参照してください。

> **Note**
>
> RUNNING ジョブのログは クラウドウォッチのログ(CloudWatch Logs) で確認できます。ロググループは /aws/batch/job、ログストリーム名形式は *first200CharsOfJobDefinitionName*/default/*ecs_task_id* です。この形式は将来変わる可能性があります。
> ジョブが RUNNING ステータスになったら、DescribeJobs API オペレーションを使用してそのログストリーム名をプログラムで取得できます。詳しくは、Amazon CloudWatch

Logs ユーザーガイドのCloudWatch Logs に送信されたログデータを表示するを参照してください。デフォルトでは、これらのログには有効期限はありません。ただし、保存期間を変更することは可能です。詳細については、Amazon CloudWatch Logs ユーザーガイドの CloudWatch Logs でのログデータ保管期間の変更 を参照してください。

`SUCCEEDED`

ジョブは、終了コード `0` で正常に完了しました。ジョブのSUCCEEDEDジョブ状態は、少なくとも 7 日間 AWS Batch に保持されます。

> **Note**
>
> SUCCEEDED ジョブのログは クラウドウォッチのログ(CloudWatch Logs) で確認できます。ロググループは /aws/batch/job、ログストリーム名形式は*first200CharsOfJobDefinitionName*/default/*ecs_task_id*です。この形式は将来に変更される可能性があります。
> ジョブが RUNNING ステータスになったら、DescribeJobs API オペレーションを使用してそのログストリーム名をプログラムで取得できます。詳しくは、Amazon CloudWatch Logs ユーザーガイドのCloudWatch Logs に送信されたログデータを表示するを参照してください。デフォルトでは、これらのログの有効期限はありません。ただし、保存期間を変更することは可能です。詳細については、Amazon CloudWatch Logs ユーザーガイドの CloudWatch Logs でのログデータ保管期間の変更 を参照してください。

`FAILED`

ジョブのすべての使用可能な試行が失敗しました。FAILED のジョブの状態は、少なくとも7日間 AWS Batch に保存されます。

> **Note**
>
> FAILED ジョブのログは クラウドウォッチのログ(CloudWatch Logs) で確認できます。ロググループは /aws/batch/job、ログストリーム名形式は*first200CharsOfJobDefinitionName*/default/*ecs_task_id*です。この形式は将来に変更される可能性があります。
> ジョブが RUNNING ステータスになったら、DescribeJobs API オペレーションを使用してそのログストリームをプログラムで取得できます。詳しくは、Amazon CloudWatch Logs

ユーザーガイドのCloudWatch Logs に送信されたログデータを表示するを参照してください。デフォルトでは、これらのログには有効期限はありません。ただし、保存期間を変更することは可能です。詳細については、Amazon CloudWatch Logs ユーザーガイドの CloudWatch Logs でのログデータ保管期間の変更 を参照してください。

AWS Batch ジョブ環境変数

AWS Batch は、コンテナジョブで特定の環境変数を設定します。これらの環境変数は、ジョブ内のコンテナに対するイントロスペクションを提供します。これらの変数の値は、アプリケーションのロジックで使用できます。が AWS Batch 設定するすべての変数はAWS_BATCH_、 プレフィックスで始まります。これは、保護された環境変数プレフィックスです。このプレフィックスは、ジョブ定義やオーバーライド内の独自の変数には使用できません。

ジョブコンテナでは、次の環境変数を使用できます。

`AWS_BATCH_CE_NAME`

この変数は、ジョブが配置されるコンピューティング環境の名前に設定されます。

`AWS_BATCH_JOB_ARRAY_INDEX`

この変数は、子配列ジョブでのみ設定されます。配列ジョブのインデックスは 0 から始まり、各子ジョブは一意のインデックス番号を受け取ります。例えば、10 の子を持つ配列ジョブのインデックス値は 0 ～ 9 です。このインデックス値を使用して、配列ジョブの子がどのように区別されるかを制御できます。詳細については、配列ジョブインデックスを使用してジョブの区別を制御するを参照してください。

`AWS_BATCH_JOB_ARRAY_SIZE`

この変数は、親配列ジョブのサイズに設定されます。親配列ジョブのサイズは、この変数の子配列ジョブに渡されます。

`AWS_BATCH_JOB_ATTEMPT`

この変数は、ジョブ試行番号に設定されます。最初の試行番号は 1 となります。詳細については、「ジョブの再試行の自動化」を参照してください。

`AWS_BATCH_JOB_ID`

この変数は AWS Batch ジョブ ID に設定されます。

AWS_BATCH_JOB_KUBERNETES_NODE_UID

この変数は、ポッドが実行されている Kubernetes クラスター内にあるノードオブジェクトの Kubernetes クラスター UID として設定されます。この変数は、Amazon EKS リソースで実行されるジョブにのみ設定されます。詳細については、「Kubernetes ドキュメンテーション」の「UID」を参照してください。

AWS_BATCH_JOB_MAIN_NODE_INDEX

この変数はマルチノード並列ジョブの子ノードでのみ設定されます。この変数は、ジョブの主要なノードのインデックス番号に設定されます。アプリケーションコードは、AWS_BATCH_JOB_MAIN_NODE_INDEX と AWS_BATCH_JOB_NODE_INDEX を単一ノードで比較して、これが主要なノードであるかを確認できます。

AWS_BATCH_JOB_MAIN_NODE_PRIVATE_IPV4_ADDRESS

この変数はマルチノード並列ジョブの子ノードでのみ設定されます。この変数はメインノードには存在しないが、ジョブのメインノードのプライベートIPv4アドレスに設定されます。子ノードのアプリケーションコードは、このアドレスを使用して主要なノードと通信できます。

AWS_BATCH_JOB_NODE_INDEX

この変数はマルチノード並列ジョブの子ノードでのみ設定されます。この変数は、ノードのノードインデックス番号に設定されます。ノードインデックスは 0 で始まり、各ノードは一意のインデックス番号を受け取ります。例えば、10 の子を持つマルチノード並列ジョブのインデックス値は 0 ～ 9 です。

AWS_BATCH_JOB_NUM_NODES

この変数はマルチノード並列ジョブの子ノードでのみ設定されます。この変数は、マルチノード並列ジョブにリクエストしたノードの数に設定されます。

AWS_BATCH_JQ_NAME

この変数は、ジョブが送信されたジョブキューの名前に設定されます。

ジョブの再試行の自動化

ジョブおよびジョブ定義に再試行戦略を適用し、失敗したジョブを自動的に再試行できます。考えられる故障のシナリオは以下の通りです：

- コンテナジョブのゼロ以外の終了コード

- Amazon EC2 インスタンスの障害または終了
- 内部 AWS サービスエラーまたは停止

ジョブがジョブキューに送信されて RUNNING 状態になると、1 回の試行とみなされます。デフォルトでは、各ジョブは 1 回の試行で SUCCEEDED または FAILED のいずれかの状態に移行します。ただし、ジョブ定義とジョブ送信の両方のワークフローで、1〜10 回の再試行戦略を指定できます。EvaluateOnExit を指定すると、最大 5 つの再試行戦略を含めることができます。EvaluateOnExitが指定されているが、いずれの再試行方法も一致しない場合、ジョブは再試行されます。終了と一致しないジョブについては、何らかの理由で終了する最後のエントリを追加します。例えば、evaluateOnExit このオブジェクトには、アクションがの 2 RETRY つのエントリと、EXIT アクションがの最後のエントリがあります。

```
"evaluateOnExit": [
    {
        "action": "RETRY",
        "onReason": "AGENT"
    },
    {
        "action": "RETRY",
        "onStatusReason": "Task failed to start"
    },
    {
        "action": "EXIT",
        "onReason": "*"
    }
]
```

ランタイムに、AWS_BATCH_JOB_ATTEMPT 環境変数がコンテナの対応するジョブ試行番号に設定されます。最初の試行は 1 番となり、後続の試行は昇順の番号 (2、3、4 など) になります。

例えば、何らかの理由でジョブの試行が失敗し、再試行設定で指定されている試行回数がその AWS_BATCH_JOB_ATTEMPT 回数よりも多いとします。その後、ジョブは元の RUNNABLE 状態に戻ります。詳細については、ジョブの状態を参照してください。

> **Note**
>
> ジョブをキャンセルまたは終了した場合、ジョブは再試行されません。無効なジョブ定義のためにジョブが失敗した場合も、ジョブは再試行されません。

詳細については、再試行戦略、シングルノードのジョブ定義を作成する、チュートリアル: ジョブを送信する および停止したタスクのエラーコード を参照してください。

ジョブの依存関係

AWS Batch ジョブを送信するときに、ジョブが依存するジョブ IDs を指定できます。これを行うと、ス AWS Batch ケジューラは、指定された依存関係が正常に完了した後にのみジョブが実行されるようにします。成功すると、依存するジョブが PENDING から RUNNABLE に移行し、その後 STARTING、RUNNING と移行します。いずれかのジョブ依存関係が失敗すると、依存するジョブは自動的に PENDING から FAILED に移行します。

例えば、ジョブ A は実行前に成功する必要がある他のジョブとの依存関係を最大 20 個記述できます。その後、ジョブ A および他の 19 個のジョブに依存する追加ジョブを送信できます。

配列ジョブの場合、ジョブ ID を指定せずに SEQUENTIAL タイプの依存関係を指定できます。こうすることで各子配列ジョブがインデックス 0 から開始して連続的に完了します。また、ジョブ ID を使用して N_TO_N タイプの依存関係を指定することもできます。この場合、このジョブの各インデックスの子は各依存関係の対応するインデックスの子が完了するまで待機してから開始されます。詳細については、「配列ジョブ」を参照してください。

依存関係を持つ AWS Batch ジョブを送信するには、「」を参照してくださいチュートリアル: ジョブを送信する。

リソース認識型スケジューリング では、ジョブの実行に必要な消費型リソースに基づいてジョブをスケジュールできます。ジョブの実行に必要な消費型リソースを指定すると、Batch はジョブをスケジュールするときにこれらのリソースの依存関係を考慮します。必要なすべてのリソースが利用可能なジョブのみを割り当てることで、コンピューティングリソースの使用率の低下を抑制できます。リソース対応スケジューリングは、FIFO スケジューリングポリシーと公平配分スケジューリングポリシーの両方で使用でき、EKS、ECS、Fargate など、Batch でサポートされているすべてのコンピューティングプラットフォームで使用できます。配列ジョブ、マルチノード並列 (MNP) ジョブ、および通常の Batch ジョブで使用できます。

ジョブのタイムアウト

この期間を超えてジョブが実行されると AWS Batch でジョブが終了するように、ジョブのタイムアウト期間を設定できます。例えば、15 分で完了することがわかっているジョブがあるとします。アプリケーションがループ状態に止まり、永続的に実行される場合に、止まったジョブを終了するためにタイムアウトを 30 分に設定できます。

> **Important**
>
> デフォルトでは、 AWS Batch にはジョブのタイムアウトはありません。ジョブタイムアウトを定義しない場合、ジョブはコンテナが終了するまで実行されます。

`attemptDurationSeconds` パラメータを指定します。ジョブ定義の際、またはジョブ送信時に行い、60 秒以上にする必要があります。ジョブ試行の`startedAt`タイムスタンプの後にこの秒数が経過すると、 はジョブ AWS Batch を終了します。コンピューティングリソースで、ジョブのコンテナは `SIGTERM` シグナルを受け取り、アプリケーションが適切にシャットダウンできるようにします。30 秒後にコンテナがまだ実行されている場合、`SIGKILL` シグナルが送信されてコンテナを強制的にシャットダウンします。

タイムアウトの終了はベストエフォートベースで処理されます。ジョブ試行がタイムアウトするタイミングでタイムアウトが終了するとは限りません (数秒長くかかることがあります)。アプリケーションで正確にタイムアウトを実行する必要がある場合は、アプリケーション内にこのロジックを実装します。多数のジョブを同時にタイムアウトする場合、タイムアウトの終了は先入れ先出しキューとして行われ、ジョブはバッチで終了します。

> **Note**
>
> AWS Batch ジョブの最大タイムアウト値はありません。

タイムアウト期間の超過で終了したジョブは再試行されません。ジョブ自体が原因でジョブ試行に失敗した場合、再試行が有効であれば再試行され、新しいジョブ試行のタイムアウトカウントダウンが開始します。

> **Important**
>
> Fargate リソースで実行されるジョブは、14 日を超えて実行することはできません。タイムアウト期間が 14 日を超えると、Fargate リソースが使用できなくなり、ジョブが終了します。

配列ジョブの場合、子ジョブは、親ジョブと同様にタイムアウト設定されています。

タイムアウト設定で AWS Batch ジョブを送信する方法については、「」を参照してくださいチュートリアル: ジョブを送信する。

Amazon EKS ジョブ

ジョブは、作業の最小単位です AWS Batch。Amazon EKS の AWS Batch ジョブには、ポッドへの one-to-one Kubernetes のマッピングがあります。 AWS Batch ジョブ定義は、 AWS Batch ジョブのテンプレートです。 AWS Batch ジョブを送信するときは、ジョブ定義を参照し、ジョブキューをターゲットにして、ジョブの名前を指定します。Amazon EKS の AWS Batch ジョブのジョブ定義では、eksProperties パラメータは、Amazon EKS AWS Batch ジョブで がサポートするパラメータのセットを定義します。SubmitJob リクエストでは、eksPropertiesOverride パラメーターを使用するといくつかの一般的なパラメーターをオーバーライドできます。これにより、複数のジョブにジョブ定義のテンプレートを使用できます。ジョブが Amazon EKS クラスターにディスパッチされると、はジョブを podspec () AWS Batch に変換しますKind： Pod。は、いくつかの追加の AWS Batch パラメータpodspecを使用して、ジョブが正しくスケーリングおよびスケジュールされるようにします。 AWS Batch はラベルとテイントを組み合わせて、ジョブが AWS Batch マネージドノードでのみ実行されるようにし、他のポッドがそれらのノードで実行されないようにします。

> **Important**
>
> - Amazon EKS ジョブ定義で hostNetworkパラメータが明示的に設定されていない場合、のポッドネットワークモードは AWS Batch デフォルトでホストモードになります。具体的には、hostNetwork=true と dnsPolicy=ClusterFirstWithHostNet という設定が適用されます。
> - AWS Batch は、ポッドがジョブを完了するとすぐにジョブポッドをクリーンアップします。ポッドアプリケーションログを表示するには、クラスターのロギングサービスを設定します。詳細については、「CloudWatch Logs を使用して Amazon EKS ジョブにおける AWS Batch をモニタリングする」を参照してください。

トピック

チュートリアル: 実行中のジョブをポッドとノードにマップする

実行中のジョブの podProperties には現在のジョブ試行用に podName と nodeName のパラメータが設定されています。これらのパラメータを表示するには、DescribeJobs API オペレーションを使用してください。

以下は出力の例です。

```
$ aws batch describe-jobs --job 2d044787-c663-4ce6-a6fe-f2baf7e51b04
{
 "jobs": [
  {
   "status": "RUNNING",
   "jobArn": "arn:aws:batch:us-east-1:123456789012:job/2d044787-c663-4ce6-a6fe-
f2baf7e51b04",
   "jobDefinition": "arn:aws:batch:us-east-1:123456789012:job-definition/
MyJobOnEks_SleepWithRequestsOnly:1",
   "jobQueue": "arn:aws:batch:us-east-1:123456789012:job-queue/My-Eks-JQ1",
   "jobId": "2d044787-c663-4ce6-a6fe-f2baf7e51b04",
   "eksProperties": {
    "podProperties": {
     "nodeName": "ip-192-168-55-175.ec2.internal",
     "containers": [
      {
       "image": "public.ecr.aws/amazonlinux/amazonlinux:2",
       "resources": {
        "requests": {
         "cpu": "1",
         "memory": "1024Mi"
        }
       }
      }
     ],
     "podName": "aws-batch.b0aca953-ba8f-3791-83e2-ed13af39428c"
    }
   }
  }
 ]
}
```

リトライが有効なジョブの場合、podName、nodeName の完了したすべての試行の終了は DescribeJobs API eksAttempts オペレーションのリストパラメータに含まれます。現在実行中の podName、nodeName の試行の終了は podProperties オブジェクト内にあります。

チュートリアル: 実行中のポッドをそのジョブにマップし直す

ポッドには、それが属するコンピューティング環境uuidの jobIdと を示すラベルがあります。は、ジョブのランタイムがジョブ情報を参照できるように環境変数 AWS Batch を挿入します。詳細

については、「AWS Batch ジョブ環境変数」を参照してください。以下のコマンドを実行すれば、この情報を見ることができる。出力は次のとおりです。

```
$ kubectl describe pod aws-batch.14638eb9-d218-372d-ba5c-1c9ab9c7f2a1 -n my-aws-batch-namespace
Name:         aws-batch.14638eb9-d218-372d-ba5c-1c9ab9c7f2a1
Namespace:    my-aws-batch-namespace
Priority:     0
Node:         ip-192-168-45-88.ec2.internal/192.168.45.88
Start Time:   Wed, 26 Oct 2022 00:30:48 +0000
Labels:       batch.amazonaws.com/compute-environment-uuid=5c19160b-
d450-31c9-8454-86cf5b30548f
              batch.amazonaws.com/job-id=f980f2cf-6309-4c77-a2b2-d83fbba0e9f0
              batch.amazonaws.com/node-uid=a4be5c1d-9881-4524-b967-587789094647
...
Status:       Running
IP:           192.168.45.88
IPs:
  IP:  192.168.45.88
Containers:
  default:
    Image:         public.ecr.aws/amazonlinux/amazonlinux:2
    ...
    Environment:
      AWS_BATCH_JOB_KUBERNETES_NODE_UID:  a4be5c1d-9881-4524-b967-587789094647
      AWS_BATCH_JOB_ID:                   f980f2cf-6309-4c77-a2b2-d83fbba0e9f0
      AWS_BATCH_JQ_NAME:                  My-Eks-JQ1
      AWS_BATCH_JOB_ATTEMPT:              1
      AWS_BATCH_CE_NAME:                  My-Eks-CE1

...
```

AWS Batch Amazon EKS ジョブがサポートする機能

以下は、Amazon EKS で実行されるKubernetesジョブにも共通する AWS Batch 特定の機能です。

- ジョブの依存関係
- 配列ジョブ
- ジョブのタイムアウト
- ジョブの再試行の自動化
- 公平配分スケジューリングを使用してジョブをスケジュールする

KubernetesSecrets および ServiceAccounts

AWS Batch は、 KubernetesSecretsおよび の参照をサポートしていますServiceAccounts。サービスアカウントの Amazon EKS IAM ロールを使用するようにポッドを設定できます。詳細については、Amazon EKS ユーザーガイド の Kubernetes サービスアカウントを使用するポッドの設定を参照してください。

関連ドキュメント

- Amazon EKS の AWS Batch におけるメモリと vCPU に関する考慮事項
- GPU ジョブを実行する
- RUNNABLE 状態でジョブが止まる

マルチノード並列ジョブ

マルチノード並列ジョブでは、複数の Amazon EC2 インスタンスにまたがる単一のジョブを実行できます。 AWS Batch のマルチノード並列ジョブ (ギャングスケジューリングとも呼ばれる) では、Amazon EC2 リソースを直接起動、設定、管理する必要なく、ラージスケールで密結合された高パフォーマンスのコンピューティングアプリケーションと分散された GPU モデルトレーニングを実行できます。 AWS Batch マルチノード並列ジョブは、IP ベースのノード間通信をサポートするフレームワークと互換性があります。例としては、アパッチ MX ネット(Apache MXNet)、TensorFlow、Caffe2、メッセージパッシングインターフェイス (MPI) などがあります。

マルチノード並列ジョブは、単一のジョブとして送信されます。ただし、ジョブ定義 (あるいは、ジョブ送信ノードの上書き) は、ジョブに作成するノードの数および作成するノードグループを指定します。各マルチノード並列ジョブには主要なノードが含まれ、まずこれが起動されます。主要なノードが確立したら、子ノードが起動されて開始します。ジョブは、メインノードが終了した場合にのみ終了します。その後、すべての子ノードが停止します。詳細については、ノードグループを参照してください。

マルチノード並列ジョブは、シングルテナントです。つまり、各 Amazon EC2 インスタンスごとに、単一のジョブコンテナのみが実行されます。

最終的なジョブステータス (SUCCEEDED あるいは FAILED) は、主要なノードの最終的なジョブステータスによって決定されます。マルチノード並列ジョブのステータスを取得するには、ジョブの送信時に返されるジョブ ID を使用して、ジョブを記述できます。子ノードの詳細が必要な場合には、各子ノードごとに個別に記述する必要があります。ノードは #*N* 表記を使用して対処されます (0 から開始)。たとえば、ジョブの 2 番目のノードの詳細にアクセスするには、 AWS Batch

DescribeJobs API オペレーションを使用して *aws_batch_job_id*#1 を記述します。マルチノード並列ジョブの started、stoppedAt、statusReason、exit 情報は、主要なノードから入力されます。

ジョブの再試行を指定した場合、メインノードに障害が発生すると、別の試行が行われます。子ノードに障害が発生しても、再試行回数は発生しません。マルチノード並列ジョブの新しい試行ごとに、関連付けられた子ノードに対応する試行が更新されます。

マルチノード並列ジョブを実行するには AWS Batch、アプリケーションコードに分散通信に必要なフレームワークとライブラリが含まれている必要があります。

トピック

- 環境変数
- ノードグループ
- MNP ジョブのライフサイクル
- を使用した MNP のコンピューティング環境に関する考慮事項 AWS Batch

環境変数

実行時に、各ノードには、すべての AWS Batch ジョブが受け取る標準環境変数が設定されます。さらに、ノードは、マルチノード並列ジョブに固有の次の環境変数で構成されます。

`AWS_BATCH_JOB_MAIN_NODE_INDEX`

この変数は、ジョブの主要なノードのインデックス番号に設定されます。アプリケーションコードは、AWS_BATCH_JOB_MAIN_NODE_INDEX と AWS_BATCH_JOB_NODE_INDEX を単一ノードで比較して、これが主要なノードであるかを確認できます。

`AWS_BATCH_JOB_MAIN_NODE_PRIVATE_IPV4_ADDRESS`

この変数はマルチノード並列ジョブの子ノードでのみ設定されます。この変数は、メインノードには存在しません。この変数はメインノードには存在しないが、ジョブのメインノードのプライベートIPv4アドレスに設定されます。子ノードのアプリケーションコードは、このアドレスを使用して主要なノードと通信できます。

`AWS_BATCH_JOB_NODE_INDEX`

この変数は、ノードのノードインデックス番号に設定されます。ノードインデックスは 0 で始まり、各ノードは一意のインデックス番号を受け取ります。例えば、10 の子を持つマルチノード並列ジョブのインデックス値は 0 ～ 9 です。

AWS_BATCH_JOB_NUM_NODES

この変数は、マルチノード並列ジョブにリクエストしたノードの数に設定されます。

ノードグループ

ノードグループとは、同じコンテナプロパティを共有するジョブノードの同一グループです。 では、ジョブごとに 5 つまでの個別のノードグループを指定できます。を使用して AWS Batch 、ジョブごとに最大 5 つの異なるノードグループを指定できます。

各グループでは、独自のコンテナイメージ、コマンド、環境変数などを持つことができます。例えば、c5.xlarge メインノードに 1 つのインスタンス、5 c5.xlarge つのインスタンスの子ノードに必要なジョブを送信できます。これらのノードグループはそれぞれ、ジョブごとに異なるコンテナイメージやコマンドを実行するように指定できます。

あるいは、ジョブ内のすべてのノードで 1つのノードグループを使用することもできます。さらに、アプリケーションコードではメインノードや子ノードなどのノードロールを区別できます。そのためには、AWS_BATCH_JOB_MAIN_NODE_INDEX 環境変数を AWS_BATCH_JOB_NODE_INDEX の独自の値と比較します。単一のジョブでは最大で 1000 までのノードを使用できます。これは、Amazon ECS クラスターのインスタンスのデフォルト制限です。この制限はリクエストに応じて増やすことができます。この制限の引き上げをリクエストすることができます。

> **Note**
>
> 現在のところ、マルチノード並列ジョブのすべてのノードグループでは、同じインスタンスタイプを使用する必要があります。

MNP ジョブのライフサイクル

マルチノード並列ジョブを送信すると、ジョブは SUBMITTED ステータスになります。その後、ジョブは、ジョブの依存関係がすべて終了するのを待ちます。ジョブも RUNNABLE ステータスに移行します。最後に、 はジョブの実行に必要なインスタンス容量を AWS Batch プロビジョニングし、これらのインスタンスを起動します。

各マルチノード並列ジョブには主要なノードが含まれます。メインノードは、 が AWS Batch 監視して送信されたマルチノードジョブの結果を決定する単一のサブタスクです。主要なノードは最初に起動され、STARTING ステータスに移行します。attemptDurationSeconds パラメータで指定されたタイムアウト値は、ジョブ全体に適用され、ノードには適用されません。

主要なノードが RUNNING ステータスに到達すると (ノードのコンテナが実行されてから)、子ノードが起動され、これもまた STARTING ステータスに移行します。子ノードはランダムな順序で始まります。子ノードの起動のタイミングや順序は保証できません。ジョブのすべてのノードが RUNNING ステータスにあること (ノードのコンテナが実行されたあと) を確認するには、アプリケーションコードで AWS Batch API をクエリして主要なノードおよび子ノードの情報を取得するか、あるいはアプリケーションコード内で整合して、分散された処理タスクを開始する前にすべてのノードがオンラインになるまで待機することができます。あるいは、アプリケーションコードは、すべてのノードがオンラインになるまで待ってから、分散処理タスクを開始することもできます。主要なノードのプライベート IP アドレスは、子ノードごとの AWS_BATCH_JOB_MAIN_NODE_PRIVATE_IPV4_ADDRESS 環境変数で利用可能です。アプリケーションコードでは、この情報を各タスク間の整合と通信に使用できます。

個別のノードが存在する場合、これらは終了コードに応じて SUCCEEDED あるいは FAILED に移行します。主要なノードが終了すると、ジョブは完了したと見なされ、すべての子ノードは停止します。子ノードが死亡した場合、 AWS Batch はジョブ内の他のノードに対してアクションを実行しません。ノード数を減らしてジョブを続行しない場合、これをアプリケーションコードに組み来み、ジョブを終了あるいはキャンセルする必要があります。これを行うと、ジョブは終了またはキャンセルされます。

を使用した MNP のコンピューティング環境に関する考慮事項 AWS Batch

AWS Batchでマルチノード並列ジョブを実行するためのコンピューティング環境を設定するときに、いくつかの考慮事項があります。

- マルチノード並列ジョブは、スポットインスタンスを使用する UNMANAGED コンピューティング環境ではサポートされていません。
- マルチノード並列ジョブをコンピューティング環境に送信する場合、単一のアベイラビリティゾーンでクラスタープレースメントグループを作成して、これをコンピューティングリソースに関連付けることを検討します。これにより、論理的グループ化されたインスタンス上のマルチノード並列ジョブが、潜在的に高度なネットワークフローにより近くなります。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。
- マルチノード並列ジョブは、スポットインスタンスを使用するコンピューティング環境ではサポートされていません。
- AWS Batch マルチノード並列ジョブは Amazon ECS awsvpcネットワークモードを使用します。これにより、マルチノード並列ジョブコンテナに Amazon EC2 インスタンスと同じネットワークプロパティが提供されます。各マルチノード並列ジョブコンテナは、独自の Elastic Network Interface、プライマリプライベート IP アドレス、および内部の DNS ホスト名を取得します。

ネットワークインターフェイスは、ホストコンピューティングリソースと同じ VPC サブネットで作成されます。

- コンピューティング環境には、最大で 5 つまでのセキュリティグループが関連付けられている場合があります。MNP タスクに作成およびアタッチされたエラスティックネットワークインターフェイスは、コンピューティング環境で指定されたセキュリティグループを使用します。セキュリティグループを指定しない場合、VPC のデフォルトのセキュリティグループが使用されます。
- awsvpc ネットワークモードでは、マルチノード並列ジョブ用にパブリック IP アドレスを使用する Elastic Network Interface を用意していません。インターネットにアクセスするには、NAT ゲートウェイを使用するよう設定されたプライベートサブネットでコンピューティングリソースを起動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 NAT ゲートウェイ を参照してください。ノード間の通信には、プライベート IP アドレス、あるいはノードの DNS ホスト名を使用する必要があります。パブリックサブネット内のコンピューティングリソースで実行されるマルチノード並列ジョブには、アウトバウンドのネットワークアクセスがありません。プライベートサブネットの VPC および NAT ゲートウェイを作成するには、仮想プライベートクラウドを作成するを参照してください。
- 作成されてコンピューティングリソースにアタッチされた Elastic Network Interface は、手動でデタッチしたり、ユーザーのアカウントを使用して変更することはできません。これは、実行中のジョブに関連付けられている Elastic Network Interface が誤って削除されることを回避するためです。Elastic Network Interface を解放するには、ジョブを終了します。
- コンピューティング環境には、マルチノード並列ジョブをサポートするために十分な最大数の vCPU があることが必要です。
- Amazon EC2 インスタンスクォータには、ジョブを実行するために必要なインスタンスの数が含まれます。例えば、30 個のインスタンスを必要とするジョブで、リージョンではアカウントが 20 個のインスタンスのみを実行できる場合、このジョブは ステータスで停止します。そうなると、RUNNABLE ジョブは、ステータスのままになります。
- マルチノード並列ジョブでノードグループにインスタンスタイプを指定する場合、コンピューティング環境がそのインスタンスタイプを起動できることが必要です。

Amazon EKS のマルチノード並列ジョブ

Amazon Elastic Kubernetes Service AWS Batch で を使用して、マネージドKubernetesクラスターでマルチノード並列 (MNP) ジョブ (ギャングスケジューリングとも呼ばれます) を実行できます。このオプションは一般に、単一の Amazon Elastic Compute Cloud インスタンスでは実行できない、大規模で密結合されたハイパフォーマンスのジョブに使用されます。詳細については、「マルチノード並列ジョブ」を参照してください。

この機能を使用して、Amazon EKS マネージドの Kubernetes 固有の高性能コンピューティングアプリケーション、大規模言語モデルトレーニング、その他の人工知能 (AI)/機械学習 (ML) ジョブを実行できます。

トピック

MNP ジョブの実行

AWS Batch は、Amazon EC2 を使用した Amazon Elastic Container Service および Amazon EKS での MNP ジョブをサポートします。以下で、この機能のインスタンスとコンテナパラメータの詳細を説明します。

Amazon EKS での MNP のインスタンスクォータ

- 1 つの MNP ジョブに、最大 1,000 個のインスタンスを使用できます。
- 1 つの Amazon EKS クラスターに最大 5,000 個のインスタンスが参加できます。
- 最大 5 つのコンピューティング環境をクラスター化して、ジョブキューにアタッチできます。

例えば、1 つのジョブキューにクラスター化されたコンピューティング環境を 5 つまで、各コンピューティング環境に 1,000 インスタンスまでスケールアップできます。

インスタンスのパラメータに加えて、どちらのサービスでも MNP ジョブに Fargate を使用できないことに注意してください。

MNP ジョブごとに使用できるインスタンスタイプは 1 つのみです。インスタンスタイプは、コンピューティング環境の更新時、または新しいコンピューティング環境の定義時に変更できます。また、インスタンスタイプを指定し、ジョブ定義の作成時に vCPU とメモリの要件を指定することもできます。

Amazon EKS での MNP コンテナクォータ

- マルチノード並列ジョブでは、ノードごとに 1 つのポッドがサポートされます。

- ポッドごとに最大 10 個のコンテナ (または 10 個の init コンテナ。詳細については、Kubernetes のドキュメントの「Init Containers」を参照してください)。
- MNP ジョブごとに最大 5 つのノード範囲。
- ノード範囲ごとに最大 10 個の個別のコンテナイメージ。

例えば、5 つのノード範囲と合計 50 個の一意のイメージを含む 1 つの MNP ジョブでは最大 10,000 個のコンテナを実行できます。

プライベート Amazon VPC と Amazon EKS クラスターでの MNP ジョブの実行

MNP ジョブは任意の Amazon EKS クラスターで、パブリックインターネットの有無にかかわらず実行できます。プライベートネットワークアクセスのみを持つ Amazon EKS クラスターを使用する場合は、 AWS Batch が Amazon EKS コントロールプレーンとマネージド Kubernetes API サーバーにアクセスできることを確認してください。Amazon Virtual Private Cloud エンドポイントを通じて必要なアクセス権限を付与できます。詳細については、「エンドポイントサービスを設定する」を参照してください。

プライベート VPC にはインターネットアクセスがないため、Amazon EKS のクラスターポッドはパブリックソースからイメージをダウンロードできません。Amazon EKS クラスターは Amazon 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。Amazon VPC に Amazon Elastic Container Registry (Amazon ECR) を作成し、コンテナイメージをここにコピーすることでノードにアクセスできます。

また、Amazon ECR を使用してプルスルーキャッシュルールを作成することもできます。外部パブリックレジストリのプルスルーキャッシュルールが作成されたら、Amazon ECR プライベートレジストリ URI を使用して、その外部パブリックレジストリからイメージをプルします。その後、Amazon ECR でリポジトリが作成され、イメージがキャッシュされます。キャッシュされたイメージが Amazon ECR プライベートレジストリ URI を使用してプルされると、Amazon ECR はリモートレジストリをチェックしてイメージの新しいバージョンがあるかどうかを確認し、24 時間ごとに 1 回までプライベートレジストリを更新します。詳細については、「Creating a pull through cache rule in Amazon ECR」を参照してください。

エラー通知

MNP ジョブがブロックされると、 AWS マネジメントコンソール と Amazon EventBridge を通じて通知を受け取ることができます。例えば、MNP ジョブがキューの先頭で停止した場合、問題とその原因に関する情報の通知を受け取り、迅速にアクションを起こしてジョブキューのブロックを解除できます。または、特定の時間内にアクションを実行しない場合、MNP ジョブを自動的に終了する

こともできます。この時間はジョブキューテンプレートで定義できます。詳細については、ジョブキューのブロックイベント を参照してください。

Amazon EKS MNP ジョブ定義を作成する

Amazon EKS で MNP ジョブを定義して実行するため、RegisterJobDefinition と SubmitJob API オペレーション内に新しいパラメータができました。

- nodeProperties セクションの eksProperties を使用すると MNP ジョブ定義を定義できます。
- nodePropertyOverrides セクションの eksPropertiesOverride を使用すると、MNP ジョブを送信するときに、ジョブ定義で定義されたパラメータを上書きできます。

これらのアクションは、API オペレーションと AWS マネジメントコンソールを使用して定義できます。

リファレンス: Amazon EKS MNP ジョブ定義リクエストのペイロードを登録する

以下の例は、Amazon EKS MNP ジョブ定義を 2 つのノードに登録する方法を示しています。

```
{
  "jobDefinitionName": "MyEksMnpJobDefinition",
  "type": "multinode",
  "nodeProperties": {
    "numNodes": 2,
    "mainNode": 0,
    "nodeRangeProperties": [
      {
        "targetNodes" : "0:",
        "eksProperties": {
          "podProperties": {
            "containers": [
              {
                "name": "test-eks-container-1",
                "image": "public.ecr.aws/amazonlinux/amazonlinux:2",
                "command": [
                  "sleep",
                  "60"
                ],
                "resources": {
                  "limits": {
```

```
                    "cpu": "1",
                    "memory": "1024Mi"
                  }
                },
                "securityContext":{
                  "runAsUser":1000,
                  "runAsGroup":3000,
                  "privileged":true,
                  "readOnlyRootFilesystem":true,
                  "runAsNonRoot":true
               }
              }
            ],
            "initContainers": [
               {
                  "name":"init-ekscontainer",
                  "image": "public.ecr.aws/amazonlinux/amazonlinux:2",
                  "command": [
                     "echo",
                     "helloWorld"
                   ],
                   "resources": {
                     "limits": {
                       "cpu": "1",
                       "memory": "1024Mi"
                     }
                  }
               }
            ],
            "metadata": {
               "labels": {
                  "environment" : "test"
               }
            }
          }
        }
      }
    ]
  }
}
```

を使用してジョブ定義を登録するには AWS CLI、MyEksMnpJobDefinition.json という名前のローカルファイルに定義をコピーし、次のコマンドを実行します。

```
aws batch register-job-definition --cli-input-json file://MyEksMnpJobDefinition.json
```

以下のような JSON レスポンスを受け取ります。

```
{
    "jobDefinitionName": "MyEksMnpJobDefinition",
    "jobDefinitionArn": "arn:aws:batch:us-east-1:0123456789:job-definition/
MyEksMnpJobDefinition:1",
    "revision": 1
}
```

Amazon EKS MNP ジョブを送信する

登録されたジョブ定義を使用してジョブを送信するには、以下のコマンドを入力します。<EKS_JOB_QUEUE_NAME> の値を、Amazon EKS コンピューティング環境に関連付けられた既存のジョブキューの名前または ARN に置き換えます。

```
aws batch submit-job --job-queue <EKS_JOB_QUEUE_NAME> \
    --job-definition MyEksMnpJobDefinition \
    --job-name myFirstEksMnpJob
```

以下のような JSON レスポンスを受け取ります。

```
{
    "jobArn": "arn:aws:batch:region:account:job/9b979cce-9da0-446d-90e2-ffa16d52af68",
    "jobName": "myFirstEksMnpJob",
    "jobId": "<JOB_ID>"
}
```

返された jobId と以下のコマンドを使用して、ジョブのステータスをチェックできます。

```
aws batch describe-jobs --jobs <JOB_ID>
```

Amazon EKS MNP ジョブ定義を上書きする

必要に応じて、ジョブ定義の詳細を上書きできます (MNP ジョブサイズや子ジョブの詳細の変更など)。以下は、5 ノードの MNP ジョブを送信するための JSON リクエストのペイロード例と、`test-eks-container-1` コンテナのコマンドの変更を示しています。

```
{
```

```
  "numNodes": 5,
  "nodePropertyOverrides": [
    {
      "targetNodes": "0:",
      "eksPropertiesOverride": {
        "podProperties": {
          "containers": [
            {
              "name": "test-eks-container-1",
              "command": [
                "sleep",
                "150"
              ]
            }
          ]
        }
      }
    }
  ]
}
```

これらのオーバーライドを含むジョブを送信するには、例をローカルファイル eks-mnp-job-nodeoverride.json に保存し、 AWS CLI を使用してオーバーライドを含むジョブを送信します。

配列ジョブ

配列ジョブは、ジョブ定義、vCPU、メモリなどの共通パラメータを共有するジョブです。これは、関連しているが個別の基本ジョブのコレクションとして実行されます。複数のホストに分散されたり、同時に実行される場合もあります。配列ジョブは、モンテカルロシミュレーションジョブ、パラメータスイープジョブ、大規模なレンダリングジョブなど、大量の並列ジョブを実行するもっとも効率的な方法です。

AWS Batch 配列ジョブは、通常のジョブと同様に送信されます。ただし、配列内で実行する子ジョブの数を定義する配列サイズ (2 ~ 10,000) を指定します。配列サイズが 1,000 以内のジョブを送信する場合は、単一ジョブが実行され 1,000 個の子ジョブが生成されます。配列ジョブは、すべての子ジョブを管理するリファレンスまたはポインタです。これにより、1つのクエリで大量のワークロードを送信することができます。`attemptDurationSeconds`パラメータで指定されたタイムアウトは、それぞれの子ジョブに適用されます。親アレイジョブには、タイムアウトはありません。

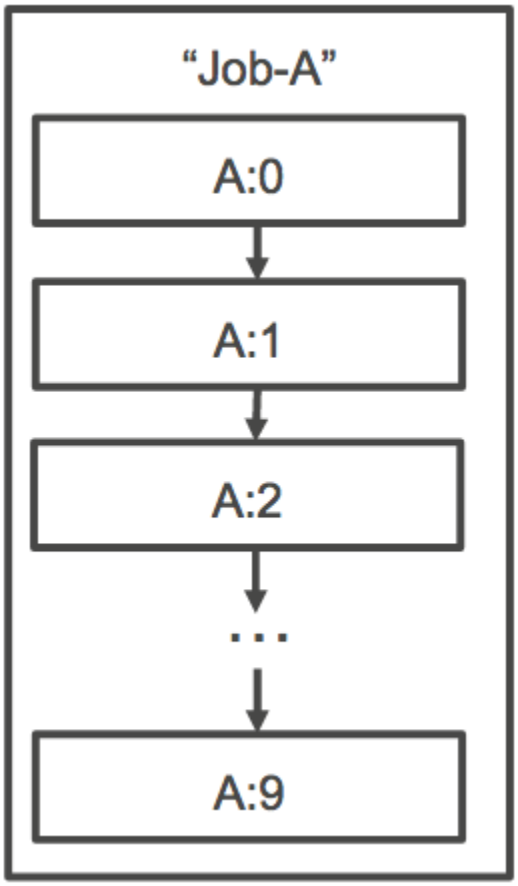
配列ジョブを送信すると、親配列ジョブは通常の AWS Batch ジョブ ID を取得します。子ジョブのベース ID は、それぞれ同じです。各子ジョブは同じベース ID を持ちますが、子ジョブの配列イン

デックスが親 ID の末尾に付加されます。例えば、配列の最初の子ジョブは*example_job_ID*:0です。

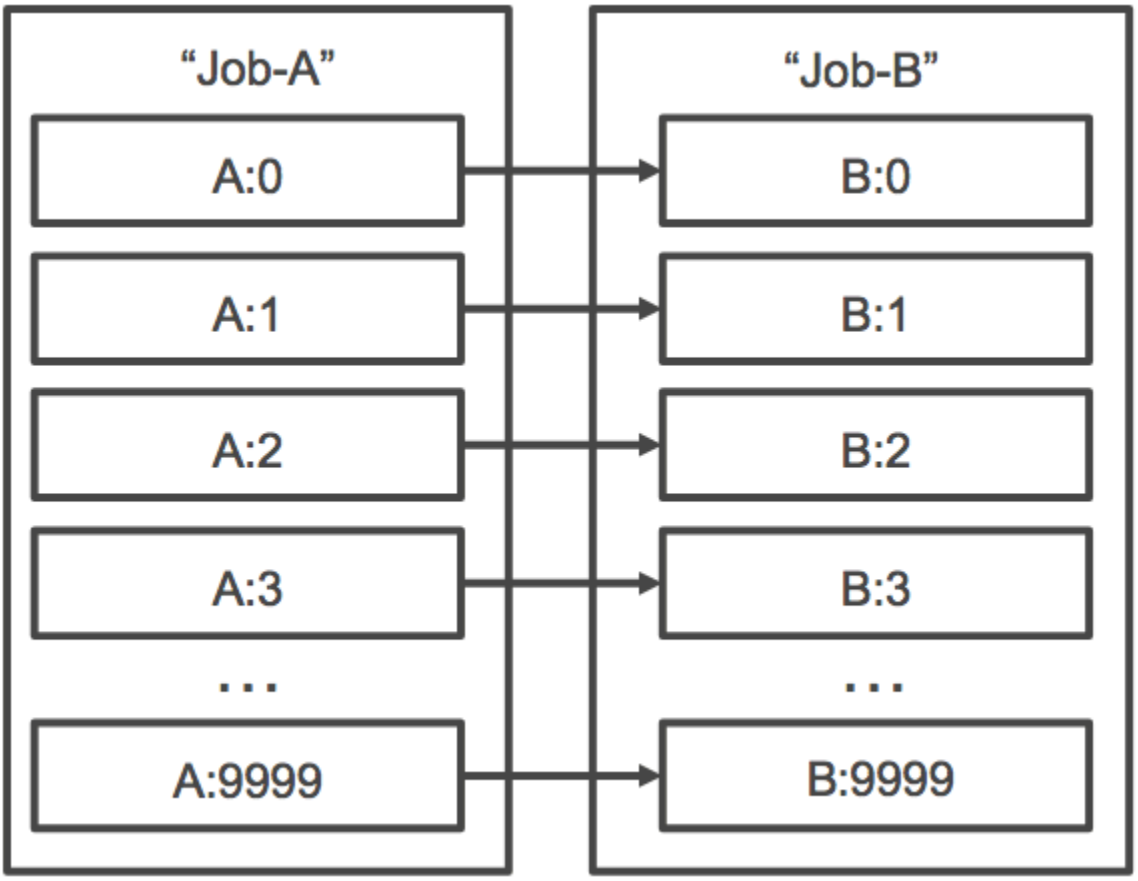
親アレイジョブは、 SUBMITTED、 PENDING、FAILED、または SUCCEEDED ステータスを入力できます。アレイの親ジョブは、PENDING 子ジョブが RUNNABLE に更新されるとに更新されます。これらの依存関係の詳細については、ジョブの依存関係を参照してください。

実行時、AWS_BATCH_JOB_ARRAY_INDEX 環境変数がコンテナの対応するジョブ配列インデックス番号に設定されます。最初の配列ジョブインデックスは 0 番となり、後続の試行は昇順の番号 (1、2、3 など) になります。このインデックス値を使用して、配列ジョブの子がどのように区別されるかを制御できます。詳細については、配列ジョブインデックスを使用してジョブの区別を制御するを参照してください。

配列ジョブの依存関係では、依存関係のタイプを指定できます (SEQUENTIAL または N_TO_N など)。SEQUENTIAL タイプの依存関係 (ジョブ ID を指定しない) を指定できます。こうすることで各子配列ジョブがインデックス 0 から開始して連続的に完了します。例えば、配列サイズが 100 の配列ジョブを送信する場合、依存関係を SEQUENTIAL タイプに指定すると、100 個の子ジョブが連続押して生成され、最初の子ジョブが成功してから次の子ジョブが開始されます。以下の図で示すジョブ A は、配列サイズが 10 である配列ジョブです。ジョブ A の子インデックスの各ジョブは、前の子ジョブに依存します。ジョブ A:1 はジョブ A:0 が完了するまで開始できません。



また、アレイジョブのジョブ ID を使用して N_TO_N タイプの依存関係を指定することもできます。この場合、このジョブの各インデックスの子は各依存関係の対応するインデックスの子が完了するまで待機してから開始されます。以下の図で示すジョブ A およびジョブ B は、配列サイズがそれぞれ 10,000 である 2 つの配列ジョブです。ジョブ B の子インデックスの各ジョブは、ジョブ A の対応するインデックスに依存します。ジョブ B:1 はジョブ A:1 が完了するまで開始できません。



親配列ジョブをキャンセルまたは終了した場合、子ジョブもすべてキャンセルまたは終了します。個々の子ジョブを、他の子ジョブに影響を与えずにキャンセルまたは終了できます (FAILED ステータスに移動させる)。ただし、子配列ジョブが失敗した場合 (それ自身の失敗または手動でキャンセルもしくは終了した場合)、親ジョブも失敗します。このシナリオでは、すべての子ジョブが完了すると、親ジョブは FAILED に移行します。

配列ジョブの検索とフィルタリングの詳細については、「」を参照してくださいジョブキュー内のジョブを検索する。

トピック

- 配列ジョブワークフローの例
- 配列ジョブインデックスを使用してジョブの区別を制御する

配列ジョブワークフローの例

AWS Batch お客様にとって一般的なワークフローは、前提条件のセットアップジョブを実行し、多数の入力タスクに対して一連のコマンドを実行し、結果を集約して概要データを Amazon S3、DynamoDB、Amazon Redshift、または Aurora に書き込むジョブで終了することです。

例えば、次のようになります。

- JobA: 配列ではない標準的なジョブです。Amazon S3 バケット、BucketA 内のオブジェクトの高速リスト化およびメタデータ検証を実行します。ジョブの JSON 送信構文 は、次のとおりです。

```
{
    "jobName": "JobA",
    "jobQueue": "ProdQueue",
    "jobDefinition": "JobA-list-and-validate:1"
}
```

- JobB : JobA に依存する 10,000 個のコピーを持つ配列ジョブです。CPU 負荷の高いコマンドを BucketA の各オブジェクトに対して実行し、結果を BucketB にアップロードします。ジョブの JSON 送信構文 は、次のとおりです。

```
{
    "jobName": "JobB",
    "jobQueue": "ProdQueue",
    "jobDefinition": "JobB-CPU-Intensive-Processing:1",
    "containerOverrides": {
        "resourceRequirements": [
            {
                "type": "MEMORY",
                "value": "4096"
            },
            {
                "type": "VCPU",
                "value": "32"
            }
        ]
   }
    "arrayProperties": {
        "size": 10000
    },
    "dependsOn": [
        {
```

```
            "jobId": "JobA_job_ID"
  }
    ]
}
```

- JobC: JobB に N_TO_N 依存関係モデルで依存する 10,000 個のコピーを持つ別の配列ジョブです。メモリ負荷の高いコマンドを BucketB の各項目に対して実行し、メタデータを DynamoDB に書き込んで、結果の出力を BucketC にアップロードします。ジョブの JSON 送信構文 は、次のとおりです。

```
{
    "jobName": "JobC",
    "jobQueue": "ProdQueue",
    "jobDefinition": "JobC-Memory-Intensive-Processing:1",
    "containerOverrides": {
        "resourceRequirements": [
            {
                "type": "MEMORY",
                "value": "32768"
            },
            {
                "type": "VCPU",
                "value": "1"
            }
        ]
   }
    "arrayProperties": {
        "size": 10000
    },
    "dependsOn": [
        {
            "jobId": "JobB_job_ID",
            "type": "N_TO_N"
        }
    ]
}
```

- JobD: 10 個の検証ステップを実行する配列ジョブです。検証ステップはそれぞれ DynamoDB にクエリする必要があり、上記の Amazon S3 バケットのいずれかとやり取りする可能性があります。の各ステップは同じコマンドを JobD 実行します。しかし、ジョブのコンテナ内の環境変数AWS_BATCH_JOB_ARRAY_INDEXの値によって動作は異な異なります。これらの検証ステップは

順番に実行されます (例えば、JobD:0 の次に JobD:1)。ジョブの JSON 送信構文 は、次のとおりです。

```
{
    "jobName": "JobD",
    "jobQueue": "ProdQueue",
    "jobDefinition": "JobD-Sequential-Validation:1",
    "containerOverrides": {
        "resourceRequirements": [
            {
                "type": "MEMORY",
                "value": "32768"
            },
            {
                "type": "VCPU",
                "value": "1"
            }
        ]
   }
    "arrayProperties": {
        "size": 10
    },
    "dependsOn": [
        {
            "jobId": "JobC_job_ID"
        },
        {
            "type": "SEQUENTIAL"
        },

    ]
}
```

- JobE: 最終的な配列ではないジョブです。シンプルなクリーンアップオペレーションをいくつか実行し、パイプラインが完了したことおよび出力 URL へのリンクを記載したメッセージを含む Amazon SNS 通知を送信します。ジョブの JSON 送信構文 は、次のとおりです。

```
{
    "jobName": "JobE",
    "jobQueue": "ProdQueue",
    "jobDefinition": "JobE-Cleanup-and-Notification:1",
    "parameters": {
        "SourceBucket": "s3://amzn-s3-demo-source-bucket",
```

```
        "Recipient": "pipeline-notifications@mycompany.com"
    },
    "dependsOn": [
        {
            "jobId": "JobD_job_ID"
        }
    ]
}
```

配列ジョブインデックスを使用してジョブの区別を制御する

このチュートリアルでは、AWS_BATCH_JOB_ARRAY_INDEX 環境変数を使用して子ジョブを区別する方法を説明します。各子ジョブは、この変数に割り当てられます。この例では、子ジョブのインデックス番号を使用して、ファイル内の特定の行を読み込みます。次に、その行番号に関連付けられたパラメータを、ジョブのコンテナ内のコマンドで置き換えます。その結果、同じ Docker イメージとコマンド引数を実行する複数の AWS Batch ジョブを持つことができます。ただし、配列ジョブインデックスが修飾子として使用されるため、結果が異なります。

このチュートリアルでは、虹のすべての色を持つテキストファイルを作成します。次に、インデックスをカラーファイルの行番号に使用できる値に変換する Dockerコンテナ 用のエントリポイントスクリプトを作成します。インデックスはゼロから始まりますが、行番号は 1 から始まります。カラーファイルとインデックスファイルをコンテナイメージにコピーし、イメージの ENTRYPOINT をエントリポイントスクリプトに設定する Dockerfile を作成します。Dockerfile とリソースは Amazon ECR にプッシュされる Docker イメージに組み込まれています。次に、新しいコンテナイメージを使用するジョブ定義を登録し、そのジョブ定義で AWS Batch 配列ジョブを送信して、結果を表示します。

トピック

- 前提条件
- コンテナイメージの構築
- Amazon ECR にイメージをプッシュします
- ジョブ定義を作成および登録する
- AWS Batch 配列ジョブを送信する
- 配列ジョブログを表示する

前提条件

このチュートリアルのワークフローには、次のような前提条件があります。

- AWS Batch コンピューティング環境。詳細については、「コンピューティング環境を作成する」を参照してください。
- AWS Batch ジョブキューと関連するコンピューティング環境。詳細については、「ジョブキューを作成する」を参照してください。
- ローカルシステムに AWS CLI インストールされている 。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。
- Docker は、ローカルシステムにインストールされます。詳細については、Docker ドキュメントのDocker CE についてを参照してください。

コンテナイメージの構築

コマンドパラメータのジョブ定義を AWS_BATCH_JOB_ARRAY_INDEX で使用できます。 ただし、エントリポイントスクリプトで変数を使用するコンテナイメージを作成することをお勧めします。このセクションでは、そのようなコンテナイメージを作成する方法について説明します。

Docker コンテナイメージを構築するには

1. Docker イメージワークスペースとして使用する新しいディレクトリを作成し、そのディレクトリに移動します。
2. WorkSpace ディレクトリで、colors.txtという名前のファイルを作成し、以下を貼り付けます。

```
red
orange
yellow
green
blue
indigo
violet
```

3. WorkSpace ディレクトリで、print-color.shという名前のファイルを作成し、以下を貼り付けます。

> **Note**
>
> 配列インデックスは 0 から始まり、行番号は 1 から始まるため、LINE 変数は AWS_BATCH_JOB_ARRAY_INDEX +1 に設定されます。COLOR 変数は、行番号に関連付けられているcolors.txtの色に設定されます。

```
#!/bin/sh
LINE=$((AWS_BATCH_JOB_ARRAY_INDEX + 1))
COLOR=$(sed -n ${LINE}p /tmp/colors.txt)
echo My favorite color of the rainbow is $COLOR.
```

4. WorkSpace ディレクトリで、Dockerfileという名前のファイルを作成し、以下を貼り付けます。この Dockerfile は、以前のファイルをコンテナにコピーし、コンテナの起動時に実行するようにエントリポイントスクリプトを設定します。

```
FROM busybox
COPY print-color.sh /tmp/print-color.sh
COPY colors.txt /tmp/colors.txt
RUN chmod +x /tmp/print-color.sh
ENTRYPOINT /tmp/print-color.sh
```

5. Docker イメージをビルドします。

```
$ docker build -t print-color .
```

6. 次のスクリプトを使用してコンテナをテストします。このスクリプトは、AWS_BATCH_JOB_ARRAY_INDEX 変数をローカルで 0 に設定し、それをインクリメントして 7 つの子 が を行う配列ジョブが何をするのかをシミュレートします。

```
$ AWS_BATCH_JOB_ARRAY_INDEX=0
while [ $AWS_BATCH_JOB_ARRAY_INDEX -le 6 ]
do
    docker run -e AWS_BATCH_JOB_ARRAY_INDEX=$AWS_BATCH_JOB_ARRAY_INDEX print-color
    AWS_BATCH_JOB_ARRAY_INDEX=$((AWS_BATCH_JOB_ARRAY_INDEX + 1))
done
```

出力を次に示します。

```
My favorite color of the rainbow is red.
My favorite color of the rainbow is orange.
My favorite color of the rainbow is yellow.
My favorite color of the rainbow is green.
My favorite color of the rainbow is blue.
My favorite color of the rainbow is indigo.
My favorite color of the rainbow is violet.
```

Amazon ECR にイメージをプッシュします

Docker コンテナを構築してテストしたので、それをイメージリポジトリにプッシュする必要があります。この例では Amazon ECR を使用していますが、DockerHub などの別のレジストリを使用することもできます。

1. コンテナイメージを保存する Amazon ECR イメージを作成します。この例では のみを使用していますが AWS CLI、 を使用することもできます AWS マネジメントコンソール。詳細については、Amazon Elastic Container Registry ユーザーガイドのリポジトリの作成を参照してください。

   ```
   $ aws ecr create-repository --repository-name print-color
   ```

2. 前のステップから返されたAmazon ECR リポジトリ URI を使用して、`print-color` イメージにタグを付けます。

   ```
   $ docker tag print-color aws_account_id.dkr.ecr.region.amazonaws.com/print-color
   ```

3. Amazon ECR レジストリにログインします。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 | docker login \
       --username AWS \
       --password-stdin aws_account_id.dkr.ecr.region.amazonaws.com
   ```

4. Amazon ECR にイメージをプッシュします。

   ```
   $ docker push aws_account_id.dkr.ecr.region.amazonaws.com/print-color
   ```

ジョブ定義を作成および登録する

Docker イメージがイメージレジストリにあるので、 AWS Batch ジョブ定義で指定できます。次に、配列ジョブを実行するために後でそれを使用できます。この例では、 AWS CLIを使用します。ただし、 AWS マネジメントコンソールを使用することもできます。詳細については、「シングルノードのジョブ定義を作成する」を参照してください。

ジョブ定義を作成するには

1. WorkSpace ディレクトリで、print-color-job-def.json という名前のファイルを作成し、以下を貼り付けます。イメージリポジトリの URI を自分のイメージの URI に置き換えます。

```
{
  "jobDefinitionName": "print-color",
  "type": "container",
  "containerProperties": {
    "image": "aws_account_id.dkr.ecr.region.amazonaws.com/print-color",
    "resourceRequirements": [
        {
            "type": "MEMORY",
            "value": "250"
        },
        {
            "type": "VCPU",
            "value": "1"
        }
    ]
  }
}
```

2. ジョブ定義を に登録します AWS Batch。

```
$ aws batch register-job-definition --cli-input-json file://print-color-job-def.json
```

AWS Batch 配列ジョブを送信する

ジョブ定義を登録したら、新しいコンテナイメージを使用する AWS Batch 配列ジョブを送信できます。

AWS Batch 配列ジョブを送信するには

1. WorkSpace ディレクトリで、print-color-job.jsonという名前のファイルを作成し、以下を貼り付けます。

   > **Note**
   >
   > この例では、the section called “前提条件” セクションで説明したジョブキューを使用しています。

   ```
   {
     "jobName": "print-color",
     "jobQueue": "existing-job-queue",
     "arrayProperties": {
       "size": 7
     },
     "jobDefinition": "print-color"
   }
   ```

2. ジョブを AWS Batch ジョブキューに送信します。出力で返されるジョブ ID を書き留めておいてください。

   ```
   $ aws batch submit-job --cli-input-json file://print-color-job.json
   ```

3. ジョブのステータスを記述し、ジョブが SUCCEEDED に移動するのを待ちます。

配列ジョブログを表示する

ジョブが SUCCEEDED ステータスになったら、ジョブのコンテナから CloudWatch Logs を表示できます。

CloudWatch Logs でジョブのログを表示するには

1. https://console.aws.amazon.com/batch/ で AWS Batch コンソールを開きます。
2. 左のナビゲーションペインで ジョブ を選択します。
3. Job queue] (ジョブキュー) で、キューを選択します。
4. Status] (ステータス) セクションで、succeeded] (成功) を選択します。

5. 配列ジョブのすべての子ジョブを表示するには、前のセクションで返されたジョブ ID を選択します。
6. ジョブのコンテナからログを表示するには、子ジョブのいずれかを選択し、View logs] (ログの表示) を選択します。

7. 他の子ジョブのログを表示します。各ジョブは、虹の別の色を返します。

GPU ジョブを実行する

GPU ジョブを使用して、インスタンスの GPU を使用するジョブを実行できます。

以下の Amazon EC2 GPU ベースのインスタンスタイプがサポートされています。詳細については、Amazon EC2 G3 インスタンス、Amazon EC2 G4 インスタンス、Amazon EC2 G5 インスタンス、Amazon EC2 G6 インスタンス、Amazon EC2 P2 インスタンス、Amazon EC2 P3 インスタンス、Amazon EC2 P4d インスタンス、Amazon EC2 P5 インスタンス、Amazon EC2 P6 インスタンス、Amazon EC2 Trn1 インスタンス、Amazon EC2 Trn2 インスタンス、Amazon EC2 Inf1インスタンス、Amazon EC2 Inf2 インスタンス、Amazon EC2 Dl1 インスタンス、Amazon EC2 Dl2 インスタンス を参照してください。

インスタンスタイプ	GPUs	GPU メモリ	vCPU	メモリ	ネットワーク帯域幅
g3s.xlarge	1	8 GiB	4	30.5 GiB	10 Gbps
g3.4xlarge	1	8 GiB	16	122 GiB	最大 10 Gbps
g3.8xlarge	2	16 GiB	32	244 GiB	10 Gbps
g3.16xlarge	4	32 GiB	64	488 GiB	25 Gbps

インスタンスタイプ	GPUs	GPU メモリ	vCPU	メモリ	ネットワーク帯域幅
g4dn.xlarge	1	16 GiB	4	16 GiB	最大 25 Gbps
g4dn.2xlarge	1	16 GiB	8	32 GiB	最大 25 Gbps
g4dn.4xlarge	1	16 GiB	16	64 GiB	最大 25 Gbps
g4dn.8xlarge	1	16 GiB	32	128 GiB	50 Gbps
g4dn.12xlarge	4	64 GiB	48	192 GiB	50 Gbps
g4dn.16xlarge	1	16 GiB	64	256 GiB	50 Gbps
g5.xlarge	1	24 GiB	4	16 GiB	最大 10 Gbps
g5.2xlarge	1	24 GiB	8	32 GiB	最大 10 Gbps
g5.4xlarge	1	24 GiB	16	64 GiB	最大 25 Gbps
g5.8xlarge	1	24 GiB	32	128 GiB	25 Gbps
g5.16xlarge	1	24 GiB	64	256 GiB	25 Gbps
g5.12xlarge	4	96 GiB	48	192 GiB	40 Gbps
g5.24xlarge	4	96 GiB	96	384 GiB	50 Gbps
g5.48xlarge	8	192 GiB	192	768 GiB	100 Gbps
g5g.xlarge	1	16 GiB	4	8 GiB	最大 10 Gbps
g5g.2xlarge	1	16 GiB	8	16 GiB	最大 10 Gbps
g5g.4xlarge	1	16 GiB	16	32 GiB	最大 10 Gbps
g5g.8xlarge	1	16 GiB	32	64 GiB	12 Gbps
g5g.16xlarge	2	32 GiB	64	128 GiB	25 Gbps
g5g.metal	2	32 GiB	64	128 GiB	25 Gbps

インスタンスタイプ	GPUs	GPU メモリ	vCPU	メモリ	ネットワーク帯域幅
g6.xlarge	1	24 GiB	4	16 GiB	最大 10 Gbps
g6.2xlarge	1	24 GiB	8	32 GiB	最大 10 Gbps
g6.4xlarge	1	24 GiB	16	64 GiB	最大 25 Gbps
g6.8xlarge	1	24 GiB	32	128 GiB	25 Gbps
g6.16xlarge	1	24 GiB	64	256 GiB	25 Gbps
g6.12xlarge	4	96 GiB	48	192 GiB	40 Gbps
g6.24xlarge	4	96 GiB	96	384 GiB	50 Gbps
g6.48xlarge	8	192 GiB	192	768 GiB	100 Gbps
g6e.xlarge	1	48 GiB	4	32 GiB	最大 20 Gbps
g6e.2xlarge	1	48 GiB	8	64 GiB	最大 20 Gbps
g6e.4xlarge	1	48 GiB	16	128 GiB	20 Gbps
g6e.8xlarge	1	48 GiB	32	256 GiB	25 Gbps
g6e.16xlarge	1	48 GiB	64	512 GiB	35 Gbps
g6e.12xlarge	4	192 GiB	48	384 GiB	100 Gbps
g6e.24xlarge	4	192 GiB	96	768 GiB	200 Gbps
g6e.48xlarge	8	384 GiB	192	1536 GiB	400 Gbps
gr6.4xlarge	1	24 GiB	16	128 GiB	最大 25 Gbps
gr6.8xlarge	1	24 GiB	32	256 GiB	25 Gbps
p2.xlarge	1	12 GiB	4	61 GiB	高
p2.8xlarge	8	96 GiB	32	488 GiB	10 Gbps

インスタンスタイプ	GPUs	GPU メモリ	vCPU	メモリ	ネットワーク帯域幅
p2.16xlarge	16	192 GiB	64	732 GiB	20 Gbps
p3.2xlarge	1	16 GiB	8	61 GiB	最大 10 Gbps
p3.8xlarge	4	64 GiB	32	244 GiB	10 Gbps
p3.16xlarge	8	128 GiB	64	488 GiB	25 Gbps
p3dn.24xlarge	8	256 GiB	96	768 GiB	100 Gbps
p4d.24xlarge	8	320 GiB	96	1152 GiB	400 Gbps
p4de.24xlarge	8	640 GiB	96	1152 GiB	400 Gbps
p5.48xlarge	8	640 GiB	192	2 TiB	3200 Gbps
p5e.48xlarge	8	1128 GiB	192	2 TiB	3200 Gbps
p5en.48xlarge	8	1128 GiB	192	2 TiB	3200 Gbps
p6-b200.48xlarge	8	1440 GiB	192	2 TiB	100 Gbps
trn1.2xlarge	1	32 GiB	8	32 GiB	最大 12.5 Gbps
trn1.32xlarge	16	512 GiB	128	512 GiB	800 Gbps
trn1n.32xlarge	16	512 GiB	128	512 GiB	1600 Gbps
trn2.48xlarge	16	1.5 TiB	192	2 TiB	3.2 Tbps
inf1.xlarge	1	8 GiB	4	8 GiB	最大 25 Gbps
inf1.2xlarge	1	8 GiB	8	16 GiB	最大 25 Gbps
inf1.6xlarge	4	32 GiB	24	48 GiB	25 Gbps
inf1.24xlarge	16	128 GiB	96	192 GiB	100 Gbps
inf2.xlarge	1	32 GiB	4	16 GiB	最大 15 Gbps

インスタンスタイプ	GPUs	GPU メモリ	vCPU	メモリ	ネットワーク帯域幅
inf2.8xlarge	1	32 GiB	32	128 GiB	最大 25 Gbps
inf2.24xlarge	6	192 GiB	96	384 GiB	50 Gbps
inf2.48xlarge	12	384 GiB	192	768 GiB	100 Gbps
dl1.24xlarge	8	256 GiB	96	768 GiB	400 Gbps
dl2q.24xlarge	8	128 GiB	96	768 GiB	100 Gbps

Note

GPU ジョブでは、NVIDIA GPUs を持つインスタンスタイプ AWS Batch のみをサポートします。例えば、G4ad ファミリーは GPU スケジューリングではサポートされていません。ジョブ定義で vcpu とメモリの要件のみを定義し、Amazon ECS または Amazon EKS コンピューティング最適化 AMI、または AMD GPUs を使用するためのカスタマイズされた AMI を使用して Amazon EC2 起動テンプレートのユーザーデータをカスタマイズすることでホスト GPUs に直接アクセス AWS Batch することで、 G4adで を引き続き使用できます。Amazon EC2
ARM64 アーキテクチャを使用するインスタンスタイプは、カスタマイズされたコードと設定によって GPU にアクセスするために AWS Batch または Amazon EC2 ユーザーデータに提供されるカスタム AMIs の GPUs ジョブでサポートされています。例えば、G5g インスタンスファミリーです。

ジョブ定義の [resourceRequirements] パラメータは、コンテナに固定される GPU の数を指定します。この GPU の数は、そのジョブの期間中にインスタンスで実行される他のジョブでは使用できません。GPU ジョブを実行するコンピューティング環境のすべてのインスタンスタイプは p6、p3、p4、p5、g3、g3s、g4、g5、g6 インスタンスファミリーのいずれかにする必要があります。これを行わないと、GPU ジョブが RUNNABLE 状態で固まる可能性があります。

GPU を使用しないジョブは GPU インスタンスで実行できます。ただし、類似の GPU 以外のインスタンスで実行するよりも、GPU インスタンスで実行する方がコストがかかる場合があります。特定の vCPU、メモリ、および所要時間によっては、このような GPU を使用しないジョブによって GPU ジョブの実行がブロックされる場合があります。

トピック

- Amazon EKS で GPU ベースの Kubernetes クラスターを作成する
- Amazon EKS GPU ジョブ定義を作成する
- Amazon EKS クラスターで GPU ジョブを実行する

Amazon EKS で GPU ベースの Kubernetes クラスターを作成する

Amazon EKS で GPU ベースの Kubernetes クラスターを作成する前に、Amazon EKS で AWS Batch のご利用開始にあたって のステップを完了しておく必要があります。また、次の操作を行います。

- AWS Batch では、NVIDIA GPUs。
- デフォルトでは、 は Amazon EKS クラスターコントロールプレーンKubernetesのバージョンに一致するバージョンで Amazon EKS 高速 AMI AWS Batch を選択します。

```
$ cat <<EOF > ./batch-eks-gpu-ce.json
{
  "computeEnvironmentName": "My-Eks-GPU-CE1",
  "type": "MANAGED",
  "state": "ENABLED",
  "eksConfiguration": {
    "eksClusterArn": "arn:aws:eks:<region>:<account>:cluster/<cluster-name>",
    "kubernetesNamespace": "my-aws-batch-namespace"
  },
  "computeResources": {
    "type": "EC2",
    "allocationStrategy": "BEST_FIT_PROGRESSIVE",
    "minvCpus": 0,
    "maxvCpus": 1024,
    "instanceTypes": [
      "p3dn.24xlarge",
      "p4d.24xlarge"
    ],
    "subnets": [
        "<eks-cluster-subnets-with-access-to-internet-for-image-pull>"
    ],
    "securityGroupIds": [
        "<eks-cluster-sg>"
    ],
```

```
    "instanceRole": "<eks-instance-profile>"
  }
}
EOF

$ aws batch create-compute-environment --cli-input-json file://./batch-eks-gpu-ce.json
```

AWS Batch はユーザーに代わって NVIDIA GPU デバイスプラグインを管理しません。このプラグインを Amazon EKS クラスターにインストールし、 AWS Batch ノードをターゲットにする必要があります。詳細については、GitHub Kubernetes のGPU Support の有効化 を参照してください。

AWS Batch ノードをターゲットにするようにNVIDIAデバイスプラグイン (DaemonSet) を設定するには、次のコマンドを実行します。

```
# pull nvidia daemonset spec
$ curl -O https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/v0.12.2/nvidia-
device-plugin.yml
# using your favorite editor, add Batch node toleration
# this will allow the DaemonSet to run on Batch nodes
- key: "batch.amazonaws.com/batch-node"
  operator: "Exists"

$ kubectl apply -f nvidia-device-plugin.yml
```

コンピューティングベース (CPU とメモリ) のワークロードと GPU ベースのワークロードを、コンピューティング環境とジョブキューを同じ組み合わせで混在させることはお勧めしません。これは、コンピューティングジョブが GPU の容量を使い果たす可能性があるためです。

ジョブキューをアタッチするには、以下のコマンドを実行します。

```
$ cat <<EOF > ./batch-eks-gpu-jq.json
 {
    "jobQueueName": "My-Eks-GPU-JQ1",
    "priority": 10,
    "computeEnvironmentOrder": [
      {
        "order": 1,
        "computeEnvironment": "My-Eks-GPU-CE1"
      }
    ]
  }
EOF
```

```
$ aws batch create-job-queue --cli-input-json file://./batch-eks-gpu-jq.json
```

Amazon EKS GPU ジョブ定義を作成する

現時点では nvidia.com/gpu のみサポートされており、設定するリソース値は整数でなければなりません。GPU の一部を使用することはできません。詳細については、Kubernetes ドキュメント の GPUのスケジュール を参照してください。

Amazon EKS に GPU ジョブ定義を登録するには、以下のコマンドを実行します。

```
$ cat <<EOF > ./batch-eks-gpu-jd.json
{
    "jobDefinitionName": "MyGPUJobOnEks_Smi",
    "type": "container",
    "eksProperties": {
        "podProperties": {
            "hostNetwork": true,
            "containers": [
                {
                    "image": "nvcr.io/nvidia/cuda:10.2-runtime-centos7",
                    "command": ["nvidia-smi"],
                    "resources": {
                        "limits": {
                            "cpu": "1",
                            "memory": "1024Mi",
                            "nvidia.com/gpu": "1"
                        }
                    }
                }
            ]
        }
    }
}
EOF

$ aws batch register-job-definition --cli-input-json file://./batch-eks-gpu-jd.json
```

Amazon EKS クラスターで GPU ジョブを実行する

GPU リソースは非圧縮です。 は GPU ジョブのポッド仕様 AWS Batch を作成します。リクエストの値は制限の値と等しくなります。Kubernetes は必須です。

ジョブを再起動するには、以下のコマンドを実行します。

```
$ aws batch submit-job --job-queue My-Eks-GPU-JQ1 --job-definition MyGPUJobOnEks_Smi --
job-name My-Eks-GPU-Job

# locate information that can help debug or find logs (if using Amazon CloudWatch Logs
 with Fluent Bit)
$ aws batch describe-jobs --job <job-id> | jq '.jobs[].eksProperties.podProperties |
 {podName, nodeName}'
{
  "podName": "aws-batch.f3d697c4-3bb5-3955-aa6c-977fcf1cb0ca",
  "nodeName": "ip-192-168-59-101.ec2.internal"
}
```

AWS Batch ジョブキューでジョブを表示する

AWS Batchでジョブを表示およびフィルタリングできます。この機能は、既存のジョブキューを表示し、そのジョブを 3 つのオプションのいずれかでフィルタリングするオプションを提供します。

検索とフィルターは、終了状態 (SUCCEEDED または FAILED) ではないジョブを取得できます。ジョブの状態が SUCCEEDED または FAILED になると、最大 7 日間ジョブを取得できるようになります。ジョブの CloudWatch または Amazon EventBridge ログは引き続き表示できます。

コンソールでジョブキュー内のすべてのジョブを一覧表示するには、この手順を使用します AWS Batch 。必要に応じて、[フィルター結果] フィールドを使用して、指定した基準で結果を絞り込みます。

1. AWS Batch コンソールに移動します。
2. ナビゲーションペインで [ジョブ] を選択します。
3. [ジョブキュー] ドロップダウンリストを展開し、検索するジョブキューを選択します。

   > **Note**
   >
   > 一度に検索できるジョブキューは 1 つのみです。

4. Filter results フィールドに、結果に含めるキーワードを入力します。このフィールドを使用して、ジョブ名、ステータス、またはジョブ ID でフィルタリングできます。プロパティによっては、等号 (=) や包含 (:) など、定義する必要のある追加の演算子がある場合があります。

> **Note**
>
> SageMaker トレーニングジョブキューは、[ジョブ名] と [ジョブ ID] によるフィルタリングのみをサポートします

5. [検索] を選択してください。

ジョブキュー内のジョブ AWS Batch を検索する

ジョブ検索 AWS Batch を使用して、 でジョブを検索およびフィルタリングできます。この機能は、既存のジョブキュー内で検索し、そのジョブをフィルタリングするオプションを提供します。

検索とフィルターは、終了状態 (SUCCEEDED または FAILED) ではないジョブを取得できます。ジョブの状態が SUCCEEDED または FAILED になると、最大 7 日間ジョブを取得できるようになります。ジョブの CloudWatch または Amazon EventBridge ログは引き続き表示できます。

複数の条件を同時に使用して検索するには、[高度な検索] 機能を使用します。例えば、[ステータス]、[日付範囲]、[追加条件] ([ジョブ名]、[ジョブ定義]、[ジョブ ID] など) のいずれかまたはすべてを含めることができます。

検索 AWS Batch ジョブ (AWS コンソール)

コンソールでジョブキュー内のジョブを検索するには、この手順を使用します AWS Batch 。

1. AWS Batch コンソールに移動します。
2. ナビゲーションペインで [ジョブ] を選択します。
3. [高度な検索] をオンにします。
4. [ジョブキュー] ドロップダウンリストを展開し、検索するジョブキューを選択します。

> **Note**
>
> 一度に検索できるジョブキューは 1 つのみです。

5. [検索オプション] の場合:

a. [ステータス] ドロップダウンリストでは、フィルタリングするステータスを 1 つ以上選択できます。詳細については、「ジョブの状態」および「サービスジョブのステータス」を参照してください。

> **Note**
>
> 配列ジョブの親は、子ジョブが に更新PENDINGされると に更新RUNNABLEされ、子ジョブの実行中に PENDINGステータスのままになります。これらのジョブを表示するには、すべての子ジョブが終了状態になるまでPENDINGステータスでフィルタリングします。

b. [日付範囲] を選択して、日付と時刻の範囲に基づいて結果をフィルタリングします。

- [相対モード] を選択すると、現在の日付と時刻からさかのぼった時間範囲に含まれる作成日を持つジョブを検索できます。
- [絶対モード] を選択すると、指定した日時の範囲に含まれる作成日を持つジョブを検索できます。

c. [追加条件] フィールドに、検索結果に含めるキーワードを入力します。たとえば、このフィールドを使用して、ジョブ名、ジョブ定義、ジョブ ID、または共有識別子で検索できます。プロパティによっては、等号 (=) や包含 (:) など、定義する必要のある追加の演算子がある場合があります。

> **Note**
>
> SageMaker トレーニングジョブキューは、[ジョブ名] と [ジョブ ID] によるフィルタリングのみをサポートします

> **Note**
>
> 共有識別子でフィルタリングする場合、ジョブのステータスを指定することもできます。これは、他のフィルターがジョブステータスフィルタリングを除外する制限の例外です。

6. [検索] を選択してください。

AWS Batch ジョブの検索とフィルタリング (AWS CLI)

この手順を使用して、 AWS CLIでジョブキュー内のすべてのジョブを一覧表示します。必要に応じて、-filters パラメータを使用して、指定した基準で結果を絞り込みます。

Search job queue (AWS CLI)

list-jobs コマンドを使用して、ジョブキューを検索およびフィルタリングできます。

例えば、ジョブ名に基づいてジョブキューを検索できます。

```
aws batch list-jobs \
    --job-queue my-job-queue \
    --filters name=JOB_NAME,values="my-job"
```

共有識別子でジョブをフィルタリングします。

```
aws batch list-jobs \
    --job-queue my-job-queue \
    --filters name=SHARE_IDENTIFIER,values="my-share"
```

共有識別子でフィルタリングする場合、ジョブステータスを含めることができます。

```
aws batch list-jobs \
    --job-queue my-job-queue \
    --job-status RUNNING \
    --filters name=SHARE_IDENTIFIER,values="my-share"
```

上のコマンドに、以下の変更を加えます。

- *my-job-queue* をジョブキューの名前に置き換えます。
- *my-job* をジョブの名前に置き換えます。
- *my-share* をフィルタリングする共有識別子に置き換えます。

Search service job queue (AWS CLI)

list-service-jobs コマンドを使用して、サービスジョブキューを検索およびフィルタリングできます。

例えば、ジョブ名に基づいてサービスジョブキューを検索できます。

```
aws batch list-service-jobs \
    --job-queue my-sm-queue \
    --filters name=JOB_NAME,values="my-sm-job"
```

共有識別子でサービスジョブをフィルタリングします。

```
aws batch list-service-jobs \
    --job-queue my-sm-queue \
    --filters name=SHARE_IDENTIFIER,values="my-share"
```

上のコマンドに、以下の変更を加えます。

- *my-sm-queue* をサービスジョブキューの名前に置き換えます。
- *my-sm-job* をサービスジョブに置き換えます。
- *my-share* をフィルタリングする共有識別子に置き換えます。

AWS Batch ジョブのネットワークモード

次の表に、 AWS Batch ジョブタイプのネットワークモードと一般的な使用方法を示します。考慮事項と動作の詳細については、「ジョブタイプ」列のリンクを参照してください。

ジョブタイプ	サポートされているネットワークモード	一般的な使用法
ECS-EC2 シンプルジョブ	host	コンピューティング環境で定義された vpc への出力のみを必要とする、最もスケーラブルで厄介な並列バッチワークロードに使用されます。
ECS-EC2 マルチノード並列ジョブ	awsvpc	タスクノード間の通信が調整された単一ジョブとしてモデル化された、緊密に結合されたマルチホスト (ノード) 分散ワークロードに使用されます。

ジョブタイプ	サポートされているネットワークモード	一般的な使用法
ECS-Fargate シンプルジョブ	awsvpc	厄介な並列バッチワークロード向けの真のサーバーレス。通常、TCO が最も低く、最も高度なコンテナ分離ジョブモデルです。
EKS-EC2 シンプルジョブ	ホストとポッド	コンピューティング環境で定義された vpc への出力のみを必要とする、極めてスケーラブルで厄介な並列バッチワークロードに使用されます。デフォルトはホストネットワークです。
EKS-EC2 マルチノード並列ジョブ	ホストとポッド	ポッドノード間の通信が調整された単一ジョブとしてモデル化された、緊密に結合されたマルチホスト (ノード) 分散ワークロードに使用されます。デフォルトはホストネットワークです。

CloudWatch Logs で AWS Batch ジョブログを表示する

Amazon CloudWatch Logs にログ情報を送信するようにAWS Batch ジョブを設定できます。これにより、1 つの便利な場所でジョブからのさまざまなログを表示できます。詳細については、「でのCloudWatch Logs の使用 AWS Batch」を参照してください。

AWS Batch コンソールのジョブログを使用して、 AWS Batch ジョブをモニタリングまたはトラブルシューティングすることもできます。

1. AWS Batch コンソール を開きます。

2. ジョブ を選択します。ジョブキューでのジョブのソートとフィルタリングの詳細については、「AWS Batch ジョブキューでジョブを表示する」および「ジョブキュー内のジョブを検索する」を参照してください。
3. Job キュー で、目的のジョブキューを選択します。

> **Tip**
>
> ジョブキューに複数のジョブがある場合は、検索とフィルタリングをオンにすると、ジョブをすばやく検出できます。詳細については、「ジョブキュー内のジョブ AWS Batch を検索する」を参照してください。

4. ステータス では、目的のジョブステータスを選択します。
5. 目的のジョブを選択すると、[詳細] ページが開きます。
6. [詳細] ページで [ログストリーム名] までスクロールし、リンクを選択します。リンクをクリックすると、ジョブの Amazon CloudWatch Logs ページが開きます。
7. (オプション) ログを初めて表示する場合は、許可を求められることがあります。

   [権限が必要です] には「**OK**」と入力し、[承認] を選択して Amazon CloudWatch の料金を承認します。

> **Note**
>
> CloudWatch 料金の承認を取り消すには:
>
> 1. 左側のナビゲーションペインで、を選択します。
> 2. ジョブ・ログ の場合は、編集 を選択します。
> 3. CloudWatch を使用するようにBatch を承認 チェックボックスをオフにします。
> 4. [Save changes] (変更の保存) をクリックします。

AWS Batch ジョブ情報の確認

ステータス、 AWS Batch ジョブ定義、コンテナ情報などのジョブ情報を確認できます。

1. AWS Batch コンソール を開きます。
2. ジョブ を選択します。

3. Job キュー で、目的のジョブキューを選択します。

> **Tip**
>
> ジョブキューに複数のジョブがある場合は、[検索とフィルター] をオンにすると、ジョブをすばやく検索できます。詳細については、「ジョブキュー内のジョブ AWS Batch を検索する」を参照してください。

4. 必要な色を選択します。

> **Note**
>
> AWS Command Line Interface (AWS CLI) を使用して、 AWS Batch ジョブの詳細を表示することもできます。詳細については、AWS CLI コマンドリファレンス の describe-domain を参照してください。

のセキュリティ AWS Batch

でのクラウドセキュリティが最優先事項 AWS です。お客様は AWS 、最もセキュリティの影響を受けやすい組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャからメリットを得られます。

セキュリティは、 AWS とお客様の間の責任共有です。責任共有モデルではこれを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説明しています。

- クラウドのセキュリティ – AWS は、 で AWS 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 AWS クラウド。 は、お客様が安全に使用できるサービス AWS も提供します。サードパーティーの監査者は、AWS コンプライアンスプログラムコンプライアンスプログラムの一環として、当社のセキュリティの有効性を定期的にテストおよび検証。が適用されるコンプライアンスプログラムの詳細については AWS Batch、「コンプライアンスプログラムAWS による対象範囲内のサービスコンプライアンスプログラム」を参照してください。
- クラウド内のセキュリティ – お客様の責任は使用する AWS のサービスによって決まります。また、お客様は、お客様のデータの機密性、企業の要件、および適用可能な法律および規制などの他の要因についても責任を担います。

このドキュメントは、 を使用する際の責任共有モデルの適用方法を理解するのに役立ちます AWS Batch。以下のトピックでは、セキュリティおよびコンプライアンスの目的を達成する AWS Batch ように を設定する方法を示します。また、 AWS Batch リソースのモニタリングや保護に役立つ他の AWS サービスの使用方法についても説明します。

トピック

- IAM AWS Batch のトラブルシューティング

の Identity and Access Management AWS Batch

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 AWS のサービス するのに役立つ です。IAM 管理者は、誰を認証 (サインイン) し、誰に AWS Batch リソースの使用を許可する (アクセス許可を付与する) かを制御します。IAM は、追加料金なしで使用できる AWS のサービス です。

トピック

- オーディエンス
- アイデンティティを使用した認証
- ポリシーを使用したアクセスの管理
- が IAM と AWS Batch 連携する方法
- のアイデンティティベースのポリシーの例 AWS Batch
- AWS の 管理ポリシー AWS Batch

オーディエンス

AWS Identity and Access Management (IAM) の使用方法は、ロールによって異なります。

- サービスユーザー - 機能にアクセスできない場合は、管理者にアクセス許可をリクエストします (「IAM AWS Batch のトラブルシューティング」を参照)。
- サービス管理者 - ユーザーアクセスを決定し、アクセス許可リクエストを送信します (「が IAM と AWS Batch 連携する方法」を参照)
- IAM 管理者 - アクセスを管理するためのポリシーを作成します (「のアイデンティティベースのポリシーの例 AWS Batch」を参照)

アイデンティティを使用した認証

認証は、ID 認証情報 AWS を使用して にサインインする方法です。、IAM ユーザー AWS アカウントのルートユーザー、または IAM ロールを引き受けることで認証される必要があります。

AWS IAM アイデンティティセンター (IAM Identity Center)、シングルサインオン認証、Google/Facebook 認証情報などの ID ソースからの認証情報を使用して、フェデレーティッド ID としてサイ

ンインできます。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。

プログラムによるアクセスの場合、 は SDK と CLI AWS を提供してリクエストを暗号化して署名します。詳細については、「IAM ユーザーガイド」の「API リクエストに対するAWS 署名バージョン 4」を参照してください。

AWS アカウント ルートユーザー

を作成するときは AWS アカウント、すべての AWS のサービス および リソースへの完全なアクセス権を持つ AWS アカウント ルートユーザーと呼ばれる 1 つのサインインアイデンティティから始めます。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。ルートユーザー認証情報を必要とするタスクについては、「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。

フェデレーテッドアイデンティティ

ベストプラクティスとして、人間のユーザーが一時的な認証情報 AWS のサービス を使用して 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を使用する必要があります。

フェデレーティッド ID は、エンタープライズディレクトリ、ウェブ ID プロバイダー、または ID Directory Service ソースの認証情報 AWS のサービス を使用して にアクセスするユーザーです。フェデレーテッドアイデンティティは、一時的な認証情報を提供するロールを引き受けます。

アクセスを一元管理する場合は、 AWS IAM アイデンティティセンターをお勧めします。詳細については、「AWS IAM アイデンティティセンター ユーザーガイド」の「IAM アイデンティティセンターとは」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、特定の個人やアプリケーションに対する特定のアクセス許可を持つアイデンティティです。長期認証情報を持つ IAM ユーザーの代わりに一時的な認証情報を使用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「ID プロバイダーとのフェデレーションを使用して にアクセスすることを人間 AWS のユーザーに要求する」を参照してください。

IAM グループは、IAM ユーザーの集合を指定し、大量のユーザーに対するアクセス許可の管理を容易にします。詳細については、「IAM ユーザーガイド」の「IAM ユーザーに関するユースケース」を参照してください。

IAM ロール

IAM ロールは、特定のアクセス許可を持つアイデンティであり、一時的な認証情報を提供します。ユーザーから IAM ロール (コンソール) に切り替えるか、 または API オペレーションを呼び出すことで、ロールを引き受けることができます。 AWS CLI AWS 詳細については、「IAM ユーザーガイド」の「ロールを引き受けるための各種方法」を参照してください。

IAM ロールは、フェデレーションユーザーアクセス、一時的な IAM ユーザーのアクセス許可、クロスアカウントアクセス、クロスサービスアクセス、および Amazon EC2 で実行するアプリケーションに役立ちます。詳細については、IAM ユーザーガイド の IAM でのクロスアカウントリソースアクセス を参照してください。

ポリシーを使用したアクセスの管理

でアクセスを制御する AWS には、ポリシーを作成し、ID AWS またはリソースにアタッチします。ポリシーは、ID またはリソースに関連付けられたときにアクセス許可を定義します。 は、プリンシパルがリクエストを行うときにこれらのポリシー AWS を評価します。ほとんどのポリシーは JSON ドキュメント AWS として に保存されます。JSON ポリシードキュメントの詳細については、「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、ポリシーを使用して、どのプリンシパルがどのリソースに対して、どのような条件でアクションを実行できるかを定義することで、誰が何にアクセスできるかを指定します。

デフォルトでは、ユーザーやロールにアクセス許可はありません。IAM 管理者は IAM ポリシーを作成してロールに追加し、このロールをユーザーが引き受けられるようにします。IAM ポリシーは、オペレーションの実行方法を問わず、アクセス許可を定義します。

アイデンティティベースのポリシー

アイデンティティベースのポリシーは、アイデンティティ (ユーザー、グループ、またはロール) にアタッチできる JSON アクセス許可ポリシードキュメントです。これらのポリシーは、アイデンティティがどのリソースに対してどのような条件下でどのようなアクションを実行できるかを制御します。アイデンティティベースポリシーの作成方法については、IAM ユーザーガイド の カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する を参照してください。

アイデンティティベースのポリシーは、インラインポリシー (単一の ID に直接埋め込む) または管理ポリシー (複数の ID にアタッチされたスタンドアロンポリシー) にすることができます。管理ポリシーと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「管理ポリシーとインラインポリシーのいずれかを選択する」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。例としては、IAM ロール信頼ポリシーや Amazon S3 バケットポリシーなどがあります。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。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。

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を使用できません。

その他のポリシータイプ

AWS は、より一般的なポリシータイプによって付与されるアクセス許可の最大数を設定できる追加のポリシータイプをサポートしています。

- アクセス許可の境界 – アイデンティティベースのポリシーで IAM エンティティに付与することのできるアクセス許可の数の上限を設定します。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) - AWS Organizations内の組織または組織単位の最大のアクセス許可を指定します。詳細については、「AWS Organizations ユーザーガイド」の「サービスコントロールポリシー」を参照してください。
- リソースコントロールポリシー (RCP) – は、アカウント内のリソースで利用できる最大数のアクセス許可を定義します。詳細については、「AWS Organizations ユーザーガイド」の「リソースコントロールポリシー (RCP)」を参照してください。
- セッションポリシー – ロールまたはフェデレーションユーザーの一時セッションを作成する際にパラメータとして渡される高度なポリシーです。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アクセス許可を理解するのがさらに難しくなります。が複数のポリシータイプが関与する場合にリクエストを許可するかどうか AWS を決定する方法については、「IAM ユーザーガイド」の「ポリシー評価ロジック」を参照してください。

が IAM と AWS Batch 連携する方法

IAM を使用して へのアクセスを管理する前に AWS Batch、 で使用できる IAM 機能を確認してください AWS Batch。

で使用できる IAM 機能 AWS Batch

IAM の特徴量	AWS Batch サポート
アイデンティティベースのポリシー	あり
リソースベースのポリシー	なし
ポリシーアクション	あり
ポリシーリソース	あり
ポリシー条件キー	あり
ACL	なし
ABAC (ポリシー内のタグ)	あり
一時的な認証情報	あり
プリンシパルアクセス権限	あり
サービスロール	あり
サービスリンクロール	はい

AWS Batch およびその他の AWS のサービスがほとんどの IAM 機能とどのように連携するかの概要については、「IAM ユーザーガイド」のAWS 「IAM と連携する のサービス」を参照してください。

のアイデンティティベースのポリシー AWS Batch

アイデンティティベースのポリシーのサポート: あり

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 許可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

ベースポリシーの作成方法については、「IAM ユーザーガイド」の「カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する」を参照してください。

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、およびアクションを許可または拒否する条件を指定できます。JSON ポリシーで使用できるすべての要素について学ぶには、IAM ユーザーガイドのIAM JSON ポリシーの要素のリファレンスを参照してください。

AWS Batchのアイデンティティベースのポリシーの例

AWS Batch アイデンティティベースのポリシーの例を表示するには、「」を参照してくださいのアイデンティティベースのポリシーの例 AWS Batch。

のポリシーアクション AWS Batch

ポリシーアクションのサポート: あり

管理者は JSON AWS 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のリソースに対してどのような条件下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。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。

AWS Batch アクションのリストを確認するには、「サービス認可リファレンス」の「 で定義されるアクション AWS Batch」を参照してください。

のポリシーアクションは、アクションの前に次のプレフィックス AWS Batch を使用します。

```
batch
```

単一のステートメントで複数のアクションを指定するには、アクションをカンマで区切ります。

```
"Action": [
      "batch:action1",
      "batch: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": "batch:Describe*"
```

AWS Batch アイデンティティベースのポリシーの例を表示するには、「」を参照してくださいのアイデンティティベースのポリシーの例 AWS Batch。

のポリシーリソース AWS Batch

ポリシーリソースのサポート: あり

管理者は JSON AWS 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のリソースに対してどのような条件下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。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。リソースレベルのアクセス許可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": "*"
```

AWS Batch リソースタイプとその ARNs「 で定義されるリソース AWS Batch」を参照してください。 どのアクションで各リソースの ARN を指定できるかについては、「AWS Batchで定義されるアクション」を参照してください。

AWS Batch向けのポリシー条件キー

サービス固有のポリシー条件キーのサポート: あり

管理者は JSON AWS 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のリソースに対してどのような条件下でアクションを実行できるかということです。

`Condition` 要素は、定義された基準に基づいてステートメントが実行される時期を指定します。イコールや未満などの条件演算子を使用して条件式を作成して、ポリシーの条件とリクエスト内の値を一致させることができます。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。

AWS Batch 条件キーのリストを確認するには、「サービス認可リファレンス」の「 の条件キー AWS Batch」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、「 で定義されるアクション AWS Batch」を参照してください。

を使用した属性ベースのアクセスコントロール (ABAC) AWS Batch

ABAC (ポリシー内のタグ) のサポート: あり

属性ベースのアクセス制御 (ABAC) は、タグと呼ばれる属性に基づいてアクセス許可を定義する認可戦略です。IAM エンティティと AWS リソースにタグをアタッチし、プリンシパルのタグがリソースのタグと一致するときにオペレーションを許可するように ABAC ポリシーを設計できます。

タグに基づいてアクセスを管理するには、`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。

サービスがすべてのリソースタイプに対して 3 つの条件キーすべてをサポートする場合、そのサービスの値はありです。サービスが一部のリソースタイプに対してのみ 3 つの条件キーのすべてをサポートする場合、値は「部分的」になります。

ABAC の詳細については、「IAM ユーザーガイド」の「ABAC 認可でアクセス許可を定義する」を参照してください。ABAC をセットアップする手順を説明するチュートリアルについては、「IAM ユーザーガイド」の「属性ベースのアクセスコントロール (ABAC) を使用する」を参照してください。

で一時的な認証情報を使用する AWS Batch

一時的な認証情報のサポート: あり

一時的な認証情報は、 AWS リソースへの短期的なアクセスを提供し、フェデレーションまたはスイッチロールの使用時に自動的に作成されます。長期的なアクセスキーを使用する代わりに、一時的な認証情報を動的に生成 AWS 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「IAM の一時的な認証情報」および「AWS のサービス と IAM との連携」を参照してください。

AWS Batchのクロスサービスプリンシパル権限

転送アクセスセッション (FAS) のサポート: あり

転送アクセスセッション (FAS) は、 を呼び出すプリンシパルのアクセス許可と AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストをリクエストする を使用します。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。

のサービスロール AWS Batch

サービスロールのサポート: あり

サービスロールとは、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 のサービスに許可を委任するロールを作成するを参照してください。

> **Warning**
>
> サービスロールのアクセス許可を変更すると、 AWS Batch 機能が破損する可能性があります。 AWS Batch が指示する場合以外は、サービスロールを編集しないでください。

のサービスにリンクされたロール AWS Batch

サービスリンクロールのサポート: あり

サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、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アクセス許可を表示できますが、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の作成または管理の詳細については、「IAM と提携するAWS のサービス」を参照してください。表の「サービスリンクロール」列に Yes と記載されたサービスを見つけます。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。

のアイデンティティベースのポリシーの例 AWS Batch

デフォルトでは、 ユーザーおよびロールには、 AWS Batch リソースを作成または変更する権限はありません。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。

これらのサンプルの JSON ポリシードキュメントを使用して IAM 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 ポリシーを作成する (コンソール)」を参照してください。

各リソースタイプの ARN の形式など AWS Batch、 で定義されるアクションとリソースタイプの詳細については、「サービス認可リファレンス」の「 のアクション、リソース、および条件キー AWS Batch」を参照してください。 ARNs

トピック

- ポリシーに関するベストプラクティス
- AWS Batch コンソールの使用
- 自分の権限の表示をユーザーに許可する

ポリシーに関するベストプラクティス

ID ベースのポリシーは、アカウント内の AWS Batch リソースを誰かが作成、アクセス、または削除できるかどうかを決定します。これらのアクションでは、 AWS アカウントに費用が発生する場合があります。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 管理ポリシーを開始し、最小特権のアクセス許可に移行する – ユーザーとワークロードにアクセス許可の付与を開始するには、多くの一般的なユースケースにアクセス許可を付与するAWS 管理ポリシーを使用します。これらは で使用できます AWS アカウント。ユースケースに固有の AWS カスタマー管理ポリシーを定義することで、アクセス許可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 ジョブ機能のAWS マネージドポリシー を参照してください。
- 最小特権を適用する – IAM ポリシーでアクセス許可を設定する場合は、タスクの実行に必要な許可のみを付与します。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、最小特権アクセス許可とも呼ばれています。IAM を使用して許可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 でのポリシーとアクセス許可 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。たとえば、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。条件を使用して、サービスアクションが などの特定の を通じて使用されている場合に AWS のサービス、サービスアクションへのアクセスを許可することもできます CloudFormation。詳細については、IAM ユーザーガイド の IAM JSON ポリシー要素:条件 を参照してください。
- IAM アクセスアナライザー 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 アクセスアナライザー 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 でポリシーを検証する を参照してください。

- 多要素認証 (MFA) を要求する – で IAM ユーザーまたはルートユーザーを必要とするシナリオがある場合は AWS アカウント、MFA をオンにしてセキュリティを強化します。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 を使用した安全な API アクセス 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。

AWS Batch コンソールの使用

AWS Batch コンソールにアクセスするには、最小限のアクセス許可のセットが必要です。これらのアクセス許可により、 の AWS Batch リソースの詳細を一覧表示および表示できます AWS アカウント。最小限必要な許可よりも制限が厳しいアイデンティティベースのポリシーを作成すると、そのポリシーを持つエンティティ (ユーザーまたはロール) に対してコンソールが意図したとおりに機能しません。

AWS CLI または AWS API のみを呼び出すユーザーには、最小限のコンソールアクセス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。

ユーザーとロールが AWS Batch 引き続きコンソールを使用できるようにするには、エンティティに `ConsoleAccess`または AWS Batch `ReadOnly` AWS 管理ポリシーもアタッチします。詳細については、「IAM ユーザーガイド」の「ユーザーへのアクセス許可の追加」を参照してください。

自分の権限の表示を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 または AWS API を使用してプログラムでこのアクションを実行するアクセス許可が含まれています。

```
{
    "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 の 管理ポリシー AWS Batch

AWS 管理ポリシーを使用して、チームおよびプロビジョニングされた AWS リソースの ID アクセス管理を簡素化できます。 AWS 管理ポリシーは、さまざまな一般的なユースケースをカバーし、アカウントでデフォルトで使用でき AWS 、ユーザーに代わって維持および更新されます。 AWS 管理ポリシーのアクセス許可は変更できません。より柔軟性が必要な場合は、IAM カスタマー管理ポリシーを作成することもできます。このようにして、チームのプロビジョニングされたリソースに、必要な権限のみを付与できます。

AWS 管理ポリシーの詳細については、IAM ユーザーガイドの「 AWS 管理ポリシー」を参照してください。

AWS サービスは、ユーザーに代わって AWS 管理ポリシーを維持および更新します。定期的に、AWS サービスは AWS マネージドポリシーに追加のアクセス許可を追加します。 AWS マネージド

ポリシーは、新機能の起動またはオペレーションが利用可能になったときに更新されます。この更新は、ポリシーが添付されているすべてのアイデンティティ (ユーザー、グループ、ロール) に自動的に影響します。ただし、権限を削除したり、既存の権限を破棄することはありません。

さらに、 は、複数のサービスにまたがるジョブ関数の 管理ポリシー AWS をサポートしています。たとえば、 ReadOnlyAccess AWS マネージドポリシーは、すべての AWS サービスとリソースへの読み取り専用アクセスを提供します。サービスが新機能を起動すると、 は新しいオペレーションとリソースの読み取り専用アクセス許可 AWS を追加します。ジョブ機能のポリシーの一覧および詳細については、「IAM ユーザーガイド」の「AWS のジョブ機能のマネージドポリシー」を参照してください。

AWS マネージドポリシー: BatchServiceRolePolicy

BatchServiceRolePolicy 管理 IAM ポリシーは、AWSServiceRoleForBatch サービスにリンクされたロールによって使用されます。これにより、 AWS Batch はユーザーに代わってアクションを実行できます。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。詳細については、「AWS Batch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。

このポリシーにより AWS Batch 、 は特定のリソースに対して次のアクションを実行できます。

- autoscaling – AWS Batch が Amazon EC2 Auto Scaling リソースを作成および管理できるようにします。 は、ほとんどのコンピューティング環境の Amazon EC2 Auto Scaling グループ AWS Batch を作成および管理します。
- ec2 – AWS Batch が Amazon EC2 インスタンスのライフサイクルを制御し、起動テンプレートとタグを作成および管理できるようにします。 は、一部の EC2 スポットコンピューティング環境の EC2 スポットフリートリクエスト AWS Batch を作成および管理します。
- ecs - AWS Batch ジョブ実行用の Amazon ECS クラスター、タスク定義、タスクの作成と管理を許可します。
- eks - AWS Batch 検証用の Amazon EKS クラスターリソースの記述を許可します。
- iam - 所有者が提供するロールを検証して Amazon EC2、Amazon EC2 Auto Scaling、Amazon ECS に渡すことを AWS Batch に許可します。
- logs – AWS Batch ジョブのロググループとログストリームの作成と管理を に許可します AWS Batch 。

ポリシーの JSON を表示するには、「AWS マネージドポリシーリファレンスガイド」の「BatchServiceRolePolicy」を参照してください。

AWS マネージドポリシー: AWSBatchServiceRolePolicyForSageMaker

`AWSServiceRoleForAWSBatchWithSagemaker` では AWS Batch 、ユーザーに代わってアクションを実行できます。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。詳細については、「AWS Batch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。

このポリシーにより AWS Batch 、 は特定のリソースに対して次のアクションを実行できます。

- `sagemaker` – が SageMaker AI トレーニングジョブやその他の SageMaker AI リソースを管理 AWS Batch できるようにします。
- `iam:PassRole` – ジョブ実行のために AWS Batch がカスタマー定義の実行ロールを SageMaker AI に渡すことを許可します。リソース制約により、SageMaker AI サービスにロールを渡すことができます。

ポリシーの JSON を表示するには、「AWS マネージドポリシーリファレンスガイド」の「AWSBatchServiceRolePolicyForSageMaker」を参照してください。

AWS マネージドポリシー: AWSBatchServiceRole ポリシー

AWSBatchServiceRole というロールのアクセス許可ポリシーは、指定されたリソースでの以下のアクションの実行を AWS Batch に許可します。

AWSBatchServiceRole 管理 IAM ポリシーは、AWSBatchServiceRole という名前のロールによって使用されることが多く、次のアクセス許可が含まれます。最小特権を付与する標準のセキュリティアドバイスに従って、AWSBatchServiceRole 管理ポリシーをガイドとして使用できます。マネージドポリシーで付与されているアクセス許可のいずれかがユースケースに必要でない場合、カスタムポリシーを作成し、必要なアクセス許可のみを追加します。この AWS Batch マネージドポリシーとロールは、ほとんどのコンピューティング環境タイプで使用できますが、エラーが発生しにくく、範囲が広く、マネージドエクスペリエンスを向上させるには、サービスにリンクされたロールの使用が推奨されます。

- `autoscaling` – AWS Batch が Amazon EC2 Auto Scaling リソースを作成および管理できるようにします。 は、ほとんどのコンピューティング環境の Amazon EC2 Auto Scaling グループ AWS Batch を作成および管理します。

- `ec2` – AWS Batch が Amazon EC2 インスタンスのライフサイクルを管理し、起動テンプレートとタグを作成および管理できるようにします。 は、一部の EC2 スポットコンピューティング環境の EC2 スポットフリートリクエスト AWS Batch を作成および管理します。
- `ecs` - AWS Batch ジョブ実行用の Amazon ECS クラスター、タスク定義、タスクの作成と管理を許可します。
- `iam` - 所有者が提供するロールを検証して Amazon EC2、Amazon EC2 Auto Scaling、Amazon ECS に渡すことを AWS Batch に許可します。
- `logs` – AWS Batch ジョブのロググループとログストリームの作成と管理を に許可します AWS Batch 。

ポリシーの JSON を表示するには、「AWS マネージドポリシーリファレンスガイド」の「AWSBatchServiceRole」を参照してください。

AWS マネージドポリシー: AWSBatchFullAccess

AWSBatchFullAccess ポリシーは、 AWS Batch リソースへのフルアクセスを AWS Batch アクションに付与します。また、Amazon EC2、Amazon EC2、Amazon ECS、Amazon ECS、Amazon EKS、IAM サービスへのアクションアクセスを許可します。これは、ユーザーまたはロールのいずれかの IAM ID が、ユーザーに代わって作成された AWS Batch マネージドリソースを表示できるようにするためです。最後に、このポリシーは、選択した IAM ロールをこれらのサービスに渡すことも許可します。

AWSBatchFullAccess を IAM エンティティにアタッチできます。 は、ユーザーに代わって がアクションを実行できるようにするサービスロール AWS Batch にもこのポリシー AWS Batch をアタッチします。

ポリシーの JSON を表示するには、「AWS マネージドポリシーリファレンスガイド」の「AWSBatchFullAccess」を参照してください。

AWS Batch AWS 管理ポリシーの更新

このサービスがこれらの変更の追跡を開始 AWS Batch してからの の AWS 管理ポリシー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アラートについては、 AWS Batch ドキュメント履歴ページの 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
AWSBatchServiceRolePolicyForSageMaker ポリシーを追加	がユーザーに代わって SageMaker AI を管理できるようにする AWSBatchServiceRolePolicyForSageMaker SageMaker サービスにリンクされたロールの新しい AWS 管理ポリシーを追加しました。 AWS Batch	2025 年 7 月 31 日
BatchServiceRolePolicy ポリシーの更新	スポットフリートリクエストの履歴と Amazon EC2 Auto Scaling のアクティビティの記述のサポートを追加するように更新しました。	2023 年 12 月 5 日
AWSBatchServiceRole ポリシーの追加	ステートメント IDs、 `ec2:DescribeSpotFleetRequestHistory` とにアクセス AWS Batch 許可を付与するように更新しました`autoscaling:DescribeScalingActivities` 。	2023 年 12 月 5 日
BatchServiceRolePolicy ポリシーの更新	Amazon EKS クラスターを記述するためのサポートを追加するために更新されました。	2022 年 10 月 20 日
AWSBatchFullAccess ポリシーが更新されました	Amazon EKSクラスターのリストと説明のサポートを追加しました。	2022 年 10 月 20 日
BatchServiceRolePolicy ポリシーの更新	AWS Resource Groupsによって管理される Amazon EC2 キャパシティ予約グループのサポートを追加するように更新されました。詳細については、「Amazon EC2 ユーザーガイド」の「キャパシティ予約グループ」を参照してください。	2022 年 5 月 18 日

変更	説明	日付
BatchServiceRolePolicy と AWSBatchServiceRole ポリシーが更新されました	異常なインスタンスが置き換えられるように、Amazon EC2 の AWS Batch マネージドインスタンスのステータスを記述するサポートを追加しました。	2021 年 12 月 6 日
BatchServiceRolePolicy ポリシーの更新	Amazon EC2 で配置グループ、キャパシティの予約、Elastic GPU、および Elastic Inference リソースのサポートを追加するように更新されました。	2021 年 3 月 26 日
BatchServiceRolePolicy ポリシーの追加 (2021年3月10日)	AWSServiceRoleForBatch サービスにリンクされたロール用の BatchServiceRolePolicy 管理ポリシーを使用すると、コンピューティング環境で使用する独自のロールを維持する代わりに、 AWS Batchが管理するサービスにリンクされたロールを使用することができます。このポリシーを使用すると、コンピュート環境で使用するために独自のロールを維持する必要がありません。	2021 年 3 月 10 日
AWSBatchFullAccess - サービスにリンクされたロールを追加するための権限を追加	AWSServiceRoleForBatch] サービスにリンクされたロールをアカウントに追加できるように IAM アクセス許可を追加しました。	2021 年 3 月 10 日
AWS Batch が変更の追跡を開始しました	AWS Batch は、 AWS 管理ポリシーの変更の追跡を開始しました。	2021 年 3 月 10 日

AWS Batch IAM ポリシー、ロール、アクセス許可

デフォルトでは、ユーザーには AWS Batch 、リソースを作成または変更したり、 AWS Batch API、AWS Batch コンソール、または を使用してタスクを実行したりするアクセス許可はありません AWS CLI。ユーザーがこれらのリソースを利用するには、特定のリソースと APIアクションを使用す

る許可を付与するIAM ポリシーを作成する必要があります。続いて、こうしたアクセス権限が必要なユーザーまたはグループにそのポリシーをアタッチします。

ポリシーをユーザーまたはユーザーグループにアタッチする場合、ポリシーによって特定リソースの特定タスクを実行するユーザーの権限が許可または拒否されます。詳細については、IAM ユーザーガイド の アクセス許可とポリシーを参照してください。カスタム IAM ポリシーの管理と作成の詳細については、IAM ポリシーの管理を参照してください。

AWS Batch は、 AWS のサービス ユーザーに代わって他の を呼び出します。そのため、 AWS Batch は 認証情報を使用して認証する必要があります。具体的には、これらのアクセス許可を提供する IAM ロールとポリシーを作成して AWS Batch 認証します。次に、それらの作成時に、そのロールをコンピューティング環境に関連付けます。詳細については、IAM ユーザーガイドのAmazon ECS インスタンスロール「」、「IAM ロール」、「サービスにリンクされたロールの使用」、およびAWS 「サービスにアクセス許可を委任するロールの作成」を参照してください。 https://docs.aws.amazon.com/IAM/latest/UserGuide/roles-toplevel.html

トピック

- IAM ポリシーの構造
- リソース: のポリシーの例 AWS Batch
- リソース: AWS Batch 管理ポリシー

IAM ポリシーの構造

次のトピックでは、IAM ポリシーの簡単な構造について説明します。

トピック

- ポリシー構文
- の API アクション AWS Batch
- の Amazon リソースネーム AWS Batch
- ユーザーが必要なアクセス許可を持っていることを確認する

ポリシー構文

IAM ポリシーは 1 つ以上のステートメントで構成される JSON ドキュメントです。各ステートメントの構成は以下のとおりです。

```
{
  "Statement":[{
    "Effect":"effect",
    "Action":"action",
    "Resource":"arn",
    "Condition":{
      "condition":{
    "key":"value"
    }
      }
    }
  ]
}
```

ステートメントは 4 つの主な工レメントで構成されています。

- Effect: effect は、Allow または Deny にすることができます。デフォルトでは、 ユーザーはリソースおよび API アクションを使用するアクセス許可がありません。そのため、すべてのリク工ストが拒否されます。明示的な許可はデフォルトに上書きされます。明示的な拒否はすべての許可に上書きされます。
- Action] (アクション): action は、アクセス許可を付与または拒否する対象とする、特定の API アクションです。アクションを指定する方法に関する指示は、の API アクション AWS Batch を参照してください。
- Resource] (リソース): アクションによって影響を及ぼされるリソースです。 AWS Batch API アクションの中には、アクションによって作成/変更できるリソースをポリシー内で特定できるものもあります。ステートメントでリソースを指定するには、Amazon リソースネーム (ARN) を使用します。詳細については、「AWS Batch API アクションでサポートされているリソースレベルのアクセス許可」および「の Amazon リソースネーム AWS Batch」を参照してください。 AWS Batch API オペレーションが現在リソースレベルのアクセス許可をサポートしていない場合は、ワイルドカード (*) を含めて、すべてのリソースが アクションの影響を受けるように指定します。
- Condition] (条件): condition はオプションです。ポリシーの発効条件を指定するために使用します。

の IAM ポリシーステートメントの例の詳細については AWS Batch、「」を参照してくださいリソース: のポリシーの例 AWS Batch。

の API アクション AWS Batch

IAM ポリシーステートメントで、IAM をサポートするすべてのサービスからの任意の API アクションを指定できます。の場合 AWS Batch、API アクションの名前に次のプレフィックスを使用します batch: (例: batch:SubmitJobおよび batch:CreateComputeEnvironment)。

単一のステートメントで複数のアクションを指定するには、各アクションをカンマで区切ります。

```
"Action": ["batch:action1", "batch:action2"]
```

またワイルドカード (*) を含めて、複数のアクションを指定することもできます。例えば、Describe という単語で始まる名前を用いて、すべてのアクションを指定できます。

```
"Action": "batch:Describe*"
```

すべての AWS Batch API アクションを指定するには、ワイルドカード (*) を含めます。

```
"Action": "batch:*"
```

AWS Batch アクションのリストについては、 AWS Batch API リファレンスの「アクション」を参照してください。

の Amazon リソースネーム AWS Batch

各 IAM ポリシーステートメントは、ユーザーがAmazon リソースネーム (ARN) を使用して指定したリソースに適用されます。

Amazon リソースネーム (ARN) には、次の一般的な構文があります:

```
arn:aws:[service]:[region]:[account]:resourceType/resourcePath
```

service

サービス (例: batch)。

region

リソース AWS リージョン の （ 例: us-east-2 ） 。

アカウント

AWS アカウント ID。ハイフンなし (例: 123456789012)。

resourceType

リソースの種類 (例: `compute-environment`)。

resourcePath

リソースを識別するパス。パスにワイルドカードの(*)が使用できます。

AWS Batch API オペレーションは現在、複数の API オペレーションに対するリソースレベルのアクセス許可をサポートしています。詳細については、「AWS Batch API アクションでサポートされているリソースレベルのアクセス許可」を参照してください。すべてのリソースを指定する場合、または特定の API アクションが ARN をサポートしていない場合は、`Resource` エレメントに (*) ワイルドカードを含めます。

```
"Resource": "*"
```

ユーザーが必要なアクセス許可を持っていることを確認する

IAM ポリシーを本稼働環境に置く前に、そのポリシーがユーザーに対し、特定の API アクションおよび必要なリソースを使用のアクセス許可を付与しているかどうかを確認することをお勧めします。

これを行うには、まずテスト目的のユーザーを作成して、IAM ポリシーをテストユーザーにアタッチします。次に、テストユーザーとしてリクエストを作成します。テストリクエストは、コンソールまたは AWS CLIを使用して行うことができます。

> **Note**
>
> IAM ポリシーシミュレーターを使用してポリシーをテストすることもできます。ポリシーシミュレーターの詳細については、IAM ユーザーガイドのIAM ポリシーシミュレーターで作業するを参照してください。

ポリシーが想定したアクセス許可をユーザーに付与していない場合、または過度に許可されている場合、必要に応じてポリシーを調整できます。必要な結果を得るまで再テストします。

> **Important**
>
> ポリシーの変更が反映され、有効になるには数分間かかります。したがって、ポリシーの更新をテストする前に、合格するには、少なくとも5分みておくことをお勧めします。

認可チェックが失敗した場合、リクエストでは診断情報でエンコードされたメッセージが返されます。DecodeAuthorizationMessage アクションを使用してメッセージをデコードできます。詳細については、AWS Security Token Service API リファレンス のDecodeAuthorizationMessage、およびAWS CLI コマンドリファレンスの decode-authorization-message を参照してください。

リソース: のポリシーの例 AWS Batch

アカウントのユーザーがアクセスできる呼び出しやリソースを制限する特定の IAM ポリシーを作成できます。このポリシーをユーザーにアタッチできます。

ポリシーをユーザーまたはグループにアタッチすると、ポリシーによって特定リソースについて特定タスクを実行する権限が許可または拒否されます。詳細については、IAM ユーザーガイド の アクセス許可とポリシーを参照してください。カスタム IAM ポリシーを管理および作成する方法については、IAM ポリシーの管理を参照してください。

以下の例では、ユーザーの AWS Batchに対するアクセス権限を制御するために使用できるポリシーステートメントについて説明しています。

例

- リソース: の読み取り専用アクセス AWS Batch
- リソース: POSIX ユーザー、Docker イメージ、特権レベル、ジョブ送信のロールを制限する
- リソース: ジョブ送信時、ジョブ定義プレフィックスを制限する
- リソース: ジョブキューを制限する
- すべての条件が文字列に一致した場合はアクションを拒否する
- リソース: いずれかの条件キーが文字列に一致した場合はアクションを拒否する
- リソース: batch:ShareIdentifier 条件キーを使用する
- で SageMaker AI リソースを管理する AWS Batch
- リソース: ジョブ定義とジョブキューのリソースタグによってジョブ送信を制限する

リソース: の読み取り専用アクセス AWS Batch

次のポリシーは、 Describeおよび で始まる名前のすべての AWS Batch API アクションを使用するアクセス許可をユーザーに付与しますList。

別のステートメントでアクセス権限を付与されない限り、ユーザーにはそのリソースに対してアクションを実行するアクセス権限がありません。デフォルトでは、API アクションを使用する権限は拒否されます。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "batch:Describe*",
                "batch:List*",
                "batch:Get*"
            ],
            "Resource": "*"
        }
    ]
}
```

リソース: POSIX ユーザー、Docker イメージ、特権レベル、ジョブ送信のロールを制限する

次のポリシーは、POSIX ユーザーが自身の制限されたジョブ定義のセットを管理することを許可します。

最初と 2 番目のステートメントを使用して、名前が *JobDefA_* で始まるジョブ定義を登録および登録解除します。

また、最初のステートメントでは、条件付きコンテキストキーを使用して POSIX ユーザー、特権ステータス、コンテナイメージ値をジョブ定義の `containerProperties` 内に制限します。詳細については、AWS Batch API リファレンスの RegisterJobDefinition を参照してください。この例では、POSIX ユーザーが nobody に設定されている場合にのみ、ジョブ定義を登録できます。特権フラグは `false` に設定されています。最後に、イメージは Amazon ECR レポジトリの `myImage` に設定されています。

> **Important**
>
> Docker は、コンテナイメージ内から `user` パラメータをユーザー `uid` に解決します。ほとんどの場合、これはコンテナイメージ内の `/etc/passwd` ファイルにあります。ジョブ定義と関連付けられたすべての IAM ポリシーの両方で直接 `uid` 値を使用することで、この名前

解決を回避できます。 AWS Batch API オペレーションおよび batch:User IAM 条件キーのいずれにおいても、数値がサポートされます。

3 番目のステートメントを使用して、ジョブ定義を特定のロールのみに制限します。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "batch:RegisterJobDefinition"
            ],
            "Resource": [
                "arn:aws:batch:us-east-2:999999999999:job-definition/JobDefA_*"
            ],
            "Condition": {
                "StringEquals": {
                    "batch:User": [
                        "nobody"
                    ],
                    "batch:Image": [
                        "999999999999.dkr.ecr.us-east-2.amazonaws.com/myImage"
                    ]
                },
                "Bool": {
                    "batch:Privileged": "false"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "batch:DeregisterJobDefinition"
            ],
            "Resource": [
                "arn:aws:batch:us-east-2:999999999999:job-definition/JobDefA_*"
            ]
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::999999999999:role/MyBatchJobRole"
            ]
        }
    ]
}
```

リソース: ジョブ送信時、ジョブ定義プレフィックスを制限する

次のポリシーを使用して、*JobDefA* で始まるすべてのジョブ定義のジョブキューにジョブを送信します。

> **Important**
>
> ジョブ送信へのリソースレベルアクセスに絞り込む場合、ジョブキューおよびジョブ定義の両方のリソースタイプを指定する必要があります。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "batch:SubmitJob"
            ],
            "Resource": [
                "arn:aws:batch:us-east-2:111122223333:job-definition/JobDefA_*",
                "arn:aws:batch:us-east-2:111122223333:job-queue/*"
            ]
        }
    ]
}
```

リソース: ジョブキューを制限する

次のポリシーを使用して、任意のジョブ定義名で queue1 という名前の特定のジョブキューへのジョブの送信をユーザーに許可します。

> **Important**
>
> ジョブ送信へのリソースレベルアクセスに絞り込む場合、ジョブキューおよびジョブ定義の両方のリソースタイプを指定する必要があります。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "batch:SubmitJob"
            ],
            "Resource": [
                "arn:aws:batch:us-east-2:888888888888:job-definition/*",
                "arn:aws:batch:us-east-2:888888888888:job-queue/queue1"
            ]
        }
    ]
}
```

すべての条件が文字列に一致した場合はアクションを拒否する

次のポリシーは、 batch:Image (コンテナイメージ ID) 条件キーが*#string1*」と batch:LogDriver (コンテナログドライバー) 条件キーが*#string2*」の両方の場合、RegisterJobDefinitionAPI AWS Batch オペレーションへのアクセスを拒否します。 は各コンテナの条件キーを評価します。マルチノード並列ジョブのように、ジョブが複数のコンテナにまたがる場合、コンテナの構成が異なる可能性があります。1 つのステートメントで複数の条件キーを評価する場合、条件キーは AND ロジックを使用して結合されます。そのため、1 つのコンテナで複数の条件キーのいずれかが一致しない場合、そのコンテナに Deny の効果は適用されません。それどころか、同じジョブ内の別のコンテナが拒否される可能性があります。

の条件キーのリストについては AWS Batch、「サービス認可リファレンス」の「 の条件キー AWS Batch」を参照してください。この方法は、batch:ShareIdentifier を除くすべての batch 条件キーで使用できます。batch:ShareIdentifier 条件キーは、ジョブ定義ではなく、ジョブに対して定義されます。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "batch:RegisterJobDefinition"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Deny",
      "Action": "batch:RegisterJobDefinition",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "batch:Image": "string1",
          "batch:LogDriver": "string2"
        }
      }
    }
  ]
}
```

リソース: いずれかの条件キーが文字列に一致した場合はアクションを拒否する

次のポリシーは、batch:Image (コンテナイメージ ID) 条件キーが "*string1*" または batch:LogDriver (コンテナログドライバー) 条件キーが "*string2*" の場合、RegisterJobDefinition API オペレーションへのアクセスを拒否します。マルチノード並列ジョブのように、ジョブが複数のコンテナにまたがる場合、コンテナの構成が異なる可能性があります。1 つのステートメントで複数の条件キーを評価する場合、条件キーは AND ロジックを使用して結合さ

れます。そのため、1 つのコンテナで複数の条件キーのいずれかが一致しない場合、そのコンテナに Deny の効果は適用されません。それどころか、同じジョブ内の別のコンテナが拒否される可能性があります。

の条件キーのリストについては AWS Batch、「サービス認可リファレンス」の「 の条件キー AWS Batch」を参照してください。この方法は、batch:ShareIdentifier を除くすべての batch 条件キーで使用できます。(batch:ShareIdentifier 条件キーは、ジョブ定義ではなく、ジョブに対して定義されます)。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "batch:RegisterJobDefinition"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Deny",
      "Action": [
        "batch:RegisterJobDefinition"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "batch:Image": [
            "string1"
          ]
        }
      }
    },
    {
      "Effect": "Deny",
      "Action": [
```

```
        "batch:RegisterJobDefinition"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "batch:LogDriver": [
            "string2"
          ]
        }
      }
    }
  ]
}
```

リソース: batch:ShareIdentifier 条件キーを使用する

次のポリシーを使用して、jobDefA ジョブ定義を使用するジョブを lowCpu 配分識別子で jobqueue1 ジョブキューに送信します。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "batch:SubmitJob"
      ],
      "Resource": [
        "arn:aws:batch:us-east-2:555555555555:job-definition/JobDefA",
        "arn:aws:batch:us-east-2:555555555555:job-queue/jobqueue1"
      ],
      "Condition": {
        "StringEquals": {
          "batch:ShareIdentifier": [
            "lowCpu"
          ]
        }
```

```
      }
    }
  ]
}
```

で SageMaker AI リソースを管理する AWS Batch

このポリシーにより、 AWS Batch は SageMaker AI リソースを管理できます。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "batch:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "arn:aws:iam::*:role/
*AWSServiceRoleForAWSBatchWithSagemaker",
            "Condition": {
                 "StringEquals": {
                     "iam:AWSServiceName": "sagemaker-
queuing.batch.amazonaws.com"
                 }
             }
         },
         {
             "Effect": "Allow",
             "Action": "iam:PassRole",
             "Resource": "*",
             "Condition": {
                 "StringEquals": {
                     "iam:PassedToService": [
```

```
                    "sagemaker.amazonaws.com"
                ]
            }
        }
    }
  ]
}
```

リソース: ジョブ定義とジョブキューのリソースタグによってジョブ送信を制限する

ジョブキューに タグがあり、ジョブ定義Environment=devに タグがある場合にのみ、次のポリシーを使用してジョブを送信しますProject=calc。このポリシーは、リソースタグを使用して、ジョブの送信中に AWS Batch リソースへのアクセスを制御する方法を示しています。

> **Important**
>
> ジョブ定義リソースタグを評価するポリシーを使用してジョブを送信する場合は、ジョブ定義リビジョン形式 () を使用してジョブを送信する必要がありますjob-definition:revision。リビジョンを指定せずにジョブを送信すると、ジョブ定義タグは評価されず、意図したアクセスコントロールがバイパスされる可能性があります。リソース ARN の*:*パターンでは、送信にリビジョンを含める必要があるため、タグポリシーを常に効果的に適用できます。

このポリシーは、異なるリソースタイプに異なるタグ条件を適用するため、2 つの異なるステートメントを使用します。ジョブ送信へのリソースレベルアクセスに絞り込む場合、ジョブキューおよびジョブ定義の両方のリソースタイプを指定する必要があります。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "batch:SubmitJob",
      "Resource": "arn:aws:batch:*:*:job-queue/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Environment": "dev"
        }
      }
```

```
    },
    {
      "Effect": "Allow",
      "Action": "batch:SubmitJob",
      "Resource": "arn:aws:batch:*:*:job-definition/*:*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Project": "calc"
        }
      }
    }
  ]
}
```

リソース: AWS Batch 管理ポリシー

AWS Batch には、ユーザーにアタッチできる管理ポリシーが用意されています。このポリシーは、AWS Batch リソースと API オペレーションを使用するアクセス許可を提供します。このポリシーを直接適用することも、独自のポリシーを作成する開始点として使用することもできます。これらのポリシーに記載されている各 API オペレーションの詳細については、AWS Batch API リファレンスのアクションを参照してください。

AWSBatchFullAccess

このポリシーは、 へのフル管理者アクセスを許可します AWS Batch。

ポリシーの JSON を表示するには、「AWS マネージドポリシーリファレンスガイド」の「AWSBatchFullAccess」を参照してください。

AWS Batch IAM 実行ロール

実行ロールは、ユーザーに代わって AWS API コールを行うアクセス許可を Amazon ECS コンテナと AWS Fargate エージェントに付与します。

> **Note**
>
> 実行ロールは Amazon ECS コンテナエージェントバージョン 1.16.0 以降でサポートされています。

IAM 実行ロールは、タスクの要件に応じて要求されます。さまざまな目的とサービスの実行ロールを、アカウントに複数関連付けることができます。

> **Note**
>
> 詳細については、 ユーザーガイドのAmazon ECS インスタンスロールを参照してくださいAmazon ECS インスタンスロール。サービスロールの詳細については、が IAM と AWS Batch 連携する方法を参照してください。

Amazon ECS は、AmazonECSTaskExecutionRolePolicyマネージドポリシーを提供します。は、という管理ポリシーを提供します。このポリシーには、上記の一般的ユースケースで必要なアクセス許可が含まれています。以下に説明する特別な使用例のために、実行ロールにインラインポリシーを追加する必要があるかもしれません。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecr:GetAuthorizationToken",
        "ecr:BatchCheckLayerAvailability",
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "*"
    }
  ]
}
```

AWS Batch API アクションでサポートされているリソースレベルのアクセス許可

リソースレベルのアクセス許可という用語は、ユーザーがアクションを実行できるリソースを指定する機能を指します。 AWS Batch では、リソースレベルのアクセス許可が部分的にサポートされています。一部の AWS Batch アクションでは、満たす必要がある条件に基づいて、ユーザーがそれらのアクションをいつ使用できるかを制御できます。ユーザーが使用できる特定のリソースに基づいて制御することもできます。例えば、特定のジョブ定義がある特定のジョブキューでのみ、ジョブを送信するアクセス許可をユーザーに付与できます。

各リソースタイプの ARN の形式など、 AWS バッチで定義されるアクションとリソースタイプの詳細については、AWS Batch「サービス認可リファレンス」の「 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。 ARNs

チュートリアル: IAM 実行ロールを作成する

アカウントにまだ IAM 実行ロールがない場合は、以下の手順でロールを作成してください。

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. ナビゲーションペインで Roles (ロール) を選択してください。
3. ロールの作成 を選択します。
4. 信頼できるエンティティタイプ で、を選択します。
5. [サービスまたはユースケース] では、[エラスティックコンテナサービス] を選択します。その後、[エラスティックコンテナサービスタスク] を再度選択します。
6. [次へ] を選択します。
7. アクセス権限ポリシー については、AmazonECSTaskExecutionRolePolicy を検索してください。
8. AmazonECSTaskExecutionRolePolicy ポリシーの左側にあるチェックボックスをオンにした後、ポリシーをアタッチ を選択します。
9. ロール名に `ecsTaskExecutionRole` と入力し、ロールの作成 を選択します。

チュートリアル: IAM 実行ロールを確認する

以下の手順を使用して、アカウントに IAM 実行ロールが既にあることを確認し、必要に応じて管理された IAM ポリシーをアタッチできます。

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. 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 ecsTaskExecutionRole を検索します。ロールが検出できない場合は、チュートリアル: IAM 実行ロールを作成する を参照してください。ロールが見つかった場合は、添付されたポリシーを表示するロールを選択します。
4. Permissions] (アクセス許可) タブで、AmazonECSTaskExecutionRolePolicy 管理ポリシーがロールにアタッチされていることを検証します。ポリシーがアタッチされている場合、 実行ロールは適切に設定されています。そうでない場合、次のサブステップに従ってポリシーをアタッチします。

   a. アクセス許可を追加、ポリシーをアタッチ の順に選択します。

   b. AmazonECSTaskExecutionRolePolicy。

   c. AmazonECSTaskExecutionRolePolicy ポリシーの左にあるボックスをオンにし、ポリシーをアタッチ を選択します。
5. Trust relationships (信頼関係) を選択します。
6. 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していれば、ロールは正しく設定されています。信頼関係の編集 を選択して、次のポリシーを追加し、信頼ポリシーの更新 を選択します。

   JSON

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

AWS Batchのサービスにリンクされたロールの使用

AWS Batch は AWS Identity and Access Management (IAM) 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、直接リンクされた一意のタイプの IAM ロールです AWS Batch。サービスにリンクされたロールは によって事前定義 AWS Batch されており、サービスがユーザーに代わって他の AWS 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含まれています。

AWS Batch は 2 つの異なるサービスにリンクされたロールを使用します。

- AWSServiceRoleForBatch - コンピューティング環境を含む AWS Batch オペレーション用。
- AWSServiceRoleForAWSBatchWithSagemaker - SageMaker AI ワークロードの管理およびキューイング用。

トピック

- での ロールの使用 AWS Batch
- SageMaker AI AWS Batch での のロールの使用

での ロールの使用 AWS Batch

AWS Batch は AWS Identity and Access Management (IAM) 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、直接リンクされた一意のタイプの IAM ロールです AWS Batch。サービスにリンクされたロールは によって事前定義 AWS Batch されており、サービスがユーザーに代わって他の AWS 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含まれています。

サービスにリンクされたロールを使用すると、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、 の設定 AWS Batch が簡単になります。 は、サービスにリンクされたロールのアクセス許可 AWS Batch を定義し、特に定義されている場合を除き、 のみがそのロールを引き受け AWS Batch 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。

> **Note**
>
> AWS Batch コンピューティング環境のサービスロールを指定するには、次のいずれかを実行します。

- サービスロールには空の文字列を使用します。これにより、 はサービスロール AWS Batch を作成できます。
- 以下の形式でサービスロールを指定します:`arn:aws:iam::account_number:role/aws-service-role/batch.amazonaws.com/AWSServiceRoleForBatch`。

詳細については、「 AWS Batch ユーザーガイド正しくないロール名または ARN」の「」を参照してください。

サービスリンクロールを削除するには、最初に関連リソースを削除する必要があります。これにより、 AWS Batch リソースへのアクセス許可が誤って削除されないため、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他のサービスの詳細については、AWS 「IAM と連携するサービス」を参照し、「サービスにリンクされたロール」列で「はい」があるサービスを探します。サービスリンク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、リンクで [はい] を選択します。

のサービスにリンクされたロールのアクセス許可 AWS Batch

AWS Batch は、AWSServiceRoleForBatch 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。 がユーザーに代わって AWS リソースを作成および管理 AWS Batch できるようにします。

サービスにリンクされたロール AWSServiceRoleForBatch は、次のサービスを信頼してロールを引き受けます。

- `batch.amazonaws.com`

BatchServiceRolePolicy という名前のロールアクセス許可ポリシーにより AWS Batch 、 は指定されたリソースに対して次のアクションを実行できます。

- `autoscaling` – AWS Batch が Amazon EC2 Auto Scaling resources を作成および管理できるようにします。 は、ほとんどのコンピューティング環境の Amazon EC2 Auto Scaling グループ AWS Batch を作成および管理します。
- `ec2` – AWS Batch が Amazon EC2 インスタンスのライフサイクルを制御し、起動テンプレートとタグを作成および管理できるようにします。 は、一部の EC2 スポットコンピューティング環境の EC2 スポットフリートリクエスト AWS Batch を作成および管理します。

- `ecs` - AWS Batch ジョブ実行用の Amazon ECS クラスター、タスク定義、タスクの作成と管理を許可します。
- `eks` - AWS Batch 検証用の Amazon EKS クラスターリソースの記述を許可します。
- `iam` - 所有者が提供するロールを検証して Amazon EC2、Amazon EC2 Auto Scaling、Amazon ECS に渡すことを AWS Batch に許可します。
- `logs` – AWS Batch ジョブのロググループとログストリームの作成と管理を に許可します AWS Batch 。

ユーザー、グループ、またはロールにサービスリンクロールの作成、編集、または削除を許可するには、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「Service-linked role permissions」を参照してください。

AWS Batchのサービスリンクロールの作成

サービスリンクロールを手動で作成する必要はありません。 AWS マネジメントコンソール、、AWS CLIまたは AWS API でコンピューティング環境を作成すると、 AWS Batch 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。

> **Important**
>
> このサービスリンクロール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にアカウントに表示されます。 AWS Batch サービスにリンクされたロールのサポートを開始した 2021 年 3 月 10 日より前にサービスを使用していた場合、 はアカウントに AWSServiceRoleForBatch ロール AWS Batch を作成しました。詳細については、「新しいロールが AWS アカウント」を参照してください。

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。コンピューティング環境を作成すると、 はサービスにリンクされたロールを再度 AWS Batch 作成します。

AWS Batchのサービスにリンクされたロールの編集

AWS Batch では、AWSServiceRoleForBatch 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の作成後は、さまざまなエンティティがロールを参照す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 を使用してロールの説明を編集することはできます。詳細については、「IAM ユーザーガイド」の「サービスリンクロールの編集」を参照してください。

IAM エンティティが AWSServiceRoleForBatch のサービスにリンクされたロールの説明を編集できるようにするには

以下のステートメントをアクセス許可ポリシーに追加します。IAM エンティティが サービスにリンクされたロールの説明を編集することを許可します。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:UpdateRoleDescription"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/batch.amazonaws.com/
AWSServiceRoleForBatch",
    "Condition": {"StringLike": {"iam:AWSServiceName": "batch.amazonaws.com"}}
}
```

AWS Batchのサービスリンクロールの削除

サービスリンクロールを必要とする機能やサービスが不要になった場合は、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。

IAM エンティティが AWSServiceRoleForBatch のサービスにリンクされたロールを作成するには

以下のステートメントをアクセス許可ポリシーに追加します。IAM エンティティが サービスにリンクされたロールを削除することを許可します。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:DeleteServiceLinkedRole",
        "iam:GetServiceLinkedRoleDeletionStatus"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/batch.amazonaws.com/
AWSServiceRoleForBatch",
    "Condition": {"StringLike": {"iam:AWSServiceName": "batch.amazonaws.com"}}
}
```

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前に、まずロールにアクティブなセッションがないことを確認し、1 つのパーティション内のすべての AWS リージョンでロールを使用するすべての AWS Batch コンピューティング環境を削除する必要があります。

サービスにリンクされたロールがアクティブなセッションを持っているかどうかを確認するには

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. ナビゲーションペインで、ロール を選択し、(チェックボックスではなく) AWSServiceRoleForBatch の名前を選択します。
3. Summary] (概要) ページで Access Advisor] (アクセスアドバイザー) を選択し、サービスにリンクされたロールの最新のアクティビティを確認します。

> **Note**
>
> AWS Batch が AWSServiceRoleForBatch ロールを使用しているかどうかわからない場合は、ロールを削除できます。サービスでロールが使用されている場合、ロールは削除されません。ロールが使用されているリージョンが表示されます。ロールが使用されている場合は、ロールを削除する前にセッションが終了するのを待つ必要があります。サービスにリンクされたロールのセッションを取り消すことはできません。

AWSServiceRoleForBatch サービスにリンクされたロールによって使用されている AWS Batch リソースを削除するには

AWSServiceRoleForBatch ロールを削除する前に、すべての AWS リージョンで AWSServiceRoleForBatch ロールを使用するすべての AWS Batch コンピューティング環境を削除する必要があります。

1. https://console.aws.amazon.com/batch/ で AWS Batch コンソールを開きます。
2. ナビゲーションバーから、使用するリージョンを選択します。
3. ナビゲーションペインで、Compute environments] (コンピューティング環境) を選択します。
4. コンピューティング環境を選択します。
5. Disable] (無効化) を選択します。State] (状態) が DISABLED] (無効) になるまで待ちます。
6. コンピューティング環境を選択します。

7. Delete] (削除) を選択します。Delete compute environment] (コンピューティング環境の削除) を選択し、削除したいコンピューティング環境を確認します。
8. すべてのリージョンでサービスにリンクされたロールを使用する、すべてのコンピューティング環境について、手順 1 ~ 7 を繰り返します。

IAM でのサービスにリンクされたロールの削除 (コンソール)

IAM コンソールを使用して、サービスにリンクされたロールを削除できます。

サービスにリンクされたロールを削除するには (コンソール)

1. にサインイン AWS マネジメントコンソール し、https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。
2. IAM コンソールのナビゲーションペインで Roles] (ロール) を選択します。AWSServiceRoleForBatch の横のチェックボックス (名前または行自体ではなく) を選択します。
3. Delete role] (ロールの削除) を選択します。
4. 確認ダイアログボックスで、サービスの最終アクセス時間データを確認します。これは、選択したそれぞれのロールの AWS のサービスへの最終アクセス時間を示します。これは、そのロールが現在アクティブであるかどうかを確認するのに役立ちます。先に進む場合は、[Yes, Delete] (はい、削除する) を選択し、削除するサービスにリンクされたロールを送信します。
5. IAM コンソール通知を見て、サービスにリンクされたロールの削除の進行状況をモニタリングします。IAM サービスにリンクされたロールの削除は非同期であるため、削除するロールを送信すると、削除タスクは成功または失敗する可能性があります。

   - タスクが成功した場合は、ロールがリストから削除され、成功の通知がページの上部に表示されます。
   - タスクが失敗した場合は、通知から View details] (詳細を表示) または View Resources] (リソースを表示) を選択して、削除が失敗した理由を知ることができます。ロールがサービスのリソースを使用しているために削除が失敗したとき、サービスがその情報を返す場合は、通知にはリソースのリストが含まれます。次にリソースをクリーンアップしてから、削除リクエストをもう一度送信できます。

     > **Note**
     >
     > サービスが返す情報に応じて、このプロセスを何度も繰り返す必要があります。例えば、サービスにリンクされたロールが 6 つのリソースを使用しており、サービスはそ

のうち 5 つのリソースに関する情報を返すことがあります。5 つのリソースをクリーンアップして削除するロールを再度送信すると、削除は失敗し、残りの 1 つのリソースが報告されます。サービスはすべてのリソースを返しますが、そのうちいくつかはリソースを報告しない場合もあります。

- タスクが失敗し、通知にリソースのリストが含まれていない場合、サービスはその情報を返さない可能性があります。サービスのリソースをクリーンアップする方法の詳細については、IAM と連携するAWS のサービスを参照してください。使用しているサービスをテーブルで見つけ、はいリンクを選択すると、そのサービスのサービスにリンクされたロールに関するドキュメントが表示されます。

IAM でのサービスにリンクされたロールの削除 (AWS CLI)

から IAM コマンドを使用して AWS Command Line Interface 、サービスにリンクされたロールを削除できます。

サービスにリンクされたロールを削除するには (CLI)

1. サービスにリンクされているロールは、使用されている、または関連するリソースがある場合は削除できないため、削除リクエストを送信する必要があります。これらの条件が満たされない場合、そのリクエストは拒否される可能性があります。レスポンスから deletion-task-id を取得して、削除タスクのステータスを確認する必要があります。サービスにリンクされたロールの削除リクエストを送信するには、次のコマンドを入力します：

```
$ aws iam delete-service-linked-role --role-name AWSServiceRoleForBatch
```

2. 削除タスクのステータスを確認するには、次のコマンドを入力します：

```
$ aws iam get-service-linked-role-deletion-status --deletion-task-id deletion-task-id
```

削除タスクのステータスは、NOT_STARTED、IN_PROGRESS、SUCCEEDED、または FAILED となります。削除が失敗した場合は、失敗した理由がコールによって返され、トラブルシューティングが可能になります。ロールがサービスのリソースを使用しているために削除が失敗したとき、サービスがその情報を返す場合は、通知にはリソースのリストが含まれます。次にリソースをクリーンアップしてから、削除リクエストをもう一度送信できます。

> **Note**
>
> サービスが返す情報に応じて、このプロセスを何度も繰り返す必要があります。例えば、サービスにリンクされたロールが 6 つのリソースを使用しており、サービスはそのうち 5 つのリソースに関する情報を返すことがあります。5 つのリソースをクリーンアップして削除するロールを再度送信すると、削除は失敗し、残りの 1 つのリソースが報告されます。サービスはすべてのリソースを返す場合もあれば、一部を返す場合もあります。または、リソースが報告されないこともあります。リソースを報告しないサービスのリソースをクリーンアップする方法については、IAM と連携するAWS サービスを参照してください。使用しているサービスをテーブルで見つけ、はいリンクを選択すると、そのサービスのサービスにリンクされたロールに関するドキュメントが表示されます。

サービスにリンクされた IAM (AWS API) でのロールの削除

IAM API を使用して、サービスにリンクされたロールを削除できます。

サービスにリンクされたロールを削除するには (API)

1. サービスにリンクされたロールの削除リクエストを送信するには、DeleteServiceLinkedRole を呼び出します。リクエストで、AWSServiceRoleForBatch のロール名を指定します。

   サービスにリンクされているロールは、使用されている、または関連するリソースがある場合は削除できないため、削除リクエストを送信する必要があります。これらの条件が満たされない場合、そのリクエストは拒否される可能性があります。レスポンスから `DeletionTaskId` を取得して、削除タスクのステータスを確認する必要があります。

2. 削除タスクのステータスを確認するには、GetServiceLinkedRoleDeletionStatus を呼び出します。リクエストで `DeletionTaskId` を指定します。

   削除タスクのステータスは、`NOT_STARTED`、`IN_PROGRESS`、`SUCCEEDED`、または `FAILED` となります。削除が失敗した場合は、失敗した理由がコールによって返され、トラブルシューティングが可能になります。ロールがサービスのリソースを使用しているために削除が失敗したとき、サービスがその情報を返す場合は、通知にはリソースのリストが含まれます。次にリソースをクリーンアップしてから、削除リクエストをもう一度送信できます。

> **Note**
>
> サービスが返す情報に応じて、このプロセスを何度も繰り返す必要があります。例えば、サービスにリンクされたロールが 6 つのリソースを使用しており、サービスはそのうち 5 つのリソースに関する情報を返すことがあります。5 つのリソースをクリーンアップして削除するロールを再度送信すると、削除は失敗し、残りの 1 つのリソースが報告されます。サービスはすべてのリソースを返しますが、そのうちいくつかはリソースを報告しない場合もあります。リソースを報告しないサービスのリソースをクリーンアップする方法については、IAM と連携するAWS のサービス サービスを参照してください。使用しているサービスをテーブルで見つけ、はいリンクを選択すると、そのサービスのサービスにリンクされたロールに関するドキュメントが表示されます。

AWS Batch のサービスにリンクされたロールをサポートするリージョン

AWS Batch は、サービスが利用可能な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をサポートしています。詳細については、AWS Batch エンドポイントを参照してください。

SageMaker AI AWS Batch での のロールの使用

AWS Batch は AWS Identity and Access Management (IAM) 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、直接リンクされた一意のタイプの IAM ロールです AWS Batch。サービスにリンクされたロールは によって事前定義 AWS Batch されており、サービスがユーザーに代わって他の AWS 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含まれています。

サービスにリンクされたロールを使用すると、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、 の設定 AWS Batch が簡単になります。 は、サービスにリンクされたロールのアクセス許可 AWS Batch を定義し、特に定義されている場合を除き、 のみがそのロールを引き受け AWS Batch 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、最初に関連リソースを削除する必要があります。これにより、 AWS Batch リソースへのアクセス許可が誤って削除されないため、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他のサービスの詳細については、AWS 「IAM と連携するサービス」を参照し、「サービスにリンクされたロール」列で「はい」を持つサービスを探しま

す。サービスリンク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、リンクで [はい] を選択します。

のサービスにリンクされたロールのアクセス許可 AWS Batch

AWS Batch は、AWSServiceRoleForAWSBatchWithSagemaker 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。 がユーザーに代わって SageMaker トレーニングジョブ AWS Batch をキューに入れ、管理できるようにします。

サービスリンクロール [AWSServiceRoleForAWSBatchWithSagemaker] は、次のサービスを信頼してそのロールを引き受けます。

- `sagemaker-queuing.batch.amazonaws.com`

ロールのアクセス許可ポリシーにより AWS Batch 、 は指定されたリソースに対して次のアクションを実行できます。

- `sagemaker` – AWS Batch が SageMaker トレーニングジョブ、変換ジョブ、およびその他の SageMaker AI リソースを管理できるようにします。
- `iam:PassRole` – ジョブ実行のために AWS Batch がカスタマー定義の実行ロールを SageMaker AI に渡すことを許可します。リソース制約により、SageMaker AI サービスにロールを渡すことができます。

ユーザー、グループ、またはロールにサービスリンクロールの作成、編集、または削除を許可するには、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「Service-linked role permissions」を参照してください。

AWS Batchのサービスリンクロールの作成

サービスリンクロールを手動で作成する必要はありません。 AWS マネジメントコンソール、、AWS CLIまたは AWS API `CreateServiceEnvironment`で を使用してサービス環境を作成すると、 AWS Batch 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。を使用してサービス環境を作成すると`CreateServiceEnvironment`、はサービスにリンクされたロールを再度 AWS Batch 作成します。

ポリシーの JSON を表示するには、「AWS マネージドポリシーリファレンスガイド」の「AWSBatchServiceRolePolicyForSageMaker」を参照してください。

AWS Batchのサービスにリンクされたロールの編集

AWS Batch では、AWSServiceRoleForAWSBatchWithSagemaker 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の作成後は、さまざまなエンティティがロールを参照す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 を使用してロールの説明を編集することはできます。詳細については、「IAM ユーザーガイド」の「サービスリンクロールの編集」を参照してください。

AWS Batchのサービスリンクロールの削除

サービスリンクロールを必要とする機能やサービスが不要になった場合は、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。

サービスリンク役割のクリーンアップ

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前に、まずロールにアクティブなセッションがないことを確認し、1 つのパーティション内のすべての AWS リージョンでロールを使用するすべてのサービス環境を削除する必要があります。

サービスにリンクされたロールがアクティブなセッションを持っているかどうかを確認するには

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. ナビゲーションペインで、[ロール] を選択し、(チェックボックスではなく) [AWSServiceRoleForAWSBatchWithSagemaker] の名前を選択します。
3. [Summary] (概要) ページで [Access Advisor] (アクセスアドバイザー) を選択し、サービスにリンクされたロールの最新のアクティビティを確認します。

> **Note**
>
> AWS Batch が AWSServiceRoleForAWSBatchWithSagemaker ロールを使用しているかどうかわからない場合は、ロールを削除できます。サービスでロールが使用されている場合、ロールは削除されません。ロールが使用されているリージョンが表示されます。ロールが使用されている場合は、ロールを削除する前にセッションが終了するのを待つ必要があります。サービスにリンクされたロールのセッションを取り消すことはできません。

AWSServiceRoleForAWSBatchWithSagemaker サービスにリンクされたロールによって使用されている AWS Batch リソースを削除するには

AWSServiceRoleForAWSBatchWithSagemaker ロールを削除する前に、すべての AWS リージョンで AWSServiceRoleForAWSBatchWithSagemaker ロールを使用するすべてのサービス環境を削除する必要があります。

1. https://console.aws.amazon.com/batch/ で AWS Batch コンソールを開きます。
2. ナビゲーションバーから、使用するリージョンを選択します。
3. ナビゲーションペインで、[環境]、[サービス環境] の順に選択します。
4. すべてのサービス環境を選択します。
5. [Disable] (無効化) を選択します。[State] (状態) が [DISABLED] (無効) になるまで待ちます。
6. サービス環境を選択します。
7. [削除] を選択します。[サービス環境の削除] を選択し、削除したいサービス環境を確認します。
8. すべてのリージョンでサービスにリンクされたロールを使用するすべてのサービス環境について、手順 1 ~ 7 を繰り返します。

IAM でのサービスにリンクされたロールの削除 (コンソール)

IAM コンソールを使用して、サービスにリンクされたロールを削除できます。

サービスにリンクされたロールを削除するには (コンソール)

1. にサインイン AWS マネジメントコンソール し、https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。
2. IAM コンソールのナビゲーションペインで [ロール] を選択します。[AWSServiceRoleForAWSBatchWithSagemaker] の横のチェックボックス (名前または行自体ではなく) を選択します。
3. [Delete role] (ロールの削除) を選択します。
4. 確認ダイアログボックスで、サービスの最終アクセス時間データを確認します。これは、選択したそれぞれのロールの AWS のサービスへの最終アクセス時間を示します。これは、そのロールが現在アクティブであるかどうかを確認するのに役立ちます。先に進む場合は、[Yes, Delete] (はい、削除する) を選択し、削除するサービスにリンクされたロールを送信します。
5. IAM コンソール通知を見て、サービスにリンクされたロールの削除の進行状況をモニタリングします。IAM サービスにリンクされたロールの削除は非同期であるため、削除するロールを送信すると、削除タスクは成功または失敗する可能性があります。

- タスクが成功した場合は、ロールがリストから削除され、成功の通知がページの上部に表示されます。
- タスクが失敗した場合は、通知から View details] (詳細を表示) または View Resources] (リソースを表示) を選択して、削除が失敗した理由を知ることができます。ロールがサービスのリソースを使用しているために削除が失敗したとき、サービスがその情報を返す場合は、通知にはリソースのリストが含まれます。次にリソースをクリーンアップしてから、削除リクエストをもう一度送信できます。

> **Note**
>
> サービスが返す情報に応じて、このプロセスを何度も繰り返す必要があります。例えば、サービスにリンクされたロールが 6 つのリソースを使用しており、サービスはそのうち 5 つのリソースに関する情報を返すことがあります。5 つのリソースをクリーンアップして削除するロールを再度送信すると、削除は失敗し、残りの 1 つのリソースが報告されます。サービスはすべてのリソースを返しますが、そのうちいくつかはリソースを報告しない場合もあります。

- タスクが失敗し、通知にリソースのリストが含まれていない場合、サービスはその情報を返さない可能性があります。サービスのリソースをクリーンアップする方法の詳細については、IAM と連携するAWS のサービスを参照してください。使用しているサービスをテーブルで見つけ、はいリンクを選択すると、そのサービスのサービスにリンクされたロールに関するドキュメントが表示されます。

IAM でのサービスにリンクされたロールの削除 (AWS CLI)

から IAM コマンドを使用して AWS Command Line Interface 、サービスにリンクされたロールを削除できます。

サービスにリンクされたロールを削除するには (CLI)

1. サービスにリンクされているロールは、使用されている、または関連するリソースがある場合は削除できないため、削除リクエストを送信する必要があります。これらの条件が満たされない場合、そのリクエストは拒否される可能性があります。レスポンスから `deletion-task-id` を取得して、削除タスクのステータスを確認する必要があります。サービスにリンクされたロールの削除リクエストを送信するには、次のコマンドを入力します：

```
$ aws iam delete-service-linked-role --role-name
 AWSServiceRoleForAWSBatchWithSagemaker
```

2. 削除タスクのステータスを確認するには、次のコマンドを入力します：

```
$ aws iam get-service-linked-role-deletion-status --deletion-task-id deletion-task-id
```

削除タスクのステータスは、NOT_STARTED、IN_PROGRESS、SUCCEEDED、または FAILED となります。削除が失敗した場合は、失敗した理由がコールによって返され、トラブルシューティングが可能になります。ロールがサービスのリソースを使用しているために削除が失敗したとき、サービスがその情報を返す場合は、通知にはリソースのリストが含まれます。次にリソースをクリーンアップしてから、削除リクエストをもう一度送信できます。

> **Note**
>
> サービスが返す情報に応じて、このプロセスを何度も繰り返す必要があります。例えば、サービスにリンクされたロールが 6 つのリソースを使用しており、サービスはそのうち 5 つのリソースに関する情報を返すことがあります。5 つのリソースをクリーンアップして削除するロールを再度送信すると、削除は失敗し、残りの 1 つのリソースが報告されます。サービスはすべてのリソースを返す場合もあれば、一部を返す場合もあります。または、リソースが報告されないこともあります。リソースを報告しないサービスのリソースをクリーンアップする方法については、IAM と連携するAWS サービスを参照してください。使用しているサービスをテーブルで見つけ、はいリンクを選択すると、そのサービスのサービスにリンクされたロールに関するドキュメントが表示されます。

サービスにリンクされた IAM (AWS API) でのロールの削除

IAM API を使用して、サービスにリンクされたロールを削除できます。

サービスにリンクされたロールを削除するには (API)

1. サービスにリンクされたロールの削除リクエストを送信するには、DeleteServiceLinkedRole を呼び出します。リクエストで、[AWSServiceRoleForAWSBatchWithSagemaker] のロール名を指定します。

サービスにリンクされているロールは、使用されている、または関連するリソースがある場合は削除できないため、削除リクエストを送信する必要があります。これらの条件が満たされない場合、そのリクエストは拒否される可能性があります。レスポンスから DeletionTaskId を取得して、削除タスクのステータスを確認する必要があります。

2. 削除タスクのステータスを確認するには、GetServiceLinkedRoleDeletionStatus を呼び出します。リクエストで DeletionTaskId を指定します。

   削除タスクのステータスは、NOT_STARTED、IN_PROGRESS、SUCCEEDED、または FAILED となります。削除が失敗した場合は、失敗した理由がコールによって返され、トラブルシューティングが可能になります。ロールがサービスのリソースを使用しているために削除が失敗したとき、サービスがその情報を返す場合は、通知にはリソースのリストが含まれます。次にリソースをクリーンアップしてから、削除リクエストをもう一度送信できます。

   > **Note**
   >
   > サービスが返す情報に応じて、このプロセスを何度も繰り返す必要があります。例えば、サービスにリンクされたロールが 6 つのリソースを使用しており、サービスはそのうち 5 つのリソースに関する情報を返すことがあります。5 つのリソースをクリーンアップして削除するロールを再度送信すると、削除は失敗し、残りの 1 つのリソースが報告されます。サービスはすべてのリソースを返しますが、そのうちいくつかはリソースを報告しない場合もあります。リソースを報告しないサービスのリソースをクリーンアップする方法については、IAM と連携するAWS のサービス サービスを参照してください。使用しているサービスをテーブルで見つけ、はいリンクを選択すると、そのサービスのサービスにリンクされたロールに関するドキュメントが表示されます。

AWS Batch のサービスにリンクされたロールをサポートするリージョン

AWS Batch は、サービスが利用可能な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をサポートしています。詳細については、AWS Batch エンドポイントを参照してください。

Amazon ECS インスタンスロール

AWS Batch コンピューティング環境には Amazon ECS コンテナインスタンスが入力されます。Amazon ECS コンテナエージェントがローカルで実行されます。Amazon ECS コンテナエージェントは、ユーザーに代わってさまざまな AWS API オペレーションを呼び出します。そのため、エージェントを実行するコンテナインスタンスには、エージェントがユーザーに属していることを

これらのサービスに伝える IAM ポリシーとロールが必要です。コンテナインスタンスの起動時に使用するIAMロールとインスタンスプロファイルを作成する必要があります。それ以外の場合、コンピューティング環境を作成してコンテナインスタンスを起動することはできません。この要件が適用されるコンテナインスタンスの起動には、Amazon が提供する、Amazon ECS に最適化された AMI が使用されている場合と使用されていない場合があります。詳細については、「Amazon Elastic Container Service デベロッパーガイド」の「Amazon ECS インスタンスロール」を参照してください。

トピック

- アカウントの Amazon ECS インスタンスロールを確認する

アカウントの Amazon ECS インスタンスロールを確認する

コンソールの初回実行時には、Amazon ECS インスタンスのロールおよびインスタンスプロファイルが自動的に作成されます。ただし、次の手順を使用して、アカウントに既に Amazon ECS インスタンスロールおよびインスタンスプロファイルが存在するか確認することができます。以下の手順では、マネージド IAM ポリシーをアタッチする方法についても説明します。

チュートリアル: IAM コンソールで **ecsInstanceRole** を確認する

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. 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 `ecsInstanceRole` を検索します。ロールが存在しない場合は、以下の手順でロールを作成してください。
    a. ロールの作成] を選択します。
    b. 信頼できるエンティティタイプ で、AWS のサービス を選択します。
    c. 一般的なユースケース で EC2 を選択します。
    d. 次へ をクリックします。
    e. アクセス権限ポリシー については、AmazonEC2ContainerServiceforEC2Role を検索してください。
    f. AmazonEC2ContainerServiceforEC2Role の横にあるチェックボックスを選択し、次へ を選択します。
    g. ロール名 には、`ecsInstanceRole` を入力し、そしてロールの作成] を選択します。

> **Note**
>
> を使用して Amazon EC2 のロール AWS マネジメントコンソール を作成する場合、コンソールはロールと同じ名前のインスタンスプロファイルを作成します。

または、 AWS CLI を使用して IAM `ecsInstanceRole` ロールを作成することもできます。次の例では、信頼ポリシーと AWS マネージドポリシーを使用して IAM ロールを作成します。

チュートリアル: IAM ロールおよびインスタンスプロファイルを作成する (AWS CLI)

1. 以下の信頼ポリシーを作成し、`ecsInstanceRole-role-trust-policy.json` という名前のテキストファイルに保存する。

   JSON

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

2. `ecsInstanceRole`のロールを作成するには、ロール作成コマンドを使用します。信頼ポリシーファイルの場所を `assume-role-policy-document` パラメータに指定します。

   ```
   $ aws iam create-role \
       --role-name ecsInstanceRole \
       --assume-role-policy-document file://ecsInstanceRole-role-trust-policy.json
   ```

3. [インスタンスプロファイルの作成] コマンドを使用して、`ecsInstanceRole` という名前のインスタンスプロファイルを作成します。

> **Note**
>
> ロールとインスタンスプロファイルは、 AWS CLI および AWS API で個別のアクションとして作成する必要があります。

```
$ aws iam create-instance-profile --instance-profile-name ecsInstanceRole
```

以下に、応答の例を示します。

```
{
    "InstanceProfile": {
        "Path": "/",
        "InstanceProfileName": "ecsInstanceRole",
        "InstanceProfileId": "AIPAT46P5RDITREXAMPLE",
        "Arn": "arn:aws:iam::123456789012:instance-profile/ecsInstanceRole",
        "CreateDate": "2022-06-30T23:53:34.093Z",
        "Roles": [],    }
}
```

4. [add-role-to-instance-profile] コマンドを使用して、ecsInstanceRole のロールを ecsInstanceRole インスタンスプロファイルに追加します。

```
aws iam add-role-to-instance-profile \
    --role-name ecsInstanceRole --instance-profile-name ecsInstanceRole
```

5. attachattach-role-policy コマンドを使用して、AmazonEC2ContainerServiceforEC2Role AWS 管理ポリシーをecsInstanceRoleロールにアタッチします。

```
$ aws iam attach-role-policy \
    --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role \
    --role-name ecsInstanceRole
```

Amazon EC2 スポットフリートロール

Amazon EC2 スポットフリートインスタンスを使用するマネージド型のコンピューティング環境を作成する場合は、AmazonEC2SpotFleetTaggingRole ポリシーを作成しなければなり

ません。このポリシーは、ユーザーに代わりインスタンスの起動、タグ付けおよび終了を行うためのスポットフリート許可を付与します。スポットフリートのリクエストでロールを指定します。また、Amazon EC2 スポットおよびスポットフリートにサービスにリンクされたロール AWSServiceRoleForEC2Spot] および AWSServiceRoleForEC2SpotFleet] がある必要があります。次の手順に従って、これらすべてのロールを作成します。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使用」およびAWS 「 サービスにアクセス許可を委任するロールの作成」を参照してください。

トピック

- AWS マネジメントコンソールにおいて、Amazon EC2 スポットフリートロールを作成する
- を使用して Amazon EC2 スポットフリートロールを作成する AWS CLI

AWS マネジメントコンソールにおいて、Amazon EC2 スポットフリートロールを作成する

Amazon EC2 スポットフリートの **AmazonEC2SpotFleetTaggingRole** IAM サービスにリンクされた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. アクセス管理] のために、ロール を選択します。
3. ロール] には、ロールの作成 を選択します。
4. [信頼されるエンティティを選択] (信頼されるエンティティタイプにある) から、[AWS のサービス] を選択します。
5. その他のユースケースでは AWS のサービス、EC2 を選択し、EC2 - スポットフリートのタグ付けを選択します。
6. [次へ] を選択します。
7. [ポリシー名] の [権限ポリシー] から、`AmazonEC2SpotFleetTaggingRole` を確認します 。
8. 次へ を選択します。
9. [名前、確認および作成]:
    a. ロールを識別するために、名前タグ に名前を入力します。
    b. [説明] には、ポリシーの簡単な説明を入力します。
    c. (オプション) [ステップ 1: 信頼できるエンティティの選択] では、[編集] を選択して、コードを変更します。
    d. (オプション) [ステップ 2: 権限の追加] では、[編集] を選択して、コードを変更します。

e. (オプション) [タグを追加] で [タグを追加] を選択し、リソースにタグを追加します。

f. ロールの作成] を選択します。

> **Note**
>
> これまでは、Amazon EC2 スポットフリートロールに対し 2つの管理ポリシーがありました。
>
> - AmazonEC2SpotFleetRole: これは、スポットフリートロール用のオリジナルの管理ポリシーです。ただし、 での使用は推奨されなくなりました AWS Batch。このポリシーは、`AWSServiceRoleForBatch` のサービスにリンクされたロールを使用するために必要なコンピューティング環境でのスポットインスタンスのタグ付けをサポートしていません。以前に、このポリシーを使用してスポットフリートロールを作成した場合は、新しい推奨ポリシーをそのロールに適用してください。詳細については、作成時にタグが付けられていないスポットインスタンスを参照してください。
> - AmazonEC2SpotFleetTaggingRole: このロールでは、Amazon EC2 スポットインスタンスにタグを付けるために必要なすべてのアクセス権限が提供されます。このロールを使用して、 AWS Batch コンピューティング環境でスポットインスタンスのタグ付けを許可します。

を使用して Amazon EC2 スポットフリートロールを作成する AWS CLI

スポットフリートのコンピューティング環境の AmazonEC2SpotFleetTaggingRole IAM ロールを作成するには

1. AWS CLIを使用して次のコマンドを実行します。

```
$ aws iam create-role --role-name AmazonEC2SpotFleetTaggingRole \
    --assume-role-policy-document '{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "spotfleet.amazonaws.com"
      },
```

```
      "Action": "sts:AssumeRole"
    }
  ]
}'
```

2. AmazonEC2SpotFleetTaggingRole マネージド IAM ポリシーを AmazonEC2SpotFleetTaggingRole ロールにアタッチするには、以下のコマンドを AWS CLIで実行します。

```
$ aws iam attach-role-policy \
  --policy-arn \
    arn:aws:iam::aws:policy/service-role/AmazonEC2SpotFleetTaggingRole \
  --role-name \
    AmazonEC2SpotFleetTaggingRole
```

Amazon EC2 スポットの **AWSServiceRoleForEC2Spot** IAMのサービスにリンクされたロールを作成するには

> **Note**
>
> AWSServiceRoleForEC2Spot IAM サービスにリンクされたロールがすでに存在する場合は、次のようなエラーメッセージが表示されます。
>
> ```
> An error occurred (InvalidInput) when calling the CreateServiceLinkedRole
>  operation:
> Service role name AWSServiceRoleForEC2Spot has been taken in this account,
>  please try a different suffix.
> ```

- AWS CLIを使用して次のコマンドを実行します。

```
$ aws iam create-service-linked-role --aws-service-name spot.amazonaws.com
```

Amazon EC2 スポットフリートの **AWSServiceRoleForEC2SpotFleet** IAM サービスにリンクされたロールを作成するには

> **Note**
>
> AWSServiceRoleForEC2SpotFleet IAM サービスにリンクされたロールがすでに存在する場合は、次のようなエラーメッセージが表示されます。
>
> ```
> An error occurred (InvalidInput) when calling the CreateServiceLinkedRole
>  operation:
> Service role name AWSServiceRoleForEC2SpotFleet has been taken in this account,
>  please try a different suffix.
> ```

- AWS CLIを使用して次のコマンドを実行します。

```
$ aws iam create-service-linked-role --aws-service-name spotfleet.amazonaws.com
```

EventBridge IAM ロール

Amazon EventBridge は、 AWS リソースの変更を記述するシステムイベントのほぼリアルタイムのストリームを提供します。 AWS Batch ジョブは EventBridge ターゲットとして使用できます。すぐに設定できる簡単なルールを使用して、それらのルールに対応して、イベントを一致させ AWS Batch ジョブを送信できます。EventBridge ルールとターゲットを使用して AWS Batch ジョブを送信する前に、EventBridge にはユーザーに代わって AWS Batch ジョブを実行するアクセス許可が必要です。

> **Note**
>
> AWS Batch キューをターゲットとして指定するルールを EventBridge コンソールで作成すると、このロールを作成できます。チュートリアル例については、EventBridge ターゲットとしての AWS Batch ジョブを参照してください。IAM コンソールを使って、ユーザーは EventBridge ロールをマニュアルで作成できます。詳細については、IAM ユーザーガイドのIAM ユーザーガイドの作成 (コンソール) を参照してください。

EventBridge IAMsのIAM ロールの信頼関係では、events.amazonaws.com サービスプリンシパルでロールを継承することを許可する必要があります。

JSON

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

EventBridge IAM ロールにアタッチされているポリシーが、リソースへの batch:SubmitJob 許可を与えていることを確認してください。次の例では、 AWS Batch が、AWSBatchServiceEventTargetRole の管理ポリシーを提供してこれらのアクセス権限を付与しています。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "batch:SubmitJob"
       ],
      "Resource": "*"
    }
  ]
}
```

仮想プライベートクラウドを作成する

コンピューティング環境のコンピューティングリソースは、 AWS Batch および Amazon ECS サービスエンドポイントとの通信に、外部ネットワークアクセスを必要とします。ただし、ユーザーが、プライベートサブネットで実行したいジョブがある場合があります。パブリックサブネットまたはプライベートサブネットのどちらかでジョブを実行する柔軟性を得るには、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がある VPC を作成します。

Amazon Virtual Private Cloud (Amazon VPC) を使用して、定義した仮想ネットワークに AWS リソースを起動できます。このトピックでは、Amazon VPC ウィザードへのリンクと選択できるオプションのリストを提供します。

「VPC を作成する」

Amazon VPC の作成方法の詳細については、「Amazon VPC ユーザーガイド」の「VPC のみを作成する」を参照し、次の表を使用して選択するオプションを決定します。

オプション	値
作成するためのリソース	VPC 専用
名前	オプションで、VPC の名前を指定します。
IPv4 CIDR ブロック	IPv4 CIDR 手動入力 CIDR ブロックサイズは /16 から /28 の間である必要があります。
IPv6 CIDR ブロック	IPv6 CIDR ブロックなし
テナンシー	デフォルト

Amazon VPC の詳細については、Amazon VPC ユーザーガイドのAmazon VPC とはを参照してください。

次の手順

VPC を作成したら、以下の次のステップを検討します：

- パブリックリソースおよびプライベートリソースでインバウンドネットワークアクセスが必要な場合は、そのセキュリティグループを作成します。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の操作を参照してください。
- 新しい VPC にコンピューティングリソースを起動する AWS Batch マネージドコンピューティング環境を作成します。詳細については、「コンピューティング環境を作成する」を参照してください。 AWS Batch コンソールでコンピューティング環境作成ウィザードを使用する場合は、先ほど作成した VPC と、インスタンスを起動するパブリックサブネットまたはプライベートサブネットを指定できます。
- 新しいコンピューティング環境にマッピングされた AWS Batch ジョブキューを作成します。詳細については、「ジョブキューを作成する」を参照してください。
- ジョブの実行で使用するジョブ定義を作成します。詳細については、シングルノードのジョブ定義を作成するを参照してください。
- ジョブ定義とともに、新しいジョブキューにジョブを送信します。このジョブは、新しい VPC およびサブネットで作成したコンピューティング環境に置かれます。詳細については、「チュートリアル: ジョブを送信する」を参照してください。

インターフェイスエンドポイントを使用して にアクセスする AWS Batch

を使用して AWS PrivateLink 、VPC と の間にプライベート接続を作成できます AWS Batch。インターネットゲートウェイ、NAT デバイス、VPN 接続、 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 AWS Batch にアクセスできます。VPC 内のインスタンスは AWS Batchにアクセスするためにパブリック IP アドレスを必要としません。

このプライベート接続を確立するには、 AWS PrivateLinkを利用したインターフェイスエンドポイントを作成します。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、 AWS Batch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「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。

に関する考慮事項 AWS Batch

のインターフェイスエンドポイントを設定する前に AWS Batch、「 AWS PrivateLink ガイド」の「インターフェイスエンドポイントのプロパティと制限」を参照してください。

AWS Batch は、インターフェイスエンドポイントを介したすべての API アクションの呼び出しをサポートしています。

のインターフェイス VPC エンドポイントを設定する前に AWS Batch、次の考慮事項に注意してください。

- Fargate リソース起動タイプを使用するジョブには、Amazon ECS のインターフェイス VPC エンドポイントは必要ありませんが、以下のポイントで説明する Amazon ECR AWS Batch、Secrets Manager、または Amazon CloudWatch Logs のインターフェイス VPC エンドポイントが必要になる場合があります。
  - ジョブを実行するには、Amazon ECS 用に、インターフェイス VPC エンドポイントを作成する必要があります。詳細については、「Amazon Elastic Container Service デベロッパーガイド」の「インターフェイス VPC エンドポイント (AWS PrivateLink)」を参照してください。
  - ジョブがAmazon ECRからプライベートイメージを引き出せるようにするには、Amazon ECR用のインターフェースVPCエンドポイントを作成する必要があります。詳細については、Amazon Elastic Container Registry ユーザーガイドのインターフェイス VPC エンドポイント (AWS PrivateLink)を参照してください。
  - ジョブが Secrets Manager から機密データを取得できるようにするには、Secrets Manager 用のインターフェイス VPC エンドポイントを作成する必要があります。詳細については、AWS Secrets Manager ユーザーガイドの VPC EndpointでSecrets Managerを使用する を参照してください。
  - VPCにインターネットゲートウェイがなく、ジョブが `awslogs` ログドライバーを使用してログ情報をCloudWatch Logsに送信する場合、CloudWatch Logs用のインターフェースVPCエンドポイントを作成する必要があります。詳細については、Amazon CloudWatch Logs ユーザーガイドのインターフェイス VPC エンドポイントでの CloudWatch Logs の使用を参照してください。
- EC2リソースを使用するジョブでは、起動するコンテナインスタンスがAmazon ECSコンテナエージェントのバージョン `1.25.1` 以降を実行している必要必要があります。詳細については、「Amazon Elastic Container Service デベロッパーガイド」の「Amazon ECS Linux コンテナエージェントバージョン」を参照してください。

- 現在、VPC エンドポイントはクロスリージョンリクエストをサポートしていません。 AWS Batch に対して API コールを発行するリージョンと同じリージョンにエンドポイントを作成してください。
- VPC エンドポイントでは、Amazon Route 53 を介して Amazon 提供の DNS のみがサポートされています。独自の DNS を使用したい場合は、条件付き DNS 転送を使用できます。詳細については、Amazon VPC ユーザーガイド の DHCP Options Sets を参照してください。
- VPC エンドポイントにアタッチされたセキュリティグループでは、VPC のプライベートサブネットから、ポート 443 で着信接続を許可する必要があります。
- AWS Batch は、以下の VPC インターフェイスエンドポイントをサポートしていません AWS リージョン。
  - アジアパシフィック (大阪) (ap-northeast-3)
  - アジアパシフィック (ジャカルタ) (ap-southeast-3)

のインターフェイスエンドポイントを作成する AWS Batch

Amazon VPC コンソールまたは AWS Command Line Interface () AWS Batch を使用して、 のインターフェイスエンドポイントを作成できますAWS CLI。詳細については、「AWS PrivateLink ガイド」の「インターフェイスエンドポイントを作成」を参照してください。

次のサービス名 AWS Batch を使用して、 用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* .batch
- com.amazonaws.*region* .batch-fips (FIPS 準拠のエンドポイントについては、AWS Batch 「エンドポイントとクォータ」を参照してください)

例えば、次のようになります。

```
com.amazonaws.us-east-2.batch
```

```
com.amazonaws.us-east-2.batch-fips
```

aws-cn パーティションでは形式が異なります。

```
cn.com.amazonaws.region.batch
```

例えば、次のようになります。

```
cn.com.amazonaws.cn-northwest-1.batch
```

AWS Batch インターフェイスエンドポイントのプライベート DNS 名

インターフェイスエンドポイントのプライベート DNS を有効にすると、特定の DNS 名を使用して接続できます。以下のオプション AWS Batchが用意されています。

- batch.*region* .amazonaws.com
- batch.*region* .api.aws

FIPS 準拠のエンドポイントの場合:

- batch-fips.*region* .api.aws
- fips.batch.*region* .amazonaws.com はサポートされていません

詳細については、AWS PrivateLink ガイドのインターフェイスエンドポイントを介したサービスへのアクセスを参照してください。

インターフェイスエンドポイントのエンドポイントポリシーを作成する

エンドポイントポリシーは、インターフェイスエンドポイントにアタッチできる IAM リソースです。デフォルトのエンドポイントポリシーでは、インターフェイスエンドポイント AWS Batch を介した へのフルアクセスが許可されます。VPC から AWS Batch に許可されるアクセスを制御するには、カスタム・エンドポイント・ポリシーをインターフェースのエンドポイントにアタッチします。

エンドポイントポリシーは以下の情報を指定します。

- アクションを実行できるプリンシパル (AWS アカウント、ユーザー、IAM ロール)。
- 実行可能なアクション。
- このアクションを実行できるリソース。

詳細については、AWS PrivateLink ガイドのControl access to services using endpoint policies (エンドポイントポリシーを使用してサービスへのアクセスをコントロールする)を参照してください。

例: AWS Batch アクションの VPC エンドポイントポリシー

以下は、カスタムエンドポイントポリシーの例です。このポリシーをインターフェイスエンドポイントにアタッチすると、すべてのリソースのすべてのプリンシパルに対して、リストされた AWS Batch アクションへのアクセスが許可されます。

```
{
   "Statement": [
      {
         "Principal": "*",
         "Effect": "Allow",
         "Action": [
            "batch:SubmitJob",
            "batch:ListJobs",
            "batch:DescribeJobs"
         ],
         "Resource":"*"
      }
   ]
}
```

のコンプライアンス検証 AWS Batch

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AWS のサービス プログラムによる対象範囲内」の「コンプライアンス」を参照して、関心のあるコンプライアンスプログラムを選択します。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。

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 AWS Artifact。詳細については、「Downloading Reports in AWS Artifact」を参照してください。

を使用する際のお客様のコンプライアンス責任 AWS のサービス は、お客様のデータの機密性、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。を使用する際のコンプライアンス責任の詳細については AWS のサービス、AWS 「 セキュリティドキュメント」を参照してください。

のインフラストラクチャセキュリティ AWS Batch

マネージドサービスである AWS Batch は、 AWS グローバルネットワークセキュリティで保護されています。 AWS セキュリティサービスと がインフラストラクチャ AWS 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環境を AWS 設計するには、「Security Pillar AWS Well-Architected Framework」の「Infrastructure Protection」を参照してください。

AWS が発行した API コールを使用して、ネットワーク AWS Batch 経由で 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 が必須で、TLS 1.3 をお勧めします。
- DHE (楕円ディフィー・ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。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。

これらの API オペレーションは任意のネットワークロケーションから呼び出すことができますが、AWS Batch はリソースベースのアクセスポリシーをサポートしており、ソース IP アドレスに基づく制限を含めることができます。 AWS Batch ポリシーを使用して、特定の Amazon Virtual Private Cloud (Amazon VPC) エンドポイントまたは特定の VPCs からのアクセスを制御することもできます。これにより、実質的に、ネットワーク内の特定の VPC からのみ、特定の AWS Batch リソースへの AWS ネットワークアクセスが分離されます。

サービス間の混乱した代理の防止

「混乱した代理」問題は、アクションを実行するためのアクセス許可を持たないエンティティが、より特権のあるエンティティにアクションの実行を強制できてしまう場合に生じる、セキュリティ上の問題です。では AWS、サービス間のなりすましにより、混乱した代理問題が発生する可能性があります。サービス間でのなりすましは、1 つのサービス (呼び出し元サービス) が、別のサービス (呼び出し対象サービス) を呼び出すときに発生する可能性があります。呼び出し元サービスは、本来ならアクセスすることが許可されるべきではない方法でその許可を使用して、別のお客様のリソースに対する処理を実行するように操作される場合があります。これを防ぐため、 AWS では、アカウントのリソースへのアクセス権が付与されたサービスプリンシパルで、すべてのサービスのデータを保護するために役立つツールを提供しています。

リソースポリシーで `aws:SourceArn`および `aws:SourceAccount` グローバル条件コンテキストキーを使用して、 がリソースに別のサービス AWS Batch に付与するアクセス許可を制限することをお勧めします。`aws:SourceArn` の値に Amazon S3 バケット ARN などのアカウント ID が含まれていない場合は、両方のグローバル条件コンテキストキーを使用して、アクセス許可を制限する必要があります。同じポリシーステートメントでこれらのグローバル条件コンテキストキーの両方を使用し、アカウント ID に`aws:SourceArn` の値が含まれていない場合、`aws:SourceAccount` 値と `aws:SourceArn` 値の中のアカウントには、同じアカウント ID を使用する必要があります。

クロスサービスのアクセスにリソースを 1 つだけ関連付けたい場合は、aws:SourceArn を使用します。そのアカウント内のリソースをクロスサービスの使用に関連付けることを許可する場合は、aws:SourceAccount を使用します。

の値は、 が AWS Batch 保存するリソースaws:SourceArnである必要があります。

混乱した代理問題から保護するための最も効果的な方法は、リソースの完全な ARN を指定して、aws:SourceArn グローバル条件コンテキストキーを使用することです。リソースの完全な ARN が不明な場合や、複数のリソースを指定する場合には、グローバルコンテキスト条件キー aws:SourceArn で、ARN の未知部分を示すためにワイルドカード文字 (*) を使用します。例えば、arn:aws:*servicename*:*:*123456789012*:*。

次の例は、 で aws:SourceArnおよび aws:SourceAccount グローバル条件コンテキストキーを使用して、混乱した代理問題 AWS Batch を防ぐ方法を示しています。

例: 1 つのコンピューティング環境にのみアクセスするためのロール

次のロールは、1 つのコンピュート環境へのアクセスにのみ使用できます。ジョブキューは複数のコンピューティング環境に関連付けることができるため、ジョブ名は * として指定する必要があります。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "batch.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnLike": {
          "aws:SourceArn": [
            "arn:aws:batch:us-east-1:123456789012:compute-environment/testCE",
            "arn:aws:batch:us-east-1:123456789012:job/*"
```

```
          ]
        }
      }
    }
  ]
}
```

例: 複数のコンピューティング環境にアクセスするためのロール

次のロールを使用して複数のコンピュート環境にアクセスできます。* ジョブキューは複数のコンピューティング環境に関連付けることができるため、ジョブ名はとして指定する必要があります。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "batch.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnLike": {
          "aws:SourceArn": [
            "arn:aws:batch:us-east-1:123456789012:compute-environment/*",
            "arn:aws:batch:us-east-1:123456789012:job/*"
          ]
        }
      }
    }
  ]
}
```

を使用した AWS Batch API コールのログ記録 AWS CloudTrail

AWS Batch は、ユーザー AWS CloudTrail、ロール、または のサービスによって実行されたアクションを記録する AWS サービスである と統合されています AWS Batch。CloudTrail は、 のすべての API コールをイベント AWS Batch 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、AWS Batch コンソールからの呼び出しと AWS Batch 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、イベントを含む 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 AWS Batch。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 が収集した情報を使用して、 AWS Batchに対して行われた要求、要求が行われた IP アドレス、要求を行った人、要求が行われた日時、および追加の詳細を判別できます。

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

- AWS Batch CloudTrail の情報
- リファレンス: AWS Batch ログファイルエントリについて

AWS Batch CloudTrail の情報

CloudTrail は AWS 、アカウントの作成時にアカウントで有効になります。アクティビティが発生すると AWS Batch、そのアクティビティはイベント履歴の他の AWS サービスイベントとともに 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。

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 AWS Batch、証跡を作成します。証跡により、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、証跡はすべての AWS リージョンに適用されます。証跡は、 AWS パーティション内のすべてのリージョンからのイベントを記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、CloudTrail ログで収集されたイベントデータをさらに分析して処理するように他の AWS サービスを設定できます。詳細については、次を参照してください:

- 証跡の作成のための概要
- CloudTrail がサポートするサービスと統合
- CloudTrail 用 Amazon SNS 通知の構成

- 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 AWS Batch アクションは CloudTrail によってログに記録され、https://docs.aws.amazon.com/batch/latest/APIReference/ に記載されています。例えば、SubmitJob、ListJobs、および DescribeJobs 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。

各イベントまたはログエントリには、リクエストの生成者に関する情報が含まれます。同一性情報は次の判断に役立ちます。

- リクエストが、ルートと IAM ユーザー認証情報のどちらを使用して送信されたか。
- リクエストがロールまたはフェデレーションユーザーの一時的なセキュリティ認証情報を使用して行われたかどうか。
- リクエストが別の AWS サービスによって行われたかどうか。

詳細については、「CloudTrail userIdentity 要素」を参照してください。

リファレンス: AWS Batch ログファイルエントリについて

「トレイル」は、指定した 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設定できます。CloudTrail のログファイルは、単一か複数のログエントリを含みます。イベントは、任意の出典からの単一のリクエストを表し、リクエストされたアクション、アクションの日時、リクエストパラメータなどに関する情報が含まれます。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。

以下の例は、CreateComputeEnvironment アクションを示す CloudTrail ログエントリです。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE:admin",
    "arn": "arn:aws:sts::012345678910:assumed-role/Admin/admin",
    "accountId": "012345678910",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
```

```
        "creationDate": "2017-12-20T00:48:46Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::012345678910:role/Admin",
        "accountId": "012345678910",
        "userName": "Admin"
      }
    }
  },
  "eventTime": "2017-12-20T00:48:46Z",
  "eventSource": "batch.amazonaws.com",
  "eventName": "CreateComputeEnvironment",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "203.0.113.1",
  "userAgent": "aws-cli/1.11.167 Python/2.7.10 Darwin/16.7.0 botocore/1.7.25",
  "requestParameters": {
    "computeResources": {
      "subnets": [
        "subnet-5eda8e04"
      ],
      "tags": {
        "testBatchTags": "CLI testing CE"
      },
      "desiredvCpus": 0,
      "minvCpus": 0,
      "instanceTypes": [
        "optimal"
      ],
      "securityGroupIds": [
        "sg-aba9e8db"
      ],
      "instanceRole": "ecsInstanceRole",
      "maxvCpus": 128,
      "type": "EC2"
    },
    "state": "ENABLED",
    "type": "MANAGED",
    "computeEnvironmentName": "Test"
  },
  "responseElements": {
    "computeEnvironmentName": "Test",
```

```
    "computeEnvironmentArn": "arn:aws:batch:us-east-1:012345678910:compute-environment/
Test"
  },
  "requestID": "890b8639-e51f-11e7-b038-EXAMPLE",
  "eventID": "874f89fa-70fc-4798-bc00-EXAMPLE",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "recipientAccountId": "012345678910"
}
```

IAM AWS Batch のトラブルシューティング

以下の情報は、 および IAM の使用時に発生する可能性がある一般的な問題の診断 AWS Batch と修正に役立ちます。

トピック

- AWS Batchでアクションを実行する権限がない
- iam:PassRole を実行する権限がありません
- 自分の AWS アカウント以外のユーザーに自分の AWS Batch リソースへのアクセスを許可したい

AWS Batchでアクションを実行する権限がない

でアクションを実行する権限がないと AWS マネジメントコンソール 通知された場合は、管理者に連絡してサポートを依頼する必要があります。担当の管理者はお客様のユーザー名とパスワードを発行した人です。

以下のエラー例は、mateojackson ユーザーがコンソールを使用して架空の *my-example-widget* リソースに関する詳細情報を表示しようとしているが、架空の batch:*GetWidget* 許可がないという場合に発生します。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 batch:GetWidget on resource: my-example-widget
```

この場合、Mateo は、batch:*GetWidget* アクションを使用して *my-example-widget* リソースへのアクセスが許可されるように、管理者にポリシーの更新を依頼します。ロールを渡すアクセス許可の付与の詳細については、「AWS 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。

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 AWS Batchにロールを渡すことができ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にリンクされたロールを作成する代わりに、既存のロールをそのサービスに渡すことができます。そのためには、サービスにロールを渡す権限が必要です。

以下の例のエラーは、marymajor という IAM ユーザーがコンソールを使用して AWS Batch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アクセス許可がありません。

```
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 管理者にお問い合わせください。サインイン資格情報を提供した担当者が管理者です。

自分の AWS アカウント以外のユーザーに自分の AWS Batch 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コントロールリスト (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:

- がこれらの機能 AWS Batch をサポートしているかどうかを確認するには、「」を参照してくださいが IAM と AWS Batch 連携する方法。
- 所有 AWS アカウント している のリソースへの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「所有 AWS アカウント している別の の IAM ユーザーへのアクセスを提供する」を参照してください。

- リソースへのアクセスをサードパーティーに提供する方法については AWS アカウント、IAM ユーザーガイドの「サードパーティー AWS アカウント が所有する へのアクセスを提供する」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) へのアクセスの許可 を参照してください。
- 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使用方法の違いについては、「IAM ユーザーガイド」の「IAM でのクロスアカウントのリソースへのアクセス」を参照してください。

AWS Step Functions

AWS Batch コンソールを使用して、 ステートマシンおよびこれらのス Step Functions テートマシンが使用する 関数の詳細を表示できます。

セクション

- チュートリアル: ステートマシンの詳細を確認する
- チュートリアル: ステートマシンを編集する
- チュートリアル: ステートマシンを実行する

チュートリアル: ステートマシンの詳細を確認する

AWS Batch コンソールには、AWS Batch ジョブを送信するワークフローステップが少なくとも 1 つ含まれているステートマシンが、現在の AWS リージョン から一覧表示されます。

ステートマシンを選択して、ワークフローをグラフィカル表示します。青でハイライトされたステップは、AWS Batch ジョブを表します。グラフコントロールを使用して、グラフをズームイン、ズームアウト、中央揃えにします。

> **Note**
>
> ステートマシンの定義で AWS Batch ジョブが JsonPath で動的に参照されている場合、関数の詳細を AWS Batch コンソールに表示することはできません。代わりに、ジョブ名が動的参照として表示され、グラフ内の対応するステップが灰色表示になります。

ステートマシンの詳細を表示するには

1. AWS Batch コンソールの[Workflow orchestration powered by Step Functions ページ] (Step Functions により強化されたワークフローオーケストレーション) を開きます。
2. ステートマシンを選択します。
   <result>

   AWS Batch コンソールに [詳細] ページが表示されます。
   </result>

詳細については、 AWS Step Functions デベロッパーガイドの Step Functions を参照してください。

チュートリアル: ステートマシンを編集する

ステートマシンを編集する場合、AWS Batch を使用して Step Functions コンソールの [Edit definition] (定義の編集) ページを開きます。

ステートマシンを編集するには

1. AWS Batch コンソールの[Workflow orchestration powered by Step Functions ページ] (Step Functions により強化されたワークフローオーケストレーション) を開きます。
2. ステートマシンを選択します。
3. [Edit] (編集) を選択します。

   Step Functions コンソールで [Edit definition] (定義の編集) ページが開きます。
4. ステートマシンを編集し、[Save] (保存) を選択します。

ステートマシンの編集の詳細については、AWS Step Functions デベロッパーガイドの 「Step Functions state machine language」を参照してください。

チュートリアル: ステートマシンを実行する

ステートマシンを実行する場合は、AWS Batch を使用して Step Functions コンソールの [New execution] (新しい実行) ページを開きます。

ステートマシンを実行するには

1. AWS Batch コンソールの[Workflow orchestration powered by Step Functions ページ] (Step Functions により強化されたワークフローオーケストレーション) を開きます。
2. ステートマシンを選択します。
3. [Execute] (実行) を選択します。

   Step Functions コンソールで [New execution] (新しい実行) ページが開きます。
4. (オプション) ステートマシンを編集し、[Start execution] (実行を開始) を選択します。

ステートマシンの実行の詳細については、AWS Step Functions デベロッパーガイドの 「ステップ関数ステートマシンの実行概念」を参照してください。

Amazon EventBridge の AWS Batch イベントストリーム

Amazon EventBridge の AWS Batch イベントストリームを使用して、ジョブキューにおけるジョブの現在の状態に関する通知を、ほぼリアルタイムで受け取ることができます。

EventBridgeを使用することで、AWS Batch サービスに関する更なるインサイトを得ることができます。具体的には、ジョブの進捗状況の確認、AWS Batchカスタムワークフローの構築、使用状況レポートやメトリクスの生成、独自のダッシュボードの作成などに使用できる。AWS BatchとEventBridgeがあれば、ジョブステータスの変更についてAWS Batchを継続的にポーリングするスケジューリングやモニタリングのコードは必要ない。その代わり、様々な Amazon EventBridge のターゲットを使って、AWS Batch ジョブの状態変化を非同期で処理することができます。これには、AWS Lambda、Amazon Simple Queue Service、Amazon Simple Notification Service、Amazon Kinesis Data Streams が含まれます。

AWS Batch イベントストリームのイベントは、少なくとも 1 回必ず送信されます。重複したイベントが送信された場合でも、イベントには重複を識別できるだけの十分な情報が備わっています。これにより、イベントのタイムスタンプとジョブの状態を比較できます。

AWS Batch ジョブは CloudWatch Events ターゲットとして利用可能です。簡単なルールを使用して、ルールに一致したイベントに応じて AWS Batch ジョブを送信できます。詳細については、Amazon EventBridge ユーザーガイドのEventBridge とはを参照してください。また、EventBridgeを使用して、cron またはレート式を使用して、特定の時間に自己トリガーする自動化されたアクションをスケジュールすることもできます。詳細については、Amazon EventBridge ユーザーガイドのスケジュールに従って実行する Amazon EventBridge ルールの作成を参照してください。チュートリアル例については、EventBridge ターゲットとしての AWS Batch ジョブを参照してください。詳細については、Amazon EventBridge スケジューラユーザーガイドの Amazon EventBridge スケジューラとは を参照してください。

トピック

- AWS Batch のイベント
- チュートリアル: AWS Batch で AWS ユーザー通知を使用する
- EventBridge ターゲットとしての AWS Batch ジョブ
- チュートリアル: EventBridge を使用して AWS Batch ジョブイベントをリッスンする
- チュートリアル: 失敗したジョブイベントに Amazon シンプル通知サービスアラートを送信する

AWS Batch のイベント

AWS Batch は、EventBridgeにジョブ・ステータス変更イベントを送信します。AWS Batchはジョブの状態を追跡します。以前に提出されたジョブの状態が変わった場合、イベントが起動します。例えば、RUNNING ステータスにあるジョブが FAILED ステータスに移動した場合などです。これらのイベントはジョブ状態の変更イベントとして分類されます。

> **Note**
>
> AWS Batch将来、 には他の種類のイベント、ソース、詳細が追加される場合があります。イベントJSONデータをプログラムでデシリアライズする場合、アプリケーションが未知のプロパティを処理できるように準備されていることを確認してください。これは、これらのプロパティが追加された場合に,問題が発生するのを防ぐためです。

ジョブ状態変更イベント

(以前に送信された) 既存のジョブで状態が変更されると、イベントが作成されます。AWS Batch ジョブ状態の詳細については、ジョブの状態を参照してください。

> **Note**
>
> 最初のジョブの送信では、イベントは作成されません。

Exampleジョブ状態変更イベント

ジョブ状態変更イベントは、以下のフォーマットで配信されます。ジョブ状態変更イベントは次の形式で送信されます (以下の detail セクションは、AWS Batch API リファレンスの DescribeJobs API オペレーションから返される JobDetail オブジェクトに似ています)。EventBridge パラメータの詳細については、Amazon EventBridge ユーザーガイドのイベントとイベントパターンを参照してください。

```
{
    "version": "0",
    "id": "c8f9c4b5-76e5-d76a-f980-7011e206042b",
    "detail-type": "Batch Job State Change",
    "source": "aws.batch",
    "account": "123456789012",
```

```
    "time": "2022-01-11T23:36:40Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:batch:us-east-1:123456789012:job/4c7599ae-0a82-49aa-ba5a-4727fcce14a8"
    ],
    "detail": {
        "jobArn": "arn:aws:batch:us-east-1:123456789012:job/4c7599ae-0a82-49aa-
ba5a-4727fcce14a8",
        "jobName": "event-test",
        "jobId": "4c7599ae-0a82-49aa-ba5a-4727fcce14a8",
        "jobQueue": "arn:aws:batch:us-east-1:123456789012:job-queue/
PexjEHappyPathCanary2JobQueue",
        "status": "RUNNABLE",
        "attempts": [],
        "createdAt": 1641944200058,
        "retryStrategy": {
            "attempts": 2,
            "evaluateOnExit": []
        },
        "dependsOn": [],
        "jobDefinition": "arn:aws:batch:us-east-1:123456789012:job-definition/first-
run-job-definition:1",
        "parameters": {},
        "container": {
            "image": "137112412989.dkr.ecr.us-east-1.amazonaws.com/amazonlinux:latest",
            "command": [
                "sleep",
                "600"
            ],
            "volumes": [],
            "environment": [],
            "mountPoints": [],
            "ulimits": [],
            "networkInterfaces": [],
            "resourceRequirements": [
                {
                    "value": "2",
                    "type": "VCPU"
                }, {
                    "value": "256",
                    "type": "MEMORY"
                }
            ],
            "secrets": []
```

```
        },
        "propagateTags": false,
        "platformCapabilities": []
    }
}
```

ジョブキューのブロックイベント

AWS Batch が RUNNABLE 状態のジョブを検出してキューをブロックすると、Amazon CloudWatch Events でイベントが作成されます。サポートされているキューのブロックの原因について詳しくは、「RUNNABLE 状態でジョブが止まる」を参照してください。DescribeJobs API アクションの statusReason フィールドでも同じ原因を確認できます。

Exampleジョブキューのブロックイベント

ジョブキューのブロックイベントは、以下のフォーマットで配信されます。ジョブ状態変更イベントは次の形式で送信されます (以下の detail セクションは、AWS Batch API リファレンスの DescribeJobs API オペレーションから返される JobDetail オブジェクトに似ています)。EventBridge パラメータの詳細については、Amazon EventBridge ユーザーガイドのイベントとイベントパターンを参照してください。

```
{
    "version": "0",
    "id": "c8f9c4b5-76e5-d76a-f980-7011e206042b",
    "detail-type": "Batch Job Queue Blocked",
    "source": "aws.batch",
    "account": "123456789012",
    "time": "2022-01-11T23:36:40Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:batch:us-east-1:123456789012:job/4c7599ae-0a82-49aa-
ba5a-4727fcce14a8",
        "arn:aws:batch:us-east-1:123456789012:job-queue/PexjEHappyPathCanary2JobQueue"
    ],
    "detail": {
        "jobArn": "arn:aws:batch:us-east-1:123456789012:job/4c7599ae-0a82-49aa-
ba5a-4727fcce14a8",
        "jobName": "event-test",
        "jobId": "4c7599ae-0a82-49aa-ba5a-4727fcce14a8",
        "jobQueue": "arn:aws:batch:us-east-1:123456789012:job-queue/
PexjEHappyPathCanary2JobQueue",
        "status": "RUNNABLE",
```

```
        "statusReason": "blocked-reason",
        "attempts": [],
        "createdAt": 1641944200058,
        "retryStrategy": {
            "attempts": 2,
            "evaluateOnExit": []
        },
        "dependsOn": [],
        "jobDefinition": "arn:aws:batch:us-east-1:123456789012:job-definition/first-
run-job-definition:1",
        "parameters": {},
        "container": {
            "image": "137112412989.dkr.ecr.us-east-1.amazonaws.com/amazonlinux:latest",
            "command": [
                "sleep",
                "600"
            ],
            "volumes": [],
            "environment": [],
            "mountPoints": [],
            "ulimits": [],
            "networkInterfaces": [],
            "resourceRequirements": [
                {
                    "value": "2",
                    "type": "VCPU"
                }, {
                    "value": "256",
                    "type": "MEMORY"
                }
            ],
            "secrets": []
        },
        "propagateTags": false,
        "platformCapabilities": []
    }
}
```

サービスジョブ状態変更イベント

既存のサービスジョブで状態が変更されると、イベントが作成されます。サービスジョブの状態の詳細については、AWS Batch サービスジョブのステータスを SageMaker AI ステータスにマッピングする を参照してください。

> **Note**
>
> 最初のジョブの送信では、イベントは作成されません。

Exampleサービスジョブ状態変更イベント

サービスジョブ状態変更イベントは、以下のフォーマットで配信されます。`detail` セクションは、AWS Batch API リファレンスの DescribeServiceJob API オペレーションから返される応答に似ています。EventBridge パラメータの詳細については、Amazon EventBridge ユーザーガイドのイベントとイベントパターンを参照してください。

> **Note**
>
> `tags` および `serviceRequestPayload` フィールドはイベント `detail` に含まれません。

```
{
  "version": "0",
  "id": "c8f9c4b5-76e5-d76a-f980-7011e206042b",
  "detail-type": "Batch Service Job State Change",
  "source": "aws.batch",
  "account": "123456789012",
  "time": "2022-01-11T23:36:40Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:batch:us-east-1:123456789012:service-job/4c7599ae-0a82-49aa-
ba5a-4727fcce14a8"
  ],
  "detail": {
    "attempts": [
      {
        "serviceResourceId": {
          "name": "TrainingJobArn",
          "value": "arn:aws:sagemaker:us-east-1:123456789012:training-job/AWSBatchmy-
training-job88b610a69aa8380ca5b0a7aba3f81cb8"
        },
        "startedAt": 1641944300058,
        "stoppedAt": 1641944400058,
        "statusReason": "Received status from SageMaker: Training job completed"
      }
```

```
    ],
    "createdAt": 1641944200058,
    "jobArn": "arn:aws:batch:us-east-1:123456789012:service-job/4c7599ae-0a82-49aa-
ba5a-4727fcce14a8",
    "jobId": "0bb17543-ece6-4480-b1a7-a556d344746b",
    "jobName": "event-test",
    "jobQueue": "arn:aws:batch:us-east-1:123456789012:job-queue/HappyPathJobQueue",
    "latestAttempt": {
      "serviceResourceId": {
        "name": "TrainingJobArn",
        "value": "arn:aws:sagemaker:us-east-1:123456789012:training-job/AWSBatchmy-
training-job88b610a69aa8380ca5b0a7aba3f81cb8"
      }
    },
    "serviceJobType": "SAGEMAKER_TRAINING",
    "startedAt": 1641944300058,
    "status": "SUCCEEDED",
    "statusReason": "Received status from SageMaker: Training job completed",
    "stoppedAt": 1641944400058,
    "timeoutConfig": {
      "attemptDurationSeconds": 60
    }
  }
}
```

サービスジョブキューのブロックイベント

AWS Batch がブロックされたキューを検出すると、Amazon CloudWatch Events でイベントが作成されます。`DescribeServiceJob` API アクションの `statusReason` フィールドでキューのブロック原因を確認できます。

Exampleサービスジョブキューのブロックイベント

サービスジョブキューのブロックイベントは、以下のフォーマットで配信されます。`detail` セクションは、AWS Batch API リファレンスの DescribeServiceJob API オペレーションから返される応答に似ています。EventBridge パラメータの詳細については、Amazon EventBridge ユーザーガイドのイベントとイベントパターンを参照してください。

> **Note**
>
> `tags` および `serviceRequestPayload` フィールドはイベント `detail` に含まれません。

```
{
  "version": "0",
  "id": "c8f9c4b5-76e5-d76a-f980-7011e206042b",
  "detail-type": "Batch Service Job Queue Blocked",
  "source": "aws.batch",
  "account": "123456789012",
  "time": "2022-01-11T23:36:40Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:batch:us-east-1:123456789012:service-job/4c7599ae-0a82-49aa-
ba5a-4727fcce14a8"
  ],
  "detail": {
    "attempts": [],
    "createdAt": 1641944200058,
    "jobArn": "arn:aws:batch:us-east-1:123456789012:service-job/4c7599ae-0a82-49aa-
ba5a-4727fcce14a8",
    "jobId": "6271dfdf-d8a7-41b1-a4d2-55a2224f5375",
    "jobName": "event-test",
    "jobQueue": "arn:aws:batch:us-east-1:123456789012:job-queue/HappyPathJobQueue",
    "serviceJobType": "SAGEMAKER_TRAINING",
    "status": "RUNNABLE",
    "statusReason": "blocked-reason",
    "timeoutConfig": {
      "attemptDurationSeconds": 60
    }
  }
}
```

チュートリアル: AWS Batch で AWS ユーザー通知を使用する

AWS ユーザー通知を使用すると、AWS Batch イベントに関する通知を受け取るための配信チャンネルを設定することができます。指定したルールにイベントが一致すると、通知を受け取ります。イベントの通知は、Eメール、チャットアプリケーション内の Amazon Q Developerのチャット通知、AWS Console Mobile Applicationのプッシュ通知などの複数のチャネルで受け取ることができます。「コンソール通知センター」で通知を確認することもできます。ユーザー通知は集約をサポートしているため、特定のイベント中に受け取る通知の数を減らすことができます。

AWS Batch でユーザー通知を設定するには:

1. AWS Batch コンソールを開きます。

2. Dashboard を選択します。
3. 通知の設定 を選択します。
4. AWSUser Notifications で通知設定を作成します。

ユーザー通知の設定と表示方法の詳細については、ユーザー通知 AWS 入門 を参照してください。

EventBridge ターゲットとしての AWS Batch ジョブ

Amazon EventBridge は、Amazon Web Services リソースの変更を示すシステムイベントのほぼリアルタイムのストリームを提供します。通常、Amazon Elastic Container Service、Amazon Elastic Kubernetes Service、および AWS Fargate ジョブにおける AWS Batch は EventBridge のターゲットとして利用できます。簡単なルールを使用して、ルールに一致したイベントに応じて AWS Batch ジョブを送信できます。詳細については、Amazon EventBridge ユーザーガイドのEventBridge とは を参照してください。

EventBridge を使用して、cron や rate 式により特定の時間に自己トリガーする自動アクションをスケジュールすることもできます。詳細については、Amazon EventBridge ユーザーガイドのスケジュールに従って実行する Amazon EventBridge ルールの作成を参照してください。

イベントがイベントパターンに一致したときに実行されるルールを作成する方法については、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。

EventBridge ターゲットとなる AWS Batch ジョブの一般的な使用例としては、以下のようなユースケースがあります。

- スケジュールされたジョブは、一定の時間間隔で発生します。例えば、cron ジョブは、Amazon EC2 スポットインスタンスのコストが低い時間帯にのみ発生します。
- AWS Batch ジョブは CloudTrail にログインした API オペレーションに応答して実行されます。例えば、指定したAmazon S3 バケットにオブジェクトがアップロードされるたびにジョブが送信されます。そのたびに、EventBridge インプットトランスフォーマーはバケットとオブジェクトのキー名を AWS Batch パラメータに渡します。

> **Note**
>
> このシナリオでは、関連する AWS リソースはすべて同じリージョン内に存在する必要があります。これには、Amazon S3 バケット、EventBridge ルール、CloudTrail ログなどのリソースが含まれます。

EventBridge ルールおよびターゲットを使用して AWS Batch ジョブを送信するには、AWS Batch ジョブを実行するアクセス権限が EventBridge サービスに必要です。EventBridge コンソールからAWS Batch ジョブをターゲットとして指定するルールを作成すると、このロールを作成することもできます。このロールに必要なサービスプリンシパルと IAM アクセス権限の詳細については、EventBridge IAM ロールを参照してください。

トピック

- チュートリアル: スケジュールされた AWS Batch ジョブを作成する
- チュートリアル: イベントパターンを使用してルールを作成する
- チュートリアル: EventBridge インプットトランスフォーマーを使用してイベント情報を AWS Batch ターゲットに渡す

チュートリアル: スケジュールされた AWS Batch ジョブを作成する

次の手順では、スケジュールされた AWS Batch ジョブと必要な EventBridge IAM ロールを作成する方法について説明します。

EventBridge でスケジュールされた AWS Batch ジョブを作成するには

> **Note**
>
> この手順は、Amazon ECS、Amazon EKS、および AWS Fargate ジョブにおけるすべての AWS Batch で機能します。

1. Amazon EventBridge コンソールの https://console.aws.amazon.com/events/ を開いてください。
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. ナビゲーションペインで ルール] を選択します。

4. ルールの作成 を選択します。
5. 名前 で、コンピューティング環境の一意な名前を指定します。名前は最大 64 文字まで入力できます。大文字、小文字、数字、ハイフン (-)、アンダースコア (_) を含めることができます。

> **Note**
>
> ルールには同じリージョン内および同じイベントバス上の別のルールと同じ名前を付けることはできません。

6. (オプション) 説明に、ルールの説明を入力します。
7. イベントバス] では、このルールに関連付けるイベントバスを選択します。このルールをアカウントからのイベントと一致させるには、デフォルト を選択します。アカウントの AWS のサービス で発生したイベントは、常にアカウントのデフォルトのイベントバスに移動します。
8. (オプション) すぐに実行しないバスのルールについては、そのルールをオフにします。
9. ルールタイプ では、スケジュール] を選択します。
10. 続行してルールを作成する または 次へ を選択します。
11. [スケジュールパターン] では、次のいずれかを実行します。
    - 毎月第一月曜日の太平洋標準日午前 8:00 など、特定の時間に実行される詳細なスケジュールを選択してから、cron 式を入力します。詳細については、「Amazon EventBridge ユーザーガイド」の「Cron 式」を参照してください。
    - 一定の間隔 (10 分ごとなど) で実行するスケジュールを選択してから、rate 式を入力します。
12. 次へ を選択します。
13. ターゲットタイプ] には、AWS のサービス を選択します。
14. ターゲットの選択 で、バッチジョブのキュー を選択します。次を設定します。
    - ジョブキュー]: ジョブをスケジュールするジョブキューの Amazon リソースネーム (ARN) を入力します。
    - ジョブ定義]: ジョブに使用するジョブ定義の名前、改正、または完全な ARN を入力します。
    - ジョブ名]: ジョブの名前を入力します。
    - 配列サイズ]: (オプション) 複数のコピーを実行するためのジョブの配列サイズを入力します。詳細については、配列ジョブを参照してください。
    - ジョブの試行]: (オプション) ジョブが失敗したときに再試行する回数を入力します。詳細については、ジョブの再試行の自動化を参照してください。

15. バッチジョブキュー] ターゲットタイプで、EventBridge はターゲットにイベントを送信するためのアクセス許可が必要です。EventBridge は、ルールの実行に必要な IAM ロールを作成できます。次のいずれかを行います:

    - 自動的に IAM ロールを作成するには、この特定のリソースに対して新しいロールを作成するを選択します。
    - 以前に作成した IAM ロールを使用するには、既存のロールの使用 を選択します。

16. （オプション）追加設定 を展開します。

    a. ターゲット入力の設定 で、イベントからのテキストをターゲットに渡す前にどのように処理するかを選択します。

    b. イベントの最大保存期間 では、未処理のイベントを保存する時間間隔を指定します。

    c. 再試行 では、イベントを再試行する回数を入力します。

    d. デッドレターキュー では、未処理イベントの取扱いに関するオプションを選択します。必要に応じて、デッドレターキューに Amazon SQS キューを指定します。

17. (オプション) 別のターゲットを追加] を選択して、このルールに別のターゲットを追加します。

18. 次へ を選択します。

19. (オプション) タグ で 新しいタグを追加 を選択し、ルールのリソースラベルを追加します。詳細については、Amazon EventBridge のタグを参照してください。

20. 次へ を選択します。

21. レビューと作成では、設定手順を確認してください。変更する必要がある場合は、編集] を選択します。完了したら、ルールの作成 を選択します。

ルールの作成の詳細については、「Amazon EventBridge ユーザーガイド」の「スケジュールに従って実行する Amazon EventBridge ルールの作成」を参照してください。

チュートリアル: イベントパターンを使用してルールを作成する

以下の手順では、イベントパターンを使用してルールを作成する方法について説明します。

定義されたパターンにイベントが一致したときに、イベントをターゲットに送信するルールを作成するには

> **Note**
>
> この手順は、Amazon ECS、Amazon EKS、および AWS Fargate ジョブにおけるすべての AWS Batch で機能します。

1. Amazon EventBridge コンソールの https://console.aws.amazon.com/events/ を開いてください。
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. ナビゲーションペインで ルール] を選択します。
4. ルールの作成 を選択します。
5. 名前 で、コンピューティング環境の一意な名前を指定します。名前は最大 64 文字まで入力できます。大文字、小文字、数字、ハイフン (-)、アンダースコア (_) を含めることができます。

   > **Note**
   >
   > ルールには同じリージョン内および同じイベントバス上の別のルールと同じ名前を付けることはできません。

6. (オプション) 説明に、ルールの説明を入力します。
7. イベントバス] では、このルールに関連付けるイベントバスを選択します。このルールをアカウントからのイベントと一致させるには、デフォルト を選択します。アカウントの AWS のサービス で発生したイベントは、常にアカウントのデフォルトのイベントバスに移動します。
8. (オプション) すぐに実行しないバスのルールについては、そのルールをオフにします。
9. ルールタイプ では、イベントパターンを持つルール] を選択します。
10. 次へ を選択します。
11. イベントソース で、AWS イベントまたは EventBridge パートナーイベント を選択します。
12. (オプション) サンプルイベント では:
    a. サンプルイベントタイプ では、AWSイベント を選択します。
    b. サンプルイベントでは、 バッチジョブの状態変更 を選択します。
13. 作成方法 では、パターンフォームの使用 を選択します。

14. イベントパターン では:

    a. イベントソース では、AWS のサービス を選択します。

    b. AWS のサービス では、バッチ を選択します。

    c. イベントタイプ では、バッチジョブの状態変更 を選択します。

15. 次へ を選択します。

16. ターゲットタイプ] には、AWS のサービス を選択します。

17. ターゲットタイプを選択 で、ターゲットタイプを選択します。例えば、バッチジョブのキューを選択します。以下を指定します。

    - ジョブキュー]: ジョブをスケジュールするジョブキューの Amazon リソースネーム (ARN) を入力します。
    - ジョブ定義]: ジョブに使用するジョブ定義の名前、改正、または完全な ARN を入力します。
    - ジョブ名]: ジョブの名前を入力します。
    - 配列サイズ]: (オプション) 複数のコピーを実行するためのジョブの配列サイズを入力します。詳細については、配列ジョブを参照してください。
    - ジョブの試行]: (オプション) ジョブが失敗したときに再試行する回数を入力します。詳細については、ジョブの再試行の自動化を参照してください。

18. バッチジョブキュー] ターゲットタイプで、EventBridge はターゲットにイベントを送信するためのアクセス許可が必要です。EventBridge は、ルールの実行に必要な IAM ロールを作成できます。次のいずれかを行います:

    - 自動的に IAM ロールを作成するには、この特定のリソースに対して新しいロールを作成する を選択します。
    - 以前に作成した IAM ロールを使用するには、既存のロールの使用 を選択します。

19. (オプション) 追加設定 を展開します。

    a. ターゲット入力の設定 で、イベントからのテキストの処理方法を選択します。

    b. イベントの最大保存期間 では、未処理のイベントを保存する時間間隔を指定します。

    c. 再試行 では、イベントを再試行する回数を入力します。

    d. デッドレターキュー では、未処理イベントの取扱いに関するオプションを選択します。必要に応じて、デッドレターキューに Amazon SQS キューを指定します。

20. (オプション) 別のターゲットを追加 を選択して、別のターゲットを追加します。

21. 次へ を選択します。

22. (オプション) タグ で 新しいタグを追加 を選択し、リソースラベルを追加します。詳細については、Amazon EventBridge ユーザーガイドの Amazon EventBridge のタグを参照してください。
23. 次へ を選択します。
24. レビューと作成では、設定手順を確認してください。変更する必要がある場合は、編集] を選択します。完了したら、ルールの作成 を選択します。

    ルールの作成に関する詳細については、「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。

チュートリアル: EventBridge インプットトランスフォーマーを使用してイベント情報を AWS Batch ターゲットに渡す

EventBridge インプットトランスフォーマーを使用して、ジョブ送信で AWS Batch にイベント情報を渡すことができます。これは、他の AWS イベント情報の結果としてジョブを呼び出す場合は、特に重要です。例えば、Amazon S3 バケットへのオブジェクトのアップロード時など。または、コンテナのコマンドで、パラメータの置換値を使用したジョブ定義を使用できます。EventBridge インプットトランスフォーマーは、イベントデータに基づいてパラメータ値を提供できます。

つまり、開始するイベントからの情報を解析し、`parameters` オブジェクトに変換する AWS Batch イベントターゲットを作成できます。ジョブが実行されると、トリガーイベントからのパラメータがジョブコンテナのコマンドに渡されます。

> **Note**
>
> このシナリオでは、すべての AWS リソース (Amazon S3 バケット、EventBridge ルール、CloudTrail ログなど) が同じリージョンに存在する必要があります。

インプットトランスフォーマーを使用して AWS Batch ターゲットを作成するには

1. Amazon EventBridge コンソールの https://console.aws.amazon.com/events/ を開いてください。
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. ナビゲーションペインで [ルール] を選択します。
4. ルールの作成 を選択します。

5. 名前 で、コンピューティング環境の一意な名前を指定します。名前は最大 64 文字まで入力できます。大文字、小文字、数字、ハイフン (-)、アンダースコア (_) を含めることができます。

   > **Note**
   >
   > ルールには、同じ AWS リージョン 内および同じイベントバス上の別のルールと同じ名前を付けることはできません。

6. (オプション) 説明 に、ルールの説明を入力します。
7. [イベントバス] では、このルールに関連付けるイベントバスを選択します。このルールをアカウントからのイベントと一致させるには、デフォルト を選択します。アカウントの AWS のサービス で発生したイベントは、常にアカウントのデフォルトのイベントバスに移動します。
8. (オプション) すぐに実行しないバスのルールについては、そのルールをオフにします。
9. ルールタイプ では、[スケジュール] を選択します。
10. 続行してルールを作成する または 次へ を選択します。
11. [スケジュールパターン] では、次のいずれかを実行します。

    - 毎月第一月曜日の太平洋標準日午前 8:00 など、特定の時間に実行される詳細なスケジュールを選択してから、cron 式を入力します。詳細については、「Amazon EventBridge ユーザーガイド」の「Cron 式」を参照してください。
    - 一定の間隔 (10 分ごとなど) で実行するスケジュールを選択してから、rate 式を入力します。

12. 次へ を選択します。
13. [ターゲットタイプ] には、AWS のサービス を選択します。
14. ターゲットの選択 で、バッチジョブのキュー を選択します。次を設定します。

    - [ジョブキュー]: ジョブをスケジュールするジョブキューの Amazon リソースネーム (ARN) を入力します。
    - [ジョブ定義]: ジョブに使用するジョブ定義の名前、改正、または完全な ARN を入力します。
    - [ジョブ名]: ジョブの名前を入力します。
    - [配列サイズ]: (オプション) 複数のコピーを実行するためのジョブの配列サイズを入力します。詳細については、配列ジョブを参照してください。
    - [ジョブの試行]: (オプション) ジョブが失敗したときに再試行する回数を入力します。詳細については、ジョブの再試行の自動化を参照してください。

15. [バッチジョブキュー] ターゲットタイプで、EventBridge はターゲットにイベントを送信するためのアクセス許可が必要です。EventBridge は、ルールの実行に必要な IAM ロールを作成できます。次のいずれかを行います:

    - 自動的に IAM ロールを作成するには、この特定のリソースに対して新しいロールを作成する を選択します。
    - 以前に作成した IAM ロールを使用するには、既存のロールの使用 を選択します。

16. (オプション) 追加設定 を展開します。
17. 追加設定 セクションの ターゲット入力の設定] で [入力トランスフォーマー] を選択します。
18. 入力トランスフォーマーの設定 を選択します。
19. (オプション) サンプルイベント では:

    a. サンプルイベントタイプ では、AWSイベント を選択します。

    b. サンプルイベントでは、バッチジョブの状態変更 を選択します。

20. ターゲット入力トランスフォーマー の 入力パス で、トリガーするイベントから解析する値を指定します。例えば、バッチジョブの状態変更 イベントを解析するには、次の JSON 形式を使用します。

```
{
    "instance": "$.detail.jobId",
    "state": "$.detail.status"
}
```

21. [テンプレート] では、以下を入力します。

```
{
    "instance": <jobId> ,
    "status": <status>
}
```

22. [確認] を選択します。
23. イベントの最大保存期間 では、未処理のイベントを保存する時間間隔を指定します。
24. 再試行 では、イベントを再試行する回数を入力します。
25. デッドレターキュー では、未処理イベントの取扱いに関するオプションを選択します。必要に応じて、デッドレターキューに Amazon SQS キューを指定します。
26. (オプション) 別のターゲットを追加 を選択して、別のターゲットを追加します。

27. 次へ を選択します。
28. (オプション) タグ で 新しいタグを追加 を選択し、リソースラベルを追加します。詳細については、Amazon EventBridge ユーザーガイドの Amazon EventBridge のタグを参照してください。
29. 次へ を選択します。
30. レビューと作成では、設定手順を確認してください。変更する必要がある場合は、[編集] を選択します。完了したら、ルールの作成 を選択します。

チュートリアル: EventBridge を使用して AWS Batch ジョブイベントをリッスンする

このチュートリアルでは、AWS Batch ジョブイベントを聴いて、CloudWatch Logs ログストリームに書き込むシンプルな AWS Lambda 関数を設定します。

前提条件

このチュートリアルでは、コンピューティング作業環境と、ジョブを受け付けることができるジョブキューがあることを前提としています。イベントをキャプチャするコンピューティング作業環境とジョブキューがなければ、AWS Batch チュートリアルの開始方法のステップに従って作成します。このチュートリアルの最後に、オプションでこのジョブキューにジョブを送信して Lambda 関数が正しく設定されていることをテストできます。

トピック

- チュートリアル: Lambda 関数を作成する
- チュートリアル: イベントルールを登録する
- チュートリアル: 設定をテストする

チュートリアル: Lambda 関数を作成する

この手順では、AWS Batch イベントストリームメッセージのターゲットとなるシンプルな Lambda 関数を作成します。

ターゲットの Lambda 関数を作成するには

1. AWS Lambda コンソールの https://console.aws.amazon.com/lambda/ を開いてください。
2. 関数の作成 を選択し、一から作成 を選択します。

3. 関数名 に、batch-event-stream-handlerと入力します。
4. ランタイム] では、Python 3.8] を選択します。
5. 関数を作成 を選択します。
6. 関数コード] セクションで、以下の例に合わせてサンプルコードを編集します。

```
import json


def lambda_handler(event, _context):
    # _context is not used
    del _context
    if event["source"] != "aws.batch":
        raise ValueError("Function only supports input from events with a source
 type of: aws.batch")

    print(json.dumps(event))
```

これは、AWS Batch から送信されたイベントを印刷するシンプルな Python 3.8 関数です。すべてが正しく設定されると、このチュートリアルの最後に、この Lambda 関数に関連付けられている CloudWatch Logs ログストリームにイベントの詳細が表示されます。

7. [デプロイ] をクリックします。

チュートリアル: イベントルールを登録する

このセクションでは、AWS Batch リソースから来るジョブイベントをキャプチャする EventBridge イベントルールを作成します。このルールでは、それが定義されているアカウント内の AWS Batch から送信されるすべてのイベントがキャプチャされます。ジョブメッセージ自体に、イベントソースに関する情報 (イベントソースの送信先ジョブキューの情報など) が含まれています。この情報を使用して、プログラムでイベントをフィルタリングおよびソートできます。

> **Note**
>
> AWS マネジメントコンソール を使用してイベントルールを作成すると、コンソールは自動的に EventBridge が Lambda 関数を呼び出すための IAM 権限を追加します。ただし、AWS CLI を使用してイベントルールを作成する場合は、この権限を明示的に付与する必要があります。詳細については、Amazon EventBridge ユーザーガイドのイベントとイベントパターンを参照してください。

EventBridge ルールを作成するには

1. Amazon EventBridge コンソールの https://console.aws.amazon.com/events/ を開いてください。
2. ナビゲーションペインで [ルール] を選択します。
3. [ルールの作成] を選択します。
4. ルールの名前と説明を入力します。

   ルールには同じリージョン内および同じイベントバス上の別のルールと同じ名前を付けることはできません。
5. [イベントバス] では、このルールに関連付けるイベントバスを選択します。このルールをアカウントからのイベントと一致させるには、AWS のデフォルトのイベントバスを選択します。アカウントの AWS サービスがイベントを発行すると、常にアカウントのデフォルトのイベントバスに移動します。
6. [ルールタイプ] で、[イベントパターンを持つルール] を選択します。
7. 次へ を選択します。
8. イベントソース では、その他] を選択します。
9. イベントパターン では、カスタムパターン (JSON エディター) を選択します。
10. 次のイベントパターンをテキストエリアに貼り付けます。

```
{
  "source": [
    "aws.batch"
  ]
}
```

   このルールは、すべての AWS Batch グループとすべての AWS Batch イベントに適用されます。または、より具体的なルールを作成し、一部の結果をフィルタで除外することもできます。
11. 次へ を選択します。
12. ターゲットタイプ] では、AWSサービス] を選択します。
13. ターゲットを選択 で、Lambda 関数 を選択します。
14. (オプション) 追加設定] では、以下を実行します。

    a. 最大イベント有効期間 に、1 分 (00:01) から 24 時間 (24:00) の間の値を入力します。

    b. 再試行 で、0 ~ 185 の数値を入力します。

c. デッドレターキュー で、標準 Amazon SQS キューをデッドレターキューとして使用するかどうかを選択します。EventBridge は、このルールに一致するイベントがターゲットに正常に配信されなかった場合に、そのイベントをデッドレターキューに送信します。次のいずれかを行います:

- デッドレターキューを使用しない場合は、なし] を選択します。
- デッドレターキューとして使用する現在の AWS アカウントの Amazon SQS キューを選択 を選択し、ドロップダウンから使用するキューを選択します。
- 他の AWS アカウントの Amazon SQS キューをデッドレターキューとして選択 を選択し、使用するキューの ARN を入力します。キューにメッセージを送信するための EventBridge 許可を付与するリソースベースのポリシーをそのキューにアタッチする必要があります。詳細については、Amazon EventBridge ユーザーガイドのデッドレターキューへの許可の付与を参照してください。

15. 次へ を選択します。
16. (オプション) ルールに 1 つ以上のタグを入力します。詳細については、Amazon EventBridge ユーザーガイドのAmazon EventBridge のタグを参照してください。
17. 次へをクリックします。
18. ルールの詳細を確認し、ルールの作成 を選択します。

チュートリアル: 設定をテストする

ジョブキューにジョブを送信して EventBridge 設定をテストできます。すべてが適切に設定されている場合、Lambda 関数がトリガーされ、関数の CloudWatch Logs ログストリームにイベントデータが書き込まれます。

設定をテストするには

1. AWS Batch コンソールを https://console.aws.amazon.com/batch/ で開きます。
2. 新しい AWS Batch ジョブを送信します。詳細については、チュートリアル: ジョブを送信するを参照してください。
3. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
4. ナビゲーションペインで、ログ を選択して Lambda 関数 (例えば、/aws/lambda/*my-function##*)] のロググループを選択します。
5. イベントデータを表示するログストリームを選択します。

チュートリアル: 失敗したジョブイベントに Amazon シンプル通知サービスアラートを送信する

このチュートリアルでは、FAILED のステータスに移行したジョブのジョブイベントのみをキャプチャする Amazon EventBridge のルールを設定します。このチュートリアルの最後に、このジョブキューにジョブを送信することもできます。これは、Amazon SNS アラートが正しく設定されていることをテストするためです。

前提条件

このチュートリアルでは、コンピューティング作業環境と、ジョブを受け付けることができるジョブキューがあることを前提としています。イベントをキャプチャするコンピューティング作業環境とジョブキューがなければ、AWS Batch チュートリアルの開始方法 のステップに従って作成します。

トピック

- チュートリアル: Amazon SNS トピックを作成してサブスクライブする
- チュートリアル: イベントルールを登録する
- チュートリアル: ルールをテストする
- 代替ルール: バッチジョブキューがブロックされました

チュートリアル: Amazon SNS トピックを作成してサブスクライブする

このチュートリアルでは、新しいイベントルールのイベントターゲットとして使用する Amazon SNS トピックを設定します。

Amazon SNS トピックを作成するには

1. Amazon SNS コンソール (https://console.aws.amazon.com/sns/v3/home) を開きます。
2. トピック、トピックの作成 の順に選択します。
3. [Type] (タイプ) で、[Standard] (標準) を選択します。
4. 名前として **JobFailedAlert** を入力し、トピックを作成 を選択します。
5. JobFailedAlert 画面で、サブスクリプションの作成 を選択します。
6. Protocol (プロトコル) として Email (E メール) を選択します。
7. エンドポイント では、現在アクセスできるメールアドレスを入力し、サブスクリプションの作成 を選択します。

8. メールアカウントを確認し、サブスクリプションの確認メールメッセージが届くのを待ちます。確認メールが届いたら、[Confirm subscription] (サブスクリプションの確認) を選択します。

チュートリアル: イベントルールを登録する

次に、失敗したジョブイベントのみをキャプチャするイベントルールを登録します。

EventBridge ルールを登録するには

1. Amazon EventBridge コンソールの https://console.aws.amazon.com/events/ を開いてください。
2. ナビゲーションペインで [ルール] を選択します。
3. [ルールの作成] を選択します。
4. ルールの名前と説明を入力します。

   ルールには同じリージョン内および同じイベントバス上の別のルールと同じ名前を付けることはできません。
5. [イベントバス] では、このルールに関連付けるイベントバスを選択します。このルールをアカウントからのイベントと一致させるには、AWS のデフォルトのイベントバスを選択します。アカウントの AWS サービスがイベントを発行すると、常にアカウントのデフォルトのイベントバスに移動します。
6. [ルールタイプ] で、[イベントパターンを持つルール] を選択します。
7. 次へ を選択します。
8. イベントソース では、[その他] を選択します。
9. イベントパターン では、カスタムパターン (JSON エディター) を選択します。
10. 次のイベントパターンをテキストエリアに貼り付けます。

```
{
  "detail-type": [
    "Batch Job State Change"
  ],
  "source": [
    "aws.batch"
  ],
  "detail": {
    "status": [
      "FAILED"
```

```
    ]
  }
}
```

このコードでは、ジョブのステータスが FAILED であるイベントに一致する EventBridge イベントルールを定義します。イベントパターンの詳細については、Amazon EventBridge ユーザーガイドのイベントとイベントパターンを参照してください。

11. 次へ をクリックします。
12. [ターゲットタイプ] では、[AWS サービス] を選択します。
13. ターゲットの選択 には SNS トピック を選択し、トピック には JobFailedAlert を選択します。
14. (オプション) 追加設定では、以下を実行します。

    a. 最大イベント有効期間 に、1 分 (00:01) から 24 時間 (24:00) の間の値を入力します。

    b. 再試行 で、0 ~ 185 の数値を入力します。

    c. デッドレターキュー で、標準 Amazon SQS キューをデッドレターキューとして使用するかどうかを選択します。EventBridge は、このルールに一致するイベントがターゲットに正常に配信されなかった場合に、そのイベントをデッドレターキューに送信します。次のいずれかを行います:

    - デッドレターキューを使用しない場合は、[なし] を選択します。
    - デッドレターキューとして使用する現在の AWS アカウントの Amazon SQS キューを選択 を選択し、ドロップダウンから使用するキューを選択します。
    - 他の AWS アカウントの Amazon SQS キューをデッドレターキューとして選択 を選択し、使用するキューの ARN を入力します。キューにメッセージを送信するための EventBridge 許可を付与するリソースベースのポリシーをそのキューにアタッチする必要があります。詳細については、Amazon EventBridge ユーザーガイドのデッドレターキューへの許可の付与を参照してください。

15. 次へ を選択します。
16. (オプション) ルールに 1 つ以上のタグを入力します。詳細については、Amazon EventBridge ユーザーガイドのAmazon EventBridge のタグを参照してください。
17. 次へをクリックします。
18. ルールの詳細を確認し、ルールの作成 を選択します。

チュートリアル: ルールをテストする

ルールをテストするには、ゼロ以外の終了コードで開始直後に終了するジョブを送信します。イベントルールが正しく設定されていれば、数分以内にイベントテキストが記載されたメールメッセージが届きます。

ルールをテストするには

1. AWS Batch コンソールを https://console.aws.amazon.com/batch/ で開きます。
2. 新しい AWS Batch ジョブを送信します。詳細については、チュートリアル: ジョブを送信するを参照してください。ジョブのコマンドでは、このコマンドを置き換え、終了コード 1 でコンテナを終了します。

   ```
   /bin/sh, -c, 'exit 1'
   ```

3. 失敗したジョブの通知に関するアラートのメールが届いていることを確認します。

代替ルール: バッチジョブキューがブロックされました

ブロックされたバッチジョブキューをモニタリングするイベントルールを作成するには、次の変更を加えてこのチュートリアルを繰り返します。

1. チュートリアル: Amazon SNS トピックを作成してサブスクライブする では、トピック名として *BlockedJobQueue* を使用します。
2. チュートリアル: イベントルールを登録する では、JSON エディタで次のパターンを使用します。

   ```
   {
      "detail-type": [
        "Batch Job Queue Blocked"
      ],
      "source": [
        "aws.batch"
      ]
   }
   ```

Elastic Fabric Adapter

Elastic Fabric Adapter (EFA) は、ハイパフォーマンスコンピューティング (HPC) アプリケーションを高速化するネットワークデバイスです。AWS Batch は、以下の条件が満たされている場合、EFA を使用するアプリケーションをサポートします。

- EFA をサポートする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「サポートされるインスタンスタイプ」を参照してください。

  > **Tip**
  >
  > AWS リージョンで EFA をサポートするインスタンスタイプのリストを表示するには 、次のコマンドを実行します。次に、AWS Batchのコンソールで利用可能なインスタンスタイプの返されたリストと共に、返されたリストを相互参照します。
  >
  > ```
  > $ aws ec2 describe-instance-types  --region us-east-1  --filters Name=network-info.efa-supported,Values=true  --query "InstanceTypes[*].[InstanceType]"  --output text | sort
  > ```

- EFAをサポートするオペレーティングシステムのリストについては、 サポート対象のオペレーティングシステムを参照してください。
- AMI には EFA ドライバーがロードされています。
- EFA のセキュリティグループは、セキュリティグループとの間で送受信されるすべてのトラフィックを許可する必要があります。
- EFA を使用するすべてのインスタンスは、同じクラスタープレイスメントグループに属していなくてはなりません。
- ジョブ定義には、`hostPath` を `/dev/infiniband/uverbs0` に設定した`devices` メンバーを含めて、EFA デバイスがコンテナにパススルーされるようにする必要があります。`containerPath` が特定される場合は、それも `/dev/infiniband/uverbs0` に設定する必要があります。`permissions` を設定する場合は、`READ` | `WRITE` | `MKNOD` に設定する必要があります。

  LinuxParameters メンバーの場所は、マルチノードの並列ジョブと単一ノードのコンテナジョブとでは異なります。以下の例は、その違いを示すものですが、必要な値が不足しています。

Exampleマルチノードの並列ジョブの例

```
{
  "jobDefinitionName": "EFA-MNP-JobDef",
  "type": "multinode",
  "nodeProperties": {
    ...
    "nodeRangeProperties": [
      {
        ...
        "container": {
          ...
          "linuxParameters": {
            "devices": [
              {
                "hostPath": "/dev/infiniband/uverbs0",
                "containerPath": "/dev/infiniband/uverbs0",
                "permissions": [
                    "READ", "WRITE", "MKNOD"
                ]
              },
            ],
          },
        },
      },
    ],
  },
}
```

Example単一ノードのコンテナジョブの例

```
{
  "jobDefinitionName": "EFA-Container-JobDef",
  "type": "container",
  ...
  "containerProperties": {
    ...
    "linuxParameters": {
      "devices": [
        {
          "hostPath": "/dev/infiniband/uverbs0",
        },
```

```
      ],
    },
  },
}
```

EFA の詳細については、「Amazon EC2 ユーザーガイド」の「Elastic Fabric Adapter」を参照してください。

モニタリング AWS Batch

モニタリングは、 と AWS Batch 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部分です。

マルチポイント障害が発生した場合にデバッグしやすくするために、 AWS ソリューションのすべての部分からモニタリングデータを収集することを強くお勧めします。まず、以下の質問に答えて監視計画を作成します。どのように答えるべきかわからない場合でも、Amazon CloudWatch Logs を使用してパフォーマンスのベースラインを確立できます。

- どのような目的でモニタリングしますか?
- どのリソースをモニタリングしますか?
- どのくらいの頻度でこれらのリソースをモニタリングしますか?
- どのモニタリングツールを使用しますか?
- 誰がモニタリングタスクを実行しますか?
- 問題が発生したときに誰が通知を受け取りますか?

次のステップでは、さまざまな時間や負荷条件で AWS Batch パフォーマンスを測定することで、環境で通常のパフォーマンスのベースラインを確立します。モニタリング中は AWS Batch、現在のパフォーマンスデータと比較できるように、モニタリングデータの履歴を保持します。これにより、通常のパフォーマンスパターンとパフォーマンスの異常を特定し、問題に対処するための方法を考えることができます。

このセクションのトピックは、ログ記録と AWS Batchの監視を開始するために役立ちます。

トピック

- での CloudWatch Logs の使用 AWS Batch
- AWS Batch CloudWatchコンテナインサイト
- CloudWatch Logs を使用して Amazon EKS ジョブにおける AWS Batch をモニタリングする

での CloudWatch Logs の使用 AWS Batch

EC2 リソースで AWS Batch ジョブを設定して、詳細なログ情報とメトリクスを CloudWatch Logs に送信できます。このようにして、1つの便利な場所でジョブからのさまざまなログを表示

できます。CloudWatch Logs の詳細については、Amazon CloudWatch ユーザーガイドのAmazon CloudWatch Logs とはを参照してください。

> **Note**
>
> デフォルトでは、 AWS Fargate コンテナの CloudWatch Logs はオンになっています。

CloudWatch Logsのロギングを有効にしてカスタマイズするには、次の1回限りの設定タスクを確認してください：

- EC2 リソースに基づく AWS Batch コンピューティング環境の場合は、`ecsInstanceRole`ロールに IAM ポリシーを追加します。詳細については、「the section called “チュートリアル: CloudWatch Logs IAM ポリシーを追加する”」を参照してください。
- 詳細な CloudWatch モニタリングを含む Amazon EC2 起動テンプレートを作成し、 AWS Batch コンピューティング環境を作成するときにテンプレートを指定します。既存のイメージに CloudWatch エージェントをインストールし、 AWS Batch 初回実行ウィザードでイメージを指定することもできます。
- (オプション) awslogs ドライバーを設定します。EC2 リソースと Fargate リソースの両方で、デフォルトの動作を変更するパラメータを追加できます。詳細については、「the section called “awslogs ログドライバーを使用する”」を参照してください。

トピック

- チュートリアル: CloudWatch Logs IAM ポリシーを追加する
- CloudWatch エージェントをインストールして設定する
- チュートリアル: CloudWatch Logs を表示する

チュートリアル: CloudWatch Logs IAM ポリシーを追加する

ジョブが CloudWatch Logs にログデータと詳細なメトリクスを送信する前に、CloudWatch Logs API を使用する IAM ポリシーを作成する必要があります。IAMポリシーを作成したら、`ecsInstanceRole` ロールにアタッチします。

> **Note**
>
> ECS-CloudWatchLogs ポリシーが ecsInstanceRole ロールにアタッチされていない場合でも、まだ基本メトリクスは CloudWatch Logs に送信できます。ただし、基本メトリックスには、ログデータや空きディスク容量などの詳細なメトリックスは含まれません。

AWS Batch コンピューティング環境は Amazon EC2 リソースを使用します。 AWS Batch 初回実行ウィザードを使用してコンピューティング環境を作成すると、 はecsInstanceRoleロール AWS Batch を作成し、それを使用して環境を設定します。

初回実行ウィザードを使用していない場合は、 AWS Command Line Interface または AWS Batch API でコンピューティング環境を作成するときにecsInstanceRoleロールを指定できます。詳細については、AWS CLI コマンドリファレンスまたはAWS Batch API リファレンスを参照してください。

ECS-CloudWatchLogs 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. ナビゲーションペインで、ポリシー を選択してください。
3. ポリシーの作成を選択します。
4. JSON を選択し、以下のポリシーを入力します：

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:*:*:*"
            ]
```

```
        }
    ]
}
```

5. [Next: Tags] (次へ: タグ) を選択します。
6. (オプション) タグを追加 で タグを追加 を選択し、ポリシーにタグを追加します。
7. Next: Review] (次へ: レビュー) を選択します。
8. ポリシーの確認 ページで、名前 に**ECS-CloudWatchLogs**と入力し、次にオプションで説明 に入力します。
9. ポリシーの作成を選択します。

ECS-CloudWatchLogs ポリシーを **ecsInstanceRole** にアタッチするには

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. ナビゲーションペインで Roles] (ロール) を選択します。
3. `ecsInstanceRole` を選択してください。ロールが存在しない場合は、Amazon ECS インスタンスロールの手順に従ってロールを作成します。
4. アクセス許可を追加を選択し、次にポリシーをアタッチ を選択します。
5. ECS-CloudWatchLogs ポリシーを選択してから、ポリシーのアタッチ を選択します。

CloudWatch エージェントをインストールして設定する

CloudWatch モニタリングを含む Amazon EC2 起動テンプレートを作成できます。起動テンプレートの詳細については、「Amazon EC2 ユーザーガイド」の「起動テンプレートからのインスタンスの起動」と「高度な詳細」を参照してください。

CloudWatch エージェントを既存の Amazon EC2 AMI にインストールし、 AWS Batch 初回実行ウィザードでイメージを指定することもできます。詳細については、「CloudWatch エージェントをインストールする」と「AWS Batch チュートリアルの開始方法」を参照してください。

> **Note**
>
> 起動テンプレートは AWS Fargate リソースではサポートされていません。

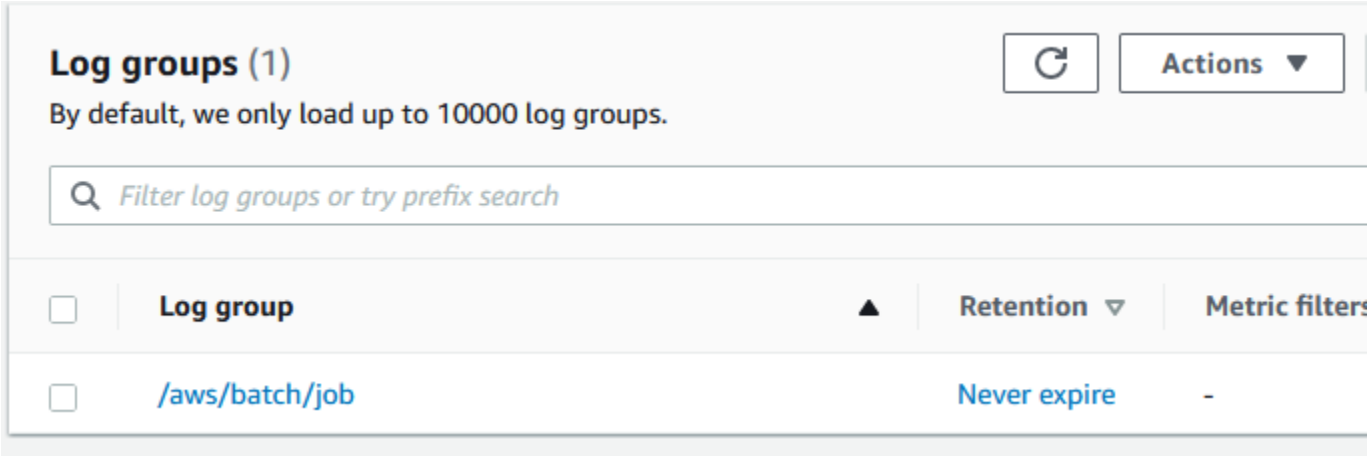
チュートリアル: CloudWatch Logs を表示する

で CloudWatch Logs ログを表示および検索できます AWS マネジメントコンソール。

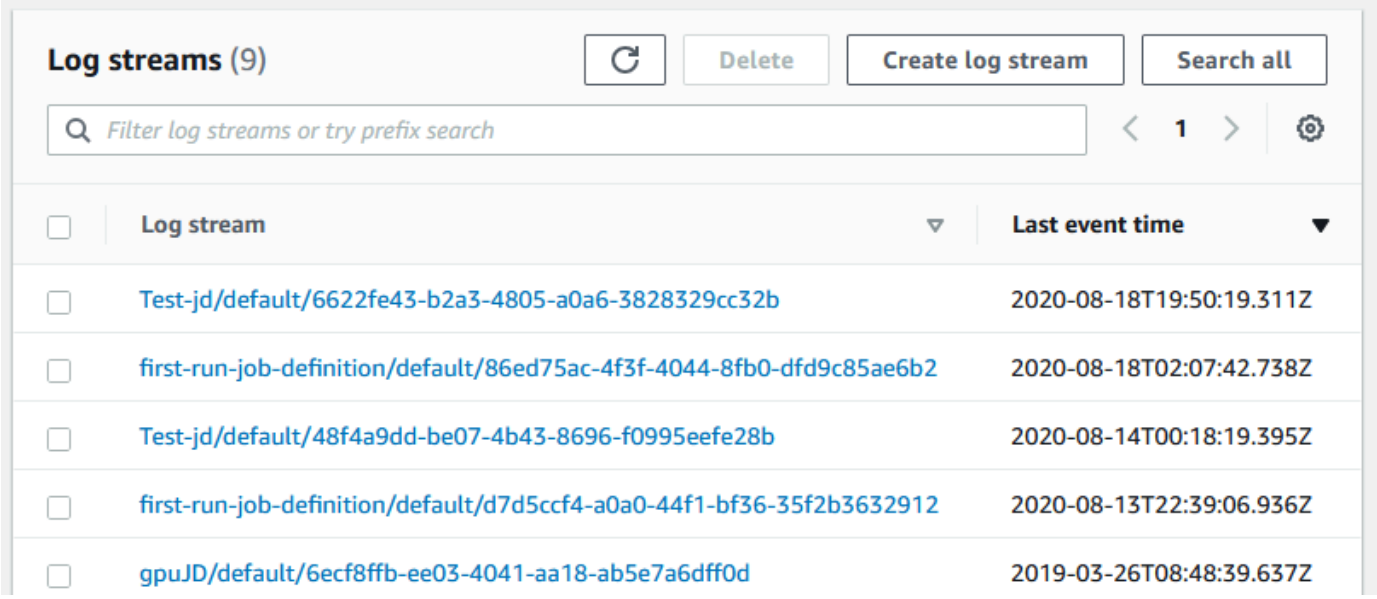
> **Note**
>
> CloudWatch Logs にデータが表示されるまでに、数分かかる場合があります。

CloudWatch Logs に送信されたログデータを表示するには

1. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
2. 左側のナビゲーションペインで ログ を選択してから ロンググループ を選択します。

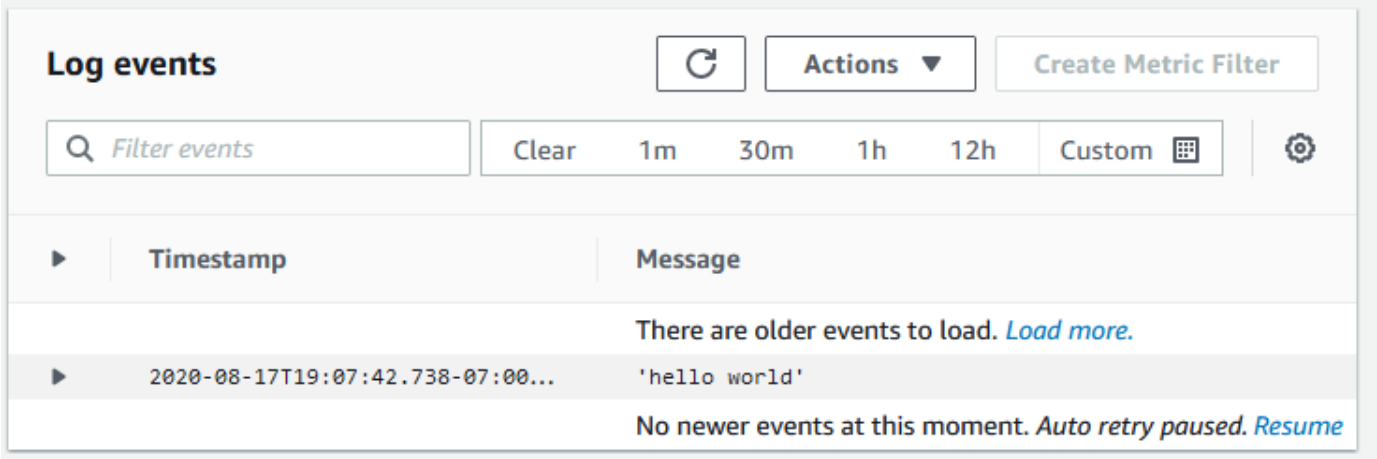


3. 表示するロググループを選択します。



4. 表示するログストリームを選択します。デフォルトでは、ストリームはジョブ名の最初の 200 文字と Amazon ECS タスク ID によって識別されます。

> **Tip**
>
> ログストリームデータをダウンロードするには、アクション を選択します。



AWS Batch CloudWatchコンテナインサイト

CloudWatchコンテナインサイトは、AWS Batch コンピュート環境とジョブからメトリクスとログを収集、集約、要約します。メトリクスには、CPU、メモリ、ディスク、ネットワークの使用率が含まれます。これらのメトリクスは、 CloudWatch ダッシュボードに追加できます。

運用データは、パフォーマンスログイベントとして収集されます。これらは、高濃度データを取り込み、大規模に保存することが可能な構造化された JSON スキーマを使用するエントリです。このデータから、CloudWatchはCloudWatchメトリクスとして、コンピュート環境とジョブレベルでより高いレベルの集約メトリクスを作成します。詳細については、Amazon CloudWatch ユーザーガイドのAmazon ECS のコンテナインサイト構造化ログを参照してください。

> **Important**
>
> CloudWatchコンテナインサイトによって収集されたメトリクスは、カスタムメトリクスとして課金されます。詳細については、Amazon CloudWatch Events の料金を参照してください。

トピック

コンテナインサイトをオンにします。

AWS Batch コンピューティング環境の Container Insights をオンにするには、次の手順を実行します。

1. AWS Batch コンソールを開きます。
2. [Environments] (環境) を選択します。
3. ご希望のコンピュート環境をお選びください。
4. [コンテナのインサイト] タブで、コンピューティング環境の [コンテナのインサイト] をオンにします。

> **Tip**
>
> デフォルトのインターバルを選択してメトリクスを集計したり、カスタムのインターバルを作成したりできます。

以下のメトリクスが表示されます。Amazon ECS コンテナインサイトメトリクスの一覧については、Amazon CloudWatch ユーザーガイドのAmazon ECS コンテナインサイトメトリクスの表示を参照してください。

- — **JobCount** コンピューティング環境で実行されるジョブの数。
- — **ContainerInstanceCount** Amazon ECS エージェントを実行し、コンピューティング環境に登録されている Amazon Elastic Compute Cloud インスタンスの数。
- — **MemoryReserved** コンピュート環境のジョブによって予約されるメモリ。このメトリクスは、ジョブ定義にメモリ予約が定義されているタスクに対してのみ収集されます。
- — **MemoryUtilized** コンピュート環境のジョブによって予約されるメモリ。このメトリクスは、ジョブ定義にメモリ予約が定義されているタスクに対してのみ収集されます。
- — **CpuReserved** コンピュート環境ジョブによって予約されている CPU ユニット。このメトリクスは、ジョブ定義にCPU予約が定義されているタスクに対してのみ収集されます。
- — **CpuUtilized** コンピュート環境のジョブが使用する CPU ユニット。このメトリクスは、ジョブ定義にCPU予約が定義されているタスクに対してのみ収集されます。

- -**NetworkRxBytes** 受信したバイト数。このメトリクスは、awsvpcまたは ネットワークモードを使用するタスクのコンテナでのみ使用できます。
- — **NetworkTxBytes** 送信されたバイト数。このメトリクスは、awsvpc またはブリッジネットワークモードを使用するジョブのコンテナでのみ使用できます。
- — **StorageReadBytes** ストレージから読み取られるバイト数。
- **StorageWriteBytes** ストレージに書き込まれるバイト数。

CloudWatch Logs を使用して Amazon EKS ジョブにおける AWS Batch をモニタリングする

Amazon CloudWatch Logs を使用して 1 つの場所ですべてのログファイルをモニタリング、保存、表示できます。CloudWatch Logs を使用すると、複数のソースからのログデータを検索、フィルタリング、分析できます。

CloudWatch Logs を使用して Amazon EKS ジョブにおける AWS Batch をモニタリングするためのプラグインを含む AWS for Fluent Bit のイメージをダウンロードできます。Fluent Bit は Docker と Kubernetes 互換性の両方を兼ね備えたオープンソースのログプロセッサおよびフォワーダーです。Fluentd よりもリソースを消費しないため、Fluent Bit をログルーターとして使用することをお勧めします。詳細については、「Amazon CloudWatch オブザーバビリティ EKS アドオンまたは Helm チャートを使用して 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。

前提条件

- ワーカーノードの AWS Identity and Access Management ポリシーに `CloudWatchAgentServerPolicy` ポリシーをアタッチします。詳細については、前提条件を確認を参照してください。

アドオンをインストールする

Fluent Bit に AWS をインストールして CloudWatch グループを作成する手順については、「Amazon CloudWatch オブザーバビリティ EKS アドオンまたは Helm チャートを使用して 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。

アドオンをインストールするときは、次の追加設定データを指定する必要があります。

- AWS マネジメントコンソール でアドオンをインストールする場合は、設定値で次の許容値を指定する必要があります。

```
{
  "tolerations": [
    {
      "key": "batch.amazonaws.com/batch-node",
      "operator": "Exists"
    }
  ]
}
```

- AWS CLI でアドオンをインストールする場合は、次の引数を追加します。

```
--configuration-values '{"tolerations":[{"key":"batch.amazonaws.com/batch-
node","operator":"Exists"}]}'
```

> **Tip**
>
> AWS Batch ノードで、0.5 CPU と 100 MB のメモリが Fluent Bit を使用することに注意してください。これにより、AWS Batch ジョブに使用できるキャパシティの合計が減少します。ジョブの規模を決定する際には、この点を考慮してください。

AWS Batch リソースのタグ付け

AWS Batch 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。

トピック

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。タグはそれぞれ、1 つのキーとオプションの 1 つの値で構成されており、どちらもお客様側が定義します。

タグを使用すると、AWS リソースを目的、所有者、環境などで分類できます。同じ型のリソースが多い場合に、割り当てたタグに基づいて特定のリソースをすばやく識別できます。たとえば、AWS Batch サービスに一連のタグを定義して、各サービス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。

タグは自動的にはリソースに割り当てられません。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、AWS Batch に関連する意味はなく、完全に文字列として解釈されます。タグの値を空の文字列に設定することはできますが、タグの値を null に設定することはできません。特定のリソースについて既存のタグと同じキーを持つタグを追加した場合、以前の値は新しい値によって上書きされます。

AWS マネジメントコンソール、AWS CLI、および AWS Batch API を使用してタグを操作できます。

AWS Identity and Access Management (IAM) を使用している場合はタグを作成、編集、削除する許可を持つ AWS アカウントのユーザーを制御できます。

リソースのタグ付け

新規または既存のAWS Batch コンピューティング環境、ジョブ、ジョブ定義、ジョブキュー、スケジューリングポリシーのタグをつけることができます。

AWS Batch コンソールを使用している場合、新規リソースには作成時にタグを適用でき、既存のリソースには関連するリソースページの Tags] (タグ) タブを使用していつでもタグを適用できます。

AWS Batch API、AWS CLI、または AWS SDK を使用している場合、新しいリソースには、関連する API アクションの `tags` パラメータを使用してタグを適用でき、既存のリソースには、`TagResource` API アクションを使用してタグを適用できます。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できます。リソースの作成時にタグを適用できない場合、リソースの作成プロセスは失敗します。これにより、作成時にタグ付けするリソースが、指定したタグで作成されるか、まったく作成されないことが確認されます。作成時にリソースにタグを付ける場合、リソースの作成後にカスタムのタグ付けスクリプトを実行する必要はありません。

次の表では、タグ付け可能な AWS Batch リソースと、作成時にタグ付け可能なリソースについて説明します。

AWS Batch リソースのタグのサポート

リソース	タグをサポート	タグの伝播をサポート	作成時のタグ付けをサポート (AWS Batch API、AWS CLI、AWS SDK)
AWS Batch コンピューティング環境	はい	いいえ。 コンピューティング環境タグは、他のリソースには伝播されません。リソースのタグは、CreateComputeEnvironment API オペレーションで渡される computeResources オブジェクト	はい

リソース	タグをサポート	タグの伝播をサポート	作成時のタグ付けをサポート (AWS Batch API、AWS CLI、AWS SDK)
		の tags メンバーで指定されます。	
AWS Batch ジョブ	可能	あり	はい
AWS Batchジョブ定義	はい	なし	はい
AWS Batchジョブキュー	はい	なし	はい
AWS Batchスケジューリングポリシー	はい	なし	はい

タグの制限

タグには以下のような基本制限があります。

- リソースあたりのタグの最大数 - 50 件
- タグキーはリソースごとにそれぞれ一意である必要があります。また、各タグキーに設定できる値は 1 つのみです。
- キーの最大長 - UTF-8 の 128 Unicode 文字
- 値の最大長 - UTF-8 の 256 Unicode 文字
- 複数の AWS サービス間およびリソース間でタグ付けスキーマを使用する場合、他のサービスでも許可される文字に制限が適用されることがあるのでご注意ください。一般的に使用が許可される文字は、UTF-8 で表現できる文字、数字、スペース、および +、-、=、.、_、:、/、@。
- 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。これらの文字列は AWS による使用のため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、リソースごとのタグ数の制限にはカウントされません。

チュートリアル: コンソールを使用してタグを管理する

AWS Batch コンソールを使って、新規または既存のコンピューティング環境、ジョブ、ジョブ定義、ジョブキューに関連するタグを管理することができます。

作成時に個々のリソースにタグを追加する

タグは、AWS Batchコンピューティング環境、ジョブ、ジョブ定義、ジョブキュー、およびスケジューリングポリシーの作成時に追加することができます。

個々のリソースのタグの追加および削除

AWS Batch では、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. AWS Batchコンソールを https://console.aws.amazon.com/batch/ で 開きます。
2. ナビゲーションバーから、使用するリージョンを選択します。
3. ナビゲーションペインでリソースタイプ (例: ジョブキュー) を選択します。
4. 特定のリソースを選択し、Edit tags] (タグの編集) を選択します。
5. 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 — リストの末尾にある空のテキストボックスにキーと値を指定します。
   - タグを削除するには — タグの横にある

     Delete icon
     ボタンを選択します。
6. 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、Edit tags] (タグの編集) を選択して終了します。

CLI または API を使用してタグを管理する

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 AWS Batch API オペレーションを使用します。

AWS Batch リソースのタグのサポート

タスク	API アクション	AWS CLI	AWS Tools for Windows PowerShell
1 つ以上のタグを追加、または上書きします。	TagResource	tag-resource	追加-BATResourceTag
1 つ以上のタグを削除します。	UntagResource	untag-resource	削除-BATResourceTag
リソースのタグの一覧表示	ListTagsForResource	list-tags-for-resource	Get-BATResourceTag

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: 既存のリソースにタグ付けする

次のコマンドでは、既存のリソースにタグ付けします。

```
aws batch tag-resource --resource-arn resource_ARN --tags team=devs
```

例 2: 既存のリソースからタグを削除する

次のコマンドでは、既存のリソースからタグを削除します。

```
aws batch 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 batch list-tags-for-resource --resource-arn resource_ARN
```

一部のリソース作成アクションでは、リソースの作成時にタグを指定できます。以下のアクションでは、作成時のタグ付けがサポートされます。

タスク	API アクション	AWS CLI	AWS Tools for Windows PowerShell
コンピューティング環境の作成	CreateComputeEnvironment	create-compute-environment	New-BATComputeEnvironment
ジョブキューの作成	CreateJobQueue	create-job-queue	New-BATJobQueue
スケジューリングポリシーの作成	CreateSchedulingPolicy	create-scheduling-policy	New-BATSchedulingPolicy
ジョブ定義を登録する	RegisterJobDefinition	register-job-definition	Register-BATJobDefinition
ジョブを送信する	SubmitJob	submit-job	Submit-BATJob
消費型リソースの作成	CreateConsumableResource	create-consumable-resource	Create-BATConsumableResource

AWS Batch のベストプラクティス

AWS Batch を使用して、複雑なアーキテクチャを管理する必要がなく、要求の厳しいさまざまな計算ワークロードを大規模に実行できます。AWS Batch ジョブは、疫学、ゲーミング、機械学習などの幅広いユースケースで使用が可能です。

このトピックでは、AWS Batchの使用時に考慮すべきベストプラクティスと、AWS Batchの使用時にワークロードを実行および最適化する方法に関するガイダンスについて説明します。

トピック

AWS Batch を使用する場合

AWS Batch は、ジョブを大規模かつ低コストで実行し、キューイングサービスとコストが最適化されたスケーリングを提供します。ただし、すべてのワークロードが AWS Batch を使用して実行するのに適しているわけではありません。

- ショートジョブ — ジョブが数秒しか実行されない場合、バッチジョブをスケジュールするためのオーバーヘッドは、ジョブ自体の実行時間よりも長くかかる可能性があります。回避策として、binpack はタスクをまとめてから、それらを AWS Batch で送信します 。次にタスクを繰り返し処理するように、ユーザーの AWS Batch のジョブを設定します。例えば、個々のタスク引数を、Amazon DynamoDB テーブルにステージするか、Amazon S3 バケット内のファイルとしてステージします。各ジョブが3～5分実行されるように、タスクをグループ化することを検討します。ジョブの binpack の後、AWS Batch ジョブ内のタスクグループをループスルーします。
- すぐに実行する必要のあるジョブ — AWS Batchはジョブを迅速に処理できます。ただし、AWS Batch はスケジューラーであり、コストパフォーマンス、ジョブの優先度、スループットを最適化します。AWS Batch は、リクエストの処理に時間がかかる場合があります。数秒以内に応答が必

要な場合は、Amazon ECS または Amazon EKS を使用するサービスベースのアプローチの方がより適しています。

大規模な実行のチェックリスト

5万以上の vCPUs で、大きなワークロードを実行する前に、次のチェックリストを検討してください。

> **Note**
>
> 100 万台以上の vCPUs で大規模なワークロードを実行する予定がある場合、または大規模な実行に関するガイダンスが必要な場合は、AWS の担当チームにお問い合わせください。

- Amazon EC2 クォータの確認 — AWS マネジメントコンソール の Service Quotas パネルで、Amazon EC2 のクォータ (制限ともいいます) を確認してください。必要に応じて、Amazon EC2 インスタンスのピーク数に合わせてクォータ引き上げをリクエストしてください。Amazon EC2 スポットインスタンスと Amazon オンデマンドインスタンスには、別々のクォータがあることを憶えていてください。詳細については、Service Quotas を開始させる を参照してください。
- リージョンごとに Amazon Elastic Block Store の割り当てを確認する — 各インスタンスはオペレーティングシステムに GP2 または GP3 ボリュームを使用します。デフォルトでは、各 AWS リージョン のクォータは 300 TiB です。ただし、各インスタンスはこのクォータの一部として力ウントを使用します。そのため、各リージョンの Amazon Elastic Block Store の割り当てを確認するときは、この点を必ず考慮に入れてください。クォータに達すると、それ以上インスタンスを作成することはできません。詳細については、Amazon Elastic Block Store エンドポイントとクォータ を参照してください。
- ストレージに Amazon S3 を使用する — Amazon S3 はハイスループットを提供し、各アベイラビリティーゾーンのジョブとインスタンスの数に基づいて、プロビジョニングするストレージの量を推測する必要がなくなります。詳細については、設計パターンのベストプラクティス: Simple Storage Service (Amazon S3) のパフォーマンスの最適化 を参照してください。
- 段階的にスケールしてボトルネックを早期に特定する — 100 万個以上の vCPUs で実行されるジョブでは、ボトルネックを早期に特定できるように、低レベルから始めて徐々に増やしていきます。例えば、5万個のvCPUs で実行することから始めます。次にその数を 20 万個の vCPUs に、次に 50 万個の vCPUs にと増やします。つまり、望ましい vCPU 数に達するまで vCPUs 数を徐々に増やし続けます。

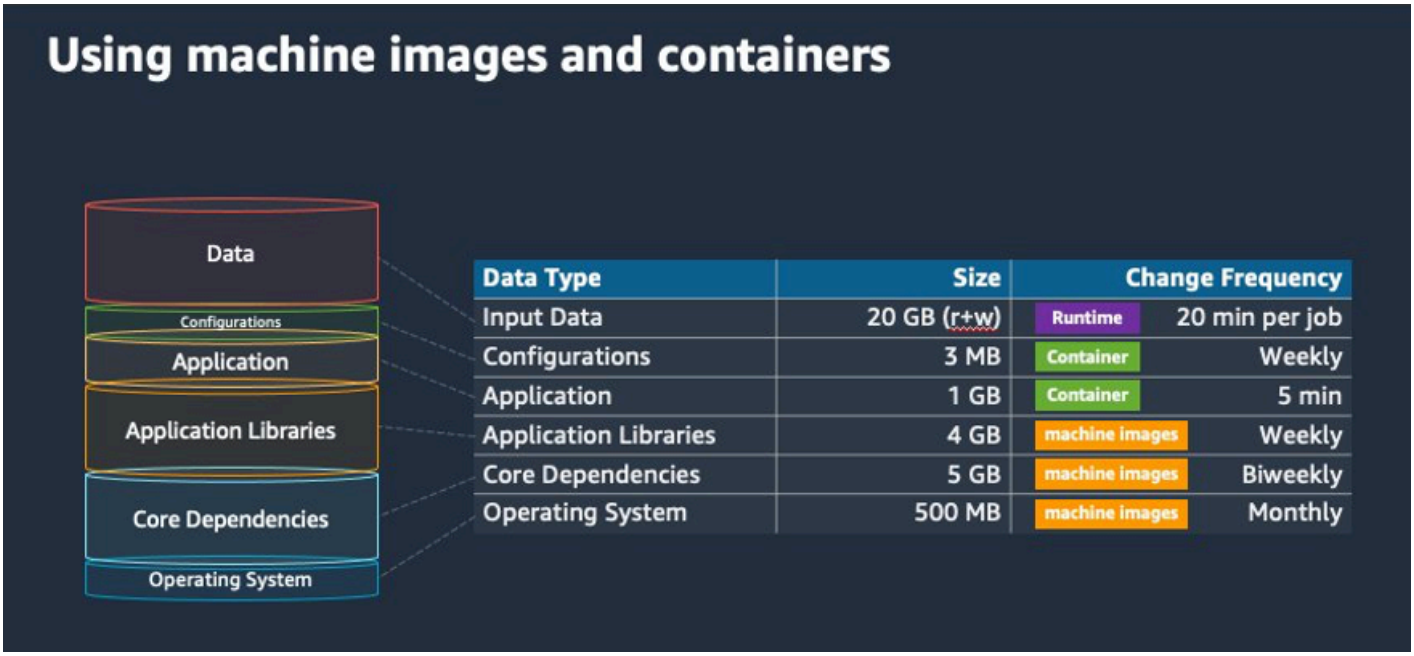
- 監視して潜在的な問題を早期に特定する — 大規模に実行する際に発生する可能性のある中断や問題を避けるために、アプリケーションとアーキテクチャの両方を必ず監視してください。仮 vCPUs を 1,000 から 5,000 にスケーリングしても、中断が発生する可能性があります。Amazon CloudWatch Logs を使用して、ログデータを確認したり、クライアントライブラリを使い CloudWatch 埋め込みメトリックスを使用したりできます。詳細については、CloudWatch Logs エージェントリファレンス と `aws-embedded-metrics` を参照してください。

コンテナと AMI の最適化

最初に実行するジョブセットでは、コンテナのサイズと構造が重要です。コンテナが 4 GBを超える場合は、特に当てはまります。コンテナイメージは、レイヤーで構築されます。レイヤーは、3つの並行スレッドを使用して Docker により並行して取得されます。`max-concurrent-downloads` パラメータを使用して、同時にスレッド数を増やすことができます。詳細については、Dockerドキュメント を参照してください。

より大きなコンテナを使用することもできますが、起動時間を短縮するために、コンテナの構造とサイズを最適化することをお勧めします。

- コンテナが小さいほどフェッチが速い — コンテナが小さいほど、アプリケーションの起動時間の高速化につながります。コンテナのサイズを小さくするには、あまり更新されないライブラリやファイルを Amazon マシンイメージ (AMI) にオフロードします。またバインドマウントを使用して、コンテナへのアクセスを可能にすることもできます。詳細については、バインドマウント を参照してください。
- サイズが同じレイヤーを作成し、大きなレイヤーを分割する — 各レイヤーは 1つのスレッドで検索されます。ですから、大きなレイヤーは、ジョブの起動時間に大きく影響する可能性があります。コンテナのサイズを大きくすることと、起動時間を短縮することの望ましいトレードオフとして、レイヤーの最大サイズを 2 GB にすることをお勧めします。`docker history your_image_id` コマンドを実行して、コンテナイメージの構造とレイヤーサイズを確認できます。詳細については、Docker ドキュメント を参照してください。
- Amazon Elastic Container Registry をコンテナリポジトリとして使用する — 何千ものジョブをパラレル実行すると、自己管理型リポジトリに障害が発生したり、スロットルされたスループットが抑制されたりする可能性があります。Amazon ECR は大規模に動作し、最大 100 万個以上の vCPUs を使用するワークロードを処理できます。



適切なコンピューティング環境リソースを選択する

AWS Fargate は、Amazon EC2 よりも初期設定や設定が少なくて済み、特に、初めて使用する場合は使いやすいでしょう。Fargate を使用すると、サーバーを管理したり、容量計画に対処したり、セキュリティのためにコンテナワークロードを分離したりする必要がなくなります。

以下の要件がある場合は、Fargate インスタンスを使用することをお勧めします:

- ジョブは、すぐに、具体的には 30 秒未満で開始する必要があります。
- ジョブの要件は、16 vCPUs 以下、GPU なし、120 GiB 以下のメモリです。

詳細については、Fargateをいつ使うべきか を参照してください。

以下の要件がある場合は、Amazon EC2 インスタンスを使用することをお勧めします。

- インスタンスの選択をより細かく制御するか、または特定のインスタンスタイプを使用する必要があります。
- ジョブには、GPU、より多くのメモリ、カスタム AMI、Amazon Elastic Fabric Adapter など、AWS Fargate が提供できないリソースが必要です。
- ハイレベルのスループット、または同時実行性が必要です。
- AMI、Amazon EC2 起動テンプレート、または特別な Linux パラメータへのアクセスをカスタマイズする必要があります。

Amazon EC2 を使用すると、特定の要件に対してワークロードをより細かく調整し、必要に応じて大規模に実行できます。

Amazon EC2 オンデマンドまたは Amazon EC2 スポット

ほとんどの AWS Batch のお客様は、Amazon EC2 スポットインスタンスを使用しています。これは、オンデマンドインスタンスよりもコストが低いからです。ただし、ワークロードが数時間にわたって実行され、中断できない場合は、オンデマンドインスタンスの方が適している可能性があります。いつでも最初にスポットインスタンスを試して、必要に応じてオンデマンドに切り替えることができます。

以下の要件と期待がある場合は、Amazon EC2 オンデマンドインスタンスを使用してください:

- ジョブの実行時間が 1 時間以上の場合、ワークロードの中断は許されません。
- ワークロード全体に対して厳格な SLO (サービスレベル目標) が設定されており、計算時間を増やすことはできません。
- 必要なインスタンスは、おそらく中断される可能性があります。

以下の要件と期待がある場合には、Amazon EC2 スポットインスタンスを使用してください:

- ジョブの実行時間は、通常 30 分以下です。
- ワークロードの一部として、潜在的な中断やジョブの再スケジュールを許容できます。詳細については、スポットインスタンスアドバイザー を参照してください。
- 実行時間の長いジョブが中断された場合、チェックポイントから再開できます。

最初にスポットインスタンスで申請し、次にオンデマンドインスタンスをフォールバックオプションとして使用することで、両方の購入モデルを混在させることができます。例えば、Amazon EC2 スポットインスタンスで実行されているコンピューティング環境に接続されているキューにジョブを送信します。ジョブが中断された場合は、Amazon EventBridge からのイベントをキャッチし、スポットインスタンスの再利用と関連付けます。次に、AWS Lambda 関数または AWS Step Functions を使用して、ジョブをオンデマンドキューに再送信します。詳細については チュートリアル: 失敗したジョブイベントに Amazon シンプル通知サービスアラートを送信する、Amazon EC2 スポットインスタンスの中断を処理するためのベストプラクティス と Step Functions AWS Batch による管理 を参照してください。

> **Important**
>
> Amazon EC2 スポットインスタンスプールの可用性を維持し、中断率を下げるには、さまざまなインスタンスタイプ、サイズ、アベイラビリティーゾーンをオンデマンドコンピューティング環境のために使用します。

AWS Batch に関する Amazon EC2 スポット利用のベストプラクティスを利用する

Amazon Elastic Compute Cloud (EC2) スポットインスタンスを選択すると、ワークフローを最適化してコスト低減化（場合によっては大幅な節約実現）の可能性が高くなります。詳しくは、Amazon S3 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。

ワークフローを最適化してコストを低減するには、以下の AWS Batch に関する Amazon EC2 スポットベストプラクティスを検討してください:

- **`SPOT_CAPACITY_OPTIMIZED`** 割り当て戦略を選択する — AWS Batch は、最も深い Amazon EC2 スポット容量プールから Amazon EC2 インスタンスを選択します。中断が心配な場合は、この方法が適切な選択です。詳細については、AWS Batch のインスタンスタイプの配分戦略 を参照してください。
- インスタンスタイプの多様化 — インスタンスタイプを多様化するには、互換性のあるサイズとファミリーを検討し、AWS Batch を価格や在庫状況に基づいて選択してください。例えば、`c5.24xlarge` を、`c5.12xlarge`、`c5a`、`c5n`、`c5d`、`m5`、`m5d` のファミリーの代替として検討してください。詳細については、インスタンスタイプとアベイラビリティーゾーンへの柔軟な対応 を参照してください。
- ジョブの実行時間またはチェックポイントの短縮 — Amazon EC2 スポットインスタンスを使用する場合は、中断を避けるために、1 時間以上かかるジョブを実行しないことをお勧めします。ジョブを 30 分以下のより小さな部分に分割したり、チェックポイントを設定したりすることで、中断の可能性を大幅に減らすことができます。
- 自動リトライを使用する — AWS Batch のジョブの中断を避けるには、ジョブには自動リトライを設定します。Batch ジョブは、ゼロ以外の終了コードが返された、サービスエラーが発生した、またはインスタンスが再利用されたなどの理由で中断される可能性があります。最大 10 個まで自動再試行を設定できます。まず、少なくとも 1～3 回の自動再試行を設定することをお勧めします。Amazon EC2 スポット中断の追跡については、スポット中断ダッシュボード を参照してください。

AWS Batch について、再試行パラメータを設定すると、ジョブはジョブキューの先頭に置かれるからです。つまり、ジョブが優先されます。ジョブ定義を作成したり、または AWS CLI でジョブを送信したりするときに、再試行方法を設定できます。詳細については、ジョブの送信 を参照してください。

```
$ aws batch submit-job --job-name MyJob \
    --job-queue MyJQ \
    --job-definition MyJD \
    --retry-strategy attempts=2
```

- カスタムリトライを使用する — 特定のアプリケーション終了コードまたはインスタンスの再利用に合わせて、ジョブの再試行方法を設定できます。次の例では、ホストが原因で障害が発生した場合、ジョブは最大 5 回まで再試行できます。ただし別の理由によりジョブが失敗すると、ジョブは終了し、ステータスは FAILED に設定されます。

```
"retryStrategy": {
    "attempts": 5,
    "evaluateOnExit":
    [{
        "onStatusReason" :"Host EC2*",
        "action": "RETRY"
    },{
      "onReason" : "*",
        "action": "EXIT"
    }]
}
```

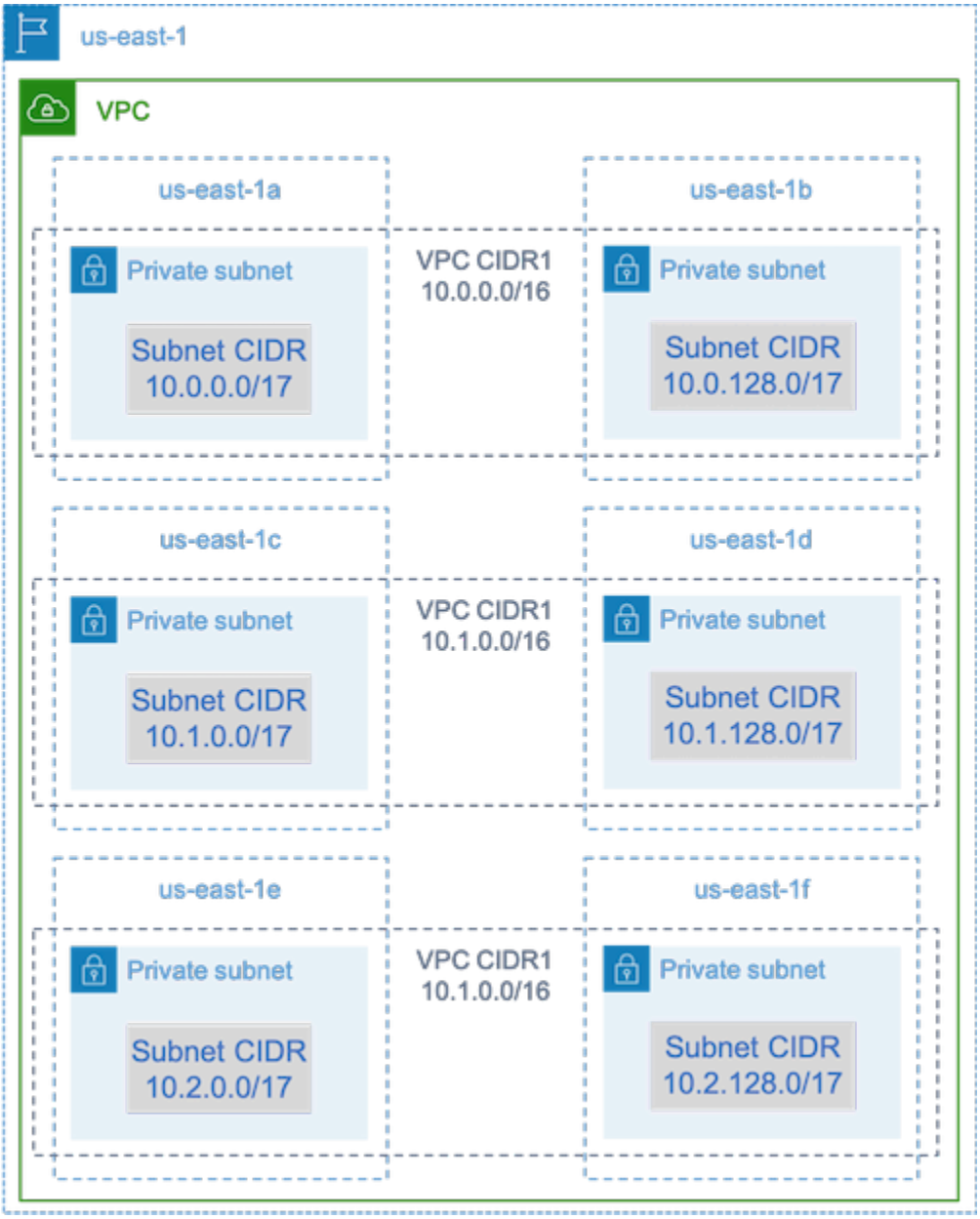
- スポット中断ダッシュボードを使用する — スポット中断ダッシュボードを使用して、スポット中断を追跡できます。このアプリケーションは、再利用された Amazon EC2 スポットインスタンスと、そのスポットインスタンスが属するアベイラビリティーゾーンに関するメトリクスを提供します。詳細については、スポット中断ダッシュボード を参照してください

一般的なエラーとトラブルシューティング

AWS Batch におけるエラーは、多くの場合、アプリケーションレベルで発生したり、ユーザーの特定のジョブ要件を満たさないインスタンス構成が原因で発生したりします。その他の問題としては、ジョブが RUNNABLE のステータス内で停止したり、INVALID の状態でコンピューティング環境が停止したりすることが挙げられます。RUNNABLE のステータスのまま動かなくなるジョブのトラブルシューティングの詳細については、RUNNABLE 状態でジョブが止まる を参照してください。ある

INVALID の状態にあるコンピューティング環境のトラブルシューティングについては、INVALID コンピューティング環境 を参照してください。

- Amazon EC2 スポット vCPU クォータの確認 — 現在のサービスクォータがジョブの要件を満たしていることを確認します。例えば、現在のサービスクォータが 256 vCPUs で、ジョブに 10,000 vCPUs が必要だとします。その場合、サービスクォータはジョブの要件を満たしていません。詳細とトラブルシューティングの手順の詳細については、Amazon EC2 Service Quotas と Amazon EC2 リソースのService Quotaを増やすには? を参照してください。
- アプリケーション実行前のジョブが失敗する — ジョブの一部には、DockerTimeoutError のエラーまたは CannotPullContainerError のエラーが原因で失敗するものがあります。トラブルシューティング情報については、AWS Batch の"DockerTimeoutError"エラーを解決する方法を参照してください。
- 不十分な IP アドレス — VPC とサブネットの IP アドレスの数によって、作成できるインスタンスの数が制限されることがあります。Classless Inter-Domain Routings (CIDR) を使用して、ワークロード実行に必要な IP アドレスよりも多くの IP アドレスを指定します。必要な場合、大きなアドレス空間を持つ専用 VPC を構築することもできます。例えば、10.x.0.0/16 において複数の CIDR を持つ VPC を作成し、すべてのアベイラビリティーゾーンに 10.x.y.0/17 の CIDR を使い サブネットを作成できます。この例では、x は 1～4 で、y は 0 または 128 です。この設定では、すべてのサブネットに 36,000 個の IP アドレスが割り当てられます。



- インスタンスが Amazon EC2 に登録されていることを確認する — Amazon EC2 コンソールにおいてインスタンスが表示されるが、Amazon ECS クラスターに Amazon Elastic Container Service コンテナインスタンスが何も表示されない場合は、Amazon ECS エージェントが Amazon マシンイメージ (AMI) にインストールされていない可能性があります。AMI 内の Amazon ECS エージェント、AMI Amazon EC2 データ、または起動テンプレートも、正しく設定されていない可能性があります。根本原因を探し出すには、別の Amazon EC2 インスタンスを作成するか、または SSH を使用して既存のインスタンスに接続します。詳細については、Amazon ECS コンテナエージェントの設定、Amazon ECS ログファイルの場所、および コンピューティングリソースの AMI を参照してください。

- AWS ダッシュボードを確認する — AWS ダッシュボードを確認して、予想されるジョブの状態とコンピューティング環境が予想どおりにスケーリングされていることを確認します。CloudWatch でジョブログを確認することもできます。
- インスタンスが作成されたことを確認する — インスタンスが作成されたら、コンピューティング環境が想定どおりにスケーリングされたことを意味します。インスタンスが作成されていない場合は、コンピューティング環境内の関連するサブネットを探して変更します。詳細については、Auto Scaling グループのスケーリングアクティビティを検証する を参照してください。

  また、インスタンスが関連するジョブ要件を満たしていることを確認するようお勧めします。例えば、ジョブに 1 TiB のメモリが必要でも、コンピューティング環境では 192 GB のメモリに制限されている C5 インスタンスタイプが使用されているとします。
- AWS Batch により、インスタンスがリクエストされていることを確認する — Auto Scaling グループの履歴をチェックして、インスタンスが AWS Batch によってリクエストされていることを確認します。これは、Amazon EC2 がどのようにインスタンスを取得しようとしているかを示しています。Amazon EC2 スポットが特定のアベイラビリティーゾーンのインスタンスを取得できない、というエラーが表示される場合は、アベイラビリティーゾーンが特定のインスタンスファミリーを提供していないことが原因である可能性があります。
- インスタンスが Amazon ECS に登録されていることを確認する — Amazon EC2 コンソールにインスタンスが表示されるが、Amazon ECS クラスターに Amazon ECS コンテナインスタンスが表示されない場合は、Amazon ECS エージェントが Amazon マシンイメージ (AMI) にインストールされていない可能性があります。さらに、Amazon ECS エージェント、AMI の Amazon EC2 データ、または起動テンプレートが正しく設定されていない可能性があります。根本原因を探し出すには、別のAmazon EC2 インスタンスを作成するか、または SSH を使用して既存のインスタンスに接続します。詳細については、CloudWatch エージェント構成ファイル: ログセクション、Amazon ECS Log File Locations、および コンピューティングリソースの AMI を参照してください。
- サポートチケットを開く — トラブルシューティングを行っても問題が解決せず、サポートプランがある場合は、サポートチケットを開いてください。サポートチケットには、問題、ワークロードの仕様、構成、およびテスト結果に関する情報を必ず含めてください。詳細については、サポートプランの比較 を参照してください。
- AWS Batch および HPC フォーラムを確認する — 詳細については、AWS Batch および HPC フォーラムを参照してください。
- AWS Batch ランタイムモニタリングダッシュボードを確認する — このダッシュボードは、サーバーレスアーキテクチャを使用して Amazon ECS、AWS Batch、Amazon EC2 からのイベントをキャプチャし、ジョブとインスタンスに関する洞察を提供します。詳細については、AWS Batch ランタイム監視ダッシュボードソリューション を参照してください。

トラブルシューティング AWS Batch

コンピューティング環境、ジョブキュー、ジョブ定義、またはジョブに関する問題のトラブルシューティングが必要となる場合があります。この章では、 AWS Batch 環境でこのような問題をトラブルシューティングして解決する方法について説明します。

AWS Batch は IAM ポリシー、ロール、アクセス許可を使用し、Amazon EC2、Amazon ECS AWS Fargate、Amazon Elastic Kubernetes Service インフラストラクチャで実行されます。これらのサービスに関連する問題のトラブルシューティングを行うには、以下を参照してください。

- IAM ユーザーガイドのIAM のトラブルシューティング
- Amazon Elastic Container Service 開発者ガイドのAmazon ECS のトラブルシューティング
- Amazon EKS ユーザーガイドの Amazon EKS のトラブルシューティング
- 「Amazon EC2 ユーザーガイド」の「Amazon EC2 インスタンスの問題をトラブルシューティングする」

目次

AWS Batch

以下のトピックを確認して、 AWS Batchを使用する際に発生する可能性のある一般的な問題に対するレビュープロセスと考えられる解決策を理解してください。

トピック

インスタンスファミリーの自動更新を受信するための最適なインスタンスタイプの設定

> **Note**
>
> 2025 年 11 月 1 日以降、optimal の動作は default_x86_64 と一致するように変更されます。変更中、インスタンスファミリーは新しい世代に更新される可能性があります。アップグレードを実行するためにアクションを実行する必要はありません。

AWS Batch は、ジョブキューの需要optimalに合わせて、 の instanceTypes で 1 つのオプションをサポートしました。default_x86_64 と default_arm64 の 2 つの新しいインスタンスタイプのオプションが導入されました。インスタンスタイプを選択しない場合、default_x86_64 が使用されます。これらの新しいオプションでは、ジョブキューの要件に基づいて、さまざまなファミリーや世代にわたって費用対効果の高いインスタンスタイプが自動的に選択されるため、ワークロードをすばやく実行できます。

で新しいインスタンスタイプの十分な容量が利用可能になると AWS リージョン、対応するデフォルトプールは新しいインスタンスタイプで自動的に更新されます。既存の optimal オプションは引き続きサポートされ、今後更新されたインスタンスを提供するために基盤となるデフォルトプールでサポートされるため、廃止されません。'optimal を使用している場合、ユーザー側でアクションは必要ありません。

ただし、ENABLED および VALID コンピューティング環境 (CEs) のみが新しいインスタンスタイプで自動的に更新されることに注意してください。DISABLED または INVALID CE がある場合は、再有効化されて VALID状態に設定されると、更新を受け取ります。

INVALID コンピューティング環境

マネージド型コンピューティング環境を誤って設定した可能性があります。設定した場合、コンピューティング環境は INVALID の状態になり、配置するジョブを受け入れられなくなります。以下のセクションでは、考えられる原因と、その原因に基づいたトラブルシューティング方法について説明します。

> **Important**
>
> AWS Batch は、Amazon EC2 起動テンプレート、Amazon EC2 Auto Scaling グループ、Amazon EC2 スポットフリート、Amazon ECS クラスターなど、ユーザーに代わってア

カウント内で複数の AWS リソースを作成および管理します。これらのマネージドリソースは、最適な AWS Batch オペレーションを確保するために特別に設定されています。これらのバッチマネージドリソースを手動で変更すると、 AWS Batch ドキュメントに明示的に記載されていない限り、予期しない動作が発生し、INVALID コンピューティング環境、最適ではないインスタンススケーリング動作、ワークロード処理の遅延、予期しないコストが発生する可能性があります。これらの手動変更を AWS Batch サービスで決定的にサポートすることはできません。コンピューティング環境を管理するには、サポートされている Batch API または Batch コンソールを必ず使用してください。

正しくないロール名または ARN

コンピューティング環境が INVALID状態になる最も一般的な原因は、 AWS Batch サービスロールまたは Amazon EC2 スポットフリートロールの名前または Amazon リソースネーム (ARN) が正しくないことです。これは、 AWS CLI または AWS SDKs。でコンピューティング環境を作成する AWS Batch と AWS マネジメントコンソール、正しいサービスまたはスポットフリートロールを選択できます。ただし、名前または ARN を手動で入力し、またはそれを間違って入力したとします。そうすると、生成されるコンピューティング環境も INVALID になります。

しかし、IAM リソースの名前または ARN を AWS CLI コマンドまたは SDK コードに手動で入力すると仮定します。この場合、 AWS Batch は文字列を検証できません。代わりに、 AWS Batch は不正な値を受け入れ、環境作成を試みる必要があります。が環境の作成に AWS Batch 失敗すると、環境は INVALID状態に移行し、次のエラーが表示されます。

無効なサービスロールの場合:

```
CLIENT_ERROR - Not authorized to perform sts:AssumeRole (Service:
AWSSecurityTokenService; Status Code: 403; Error Code: AccessDenied;
Request ID: dc0e2d28-2e99-11e7-b372-7fcc6fb65fe7)
```

無効なスポットフリートロールの場合:

```
CLIENT_ERROR - Parameter: SpotFleetRequestConfig.IamFleetRole
is invalid. (Service: AmazonEC2; Status Code: 400; Error Code:
InvalidSpotFleetRequestConfig; Request ID: 331205f0-5ae3-4cea-
bac4-897769639f8d) Parameter: SpotFleetRequestConfig.IamFleetRole is
invalid
```

この問題の一般的な原因の 1 つには、次のシナリオがあります。 AWS CLI または AWS SDKs を使用する場合にのみ、完全な Amazon リソースネーム (ARN) ではなく、IAM ロールの名前を指定しま

す。ロールを作成した方法によって、ARN に aws-service-role パスプレフィックスが含まれることがあるからです。例えば、 AWS Batch サービスロールを、AWS Batchのサービスにリンクされたロールの使用 の手順を使用して手動で作成すると、サービスロール ARN は次のようになります。

```
arn:aws:iam::123456789012:role/AWSBatchServiceRole
```

ただし、コンソールの最初の実行ウィザードの一環としてサービスロールを作成した場合、サービスロール ARN は次のようになります。

```
arn:aws:iam::123456789012:role/aws-service-role/AWSBatchServiceRole
```

この問題は、 AWS Batch サービスレベルポリシー (AWSBatchServiceRole) をサービス以外のロールにアタッチする場合にも発生する可能性があります。例えば、このシナリオでは、次のようなエラーメッセージが表示される場合があります。

```
CLIENT_ERROR - User: arn:aws:sts::account_number:assumed-role/batch-replacement-role/
aws-batch is not
   authorized to perform: action on resource ...
```

問題を解決するには、次のいずれかを実行します。

- AWS Batch コンピューティング環境を作成するときは、サービスロールに空の文字列を使用します。
- 以下の形式でサービスロールを指定します: arn:aws:iam::*account_number*:role/aws-service-role/batch.amazonaws.com/AWSServiceRoleForBatch。

AWS CLI または AWS SDKs を使用するときに IAM ロールの名前のみを指定する場合、 は ARN がaws-service-roleパスプレフィックスを使用しないことを AWS Batch 前提としています。このため、コンピューティング環境を作成するときには、IAM ロールに完全 ARN を指定することが推奨されます。

この設定の誤りがあるコンピューティング環境を修復するには、INVALID のコンピューティング環境を修復する を参照してください。

INVALID のコンピューティング環境を修復する

コンピューティング環境が INVALID の状態にある場合、無効なパラメータを修復して更新する必要があります。正しくないロール名または ARN については、正しいサービスロールを使ってコンピューティング環境を更新できます。

誤って設定されたコンピューティング環境を修復するには

1. https://console.aws.amazon.com/batch/ で AWS Batch コンソールを開きます。
2. ナビゲーションバーから、 AWS リージョン 使用する を選択します。
3. ナビゲーションペインで、[Compute environments] (コンピューティング環境) を選択します。
4. [Compute environments] (コンピューティング環境) ページで、編集するコンピューティング環境の横にあるラジオボタンを選択し、[Edit] (編集) を選択します。
5. [Update compute environment] (コンピューティング環境の更新) ページの [Service role] (サービスロール) で、コンピューティング環境に使用する IAM ロールを選択します。 AWS Batch コンソールには、コンピューティング環境と正しい信頼関係があるロールのみが表示されます。

   > **Tip**
   >
   > サービスにリンクされたロールを作成する方法に関する説明については、「での ロールの使用 AWS Batch」を参照してください。

6. [Save] (保存) を選択して、コンピューティング環境を更新します。

RUNNABLE 状態でジョブが止まる

コンピューティング環境にコンピューティングリソースがあるのに、ジョブが RUNNABLE の状態よりも先に進まないとします。その場合、ジョブがコンピューティングリソースに配置されるのが何かの原因で妨げられ、ジョブキューがブロックされている可能性があります。ここでは、ジョブが順番を待っているのか、スタックしてキューをブロックしているのか確認する方法を説明します。

が先頭にRUNNABLEジョブがあり、キューをブロックしていること AWS Batch を検出すると、Amazon CloudWatch Events から理由を含むジョブキューのブロックイベントイベントを受け取ります。同じ理由が、`ListJobs` と `DescribeJobs` の API コールの一部として `statusReason` フィールドに挿入されます。

必要に応じて、`CreateJobQueue` と `UpdateJobQueue` の API アクションを使用して `jobStateTimeLimitActions` パラメータを設定できます。

> **Note**
>
> 現在、`jobStateLimitActions.action` で使用できるアクションはジョブのキャンセルのみです。

jobStateTimeLimitActions パラメータは、特定の状態のジョブに対して が AWS Batch 実行する一連のアクションを指定するために使用されます。maxTimeSeconds フィールドを使用して時間のしきい値を秒単位で設定できます。

ジョブが定義された RUNNABLEの状態にある場合statusReason、 maxTimeSeconds は経過後に指定されたアクションを実行 AWS Batch します。

例えば、jobStateTimeLimitActions パラメータを使用すると、ジョブが十分な容量を利用できるようになるまで、RUNNABLE 状態で最大 4 時間待機するように設定できます。これを行うには、ジョブがキャンセルされて次のジョブがジョブキューの先頭に進む前に、statusReason を CAPACITY:INSUFFICIENT_INSTANCE_CAPACITY に、maxTimeSeconds を 144000 に設定します。

ジョブキューがブロックされていることを検出する AWS Batch と、 が に提供する理由は次のとおりです。このリストでは ListJobs と DescribeJobs の API アクションから返されるメッセージを示しています。また、これらは jobStateLimitActions.statusReason パラメータに定義できる値と同じです。

1. 理由: 接続されているすべてのコンピューティング環境で、容量不足のエラーが出ています。リクエストされると、 は容量不足エラーが発生した Amazon EC2 インスタンス AWS Batch を検出します。ジョブを手動でキャンセルすると、後続のジョブはキューの先頭に移動できますが、サービスロールの問題は解決されず、次のジョブもブロックされる可能性があります。この問題は手動で調査して解決することをお勧めします。
   - ジョブがスタックしているときの **statusReason** メッセージ:
     `CAPACITY:INSUFFICIENT_INSTANCE_CAPACITY - Service cannot fulfill the capacity requested for instance type [instanceTypeName]`
   - **jobStateTimeLimitActions** に使用される **reason**:
     `CAPACITY:INSUFFICIENT_INSTANCE_CAPACITY`
   - ジョブがキャンセルされた後の **statusReason** メッセージ:
     `Canceled by JobStateTimeLimit action due to reason: CAPACITY:INSUFFICIENT_INSTANCE_CAPACITY`

   メモ:

   a. AWS Batch サービスロールには、この検出が機能するためのautoscaling:DescribeScalingActivitiesアクセス許可が必要です。AWS Batchのサービスにリンクされたロールの使用 のサービスにリンクされたロール (SLR) または AWS マネージドポリシー: AWSBatchServiceRole ポリシー の管理ポリシーを使用する場合、アクセス許可ポリシーが更新されるためアクションは必要ありません。

b. SLR または管理ポリシーを使用する場合は、autoscaling:DescribeScalingActivities と ec2:DescribeSpotFleetRequestHistory のアクセス許可を追加して、RUNNABLE のときにブロックされたジョブキューイベントと更新されたジョブステータスを受信できるようにする必要があります。さらに、 AWS Batch では、ジョブキューで設定されていても jobStateTimeLimitActions パラメータを使用して cancellation アクションを実行するため、これらのアクセス許可が必要です。

c. マルチノード並列 (MNP) ジョブの場合、アタッチされた高優先度の Amazon EC2 コンピューティング環境で insufficient capacity エラーが発生すると、優先度の低いコンピューティング環境でこのエラーが発生してもキューがブロックされます。

2. 理由: すべてのコンピューティング環境に、ジョブ要件よりも小さい maxvCpus パラメータがあります。手動でジョブをキャンセルするか、statusReason に jobStateTimeLimitActions パラメータを設定してキャンセルすると、後続のジョブがキューの先頭に移動できるようになります。必要に応じて、プライマリコンピューティング環境の maxvCpus パラメータを増やして、ブロックされたジョブのニーズを満たすことができます。

   - ジョブがスタックしているときの **statusReason** メッセージ:
     `MISCONFIGURATION:COMPUTE_ENVIRONMENT_MAX_RESOURCE - CE(s) associated with the job queue cannot meet the CPU requirement of the job.`
   - **jobStateTimeLimitActions** に使用される **reason**:
     `MISCONFIGURATION:COMPUTE_ENVIRONMENT_MAX_RESOURCE`
   - ジョブがキャンセルされた後の **statusReason** メッセージ:
     `Canceled by JobStateTimeLimit action due to reason: MISCONFIGURATION:COMPUTE_ENVIRONMENT_MAX_RESOURCE`

3. 理由: ジョブの要件を満たすインスタンスが、どのコンピューティング環境にもありません。ジョブがリソースをリクエストすると、 はアタッチされたコンピューティング環境が受信ジョブに対応できないことを AWS Batch 検出します。手動でジョブをキャンセルするか、statusReason に jobStateTimeLimitActions パラメータを設定してキャンセルすると、後続のジョブがキューの先頭に移動できるようになります。必要に応じて、コンピューティング環境で許可されるインスタンスタイプを再定義して、必要なジョブリソースを追加できます。

   - ジョブがスタックしているときの **statusReason** メッセージ:
     `MISCONFIGURATION:JOB_RESOURCE_REQUIREMENT - The job resource requirement (vCPU/memory/GPU) is higher than that can be met by the CE(s) attached to the job queue.`
   - **jobStateTimeLimitActions** に使用される **reason**:
     `MISCONFIGURATION:JOB_RESOURCE_REQUIREMENT`

- ジョブがキャンセルされた後の **statusReason** メッセージ:
  `Canceled by JobStateTimeLimit action due to reason: MISCONFIGURATION:JOB_RESOURCE_REQUIREMENT`

4. 理由: すべてのコンピューティング環境にサービスロールの問題があります。これを解決するには、サービスロールのアクセス許可を AWS の 管理ポリシー AWS Batch と比較して、ギャップがあれば対処します。注意: このエラーを解決するために、jobStateTimeLimitActions パラメータを使用してプログラム可能なアクションを設定することはできません。

   同様のエラーを避けるため、AWS Batchのサービスにリンクされたロールの使用 を使用するのがベストプラクティスです。

   手動でジョブをキャンセルするか、statusReason に jobStateTimeLimitActions パラメータを設定してキャンセルすると、後続のジョブがキューの先頭に移動できるようになります。サービスロールの問題をすべて解決しないと、次のジョブもブロックされる可能性があります。この問題は手動で調査して解決することをお勧めします。

   - ジョブがスタックしているときの **statusReason** メッセージ:
     `MISCONFIGURATION:SERVICE_ROLE_PERMISSIONS - Batch service role has a permission issue.`

5. 理由: コンピューティング環境にサポートされていないインスタンスタイプの設定があります。これは、選択したアベイラビリティーゾーンでインスタンスタイプが使用できない場合、または起動テンプレートまたは起動設定に指定されたインスタンスタイプと互換性のない設定が含まれている場合に発生する可能性があります。これを解決するには、指定した AWS リージョン およびアベイラビリティーゾーンでインスタンスタイプがサポートされていること、起動テンプレート設定がインスタンスタイプと互換性があることを確認し、新しい世代のインスタンスタイプへの更新を検討してください。サポートされているインスタンスタイプの検索の詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EC2 インスタンスタイプの検索」を参照してください。

   - ジョブがスタックしているときの **statusReason** メッセージ:
     `MISCONFIGURATION:EC2_INSTANCE_CONFIGURATION_UNSUPPORTED - Your compute environment associated with this job queue has an unsupported instance type configuration.`

6. 理由: すべてのコンピューティング環境が無効です。詳細については、「INVALID コンピューティング環境」を参照してください。注意: このエラーを解決するために、jobStateTimeLimitActions パラメータを使用してプログラム可能なアクションを設定することはできません。

- ジョブがスタックしているときの **statusReason** メッセージ: `ACTION_REQUIRED - CE(s) associated with the job queue are invalid.`

7. 理由: AWS Batch がブロックされたキューを検出しましたが、理由を特定できません。注意: このエラーを解決するために、`jobStateTimeLimitActions` パラメータを使用してプログラム可能なアクションを設定することはできません。トラブルシューティングの詳細については、re:Post の「AWSAWS Batch ジョブが [RUNNABLE] ステータスでスタックしているのはなぜですか?」を参照してください。
   - ジョブがスタックしているときの **statusReason** メッセージ: `UNDETERMINED - Batch job is blocked, root cause is undetermined.`

CloudWatch Events からイベントを受信しなかった場合、または原因不明のイベントを受信した場合、一般に以下のようないくつかの原因が考えられます。

`awslogs` のログドライバーが、コンピューティングリソースで設定されていない

AWS Batch ジョブはログ情報を CloudWatch Logs に送信します。この機能を有効にするには、`awslogs`のログドライバーを使用するようにコンピューティングリソースを設定する必要があります。コンピューティングリソース AMI を、Amazon ECSに最適化されたAMI (またはAmazon Linux) に基づいて作成すると仮定します。次に、このドライバーはデフォルトにより`ecs-init`のパッケージに登録されます。ここで、別のベース AMI を使用すると仮定します。別の基本 AMI を使用する場合、Amazon ECS コンテナエージェントが開始するときに、`awslogs` ログドライバーが、 `ECS_AVAILABLE_LOGGING_DRIVERS` 環境変数で使用可能なログドライバーとして指定されていることを検証する必要があります。詳細については、コンピューティングリソースの AMI 仕様 およびチュートリアル: コンピューティングリソース AMI を作成するを参照してください。

リソースが不十分である

コンピューティングリソースが割り当てることができるリソースを超えた CPU、またはメモリリソースがジョブ定義に指定されている場合、ジョブは配置されません。例えば、ジョブが 4 GiB のメモリを指定し、コンピューティングリソースで使用できるのがそれ以下の場合を考えます。そうなると、そのジョブをそれらのコンピューティングリソースに配置できない場合があります。この場合、ジョブ定義に指定するメモリを減らすか、あるいは環境のコンピューティングリソースを追加する必要があります。一部のメモリは、Amazon ECS コンテナエージェントやその他の重要なシステムプロセス用に予約されています。詳細については、コンピューティングリソースメモリの管理を参照してください。

コンピューティングリソースのインターネットアクセスがない

コンピューティングリソースには、Amazon ECS サービスエンドポイントと通信するために外部ネットワークアクセスが必要です。これは、インターフェイス VPC エンドポイントを介して、またはパブリック IP アドレスを持つコンピュートリソースを通じて可能になります。

インターフェイス VPC エンドポイントの詳細については、Amazon Elastic Container Service 開発者ガイドのAmazon ECS インターフェイス VPC エンドポイント (AWS PrivateLink) を参照してください。

インターフェイス VPC エンドポイントが設定されておらず、コンピューティングリソースがパブリック IP アドレスを持たない場合は、ネットワークアドレス変換 (NAT) を使用してこのアクセスを提供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 NAT ゲートウェイを参照してください。詳細については、the section called “VPC を作成する”を参照してください。

Amazon EC2 インスタンス制限に達している

アカウントが で起動できる Amazon EC2 インスタンスの数 AWS リージョン は、EC2 インスタンスのクォータによって決まります。特定のインスタンスタイプには、インスタンスタイプごとの制限もあります。ユーザーのアカウントの Amazon EC2 インスタンスクォータの詳細 (制限の引き上げをリクエストする方法を含む) については、「Amazon EC2 ユーザーガイド」の「Amazon EC2 の Service Quotas」を参照してください。

Amazon ECS コンテナエージェントが存在しない

AWS Batch ジョブを実行するには、Amazon ECS コンテナエージェントは Amazon マシンイメージ (AMI) にインストールされる必要があります。Amazon ECS コンテナエージェントが Amazon ECSに最適化 AMI にデフォルトでインストールされます。Amazon ECS コンテナエージェントの詳細については、「Amazon Elastic Container Service デベロッパーガイド」の「Amazon ECS コンテナエージェント」を参照してください。

詳細については、「re:Post」の「Why is my AWS Batch job stuck in RUNNABLE status?」を参照してください。

作成時にタグが付けられていないスポットインスタンス

AWS Batch のコンピューティングリソースのスポットインスタンスのタグ付けは、2017 年 10 月 25 日にサポートが開始されました。以前には、Amazon EC2 スポットフリートロール用の推奨 IAM 管理ポリシー (`AmazonEC2SpotFleetRole`) には、起動時にスポットインスタンスにタグを付けるアクセス権限が含まれていませんでした。新しい推奨の IAM 管理ポリシー

は、AmazonEC2SpotFleetTaggingRole と呼ばれます。起動時のスポットインスタンスへのタグ付けをサポートします。

作成時にスポットインスタンスのタグ付けを修正するには、以下の手順に従って、現在推奨の IAM 管理ポリシーを Amazon EC2 スポットフリートロールに割り当てます。その方法で、そのロールで今後作成されるスポットインスタンスには、作成時にインスタンスタグを適用する権限が付与されます。

現在の IAM 管理ポリシーを Amazon EC2 スポットフリートロールに割り当てるには

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. [Roles] (ロール) を選択し、Amazon EC2 スポットフリートロールを選択します。
3. Attach policy] (ポリシーのアタッチ) を選択します。
4. [AmazonEC2SpotFleetTaggingRole] を選択し、[Attach policy] (ポリシーのアタッチ) を選択します。
5. Amazon EC2 スポットフリートロールを再度選択し、前のポリシーを削除します。
6. AmazonEC2SpotFleetRole ポリシーの右側にある [x] を選択し、[Detach] (デタッチ) を選択します。

スポットインスタンスがスケールダウンしない

AWS Batch は、2021 年 3 月 10 日に [AWSServiceRoleForBatch] というサービスにリンクされたロールを導入しました。コンピューティング環境の `serviceRole` パラメータにロールが指定されていない場合、このサービスにリンクされたロールが、サービスロールとして使用されます。EC2 Spot のコンピューティング環境でサービスにリンクされたロールが使用されているが、使用されているスポットロールに AmazonEC2SpotFleetTaggingRole 管理ポリシーが含まれていない場合、Spotインスタンスがスケールダウンされません。そうであれば、スポットインスタンスはスケールダウンしません。その結果、このオペレーションを実行する権限がありませんというエラーメッセージを受け取ります。以下の手順で、`spotIamFleetRole` パラメータで使用するスポットフリートロールを更新してください。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使用およびAWS サービスに対してアクセス許可を委譲するロールの作成を参照してください。

トピック

- AmazonEC2SpotFleetTaggingRole 管理ポリシーを、AWS マネジメントコンソールにあるユーザーのスポットフリートロールにアタッチします

- AmazonEC2SpotFleetTaggingRole 管理ポリシーを、AWS CLIを使用して、ユーザーのスポットフリートロールにアタッチする

AmazonEC2SpotFleetTaggingRole 管理ポリシーを、AWS マネジメントコンソールにあるユーザーのスポットフリートロールにアタッチします

現在の IAM 管理ポリシーを Amazon EC2 スポットフリートロールに割り当てるには

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. [Roles] (ロール) を選択し、Amazon EC2 スポットフリートロールを選択します。
3. Attach policy] (ポリシーのアタッチ) を選択します。
4. [AmazonEC2SpotFleetTaggingRole] を選択し、[Attach policy] (ポリシーのアタッチ) を選択します。
5. Amazon EC2 スポットフリートロールを再度選択し、前のポリシーを削除します。
6. AmazonEC2SpotFleetRole ポリシーの右側にある [x] を選択し、[Detach] (デタッチ) を選択します。

AmazonEC2SpotFleetTaggingRole 管理ポリシーを、AWS CLIを使用して、ユーザーのスポットフリートロールにアタッチする

コマンド例では、Amazon EC2 スポットフリートのロール名が *AmazonEC2SpotFleetRole]* であることを想定しています。ロールで別の名前が使用されている場合は、一致するようにコマンドを調整します。

AmazonEC2SpotFleetTaggingRole 管理ポリシーを、ユーザースポットフリートロールにアタッチするには

1. AmazonEC2SpotFleetTaggingRole] 管理 IAM ポリシーを *AmazonEC2SpotFleetRole]* ロールにアタッチするには、AWS CLIを使用して次のコマンドを実行します。

```
$ aws iam attach-role-policy \
    --policy-arn arn:aws:iam::aws:policy/service-role/AmazonEC2SpotFleetTaggingRole
 \
    --role-name AmazonEC2SpotFleetRole
```

2. [AmazonEC2SpotFleetRole] のマネージド IAM ポリシーを *[AmazonEC2SpotFleetRole]* ロールからデタッチするには、AWS CLI を使用して次のコマンドを実行します。

```
$ aws iam detach-role-policy \
    --policy-arn arn:aws:iam::aws:policy/service-role/AmazonEC2SpotFleetRole \
    --role-name AmazonEC2SpotFleetRole
```

Secrets Manager のシークレットを取得できない

バージョン 1.16.0-1 より以前の Amazon ECS エージェントで AMI を使用している場合は、この機能を使用するため、Amazon ECS エージェント設定変数 `ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE=true` を使用する必要があります。そのインスタンスを作成するときに、新しいコンテナインスタンスの `./etc/ecs/ecs.config` ファイルに追加します。または、既存のインスタンスに追加することもできます。既存のインスタンスに追加する場合は、追加後に ECS エージェント を再起動する必要があります。詳細については、Amazon Elastic Container Service デベロッパーガイドの「Amazon ECS コンテナエージェントの設定」を参照してください。

ジョブ定義リソース要件を上書きできない

SubmitJob に渡される containerOverrides 構造の `memory` および `vcpus` メンバーで指定されるメモリと vCPU の上書きは、ジョブ定義の resourceRequirements 構造で指定されるメモリと vCPU の要件を上書きできません。

これらのリソース要件を上書きしようとすると、次のエラーメッセージが表示されることがあります:

この値は非推奨のキーで送信され、ジョブ定義のリソース要件で提供される値と競合する可能性があります。

これを修正するには、containerOverrides] の resourceRequirements] メンバーにメモリと vCPU の要件を指定します。例えば、メモリと vCPU の上書きが次の行で指定されているとします。

```
"containerOverrides": {
   "memory": 8192,
   "vcpus": 4
}
```

それらを次に変更します:

```
"containerOverrides": {
```

```
   "resourceRequirements": [
      {
         "type": "MEMORY",
         "value": "8192"
      },
      {
         "type": "VCPU",
         "value": "4"
      }
   ],
}
```

ジョブ定義の containerProperties] オブジェクトで指定されているメモリと vCPU の要件と同じ変更を行います。例えば、メモリと vCPU の要件が次の行で指定されているとします。

```
{
   "containerProperties": {
      "memory": 4096,
      "vcpus": 2,
}
```

それらを次に変更します:

```
"containerProperties": {
   "resourceRequirements": [
      {
         "type": "MEMORY",
         "value": "4096"
      },
      {
         "type": "VCPU",
         "value": "2"
      }
   ],
}
```

desiredvCpus 設定を更新すると、エラーメッセージが表示されます

AWS Batch API を使用して目的の vCPUs (desiredvCpus) 設定を更新すると、次のエラーメッセージが表示されます。

`Manually scaling down compute environment is not supported. Disconnecting job queues from compute environment will cause it to scale-down to minvCpus.`

この問題は、desiredvCpus 更新された desiredvCpus 値が現在の値より小さい場合に発生します。desiredvCpus 値を更新するには、次のいずれかが当てはまる必要があります：

- desiredvCpus 値は minvCpus と maxvCpus 値の間になければなりません。
- 値は、desiredvCpusStart Frame (スタートフレーム) desiredvCpus 値と同等かそれ以上である必要があります。

AWS Batch Amazon EKS での

トピック

Amazon Elastic Kubernetes Service AWS Batch で を使用する際に発生する可能性がある一般的な問題に対するレビュープロセスと潜在的な解決策については、以下のトピックを参照してください。

INVALID コンピューティング環境

マネージド型コンピューティング環境を誤って設定した可能性があります。設定した場合、コンピューティング環境は INVALID の状態になり、配置するジョブを受け入れられなくなります。以下のセクションでは、考えられる原因と、その原因に基づいトラブルシューティング方法について説明します。

サポート対象外の Kubernetes バージョン

CreateComputeEnvironment API オペレーションまたは API オペレーション UpdateComputeEnvironment を使用してコンピュート環境を作成または更新すると、次のようなエラーメッセージが表示されることがあります。この問題は、Kubernetes でサポートされていないバージョンを EC2Configuration で指定した場合に発生します。

```
At least one imageKubernetesVersion in EC2Configuration is not supported.
```

この問題を解決するには、コンピュート環境を削除し、サポートされている Kubernetes バージョンで再作成してください。

Amazon EKS クラスターでマイナーバージョンアップグレードを実行できます。例えば、イナーバージョンがサポートされていない場合でも、クラスターを 1.xx から 1.yy マにアップグレードできます。

ただし、INVALID メジャーバージョンを更新すると、コンピュート環境のステータスがに変わることがあります。例えば、1.xx から 2.yy へのメジャーバージョンアップグレードを実行する場合などです。メジャーバージョンが でサポートされていない場合は AWS Batch、次のような工ラーメッセージが表示されます。

```
reason=CLIENT_ERROR - ... EKS Cluster version [2.yy] is unsupported
```

この問題を解決するには、API オペレーションを使用してコンピューティング環境を作成または更新するときに、Kubernetes サポートされているバージョンを指定してください。

AWS Batch Amazon EKS の は現在、次のKubernetesバージョンをサポートしています。

- 1.34
- 1.33
- 1.32
- 1.31
- 1.30
- 1.29

インスタンスプロファイルが存在しません

指定されたインスタンスプロファイルが存在しない場合、Amazon EKS コンピューティング環境 AWS Batch のステータスは に変更されますINVALID。statusReason パラメータに次のような工ラーセットが表示されます。

```
CLIENT_ERROR - Instance profile arn:aws:iam::...:instance-profile/<name> does not exist
```

この問題を解決するには、ワーキングインスタンスプロファイルを指定または作成します。詳細については、Amazon EKS ユーザーガイドのAmazon EKS ノードの IAM ロールを参照してくださない。

無効な Kubernetes 名前空間が

Amazon EKS AWS Batch でコンピューティング環境の名前空間を検証できない場合、コンピューティング環境のステータスは に変更されますINVALID。例えば、この問題は名前空間が存在しない場合に発生する可能性があります。

statusReason パラメータには、次のようなエラーメッセージが設定されています。

```
CLIENT_ERROR - Unable to validate Kubernetes Namespace
```

この問題は、次のいずれかに該当する場合に発生する可能性があります:

- CreateComputeEnvironment 呼び出し中のKubernetes 名前空間文字列は存在しません。詳細については、CreateComputeEnvironment を参照してください。
- 名前空間の管理に、必要なロールベースのアクセス制御 (RBAC) 権限が正しく設定されていません。
- AWS Batch は Amazon EKS Kubernetes API サーバーエンドポイントにアクセスできません。

この問題を解決するには、aws-auth ConfigMap のフィールドが正しく設定されていることを確認しますをご参照ください。詳細については、「Amazon EKS で AWS Batch のご利用開始にあたって」を参照してください。

削除されたコンピューティング環境

Amazon EKS コンピューティング環境でアタッチされている を削除する前に AWS Batch 、Amazon EKS クラスターを削除するとします。そうすると、コンピューティング環境のステータスは INVALID に変わります。このシナリオでは、Amazon EKS クラスターを同じ名前で再作成すると、コンピューティング環境は正しく機能しません。

この問題を解決するには、Amazon EKS コンピューティング環境で を削除してから再作成 AWS Batch します。

ノードは Amazon EKS クラスターに加わりません

AWS Batch Amazon EKS 上の は、すべてのノードが Amazon EKS クラスターに参加していないと判断した場合、コンピューティング環境をスケールダウンします。Amazon EKS AWS Batch でコン

ピューティング環境をスケールダウンすると、コンピューティング環境のステータスは に変更されますINVALID。

> **Note**
>
> AWS Batch は、問題をデバッグできるように、コンピューティング環境のステータスをすぐに変更しません。

statusReason パラメータには、次のようなエラーメッセージが設定されています。

```
Your compute environment has been INVALIDATED and scaled down because
none of the instances joined the underlying ECS Cluster. Common issues
preventing instances joining are the following: VPC/Subnet configuration
preventing communication to ECS, incorrect Instance Profile policy
preventing authorization to ECS, or customized AMI or LaunchTemplate
configurations affecting ECS agent.
```

```
Your compute environment has been INVALIDATED and scaled down because
none of the nodes joined the underlying Amazon EKS Cluster. Common issues
preventing nodes joining are the following: networking configuration
preventing communication to Amazon EKS Cluster, incorrect Amazon EKS
Instance Profile or Kubernetes RBAC policy preventing authorization
to Amazon EKS Cluster, customized AMI or LaunchTemplate configurations
affecting Amazon EKS/Kubernetes node bootstrap.
```

デフォルトの Amazon EKS AMI を使用する場合、この問題の最も一般的な原因は次のとおりです：

- インスタンスロールが正しく設定されていません。詳細については、Amazon EKS ユーザーガイドのAmazon EKS ノードの IAM ロールを参照してください。
- サブネットが正しく設定されていません。詳細については、「Amazon EKS ユーザーガイド」の「Amazon EKS VPC とサブネットの要件と考慮事項」を参照してください。
- セキュリティグループが正しく設定されていません。詳細については、「Amazon EKS ユーザーガイド」の「Amazon EKS セキュリティグループの要件と考慮事項」を参照してください。

> **Note**
>
> また、Personal Health Dashboard (PHD) にエラー通知が表示される場合があります。

Amazon EKS ジョブが **RUNNABLE** のステータスで停止した場合の AWS Batch

マネージド型ノードグループを作成する場合、または aws-auth を使用してノードグループを作成する場合、ConfigMap eksctl が自動的に作成され、クラスターに適用されます。aws-auth ConfigMap は最初に、ノードがクラスターに参加できるようにするために作成されました。ただし、aws-auth ConfigMap を使用して、ユーザーとロールにロールベースのアクセス制御 (RBAC) アクセスを追加することもできます。

aws-auth ConfigMap が正しく設定されていることを確認するには:

1. マップされたロールを以下の aws-auth ConfigMap から取得します :

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

2. roleARN が次のように設定されていることを確認します。

```
rolearn: arn:aws:iam::aws_account_number:role/AWSServiceRoleForBatch
```

> **Note**
>
> また、Amazon EKS コントロールプレーンログを確認することもできます。詳細については、「Amazon EKS ユーザーガイド」の「Amazon EKS コントロールプレーンのログ」を参照してください。

ジョブが RUNNABLE のステータスのままになる問題を解決するには、kubectl を使用してマニフェストを再適用することをお勧めします。詳細については、AWS Batch の Amazon EKS クラスターを準備するを参照してください。または、kubectl を使用して aws-auth ConfigMap を手動で編集することもできます。詳細については、「Amazon EKS ユーザーガイド」の「IAM ユーザーとロールのクラスターへのアクセスを有効にする」を参照してください。

Amazon EKS ジョブが **STARTING** のステータスで停止した場合の AWS Batch

ポッドのスタートアップの問題が解決されるか、ジョブが終了するまでに、kubelet (pull、log、exec、attach) から長時間実行されているリクエストの ContainerCreating でポッドが PENDING の状態で停止した場合、ジョブのステータスが STARTING のままになることが

あります。以下の対象となるシナリオにおいて、AWS Batch はユーザーの代わりにジョブを終了させます。そうでない場合、ジョブは TerminateJob API を使用して手動で終了する必要があります。

ジョブが STARTING で停止する原因を検証するには、チュートリアル: 実行中のジョブをポッドとノードにマップする を使用して podName を検索し、ポッドについて説明します。

```
% kubectl describe pod aws-batch.000c8190-87df-31e7-8819-176fe017a24a -n my-aws-batch-
namespace
Name:             aws-batch.000c8190-87df-31e7-8819-176fe017a24a
Namespace:        my-aws-batch-namespace
...
Containers:
  default:
...
    State:          Waiting
      Reason:       ContainerCreating
    Ready:          False
...
Conditions:
  Type                        Status
  PodReadyToStartContainers   False
  Initialized                 True
  Ready                       False
  ContainersReady             False
  PodScheduled                True
...
Events:
  Type     Reason       Age    From     Message
  ----     ------       ----   ----     -------
  Warning  FailedMount  2m32s  kubelet  Unable to attach or mount volumes: ...
```

完全な可視性を実現するために、コントロールプレーンログを CloudWatch Logs に送信するように EKS クラスターを設定することを検討してください。

シナリオ: 永続ボリュームクレームのアタッチまたはマウントの失敗

ボリュームのアタッチまたはマウントに失敗する永続ボリュームクレームを使用するジョブは、終了の対象となります。これは、ジョブ定義が正しく設定されていないことが原因である可能性があります。詳細については、「Amazon EKS リソースにシングルノードのジョブ定義を作成する」を参照してください。

aws-auth ConfigMap のフィールドが正しく設定されていることを確認します

aws-auth ConfigMap が正しく設定されていることを確認するには:

1. aws-auth ConfigMap にマップされたロールを取得します。

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

2. roleARN が次のように設定されていることを確認します。

```
rolearn: arn:aws:iam::aws_account_number:role/AWSServiceRoleForBatch
```

> **Note**
>
> パス aws-service-role/batch.amazonaws.com/が、サービスにリンクされたロールの ARN から削除されました。これは aws-auth 設定マップに問題があるためです。詳細については、パスを持つロールは、aws-authconfigmap の ARN にパスが含まれていると機能しない を参照してください。

> **Note**
>
> また、Amazon EKS コントロールプレーンログを確認することもできます。詳細については、「Amazon EKS ユーザーガイド」の「Amazon EKS コントロールプレーンのログ」を参照してください。

ジョブが RUNNABLEのステータスのままになる問題を解決するには、kubectl を使用してマニフェストを再適用することをお勧めします。詳細については、AWS Batch の Amazon EKS クラスターを準備するを参照してください。または、kubectl を使用して aws-auth ConfigMap を手動で編集することもできます。詳細については、「Amazon EKS ユーザーガイド」の「IAM ユーザーとロールのクラスターへのアクセスを有効にする」を参照してください。

RBAC の権限またはバインディングが適切に設定されていない

RBAC 権限またはバインディングの問題が発生した場合は、aws-batch Kubernetes のロールが Kubernetes 名前空間にアクセスできることを確認してください:

```
$ kubectl get namespace namespace --as=aws-batch
```

```
$ kubectl auth can-i get ns --as=aws-batch
```

kubectl describe コマンドを使用して、クラスターロールまたは Kubernetesの名前空間の権限を確認することもできます。

```
$ kubectl describe clusterrole aws-batch-cluster-role
```

以下は出力の例です。

```
Name:         aws-batch-cluster-role
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources                                       Non-Resource URLs  Resource Names
 Verbs
  ---------                                       -----------------  --------------
 -----
  configmaps                                      []                 []
 [get list watch]
  nodes                                           []                 []
 [get list watch]
  pods                                            []                 []
 [get list watch]
  daemonsets.apps                                 []                 []
 [get list watch]
  deployments.apps                                []                 []
 [get list watch]
  replicasets.apps                                []                 []
 [get list watch]
  statefulsets.apps                               []                 []
 [get list watch]
  clusterrolebindings.rbac.authorization.k8s.io   []                 []
 [get list]
  clusterroles.rbac.authorization.k8s.io          []                 []
 [get list]
  namespaces                                      []                 []
 [get]
  events                                          []                 []
 [list]
```

```
$ kubectl describe role aws-batch-compute-environment-role -n my-aws-batch-namespace
```

以下は出力の例です。

```
Name:         aws-batch-compute-environment-role
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources                                  Non-Resource URLs  Resource Names  Verbs
  ---------                                  -----------------  --------------  -----
  pods                                       []                 []              [create
 get list watch delete patch]
  serviceaccounts                            []                 []              [get list]
  rolebindings.rbac.authorization.k8s.io  []                 []              [get list]
  roles.rbac.authorization.k8s.io         []                 []              [get list]
```

この問題を解決するには、RBAC 権限と rolebinding コマンドを再度適用します。詳細については、AWS Batch の Amazon EKS クラスターを準備するを参照してください。

リソース: AWS Batch Service Quotas

次の表に、変更できない AWS Batch のサービスクォータを示します。各クォータはリージョンごとにあります。

リソース	クォータ
ジョブキューの最大数。詳細については、ジョブキューを参照してください。	50
Amazon ECS と Amazon EKS 全体のコンピューティング環境の最大数。詳細については、のコンピューティング環境 AWS Batchを参照してください。	50
Amazon EKS クラスターごとのコンピューティング環境の最大数。	5
ジョブキューあたりのコンピューティング環境の最大数	3
ジョブの依存関係の最大数	20
ジョブ定義の最大サイズ (RegisterJobDefinition API オペレーションの場合)	24 KiB
ジョブの最大ペイロードサイズ (SubmitJob API オペレーションの場合)	30 KiB
配列ジョブの最大配列サイズ	10000
SUBMITTED 状態のジョブの最大数	1000000
SubmitJob オペレーションに必要な各アカウントの最大秒間トランザクション数 (TPS)	50
消費型リソースの最大数	50,000
サービス環境の最大数。詳細については、「サービス環境」を参照してください。	50
ジョブキューあたりのコンピューティング環境の最大数	1
SubmitServiceJob リクエストの最大サイズ	30 KiB
ジョブサービスリクエストの最大ペイロードサイズ (SubmitServiceJob API オペレーションの場合)	10 KiB

リソース	クォータ
SubmitServiceJob オペレーションに必要な各アカウントの最大秒間トランザクション数 (TPS)	5
サービスジョブの再試行戦略の最大試行回数	10

AWS Batch の使用方法によっては、追加のクォータが適用される場合があります。Amazon EC2 クォータについて学習するには、AWS 全般のリファレンス の「Amazon EC2 Service Quotas」を参照してください。Amazon ECS クォータの詳細については、 AWS 全般のリファレンス の「Amazon ECS Service Quotas」を参照してください。Amazon EKS クォータの詳細については、「AWS 全般のリファレンス」の「Amazon EKS Service Quotas」を参照してください。

ドキュメント履歴

次の表に、AWS Batch の最初のリリース以後に行われた、ドキュメントの重要な変更を示します。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
default_x86_64 と default_arm64 を追加	許可されたインスタンスタイプとして `default_x86_64` と `default_arm64` を新規追加しました。	2025 年 8 月 15 日
サービス環境とサービスジョブを追加	SageMaker AI で AWS Batch を使用するためのサービス環境とサービスジョブを追加しました。	2025 年 7 月 30 日
AWSServiceRoleForAWSBatchWithSagemaker と AWSBatchServiceRolePolicyForSageMaker を追加	AWS Batch がユーザーに代わって SageMaker AI を管理できるようにする、AWS サービスにリンクされたロール `AWSServiceRoleForAWSBatchWithSagemaker` と管理ポリシー `AWSBatchServiceRolePolicyForSageMaker` を新規追加しました。	2025 年 7 月 30 日
EKS AL 2023 AMI のサポートを追加	EKS AL2 から EKS AL2023 へアップグレードする方法	2025 年 6 月 24 日
FireLens および ECS Exec コマンドのサポートを追加	FireLens および ECS Exec コマンドのサポートを追加しました。	2025 年 4 月 15 日

AWS Batch のリソース対応スケジューリングのサポートを追加	Amazon Elastic Container Service、Amazon Elastic Kubernetes Service、AWS Fargate の AWS Batch にリソース対応スケジューリングのサポートを追加しました。	2025 年 2 月 27 日
AWS Batch でサポートされている Amazon EKS バージョンの更新	AWS Batch でサポートされている Amazon EKS バージョンを更新し、バージョン 1.22 を削除しました。	2024 年 3 月 11 日
AWS Batch でサポートされている Amazon EKS バージョンの更新	AWS Batch でサポートされている Amazon EKS バージョンに、バージョン 1.29 を追加して更新しました。	2024 年 2 月 29 日
ジョブの再試行の自動化	コードのサンプルを修正しました。	2024 年 2 月 29 日
にマルチコンテナジョブのサポートを追加AWS Batch	Amazon Elastic Container Service、Amazon Elastic Kubernetes Service、AWS Fargate の AWS Batch にマルチコンテナジョブのサポートを追加しました。	2024 年 2 月 28 日
AWS Batch でサポートされている Amazon EKS バージョンの更新	AWS Batch がサポートする Amazon EKS バージョンに、バージョン 1.28 を追加して更新しました。	2024 年 1 月 27 日

「BatchServiceRolePolicy」と「AWSBatchServiceRole」の更新	BatchServiceRolePolicy スポットフリートリクエストの履歴と Amazon EC2 Auto Scaling のアクティビティの記述のサポートを追加するように更新しました。 AWSBatchServiceRole ステートメント ID を追加し、AWS Batch に `ec2:DescribeSpotFleetRequestHistory` と `autoscaling:DescribeScalingActivities` のアクセス許可を付与するように更新しました。	2023 年 12 月 5 日
Amazon EKS の AWS Batch	AWS Batch では、Amazon EKS クラスターでのジョブの実行のサポートが追加されました。	2022 年 10 月 25 日
のクロスサービスでの混乱した代理の防止AWS Batch	エンティティ (サービスまたはアカウント) が別のエンティティによってアクションを実行するように強制された場合に発生する、混乱を招く副セキュリティ問題の回避策を、AWS Batch が現在提供するようになりました。	2022 年 6 月 6 日

インターフェイス VPC エンドポイント (AWS PrivateLink)	AWS PrivateLink を搭載したインターフェイス VPC エンドポイントを設定するためのサポートが追加されました。つまり、インターネット、NAT インスタンス、VPN 接続、または Direct Connect を行うことなく、VPC と AWS Batch をプライベートに接続できます。	2022 年 4 月 15 日
コンピューティング環境の更新機能の強化	AWS Batch により、コンピューティング環境のサポート更新が強化されました。	2022 年 4 月 14 日
AWS 管理ポリシーの更新 - 既存のポリシーへの更新	AWS Batch 既存の管理ポリシーの更新。	2021 年 12 月 6 日
公平配分のスケジューリング	AWS Batch では、ジョブキューにスケジューリングポリシーを追加するためのサポートが追加されました。	2021 年 11 月 9 日
Amazon EFS	AWS Batch では、ジョブ定義に Amazon EFS ファイルシステムを追加するためのサポートを追加しました。	2021 年 4 月 1 日
サービスにリンクされたロールが追加されました	AWS Batch では、サービスにリンクされたロール AWSServiceRoleForAmazonEKS が追加されました。	2021 年 3 月 10 日
AWS Fargate のサポート	AWS Batch では、Fargate リソースでのジョブの実行のサポートが追加されました。	2020 年 12 月 3 日

リソースのタグ付け	AWS Batch では、コンピューティング環境、ジョブ定義、ジョブキュー、ジョブにメタデータタグの追加がサポートされました。	2020 年 10 月 7 日
シークレット	AWS Batch では、ジョブにシークレットを渡す機能が追加されました。	2020 年 10 月 1 日
ログ記録	AWS Batch では、ジョブに追加のログドライバーを指定するためのサポートが追加されました。	2020 年 10 月 1 日
配分戦略	AWS Batch では、インスタンスタイプを選択する複数の戦略がサポートされています。	2019 年 10 月 16 日
EFA のサポート	AWS Batch によって Elastic Fabric Adapter (EFA) デバイスのサポートが追加されます。	2019 年 8 月 2 日
GPU スケジューリング	AWS Batch では、GPU スケジューリングが追加されました。この機能を使用すると、各ジョブが必要とする GPU の数を指定できます。AWS Batch は、それに応じてインスタンスをスケールアップします。	2019 年 4 月 4 日

マルチノード並列ジョブ	AWS Batch では、マルチノード並列ジョブのサポートが追加されました。この機能を使用すると、複数の Amazon EC2 インスタンスにまたがる単一のジョブを実行できます。	2018 年 11 月 19 日
リソースレベルのアクセス許可	AWS Batch で、複数の API オペレーションにおけるリソースレベルのアクセス許可がサポートされるようになりました。	2018 年 11 月12 日
Amazon EC2 起動テンプレートのサポート	AWS Batch にコンピューティング環境で起動テンプレートを使用するためのサポートが追加されました。	2018 年 11 月12 日
AWS Batch ジョブのタイムアウト	AWS Batch では、ジョブタイムアウトのサポートが追加されました。これにより、特定の期間を超えてジョブが実行されると AWS Batch でジョブが終了するように、ジョブのタイムアウト期間を設定できます。	2018 年 4 月 5 日
AWS Batch EventBridge ターゲットとしてのジョブ	AWS Batch ジョブは CloudWatch Events ターゲットとして利用可能になります。簡単なルールを作成することで、イベントをマッチングさせ、それに応じて AWS Batch ジョブを送信できます。	2018 年 3 月 1 日

AWS Batch の CloudTrail 監査	CloudTrail では、AWS Batch API アクションに対する呼び出しを監査できます。	2018 年 1 月 10 日
配列ジョブ	AWS Batch では、配列ジョブのサポートが追加されました。パラメータスイープおよび Monte Carlo ワークロードのパラメータに配列ジョブを使用できます。	2017 年 11 月 28 日
拡張された AWS Batch のタグ付け	AWS Batch は、タグ付け機能のサポートを拡大しました。この機能を使用して、マネージド型のコンピューティング環境内で起動された Amazon EC2 スポットインスタンスのタグを指定できます。	2017 年 10 月 26 日
AWS Batch Events Bridge のイベントストリーム	AWS Batch は、Events Bridge にイベントストリームを追加します。ユーザーは、AWS Batch のイベントストリームを使用して、ジョブキューに送信されたジョブの状態に関する通知を、ほぼリアルタイムで受け取ることができます。	2017 年 10 月 24 日
ジョブの再試行の自動化	AWS Batch では、ジョブ再試行のサポートが追加されました。この更新により、ジョブおよびジョブ定義に再試行戦略を適用し、ジョブが失敗した場合に自動的に再試行できます。	2017 年 3 月 28 日

AWS Batch の一般提供	AWS Batch は、AWS クラウド 上でバッチコンピューティングワークロードを実行するための手段として設計されています。	2017 年 1 月 5 日

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