aws

管理ガイド

Amazon EMR

Amazon EMR: 管理ガイド

Table of Contents

Amazon EMR とは

Amazon EMR (以前は Amazon Elastic と呼ばれていました MapReduce) は、Apache Hadoop や Apache Spark AWS などのビッグデータフレームワークを簡単に実行して膨大な量のデータを処理および分析できるようにするマネージド型クラスタープラットフォームです。これらのフレームワークと、関連するオープンソースプロジェクトを使用することで、分析用のデータやビジネスインテリジェンスワークロードを処理できます。Amazon EMR を使用して、大量のデータを変換し、Amazon Simple Storage Service (Amazon S3) や Amazon DynamoDB などの他の AWS データストアやデータベースにデータを出し入れすることもできます。

Amazon EMR を初めて使用する方には、このセクションと併せて、次のセクションを初めに読むことをおすすめします。

- Amazon EMR – このサービスページには、Amazon EMR のハイライト、製品詳細、料金情報が掲載されています。
- チュートリアル: Amazon EMR の使用開始 — このチュートリアルを使用すると、Amazon EMR をすぐに使い始めることができます。

このセクションの内容

Amazon EMR の概要

このトピックでは、クラスターに作業を送信する方法、データが処理される方法、処理中のクラスターの状態の変化など、Amazon EMR クラスターの概要を示します。

このトピックの内容

クラスターおよびノードについて

Amazon EMR の中心的なコンポーネントは、クラスターです。クラスターは、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスの集合です。クラスター内の各インスタンスは、ノードと呼ばれます。各ノードには、クラスター内にロールがあり、ノードタイプと呼ばれます。また、Amazon EMR は、各ノードタイプにさまざまなソフトウェアコンポーネントをインストールし、Apache Hadoop などの分散型アプリケーションでのロールを各ノードに付与します。

Amazon EMR のノードタイプは、次のとおりです。

- プライマリノード: 処理を行うために他のノード間でのデータおよびタスクの分散を調整するソフトウェアコンポーネントを実行することで、クラスターを管理するノードです。プライマリノードは、タスクのステータスを追跡し、クラスターの状態を監視します。すべてのクラスターにはプライマリノードがあり、プライマリノードのみで 1 つのノードクラスターを作成できます。
- コアノード: タスクを実行し、クラスター上の Hadoop Distributed File System (HDFS) にデータを保存するソフトウェアコンポーネントを持つノードです。マルチノードクラスターには、少なくとも 1 つのコアノードがあります。
- タスクノード: タスクを実行するのみで、HDFS にデータを保存しないソフトウェアコンポーネントを持つノードです。タスクノードはオプションです。

クラスターへのワークの送信

Amazon EMR でクラスターを実行する場合、行う必要があるワークを指定する方法についてはいくつかのオプションがあります。

- クラスターの作成時にステップとして指定する関数で実行するワークの、完全な定義を提供します。これは、通常、一定量のデータを処理し処理が完了したときに終了するクラスターに対して実行されます。
- 長時間稼働クラスターを作成し、Amazon EMR コンソール、Amazon EMR API、またはを使用してステップを送信します AWS CLI 。ステップには 1 つ以上のジョブが含まれる場合があります。詳細については、「クラスターへの作業の送信」を参照してください。
- クラスターを作成し、SSH を使用してプライマリノードや必要に応じて他のノードに接続して、インストール済みアプリケーションが提供するインターフェイスを使用します。これにより、スクリプト化を使用するか、インタラクティブにタスクを実行し、クエリを送信します。詳細については、「Amazon EMR リリースガイド」を参照してください。

データの処理

クラスターを起動するとき、データ処理の必要に合わせてインストールするフレームワークとアプリケーションを選択します。Amazon EMR クラスターでデータを処理するには、インストールされたアプリケーションにジョブまたはクエリを直接送信するか、クラスターでステップを実行することもできます。

アプリケーションへのジョブの直接送信

Amazon EMR クラスターにインストールされたソフトウェアを使用し、直接ジョブを送信して操作できます。これを行うには、通常安全な接続経由でプライマリノードに接続し、クラスターで直接実行されるソフトウェアに使用できるインターフェイスとツールにアクセスします。詳細については、「クラスターに接続する」を参照してください。

ステップの実行によるデータの処理

Amazon EMR クラスターには、1 つ以上のステップを順番に並べて送信できます。各ステップは、クラスターにインストールされたソフトウェアにより処理するためのデータを操作する指示が含まれる作業単位です。

4 つのステップを使用した処理の例を次に示します。

1. 処理のために入力データセットを送信する
2. Pig プログラムを使用して、最初のステップの出力を処理する。
3. Hive プログラムを使用して 2 番目の入力データセットを処理する
4. 出力データセットを書き込む

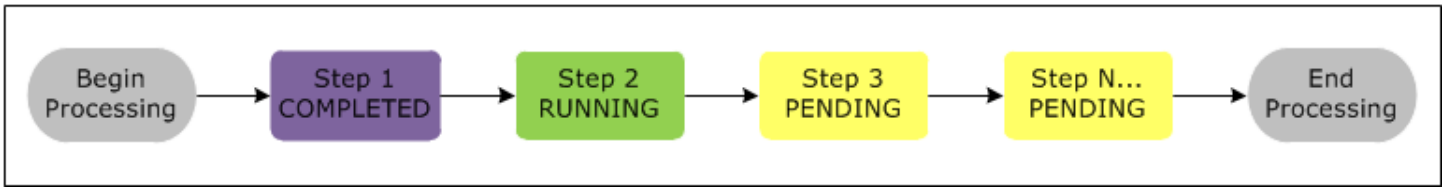
通常、Amazon EMR でデータを処理する場合、入力とは、選択した基になるファイルシステム (Amazon S3 や HDFS など) にファイルとして保存されるデータのことです。このデータは、処理シーケンスの次のステップに移動します。最後のステップで、指定された場所 (Amazon S3 バケットなど) に出力データが書き込まれます。

ステップは、次の順序で実行されます。

1. リクエストが送信され、ステップの処理が開始されます。
2. すべてのステップの状態が [PENDING (保留中)] になります。
3. シーケンスの最初のステップが開始されると、その状態が [RUNNING (実行中)] に変わります。他のステップは、[PENDING (保留中)] 状態のままです。

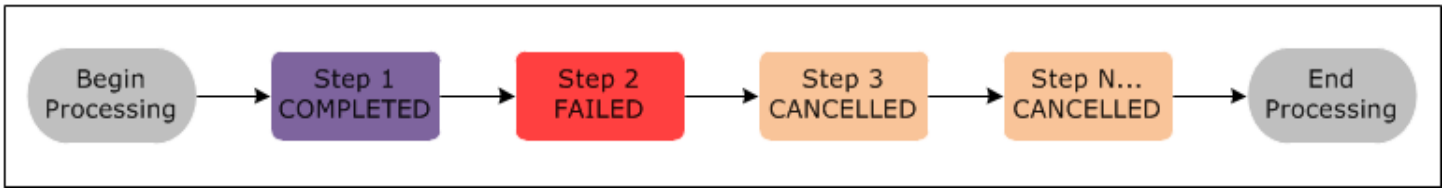
4. 最初のステップが完了すると、その状態が [COMPLETED (完了済み)] に変わります。
5. シーケンスの次のステップが開始され、その状態が [RUNNING (実行中)] に変わります。完了すると、その状態が [COMPLETED (完了済み)] に変わります。
6. すべてのステップが完了し、処理が終了するまで、ステップごとにこのパターンが繰り返されます。

次の図に、ステップが処理される際のステップシーケンスと状態の変化を示します。



ステップが処理に失敗した場合、その状態は [FAILED] (失敗) に変わります。ステップごとに、次に実行する処理を指定できます。デフォルトでは、シーケンスの残りのステップはすべて [CANCELLED] (キャンセル済み) に設定され、前のステップが失敗した場合は実行されません。エラーを無視し、残りのステップを続行するか、ただちにクラスターを終了する選択ができます。

次の図に、処理中にステップが失敗した場合のステップシーケンスと状態のデフォルトの変化を示します。



クラスターライフサイクルについて

成功した Amazon EMR クラスターは次のプロセスに従います。

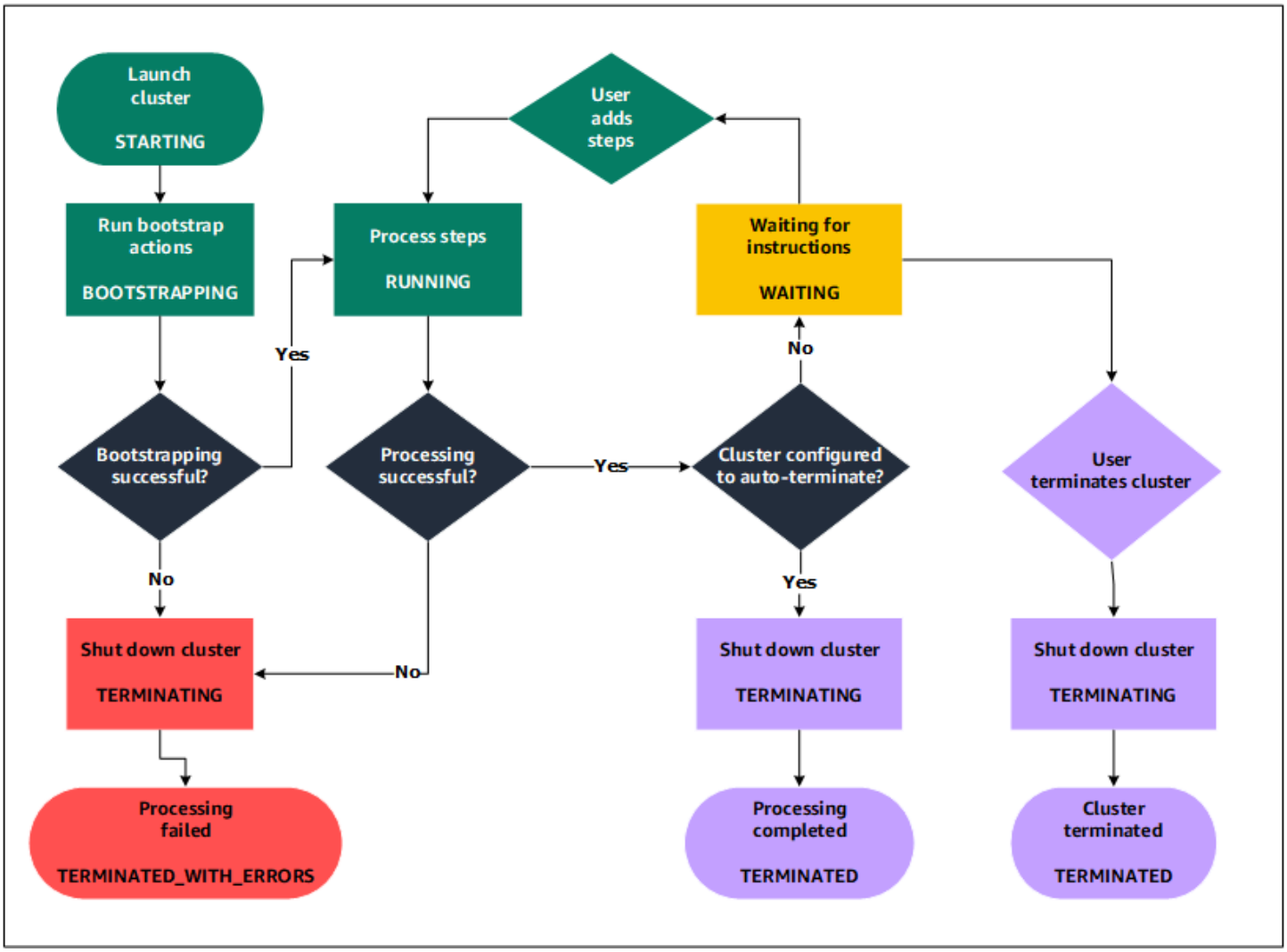
1. 最初に、Amazon EMR は指定に従って、各インスタンスのクラスターで EC2 インスタンスをプロビジョニングします。詳細については、「クラスターハードウェアとネットワークを設定する」を参照してください。Amazon EMR は、すべてのインスタンスに対して、Amazon EMR 用のデフォルト AMI または指定するカスタム Amazon Linux AMI を使用します。詳細については、「カスタム AMI の使用」を参照してください。このフェーズの間、クラスターの状態は `STARTING` です。
2. 各インスタンスで指定したブートストラップアクションが Amazon EMR によって実行されます。ブートストラップアクションを使用してカスタムアプリケーションをインストールし、必要な力

スタマイズを実行できます。詳細については、「追加のソフトウェアをインストールするためのブートストラップアクションの作成」を参照してください。このフェーズの間、クラスターの状態は BOOTSTRAPPING です。

3. Amazon EMR は、Hive、Hadoop、Spark など、クラスターの作成時に指定するネイティブアプリケーションをインストールします。
4. ブートストラップアクションが正常に完了し、ネイティブアプリケーションがインストールされると、クラスターの状態は RUNNING になります。この時点で、クラスターインスタンスに接続できます。クラスターは、クラスターの作成時に指定されたステップを順番に実行します。前のステップの完了後に実行される追加のステップを送信できます。詳細については、「クラスターへの作業の送信」を参照してください。
5. ステップが正常に実行されると、クラスターは WAITING 状態になります。最後のステップの完了後に自動終了するようクラスターが設定されている場合は、TERMINATING 状態になり、次に TERMINATED 状態になります。クラスターが待機するように設定されている場合は、不要になったときに手動でシャットダウンする必要があります。クラスターを手動でシャットダウンすると、クラスターは TERMINATING 状態になり、次に TERMINATED 状態になります。

クラスターのライフサイクル中にエラーが発生すると、削除保護を有効にしていない限り、Amazon EMR はクラスターとそのすべてのインスタンスを終了させます。エラーのためにクラスターが終了した場合、そのクラスターに保存されているデータは削除され、クラスターの状態は TERMINATED_WITH_ERRORS に設定されます。削除保護を有効にした場合、クラスターからデータを取得し、削除保護を解除してクラスターを終了できます。詳細については、「終了保護の使用」を参照してください。

次の図は、クラスターのライフサイクルと、ライフサイクルの各ステージが特定のクラスターの状態にどのようにマッピングされるかを表しています。



Amazon EMR を使用する利点

Amazon EMR を使用することには多くのメリットがあります。このセクションでは、このようなメリットの概要と、詳しい情報へのリンクを示します。

トピック

- コスト削減
- AWS インテグレーション
- デプロイ
- スケーラビリティと柔軟性
- 信頼性
- セキュリティ

- モニタリング
- 管理インターフェイス

コスト削減

Amazon EMR の料金は、デプロイするインスタンスタイプと Amazon EC2 インスタンスの数、およびクラスターを起動するリージョンによって異なります。オンデマンド料金は低価格ですが、リザーブドインスタンスまたはスポットインスタンスを購入すると、コストをさらに抑えることができます。スポットインスタンスを利用すると大幅に節約できます。場合によっては、オンデマンド料金の 10 分の 1 になります。

> **Note**
>
> EMR クラスターとともに Amazon S3、Amazon Kinesis、または DynamoDB を使用している場合、Amazon EMR の利用とは別個に請求されるサービスについては追加料金が発生します。

> **Note**
>
> プライベートサブネットに Amazon EMR クラスターを設定するときは、Amazon S3 の VPC エンドポイントも設定することをお勧めします。EMR クラスターが Amazon S3 の VPC エンドポイントのないプライベートサブネットに存在する場合、EMR クラスターと S3 間のトラフィックは VPC 内にとどまらないため、S3 トラフィックに関連する追加の NAT ゲートウェイ料金が発生します。

料金のオプションと詳細については、「Amazon EMR 料金表」を参照してください。

AWS インテグレーション

Amazon EMR AWS は他のサービスと統合して、ネットワーク、ストレージ、セキュリティなどに関連する機能をクラスターに提供します。次のリストに、この統合の例をいくつか示します。

- クラスターのノードを構成するインスタンスの Amazon EC2
- インスタンスを起動する仮想ネットワークを設定するための Amazon Virtual Private Cloud (Amazon VPC)

- 入力データと出力データを保存するための Amazon S3
- Amazon CloudWatch がクラスターのパフォーマンスを監視し、アラームを設定する
- AWS Identity and Access Management (IAM) を使用して権限を設定します。
- AWS CloudTrail サービスへのリクエストを監査するため。
- AWS Data Pipeline クラスターをスケジュールして起動する。
- AWS Lake Formation Amazon S3 データレイク内のデータを検出、カタログ化、保護する

デプロイ

EMR クラスターは、クラスターに送信した処理を実行する EC2 インスタンスで構成されます。クラスターを起動すると、選択したアプリケーション (Apache Hadoop や Spark など) を使用してインスタンスが Amazon EMR により構成されます。クラスターの処理の必要に最も合うインスタンスサイズとタイプを選択します (バッチ処理、レイテンシークエリ、ストリーミングデータ、または大容量データストレージ)。Amazon EMR で利用可能なインスタンスタイプの詳細については、「クラスターハードウェアとネットワークを設定する」を参照してください。

Amazon EMR には、クラスター上のソフトウェアを設定するためのさまざまな方法があります。たとえば、Amazon EMR リリースは、Hadoop などの多用途フレームワークと Hive、Pig、Spark などのアプリケーションを含むアプリケーションセットを選択してインストールできます。いくつかの MapR ディストリビューションの 1 つをインストールすることもできます。Amazon EMR は Amazon Linux を使用するため、yum パッケージマネージャで、またはソースから手動でクラスターにソフトウェアをインストールすることもできます。詳細については、「クラスターソフトウェアを設定する」を参照してください。

スケーラビリティと柔軟性

Amazon EMR には、コンピューティングニーズの変化に合わせてクラスターを拡大または縮小できる柔軟性が備わっています。クラスターのサイズを変更し、ピークワークロード用にインスタンスを追加したり、ピークワークロードが減少したときにインスタンスを削除してコストをコントロールしたりすることができます。詳細については、「実行中のクラスターのサイズを手動で変更する」を参照してください。

Amazon EMR には、複数のインスタンスグループを実行することにより、あるグループでオンデマンドインスタンスを使用して処理能力を保証すると同時に、別のグループでスポットインスタンスを使用してジョブを高速に低コストで完了できるようにするオプションも用意されています。異なるインスタンスタイプを混ぜて、別のインスタンスタイプよりも有利なスポットインスタンスタイプの料

金を活かすこともできます。詳細については、「スポットインスタンスを使用すべき場合」を参照してください。

さらに、Amazon EMR には入力データ、出力データ、中間データに複数のファイルシステムを使用する柔軟性も備わっています。例えば、クラスターのライフサイクル後は保存する必要がないデータを処理するために、クラスターのプライマリノードとコアノードで実行される Hadoop 分散ファイルシステム (HDFS) を選択することができます。Amazon S3 をクラスターで実行されるアプリケーションのデータレイヤーとして使用するために EMR File System (EMRFS) を選択し、コンピューティングとストレージを分離して、データをクラスターのライフサイクル外に保持することができます。EMRFS には、コンピューティングのニーズとストレージのニーズそれぞれに合わせて拡大または縮小できるという利点もあります。コンピューティングのニーズが変化した場合はクラスターのサイズを変更することができ、ストレージのニーズが変化した場合は Amazon S3 を使用することができます。詳細については、「ストレージシステムとファイルシステムで作業する」を参照してください。

信頼性

Amazon EMR は、クラスター内のノードを監視し、障害が発生した場合はインスタンスを自動的に終了して置き換えます。

Amazon EMR には、クラスターの終了方法 (自動または手動) をコントロールする設定オプションが用意されています。クラスターが自動的に終了されるように設定した場合、すべてのステップが完了すると終了されます。これは一時的なクラスターと呼ばれます。一方、クラスターが必要なくなったときに手動で終了を選択できるように、処理が完了した後もクラスターが実行され続けるように設定することもできます。または、クラスターを作成して、インストールされたアプリケーションを直接操作した後、必要なくなった時に手動で終了することもできます。このようなクラスターは長時間稼働クラスターと呼ばれます。

さらに、削除保護を設定し、処理中にエラーや問題が発生した場合にクラスター内のインスタンスが削除されないようにすることもできます。終了保護が有効になると、終了前にインスタンスからデータを回復できます。これらのオプションのデフォルト設定は、クラスターの起動方法 (コンソール、CLI、または API) によって異なります。詳細については、「終了保護の使用」を参照してください。

セキュリティ

Amazon EMR は、IAM や Amazon VPC AWS などの他のサービスや Amazon EC2 キーペアなどの機能を活用して、クラスターとデータを保護するのに役立ちます。

IAM

Amazon EMR は、アクセス権限を管理するため IAM と統合されています。アクセス権限は、ユーザーまたは IAM グループにアタッチする IAM ポリシーを使用して定義します。ポリシーで定義したアクセス権限により、それらのユーザーまたはグループのメンバーが実行できるアクションと、アクセスできるリソースが決まります。詳細については、「Amazon EMR で IAM が機能する仕組み」を参照してください。

さらに、Amazon EMR は Amazon EMR サービス自体の IAM ロールとインスタンスの EC2 インスタンスプロファイルを使用します。これらのロールは、 AWS サービスとインスタンスがユーザーに代わって他のサービスにアクセスするためのアクセス権限を付与します。Amazon EMR サービスのデフォルトロールと EC2 インスタンスプロファイルのデフォルトロールが存在します。 AWS デフォルトのロールは管理ポリシーを使用します。管理ポリシーは、コンソールから EMR クラスターを初めて起動してデフォルトの権限を選択したときに自動的に作成されます。デフォルト IAM ロールは、 AWS CLIから作成することもできます。代わりに権限を管理したい場合は AWS、サービスとインスタンスプロファイルのカスタムロールを選択できます。詳細については、「AWS のサービスおよびリソースへのアクセス許可を Amazon EMR に付与する IAM サービスロールの設定」を参照してください。

セキュリティグループ

Amazon EMR は、セキュリティグループを使用して、EC2 インスタンスの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をコントロールします。クラスターを起動すると、Amazon EMR はプライマリインスタンスのセキュリティグループと、コアインスタンス/タスクインスタンスによって共有されるセキュリティグループを使用します。Amazon EMR は、クラスター内のインスタンス間の通信を確実にするために、セキュリティグループルールを設定します。オプションで、追加のセキュリティグループを設定し、高度なルールで、プライマリインスタンスとコア/タスクインスタンスにそのグループを割り当てることができます。詳細については、「セキュリティグループを使用してネットワークトラフィックを制御する」を参照してください。

暗号化

Amazon EMR では、Amazon S3 に保存するデータを保護できるようにするため、EMRFS を使用したオプションの Amazon S3 サーバー側の暗号化とクライアント側の暗号化がサポートされます。サーバー側の暗号化を使うと、Amazon S3 はアップロード後にデータを暗号化します。

クライアント側の暗号化を使用すると、暗号化および復号プロセスは EMR クラスターの EMRFS で行われます。クライアント側の暗号化のルートキーは、 AWS Key Management Service (AWS KMS) または独自のキー管理システムを使用して管理します。

詳細については、「EMRFS プロパティを使用して Amazon S3 の暗号化を指定する」を参照してください。

Amazon VPC

Amazon EMR は、Amazon VPC の仮想プライベートクラウド (VPC) におけるクラスターの起動をサポートします。VPC は、 AWS ネットワーク設定とアクセスの高度な側面を制御できるという点で、独立した仮想ネットワークです。詳細については、「ネットワークを設定する」を参照してください。

AWS CloudTrail

Amazon EMR CloudTrail はと統合して、お客様のアカウントによって、 AWS またはお客様のアカウントに代わって行われたリクエストに関する情報を記録します。この情報を使用すると、クラスターにアクセスしたユーザーや日時に加え、リクエストの生成元 IP アドレスを追跡できます。詳細については、「での Amazon EMR API コールのログ記録 AWS CloudTrail」を参照してください。

Amazon EC2 のキーペア

リモートコンピュータとプライマリノードの間で安全な接続を確立することにより、クラスターを監視して操作できます。Secure Shell (SSH) ネットワークプロトコルを使用して接続するか、Kerberos で認証することができます。SSH を使用する場合は Amazon EC2 キーペアが必要です。詳細については、「SSH 認証情報に EC2 キーペアを使用する」を参照してください。

モニタリング

Amazon EMR 管理インターフェイスとログファイルを使用して、障害やエラーなどのクラスターの問題をトラブルシューティングできます。Amazon EMR には、ログファイルを Amazon S3 にアーカイブする機能があるため、クラスターの終了後でもログを保存し、問題のトラブルシューティングを行うことができます。Amazon EMR には、Amazon EMR コンソールで、ステップ、ジョブ、およびタスクに基づいてログファイルを参照するためのオプションのデバッグツールも用意されています。詳細については、「クラスターのログ記録とデバッグを設定する」を参照してください。

Amazon EMR CloudWatch はと統合して、クラスターとクラスター内のジョブのパフォーマンスメトリクスを追跡します。クラスターがアイドル状態かどうかや使用されているストレージの割合など、さまざまなメトリクスに基づいてアラームを設定できます。詳細については、「を使用した Amazon EMR メトリクスのモニタリング CloudWatch」を参照してください。

管理インターフェイス

Amazon EMR とやり取りする方法はいくつかあります。

- コンソール — クラスターの起動と管理に使用できるグラフィカルユーザーインターフェイス。起動するクラスターの詳細をウェブフォームに入力することで指定し、既存のクラスターの詳細を確認して、クラスターのデバッグや終了を行うことができます。コンソールは、最も簡単に Amazon EMR の使用を開始する手段です。プログラミングの知識は必要ありません。コンソールは、https://console.aws.amazon.com/elasticmapreduce/home からオンラインで使用できます。
- AWS Command Line Interface (AWS CLI) — Amazon EMR に接続してクラスターを作成および管理するためにローカルマシンで実行するクライアントアプリケーション。には、Amazon EMR AWS CLI 固有の機能豊富なコマンドセットが含まれています。これを利用すると、クラスターの起動と管理のプロセスをスクリプトで自動化できます。コマンドラインからの作業を好む場合は、を使用するのが最適なオプションです AWS CLI 。詳細については、「https://docs.aws.amazon.com/cli/latest/reference/emr/index.html コマンドリファレンス」の「AWS CLI Amazon EMR」を参照してください。
- Software Development Kit (SDK) — SDK には、Amazon EMR を呼び出してクラスターを作成し、管理する機能が備わっています。これを利用すると、クラスターの作成や管理のプロセスを自動化するアプリケーションを作成できます。Amazon EMR の機能を拡張したりカスタマイズしたりするには、SDK が最適の選択肢です。Amazon EMR は現在、次の SDK で使用可能です。Go、Java、.NET (C# および VB.NET)、Node.js、PHP、Python、および Ruby。これらの SDK の詳細については、「AWSのツール」および「Amazon EMR サンプルコード & ライブラリ」を参照してください。
- Web Service API – JSON を使用して直接ウェブサービスを呼び出すことができる低レベルインターフェイスです。Amazon EMR を呼び出すカスタム SDK を作成するには、この API が最適の選択肢です。詳細については、「Amazon EMR API リファレンス」を参照してください。

Amazon EMR アーキテクチャの概要

Amazon EMR サービスアーキテクチャは複数のレイヤーで構成されており、各レイヤーはクラスターに特定の機能を提供します。このセクションでは、各レイヤーとコンポーネントの概要を紹介します。

このトピックの内容

- データ処理フレームワーク
- アプリケーションとプログラム

[Storage (ストレージ)]

ストレージレイヤーには、クラスターで使用されるさまざまなファイルシステムが含まれています。次のように、さまざまなタイプのストレージオプションがあります。

Hadoop Distributed File System (HDFS)

Hadoop Distributed File System (HDFS) は Hadoop が採用する、分散型のスケーラブルなファイルシステムです。HDFS はデータをクラスター内の各インスタンスに分散して保存します。データの複数のコピーが複数のインスタンスに保存されるため、個々のインスタンスが障害を起こしてもデータが失われることはありません。HDFS はエフェメラルストレージであり、クラスターを終了するときに消去されます。HDFS は、 MapReduce 処理中の中間結果をキャッシュする場合や、大量のランダム I/O が発生するワークロードに役立ちます。

詳細については、このガイドの「インスタンスストレージ」か、Apache Hadoop ウェブサイトの「HDFS User Guide」を参照してください。

EMR ファイルシステム (EMRFS)

EMR ファイルシステム (EMRFS) を使用すると、Amazon EMR で Hadoop が拡張され、まるで HDFS などのファイルシステムのように、Amazon S3 に保存されたデータに直接アクセスできます。クラスターではファイルシステムとして HDFS または Amazon S3 のいずれかを使用できます。ほとんどの場合、Amazon S3 は入力データおよび出力データを格納する場合に使用され、中間結果は HDFS に格納されます。

ローカルファイルシステム

ローカルファイルシステムとは、ローカルに接続されているディスクを指します。Hadoop クラスターを作成すると、インスタンスストアと呼ばれる、あらかじめアタッチされたディスクストレージのブロックが事前設定されている Amazon EC2 インスタンスから、各ノードが作成されます。インスタンスストアボリューム上のデータは、Amazon EC2 インスタンスのライフサイクル中のみ使用できます。

クラスターリソース管理

リソース管理レイヤーは、クラスターリソースの管理とデータを処理するジョブのスケジューリングを行います。

デフォルトでは、Amazon EMR は Apache Hadoop 2.0 に導入されたコンポーネントである YARN (Yet Another Resource Negotiator) を使用し、複数のデータ処理フレームワークのクラスターリソースを集中管理します。ただし、Amazon EMR で提供されているフレームワークやアプリケーションの中には、リソースマネージャーとして YARN を使用しないものも存在します。Amazon EMR の各ノードには、YARN コンポーネントの管理、クラスターの正常な状態の維持、Amazon EMR とのやり取りを行うエージェントもあります。

スポットインスタンスはタスクノードの実行に使用されることが多いため、Amazon EMR には、タスクノードが終了しても実行中のジョブが失敗しないように YARN ジョブをスケジュールするための機能がデフォルトで備えられています。Amazon EMR は、アプリケーションマスタープロセスをコアノードでのみ実行できるようにすることで、これを実現しています。アプリケーションマスタープロセスは実行中のジョブを制御し、ジョブが有効である間は存続する必要があります。

Amazon EMR リリース 5.19.0 以降では、組み込みの YARN ノードラベル機能を使用して、これを実現しています。(以前のバージョンではコードパッチを使用していました)。`yarn-site` と `capacity-scheduler` の設定分類のプロパティは、YARN capacity-scheduler と fair-scheduler がノードラベルを利用できるように、デフォルトで設定されます。Amazon EMR は、`CORE` ラベルでコアノードに自動的にラベルを付け、アプリケーションマスターが CORE ラベルを持つノードでのみスケジュールされるようにプロパティを設定します。yarn-site および capacity-scheduler 設定分類の関連プロパティを手動で変更したり、関連する XML ファイルで直接変更したりすると、この機能が停止したり、この機能が変更されたりする可能性があります。

データ処理フレームワーク

データ処理フレームワークレイヤーは、データの処理と分析に使用されるエンジンです。YARN で実行されるフレームワークや、独自のリソース管理を持つフレームワークも多数用意 バッチ、インタラクティブ、メモリ内、ストリーミングなど、処理のニーズの種類に合わせて異なるフレームワークを使用できます。選択するフレームワークは、ユースケースによって異なります。これは、アプリケーションレイヤー (処理するデータの操作に使用されるレイヤー) から使用可能な言語とインターフェイスに影響を与えます。Amazon EMR で使用できる主な処理フレームワークは Hadoop MapReduce と Spark です。

Hadoop MapReduce

Hadoop MapReduce は分散コンピューティング用のオープンソースプログラミングモデルです。これは、Map と Reduce という 2 つの機能を提供しながら、全てのロジックを処理することで並行分散アプリケーションを書くプロセスを単純化するものです。Map 機能は「中間結果」と呼ばれるキーと値のペアにデータをマップします。Reduce 機能は中間結果を集計し、追加アルゴリズムを適用して、最終出力を発生させます。Map や Reduce プログラムを自動的に生成する Hive など MapReduce、複数のフレームワークが利用できます。

詳細については、Apache Hadoop Wiki ウェブサイトの「How map and reduce operations are actually carried out」を参照してください。

Apache Spark

Spark は、ビッグデータワークロードの処理に役立つクラスターフレームワークおよびプログラミングモデルです。Hadoop と同様に MapReduce、Spark はオープンソースの分散処理システムですが、実行プランには有向非循環グラフを使用し、データセットにはメモリ内キャッシュを使用します。Spark を Amazon EMR で実行すると、EMRFS を使用して Amazon S3 内のデータに直接アクセスできます。Spark では、SparkSQL などの複数のインタラクティブクエリモジュールがサポートされます。

詳細については、「Amazon EMR リリース ガイド」の「Amazon EMR クラスターでの Apache Spark」を参照してください。

アプリケーションとプログラム

Amazon EMR では、Hive、Pig、Spark Streaming ライブラリなどの多くのアプリケーションがサポートされ、高レベルな言語を使用したワークロード処理の作成、機械学習アルゴリズムの利用、ストリーミング処理アプリケーションの作成、データウェアハウスの構築などの機能が提供されます。加えて、Amazon EMR では、YARN を使用する代わりに独自のクラスター管理機能を持つオープンソースプロジェクトもサポートされます。

さまざまなライブラリと言語を使用して、Amazon EMR で実行したアプリケーションを操作します。たとえば、Java、Hive、Pig を Spark と一緒に使用したり、Spark ストリーミング、Spark SQL、MLlib、GraphX を Spark MapReduce と一緒に使用したりできます。

詳細については、「Amazon EMR リリースガイド」を参照してください。

Amazon EMR のセットアップ

Amazon EMR クラスターを初めて起動する前に、このセクションのタスクを完了してください。

Amazon EMR を初めて使用する場合は、事前に以下のタスクを実行してください。

にサインアップ AWS アカウント

をお持ちでない場合は AWS アカウント、次の手順を実行して作成してください。

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、通話呼び出しを受け取り、電話キーパッドで検証コードを入力するように求められます。

   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。セキュリティ上のベストプラクティスとして、ユーザーに管理アクセスを割り当て、root ユーザーアクセスを必要とするタスクを実行するときは root ユーザーのみを使用してください。

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

管理者権限を持つユーザーを作成します。

にサインアップしたら AWS アカウント、日常のタスクに root ユーザーを使用しないように AWS IAM Identity Center、管理ユーザーを保護し、有効にしてから作成してください。 AWS アカウントのルートユーザー

セキュリティを確保してください。 AWS アカウントのルートユーザー

1. [Root user] を選択し、 AWS アカウント メールアドレスを入力して、AWS Management Consoleアカウント所有者としてにログインします。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 root 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。

管理者権限を持つユーザーを作成します。

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 Identity Center ユーザーガイド」の「AWS IAM Identity Centerの有効化」を参照してください。

2. IAM Identity Center で、ユーザーに管理アクセスを付与します。

   IAM アイデンティティセンターディレクトリ をアイデンティティソースとして使用するチュートリアルについては、『ユーザーガイド』の「 IAM アイデンティティセンターディレクトリデフォルトでのユーザーアクセスの設定」を参照してください。AWS IAM Identity Center

管理者権限を持つユーザーとしてサインインします。

- 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 アクセスポータルへのサインイン」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ます。

1. IAM Identity Center で、最小権限の権限を適用するというベストプラクティスに従った権限セットを作成します。

   手順については、『ユーザーガイド』の「権限セットの作成」を参照してください。AWS IAM Identity Center

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

手順については、『AWS IAM Identity Center ユーザーガイド』の「グループの追加」を参照してください。

SSH 用の Amazon EC2 キーペアを作成する

> **Note**
>
> Amazon EMR リリースバージョン 5.10.0 以降では、ユーザーを認証してクラスターに SSH 接続できるように Kerberos を設定できます。詳細については、「Amazon EMR での認証に Kerberos を使用する」を参照してください。

Secure Shell (SSH) プロトコルを使用した安全なチャネル経由でクラスター内のノードを認証して接続するため、クラスターを起動する前に Amazon Elastic Compute Cloud (Amazon EC2) キーペアを作成します。キーペアを指定せずにクラスターを作成することもできます。この手順は、通常、自動的に起動および作動して段階的処理を実行した後に終了する一時クラスターの場合に実施します。

状況	作業内容
使用する Amazon EC2 キーペアが既にあるか、クラスターへの認証が必要ない。	この手順をスキップしてください。
キーペアを作成する必要がある。	「Amazon EC2 を使用してキーペアを作成する」を参照してください。

次のステップ

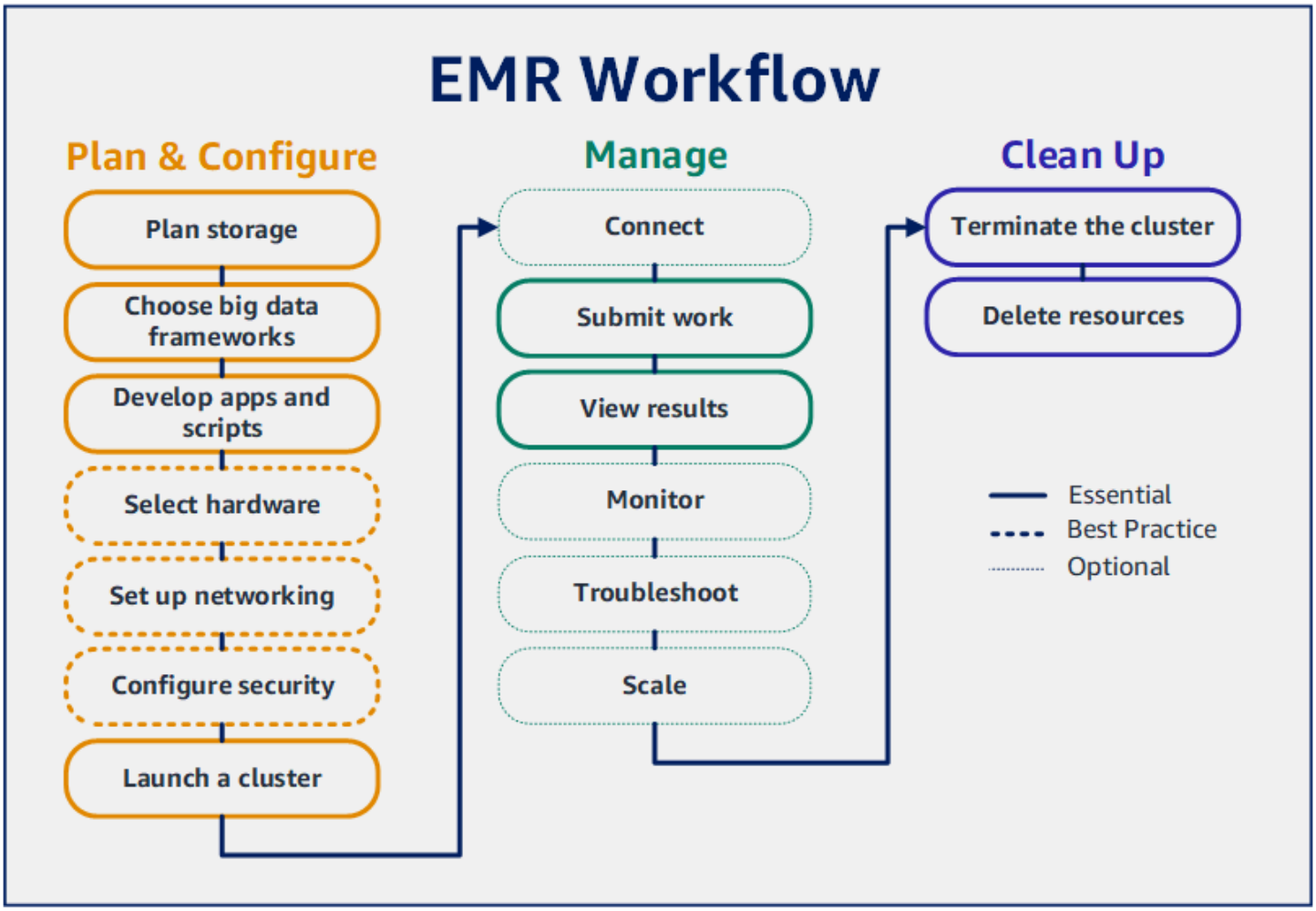
- サンプルクラスターの作成のガイダンスについては、「チュートリアル: Amazon EMR の使用開始」を参照してください。
- カスタムクラスターを設定し、カスタムクラスターへのアクセスを制御する方法の詳細については、「クラスターの計画と設定」と「Amazon EMR でのセキュリティ」を参照してください。

チュートリアル: Amazon EMR の使用開始

概要

Amazon EMR を使用すると、ビッグデータフレームワークを使用してデータを処理および分析するクラスターをわずか数分でセットアップできます。このチュートリアルでは、Spark を使用してサンプルクラスターを起動する方法と、Amazon S3 バケットに保存されているシンプルな PySpark スクリプトを実行する方法について説明します。計画と設定、管理、およびクリーンアップという 3 つの主要なワークフローカテゴリにおける Amazon EMR の必須タスクを取り上げます。

チュートリアルの途中には詳細なトピックへのリンクがあります。また、「次のステップ」セクションに追加手順の概要が記載されています。ご質問や不明点がある場合は、ディスカッションフォーラムに投稿して Amazon EMR チームにお問い合わせください。



前提条件

- Amazon EMR クラスターを起動する前に、「Amazon EMR のセットアップ」のタスクを完了していることを確認してください。

コスト

- 作成するサンプルクラスターは、ライブ環境で実行されます。クラスターには最低料金が発生します。追加料金が発生しないように、このチュートリアルの最後の手順で必ずクリーンアップタスクを完了してください。料金は、Amazon EMR の料金に従って秒単位で発生します。料金はリージョンによっても異なります。詳細については、「Amazon EMR の料金」を参照してください。
- Amazon S3 に保存する小さなファイルについて、最低料金が発生する場合があります。 AWS 無料利用枠の使用制限内であれば、Amazon S3 の一部またはすべての料金が免除される場合があります。詳細については、「Amazon S3 の料金」と「AWS 無料利用枠」を参照してください。

ステップ 1: Amazon EMR クラスターを計画して設定する

Amazon EMR 用のストレージを準備する

Amazon EMR を使用するときに、入力データ、出力データ、およびログファイルの保存先をさまざまなファイルシステムから選択できます。このチュートリアルでは、EMRFS を使用して S3 バケットにデータを保存します。EMRFS は、Amazon S3 に対する通常のファイルの読み書きを可能にする Hadoop ファイルシステムの実装です。詳細については、「ストレージシステムとファイルシステムで作業する」を参照してください。

このチュートリアル用にバケットを作成するには、「Amazon Simple Storage Service ユーザーガイド」の「S3 バケットを作成する方法」に従ってください。Amazon EMR クラスターを起動するリージョンと同じ AWS リージョンにバケットを作成します。たとえば、米国西部 (オレゴン) の us-west-2 です。

Amazon EMR で使用するバケットとフォルダには次の制限があります。

- 名前に使用できるのは、小文字、数字、ピリオド (.)、およびハイフン (-) のみです。
- 名前の末尾を数字にすることはできません。
- バケット名はすべての AWS アカウントで一意である必要があります。
- 出力フォルダは空である必要があります。

Amazon EMR の入力データを使用してアプリケーションを準備する

Amazon EMR のアプリケーションを準備する最も一般的な方法は、アプリケーションとその入力データを Amazon S3 にアップロードすることです。次に、クラスターに作業内容を送信するときに、スクリプトとデータを保存する Amazon S3 の場所を指定します。

このステップでは、サンプル PySpark スクリプトを Amazon S3 バケットにアップロードします。使用する PySpark スクリプトを用意しました。このスクリプトは食品施設の検査データを処理し、S3 バケットに結果ファイルを返します。結果ファイルには、「赤」タイプの違反が最も多い上位 10 施設がリストされます。

また、 PySpark スクリプトが処理するサンプル入力データを Amazon S3 にアップロードします。入力データは、ワシントン州キング郡にある保健局の 2006 ~ 2020 年の検査結果の修正版です。詳細については、「King County Open Data: Food Establishment Inspection Data」を参照してください。データセットのサンプル行を次に示します。

```
name, inspection_result, inspection_closed_business, violation_type, violation_points
100 LB CLAM, Unsatisfactory, FALSE, BLUE, 5
100 PERCENT NUTRICION, Unsatisfactory, FALSE, BLUE, 5
7-ELEVEN #2361-39423A, Complete, FALSE, , 0
```

EMR のサンプル PySpark スクリプトを準備するには

1. 以下のコード例を任意のエディタの新しいファイルにコピーします。

```
import argparse

from pyspark.sql import SparkSession

def calculate_red_violations(data_source, output_uri):
    """
    Processes sample food establishment inspection data and queries the data to
 find the top 10 establishments
    with the most Red violations from 2006 to 2020.

    :param data_source: The URI of your food establishment data CSV, such as 's3://
DOC-EXAMPLE-BUCKET/food-establishment-data.csv'.
    :param output_uri: The URI where output is written, such as 's3://DOC-EXAMPLE-
BUCKET/restaurant_violation_results'.
    """
```

```
    with SparkSession.builder.appName("Calculate Red Health
 Violations").getOrCreate() as spark:
        # Load the restaurant violation CSV data
        if data_source is not None:
            restaurants_df = spark.read.option("header", "true").csv(data_source)

        # Create an in-memory DataFrame to query
        restaurants_df.createOrReplaceTempView("restaurant_violations")

        # Create a DataFrame of the top 10 restaurants with the most Red violations
        top_red_violation_restaurants = spark.sql("""SELECT name, count(*) AS
 total_red_violations
          FROM restaurant_violations
          WHERE violation_type = 'RED'
          GROUP BY name
          ORDER BY total_red_violations DESC LIMIT 10""")

        # Write the results to the specified output URI
        top_red_violation_restaurants.write.option("header",
 "true").mode("overwrite").csv(output_uri)

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        '--data_source', help="The URI for you CSV restaurant data, like an S3
 bucket location.")
    parser.add_argument(
        '--output_uri', help="The URI where output is saved, like an S3 bucket
 location.")
    args = parser.parse_args()

    calculate_red_violations(args.data_source, args.output_uri)
```

2. health_violations.py という名前でファイルを保存します。
3. health_violations.py を、このチュートリアル用に作成した Amazon S3 のバケットにアップロードします。手順については、「Amazon Simple Storage Service ユーザーガイド」の「バケットにオブジェクトをアップロードする」を参照してください。

EMR 用のサンプル入力データを準備するには

1. zip ファイル food_establishment_data.zip をダウンロードします。

2. `food_establishment_data.zip` を解凍し、ご使用のマシンに `food_establishment_data.csv` として保存します。
3. この CSV ファイル を、このチュートリアル用に作成した S3 バケットにアップロードします。手順については、「Amazon Simple Storage Service ユーザーガイド」の「バケットにオブジェクトをアップロードする」を参照してください。

EMR 用データのセットアップに関する詳細は、「入力データを準備する」を参照してください。

Amazon EMR クラスターを起動する

ストレージの場所とアプリケーションを準備したら、サンプルの Amazon EMR クラスターを起動できます。このステップでは、最新の Amazon EMR リリースバージョンを使用して、Apache Spark クラスターを起動します。

Console

コンソールで Spark がインストールされたクラスターを起動するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で、[クラスター] を選択し、[クラスターの作成] を選択します
3. [クラスターの作成] ページで、[リリース]、[インスタンスタイプ]、[インスタンス数]、および [アクセス許可] のデフォルト値を書き留めます。これらのフィールドには、汎用クラスターで機能する値が自動的に入力されます。
4. [クラスター名] フィールドに、クラスターを識別しやすい一意のクラスター名 (*`My first cluster`* など) を入力します。クラスター名に <、>、$、|、` (バックティック) の文字を含めることはできません。
5. [アプリケーション] で、[Spark] オプションをクリックして、クラスターに Spark をインストールします。

> **Note**
>
> Amazon EMR クラスターを起動する前に、クラスターで必要なアプリケーションを選択してください。起動後は、クラスターに対するアプリケーションの追加や削除はできません。

6. [クラスターログ] で、[クラスター固有のログを Amazon S3 に公開] チェックボックスを選択します。[Amazon S3 ロケーション] の値は、作成した Amazon S3 バケットに置き換え、その後に **/logs** を追加します。例えば **s3://DOC-EXAMPLE-BUCKET/logs** です。**/logs** を追加すると、バケットに「logs」という新しいフォルダが作成されます。Amazon EMR によってここにクラスターのログファイルがコピーされます。
7. [セキュリティ設定とアクセス許可] で、[EC2 キーペア] を選択します。同じセクションで、Amazon EMR のサービスロールドロップダウンメニューを選択し、EMR_DefaultRoleを選択します。次に、インスタンスプロファイルの IAM ロールドロップダウンメニューを選択し、EMR_EC2_DefaultRole を選択します。
8. [クラスターの作成] を選択して、クラスターを起動し、クラスターの詳細ページを開きます。
9. クラスター名の横のクラスターの [ステータス] を見つけます。Amazon EMR によるクラスターのプロビジョニングに伴って、ステータスが [開始中] から [実行中]に、そして [待機中] に変わります。ステータスの更新を確認するには、右側にある更新アイコンを選択するか、ブラウザを更新する必要があります。

クラスターが起動して実行中になり、作業を受け付ける準備ができると、ステータスが [待機中] に変わります。クラスターの概要の読み込みの詳細については、「クラスターステータスと詳細の表示」を参照してください。クラスターのステータスの詳細については、「クラスターライフサイクルについて」を参照してください。

CLI

を使用して Spark がインストールされたクラスターを起動するには AWS CLI

1. 以下のコマンドを使用して、クラスターの作成に使用できる IAM のデフォルトロールを作成します。

```
aws emr create-default-roles
```

`create-default-roles` に関する詳細は、「AWS CLI Command Reference」を参照してください。

2. 次のコマンドを使用して Spark クラスターを作成します。`--name` オプションを使用してクラスターの名前を入力し、`--ec2-attributes` オプションを使用して EC2 キーペアの名前を指定します。

```
aws emr create-cluster \
```

```
--name "<My First EMR Cluster>" \
--release-label <emr-5.36.2> \
--applications Name=Spark \
--ec2-attributes KeyName=<myEMRKeyPairName> \
--instance-type m5.xlarge \
--instance-count 3 \
--use-default-roles
```

その他の必須値 --instance-type、--instance-count、および --use-default-roles にも注意してください。これらの値は、汎用クラスター向けに選択されています。create-cluster に関する詳細は、「AWS CLI Command Reference」を参照してください。

> **Note**
>
> 読みやすくするために、Linux 行連続文字 (\) が含まれています。Linux コマンドでは、これらは削除することも、使用することもできます。Windows の場合、削除するか、キャレット (^) に置き換えてください。

次のような出力が表示されます。新しいクラスターの ClusterId と ClusterArn が出力に表示されます。ClusterId を書き留めてください。ClusterId は、クラスターのステータスの確認と、作業の送信に使用します。

```
{
    "ClusterId": "myClusterId",
    "ClusterArn": "myClusterArn"
}
```

3. 次のコマンドを使用して、クラスターのステータスを確認します。

```
aws emr describe-cluster --cluster-id <myClusterId>
```

新しいクラスター用のオブジェクト Status がある次のような出力が表示されます。

```
{
    "Cluster": {
        "Id": "myClusterId",
        "Name": "My First EMR Cluster",
        "Status": {
```

```
            "State": "STARTING",
            "StateChangeReason": {
                "Message": "Configuring cluster software"
            }
        }
    }
}
```

Amazon EMR によるクラスターのプロビジョニングに伴って、State 値が STARTING から RUNNING に、そして WAITING に変わります。

クラスターが起動して実行中になり、作業を受け付ける準備ができると、ステータスが **WAITING** に変わります。クラスターのステータスの詳細については、「クラスターライフサイクルについて」を参照してください。

ステップ 2: Amazon EMR クラスターを管理する

Amazon EMR に作業を送信する

クラスターを起動したら、データを処理して分析するために、実行中のクラスターに作業を送信できます。作業は、ステップとして Amazon EMR クラスターに送信します。ステップとは、1 つ以上のアクションで構成される作業の単位です。たとえば、値の計算や、データの転送と処理のためにステップを送信することが考えられます。ステップは、クラスターの起動時にも、実行中のクラスターに対しても送信できます。チュートリアルのこの部分では、実行中のクラスターにステップとして health_violations.py を送信します。ステップの詳細については、「クラスターへの作業の送信」を参照してください。

Console

コンソールを使用してステップとして Spark アプリケーションを送信するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で [クラスター] を選択し、作業を送信するクラスターを選択します。クラスターの状態は [待機中] である必要があります。
3. [ステップ] タブを選択し、[ステップの追加] を選択します。
4. 次のガイドラインに従ってステップを設定します。

- [タイプ] で、[Spark アプリケーション] を選択します。[デプロイモード]、[アプリケーションの場所]、および [Spark-submit オプション] のフィールドが追加で表示されます。
- [名前] に新しい名前を入力します。クラスターに多数のステップがある場合は、それぞれに名前を付けると追跡しやすくなります。
- [デプロイモード] は、デフォルト値 [クラスターモード] のままにします。Spark のデプロイモードの詳細については、Apache Spark ドキュメントの「Cluster Mode Overview」を参照してください。
- [アプリケーションの場所] には、Amazon S3 内の `health_violations.py` スクリプトの場所を入力します (*s3://DOC-EXAMPLE-BUCKET/health_violations.py* など)。
- [Spark-submit オプション] フィールドは空白のままにします。`spark-submit` オプションの詳細については、「Launching applications with spark-submit」を参照してください。
- [引数] フィールドに、次の引数と値を入力します。

```
--data_source s3://DOC-EXAMPLE-BUCKET/food_establishment_data.csv
--output_uri s3://DOC-EXAMPLE-BUCKET/myOutputFolder
```

s3://DOC-EXAMPLE-BUCKET/food_establishment_data.csv を「Amazon EMR の入力データを使用してアプリケーションを準備する」で準備した入力データの S3 バケット URI に置き換えます。

DOC-EXAMPLE-BUCKET をこのチュートリアル用に作成したバケットの名前に置き換え、をクラスター出力フォルダの名前*myOutputFolder*に置き換えます。

- [ステップが失敗した場合のアクション] では、デフォルトのオプション [続行] を使用します。これにより、ステップが失敗してもクラスターは引き続き実行されます。

5. [追加] を選択して、ステップを送信します。ステップが、[保留中] というステータスでコンソールに表示されます。
6. ステップのステータスをモニタリングします。[保留中] から [実行中]、[完了] に変わります。コンソール内のステータスを更新するには、[フィルター] の右にある更新アイコンを選択します。スクリプトの実行には約 1 分間かかります。ステータスが [完了済み] に変わると、ステップは正常に完了しています。

CLI

を使用してステップとして Spark アプリケーションを送信するには AWS CLI

1. 「Amazon EMR クラスターを起動する」で起動したクラスターの ClusterId がわかっていることを確認します。次のコマンドを使用して、クラスター ID を取得することもできます。

```
aws emr list-clusters --cluster-states WAITING
```

2. add-steps コマンドで ClusterId を指定して、ステップとして health_violations.py を送信します。

   - ステップの名前を指定するには、*"My Spark Application"* を置き換えます。Args 配列の *s3://DOC-EXAMPLE-BUCKET/health_violations.py* を health_violations.py アプリケーションの場所に置き換えます。
   - *s3://DOC-EXAMPLE-BUCKET/food_establishment_data.csv* を food_establishment_data.csv データセットの S3 の場所に置き換えます。
   - *s3://DOC-EXAMPLE-BUCKET/MyOutputFolder* を、指定したバケットの S3 パスとクラスター出力フォルダの名前に置き換えます。
   - ActionOnFailure=CONTINUE は、ステップが失敗してもクラスターを引き続き実行することを指定します。

```
aws emr add-steps \
--cluster-id <myClusterId> \
--steps Type=Spark,Name="<My Spark
 Application>",ActionOnFailure=CONTINUE,Args=[<s3://DOC-EXAMPLE-
BUCKET/health_violations.py>,--data_source,<s3://DOC-EXAMPLE-BUCKET/
food_establishment_data.csv>,--output_uri,<s3://DOC-EXAMPLE-BUCKET/
MyOutputFolder>]
```

CLI を使用したステップの送信の詳細については、「AWS CLI コマンドリファレンス」を参照してください。

ステップを送信すると、StepIds がリストされた次のような出力が表示されます。ステップを 1 つ送信したので、1 つの ID のみがリストされます。ステップ ID をコピーします。ステップ ID は、ステップのステータスを確認するために使用します。

```
{
    "StepIds": [
        "s-1XXXXXXXXXXA"
    ]
}
```

3. describe-step コマンドを使用して、ステップのステータスのクエリを実行します。

```
aws emr describe-step --cluster-id <myClusterId> --step-id <s-1XXXXXXXXXXA>
```

ステップに関する情報が記載された、次のような出力が表示されます。

```
{
    "Step": {
        "Id": "s-1XXXXXXXXXXA",
        "Name": "My Spark Application",
        "Config": {
            "Jar": "command-runner.jar",
            "Properties": {},
            "Args": [
                "spark-submit",
                "s3://DOC-EXAMPLE-BUCKET/health_violations.py",
                "--data_source",
                "s3://DOC-EXAMPLE-BUCKET/food_establishment_data.csv",
                "--output_uri",
                "s3://DOC-EXAMPLE-BUCKET/myOutputFolder"
            ]
        },
        "ActionOnFailure": "CONTINUE",
        "Status": {
            "State": "COMPLETED"
        }
    }
}
```

ステップの State は、ステップの実行に伴って PENDING から RUNNING、さらには COMPLETED へと変わります。このステップの実行には約 1 分間かかるため、ステータスを数回確認することが必要な場合があります。

State が **COMPLETED** に変わることで、ステップが正常に終了したことがわかります。

ステップのライフサイクルに関する詳細は、「ステップの実行によるデータの処理」を参照してください。

結果を表示する

ステップが正常に実行されると、Amazon S3 の出力フォルダで出力結果を表示できます。

health_violations.py の結果を表示するには

1. https://console.aws.amazon.com/s3/でAmazon S3 コンソールを開きます。
2. [バケット名] を選択し、次に、ステップの送信時に指定した出力フォルダを選択します。例えば、*DOC-EXAMPLE-BUCKET* と ですmyOutputFolder。
3. 出力フォルダーに次の項目が表示されることを確認します。
   - _SUCCESS という名前の小さなサイズのオブジェクト。
   - 結果が含まれている、プレフィックス part- で始まる CSV ファイル。
4. 結果を含むオブジェクトを選択し、[ダウンロード] をクリックして結果をローカルファイルシステムに保存します。
5. 任意の工ディタで、結果を開きます。出力ファイルには、赤の違反が最も多い上位 10 施設がリストされます。出力ファイルには、各施設に対する赤の違反の総数も表示されます。

   health_violations.py 結果の例を次に示します。

```
name, total_red_violations
SUBWAY, 322
T-MOBILE PARK, 315
WHOLE FOODS MARKET, 299
PCC COMMUNITY MARKETS, 251
TACO TIME, 240
MCDONALD'S, 177
THAI GINGER, 153
SAFEWAY INC #1508, 143
TAQUERIA EL RINCONSITO, 134
HIMITSU TERIYAKI, 128
```

Amazon EMR クラスターの出力に関する詳細は、「出力場所を設定する」を参照してください。

(オプション) 実行中の Amazon EMR クラスターに接続する

Amazon EMR を使用する際に、ログファイルの読み取り、クラスターのデバッグ、または Spark シェルなどの CLI ツールの使用のために、実行中のクラスターへの接続が必要になることがあります。Amazon EMR では、Secure Shell (SSH) プロトコルを使用してクラスターに接続できます。このセクションでは、SSH の設定、クラスターへの接続、および Spark のログファイルの表示を行う方法について説明します。クラスターへの接続に関する詳細は、「Amazon EMR クラスターノードへのアクセス認証」を参照してください。

クラスターへの SSH 接続を許可する

クラスターに接続する前に、インバウンド SSH 接続を許可するようにクラスターセキュリティグループを変更する必要があります。Amazon EC2 セキュリティグループは、クラスターへの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を制御する仮想ファイアウォールとして機能します。このチュートリアルのためにクラスターを作成したとき、Amazon EMR によって次のセキュリティグループが作成されました。

ElasticMapReduceマスター

プライマリノードに関連付けられているデフォルトの Amazon EMR マネージドセキュリティグループ。Amazon EMR クラスターでは、プライマリノードは、クラスターを管理する Amazon EC2 インスタンスです。

ElasticMapReduce-スレーブ

コアノードとタスクノードに関連付けられているデフォルトのセキュリティグループ。

Console

コンソールを使用してプライマリセキュリティグループの信頼できるソースの SSH アクセスを許可するには

セキュリティグループを編集するには、クラスターが存在する VPC のセキュリティグループを管理する権限が必要です。詳細については、「IAM ユーザーガイド」の「IAM ユーザーのアクセス許可の変更」と、EC2 セキュリティグループを管理できるようにする「ポリシーの例」を参照してください。

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。

2. 左側のナビゲーションペインの [EMR on EC2] で [クラスター] を選択し、更新するクラスターを選択します。選択すると、クラスターの詳細ページが開きます。このページの [プロパティ] タブは事前に選択されます。
3. [プロパティ] タブの [ネットワーク] で、[EC2 セキュリティグループ (ファイアウォール)] の横にある矢印を選択してこのセクションを展開します。[プライマリノード] で、セキュリティグループリンクを選択します。以下の手順を完了したら、必要に応じてこのステップに戻り、[コアノードとタスクノード] を選択し、以下の手順を繰り返して SSH クライアントがコアノードとタスクノードにアクセスできるようにします。
4. これにより、EC2 コンソールが開きます。[Inbound rules] (インバウンドルール) タブを選択し、[Edit inbound rules] (インバウンドルールの編集) を選択します。
5. 次の設定で、パブリックアクセスを許可するインバウンドルールを確認します。存在する場合は、[Delete] (削除) をクリックして削除します。

   - タイプ

     SSH
   - [ポート]

     22
   - ソース

     Custom 0.0.0.0/0

   > ⚠ Warning
   >
   > 2020 年 12 月以前は、 ElasticMapReduce-master セキュリティグループには、すべてのソースからのインバウンドトラフィックをポート 22 で許可する事前設定済みのルールがありました。このルールは、マスターノードへの最初の SSH 接続を簡素化するために作成されたものです。このインバウンドルールを削除して、信頼できる送信元のみにトラフィックを制限することを強くお勧めします。

6. ルールのリストの下部までスクロールし、[Add Rule] (ルールの追加) を選択します。
7. [Type (タイプ)] で、[SSH] を選択します。SSH を選択すると、[Protocol] (プロトコル) に [TCP] が、[Port Range] (ポート範囲) に [22] が自動的に入力されます。
8. [source] (送信元) には、[My IP] (マイ IP) を選択して、IP アドレスを送信元アドレスとして自動的に追加します。[Custom] (カスタム) で信頼済みクライアントの IP アドレスの範囲を

追加することも、他のクライアントに追加のルールを作成することもできます。多くのネットワーク環境では IP アドレスを動的に割り当てるため、将来的に信頼済みクライアントの IP アドレスを更新することが必要になる場合があります。

9. [保存] を選択します。
10. オプションで、[コアノードとタスクノード] をリストから選択し、上記の手順を繰り返して、コアノードとタスクノードへの SSH クライアントアクセスを許可します。

Old console

コンソールを使用して信頼できるソースにプライマリセキュリティグループへの SSH アクセスを許可するには

セキュリティグループを編集するには、クラスターが存在する VPC のセキュリティグループを管理する権限が必要です。詳細については、「IAM ユーザーガイド」の「IAM ユーザーのアクセス許可の変更」と、EC2 セキュリティグループを管理できるようにする「ポリシーの例」を参照してください。

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. [Clusters] を選択します。変更するクラスターの ID を選択します。
3. ネットワークとセキュリティペインで、EC2 セキュリティグループ (ファイアウォール) ドロップダウンを展開します。
4. プライマリノード で、セキュリティグループを選択します。
5. [Edit inbound rules] (インバウンドルールの編集) を選択します。
6. 次の設定で、パブリックアクセスを許可するインバウンドルールを確認します。存在する場合は、[Delete] (削除) をクリックして削除します。

   - タイプ

     SSH
   - [ポート]

     22
   - ソース

     Custom 0.0.0.0/0

> **Warning**
>
> 2020 年 12 月以前は、ポート 22 ですべてのソースからのインバウンドトラフィックを許可する事前設定済みのルールがありました。このルールは、プライマリノードへの最初の SSH 接続を簡素化するために作成されたものです。このインバウンドルールを削除して、信頼できる送信元のみにトラフィックを制限することを強くお勧めします。

7. ルールのリストの下部までスクロールし、[Add Rule] (ルールの追加) を選択します。
8. [Type (タイプ)] で、[SSH] を選択します。

   SSH を選択すると、[Protocol] (プロトコル) に [TCP] が、[Port Range] (ポート範囲) に [22] が自動的に入力されます。
9. [source] (送信元) には、[My IP] (マイ IP) を選択して、IP アドレスを送信元アドレスとして自動的に追加します。[Custom] (カスタム) で信頼済みクライアントの IP アドレスの範囲を追加することも、他のクライアントに追加のルールを作成することもできます。多くのネットワーク環境では IP アドレスを動的に割り当てるため、将来的に信頼済みクライアントの IP アドレスを更新することが必要になる場合があります。
10. [保存] を選択します。
11. 必要に応じて、ネットワークとセキュリティペインの Core ノードとタスクノードで他のセキュリティグループを選択し、上記のステップを繰り返して、SSH クライアントがコアノードとタスクノードにアクセスできるようにします。

を使用してクラスターに接続する AWS CLI

AWS CLIを使用すると、オペレーティングシステムに関係なく、クラスターへの SSH 接続を作成できます。

を使用してクラスターに接続し、ログファイルを表示するには AWS CLI

1. 次のコマンドを使用して、クラスターへの SSH 接続を開きます。*<mykeypair.key>* を、キーペアファイルの完全修飾パスとファイル名に置き換えてください。例えば `C:\Users\<username>\.ssh\mykeypair.pem` です。

   ```
   aws emr ssh --cluster-id <j-2AL4XXXXXX5T9> --key-pair-file <~/mykeypair.key>
   ```

2. `/mnt/var/log/spark` に移動して、クラスターのマスターノード上の Spark ログにアクセスします。次に、その場所にあるファイルを表示します。マスターノード上の追加のログファイルのリストについては、「プライマリノードのログファイルを表示する」を参照してください。

```
cd /mnt/var/log/spark
ls
```

ステップ 3: Amazon EMR リソースをクリーンアップする

クラスターを終了する

クラスターに作業を送信し、PySpark アプリケーションの結果を確認したので、クラスターを終了できます。クラスターを終了すると、クラスターに関連付けられているすべての Amazon EMR 料金と Amazon EC2 インスタンスが停止します。

クラスターを終了しても、Amazon EMR ではクラスターに関するメタデータが 2 か月間無料で保持されます。アーカイブされたメタデータは、新しいジョブのためのクラスターのクローン作成や、参照目的でのクラスター設定への再アクセスに便利です。メタデータには、クラスターが S3 に書き込むデータや、クラスターの HDFS に格納されるデータは含まれません。

> **Note**
>
> クラスターを終了した後に、Amazon EMR コンソールでリストビューからクラスターを削除することはできません。Amazon EMR によってメタデータがクリアされると、終了したクラスターがコンソールから消えます。

Console

コンソールでクラスターを終了するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. [クラスター] を選択し、終了するクラスターを選択します。
3. [アクション] ドロップダウンメニューで、[クラスターの終了] を選択します。

4. ダイアログボックスで、[終了] を選択します。クラスター設定によっては、終了に 5 ~ 10 分間かかる場合があります。Amazon EMR クラスターの終了に関する詳細は、「クラスターを終了する」を参照してください。

CLI

を使用してクラスターを終了するには AWS CLI

1. 次のコマンドを使用して、クラスターの終了プロセスを開始します。*#myClusterId#* をサンプルクラスターの ID に置き換えます。このコマンドでは、出力は返されません。

```
aws emr terminate-clusters --cluster-ids <myClusterId>
```

2. クラスターの終了プロセスが進行中であることを確認するために、次のコマンドでクラスターのステータスを確認します。

```
aws emr describe-cluster --cluster-id <myClusterId>
```

JSON 形式の出力例を次に示します。クラスターの Status が **TERMINATING** から **TERMINATED** に変わります。クラスター設定によっては、終了に 5 ~ 10 分間かかる場合があります。Amazon EMR クラスターの終了に関する詳細は、「クラスターを終了する」を参照してください。

```
{
    "Cluster": {
        "Id": "j-xxxxxxxxxxxx",
        "Name": "My Cluster Name",
        "Status": {
            "State": "TERMINATED",
            "StateChangeReason": {
                "Code": "USER_REQUEST",
                "Message": "Terminated by user request"
            }
        }
    }
}
```

S3 リソースを削除する

追加料金が発生しないように、Amazon S3 バケットを削除する必要があります。バケットを削除すると、このチュートリアル用のすべての Amazon S3 リソースが削除されます。バケットに含まれている内容は次のとおりです。

- PySpark スクリプト
- 入力データセット
- 出力結果フォルダ
- ログファイルフォルダ

PySpark スクリプトまたは出力を別の場所に保存した場合、保存されたファイルを削除するための追加の手順が必要になる場合があります。

> **Note**
>
> バケットを削除する前に、クラスターを終了する必要があります。そうしないと、バケットを空にできない可能性があります。

バケットを削除するには、「Amazon Simple Storage Service ユーザーガイド」の「S3 バケットを削除する方法」に従ってください。

次のステップ

これで、最初の Amazon EMR クラスターの起動を最初から最後まで実行しました。ビッグデータアプリケーションの準備と送信、結果の表示、クラスターの終了などの、必須 EMR タスクも完了しました。

以降のトピックでは、Amazon EMR ワークフローをカスタマイズする方法について詳しく説明します。

Amazon EMR のビッグデータアプリケーションについて調べる

「Amazon EMR リリース ガイド」で、クラスターにインストールできるビッグデータアプリケーションを確認し、比較します。リリースガイドには、各 EMR リリースバージョンの詳細と、Amazon EMR で Spark や Hadoop などのフレームワークを使用するためのヒントが記載されています。

クラスターのハードウェア、ネットワーク、およびセキュリティを計画する

このチュートリアルでは、詳細オプションを設定せずにシンプルな EMR クラスターを作成しました。詳細オプションでは、Amazon EC2 インスタンスタイプ、クラスターネットワーク、およびクラスターセキュリティを指定できます。要件を満たすクラスターの計画と起動に関する詳細は、「クラスターの計画と設定」と「Amazon EMR でのセキュリティ」を参照してください。

クラスターを管理する

「クラスターを管理する」で、実行中のクラスターの操作について詳しく説明しています。クラスターを管理するために、クラスターに接続し、ステップをデバッグし、クラスターのアクティビティと状態を追跡できます。EMR マネージドスケーリングを使用して、ワークロードの需要に応じてクラスターリソースを調整することもできます。

別のインターフェースを使用する

Amazon EMR コンソールに加えて、、ウェブサービス API AWS Command Line Interface、またはサポートされている多くの AWS SDKsを管理できます。詳細については、「管理インターフェイス」を参照してください。

Amazon EMR クラスターにインストールされているアプリケーションと、さまざまな方法でやり取りすることもできます。Apache Hadoop のようないくつかのアプリケーションでは、表示可能なウェブインターフェイスを公開しています。詳細については、「Amazon EMR クラスターでホストされているウェブインターフェイスを表示する」を参照してください。

EMR テクニカルブログを参照する

新しい Amazon EMR 機能のサンプルチュートリアルと詳細な技術説明については、「AWS Big Data Blog」を参照してください。

Amazon EMR コンソール

コンソールのインターフェイスは更新されており、Amazon EMR 環境を直感的に管理でき、ドキュメント、製品情報、その他のリソースに簡単にアクセスできます。

コンソール機能

Amazon EMR コンソールは、次の URL から入手できます。

- コンソール URL — https://console.aws.amazon.com/emr

次の表は、Amazon EMR コンソールコンポーネントの主要なステータスを示しています。

Amazon EMR コンソールコンポーネント	コンソール	
EMR Studio	✓	
クラスターの作成と管理	✓	
パブリックアクセスをブロックする	✓	
Amazon CloudWatch イベントのモニタリング	✓	
セキュリティ設定	✓	
仮想クラスター (Amazon EMR on EKS)	✓	
Amazon Virtual Private Cloud サブネットの表示と管理 1	✓	
ノートブック 2	✓	

[1] コンソールでは、クラスターを作成するときに Networking セクション内の Amazon VPC サブネットを表示および管理できます。

[2] EMR Notebooks がコンソールの EMR Studio ワークスペースとして使用できます。コンソールの [ワークスペースの作成] ボタンでは、新しいノートブックを作成できます。EMR Notebooks ユーザーが Workspace にアクセスしたり作成したりするには、追加の IAM ロール権限が必要です。詳細については、「Amazon EMR Notebooks はコンソールの Amazon EMR スタジオワークスペース」および「Amazon EMR コンソール」 を参照してください。

相違点の要約

このセクションでは、Amazon EMR コンソールエクスペリエンスの機能の概要を説明します。これらの機能は次のカテゴリに分類されます。

- コンソールのクラスター互換性
- クラスターの作成
- クラスターの詳細を表示または編集する
- クラスターの表示と検索
- セキュリティ設定を操作する場合の相違点

コンソールのクラスター互換性

場合によっては、作成したクラスターがコンソールと互換性がないことがあります。次のリストでは、Amazon EMR コンソールの互換性要件について説明します。

- コンソールは Amazon EMR リリース 5.20.1 以降で作成されたクラスターをサポートします。
- コンソールでは自動スケーリングを使用するクラスターを複製できますが、新しいクラスターを作成できるのは、クラスターを手動でスケーリングする場合やマネージドスケーリングを使用する場合のみです。

リリース 5.20.1 以前のクラスターを作成して使用するには、 AWS Command Line Interface (AWS CLI) または SDK を使用できます。 AWS

クラスターの作成

機能	コンソール	
	プライマリ、コア、タスク	

機能	コンソール	
用語: Amazon EMR クラスターノードタイプ		
Amazon EMR がサポートするリリース [1]	Amazon EMR リリース 5.20.1 以降	
クラスターをすばやく起動する	[サマリー] パネルの下にある [クラスターの作成] ボタンを使用します。クラスター名には <、>、$、|、` (バックティック) の文字は使用できません。	
スポットプロビジョニングタイムアウトの設定	クラスター内のフリートごとにインスタンスをプロビジョニングするためのタイムアウト期間を定義します。	
サービスロールと Amazon EC2 インスタンスプロファイルロール	コンソールではデフォルトのロールは作成されません。IAM コンソールでロールを作成するか、すでに作成されている IAM ロールを選択する必要があります。	
クラスターの可視化	Amazon EMR コンソール内では、クラスターをすべてのユーザーに表示することはできません。クラスターアクセスは IAM ポリシーによって決定されます。	

機能	コンソール	
ネットワーク - プライベートサブネットの設定	Amazon S3 エンドポイントと NAT ゲートウェイは、それぞれの Amazon S3 コンソールと Amazon VPC コンソールから設定する必要があります。	
EMR ファイルシステムの整合性のあるビュー (EMRFS CV、EMR File System consistent view)	2020 年 12 月 1 日に Amazon S3 read-after-write の強力なコンシステンシーがリリースされるので、EMR クラスターで EMRFS CV を使用する必要はありません。	
Debugging	クラスター詳細ページのアプリケーション UI インターフェイスを使用してジョブをデバッグできます。	

[1] コンソールで Amazon EMR 5.20.1 より前のリリースを使用してクラスターを作成または編集することはできませんが、5.20.1 より前のリリースを使用して作成された既存のクラスターは引き続き機能します。5.20.1 より前の Amazon EMR リリースでクラスターを作成および編集するには、API または CLI を使用します。コンソールを使用してすべてのクラスターを表示できますが、5.20.1 より前に作成されたコンソールは新しい機能に対応していない可能性があります。

クラスターの表示と検索

次の表は、Amazon EMR コンソールを使用してクラスターを表示、表示、検索する方法を示しています。

> **Note**
>
> クラスターリストにデータフィルタを適用すると、データベース全体がクエリされます。ただし、検索ボックスにテキスト文字列を入力すると、リストがクライアント側で読み込んだ結果にのみ検索が適用されます。

機能	コンソール	
クラスターの詳細を表示する	[クラスター ID] を選択すると、設定オプション、永続アプリケーション UI、ログなど、クラスターの詳細をすべて表示できます。	
クラスターの検索	1 つの検索フィールドを使用してテキスト検索クエリを入力し、「ステータス = すべてのアクティブステータス」のようなデータフィルターを作成して適用できます。	
障害が発生したクラスターの検索	障害が発生したクラスターを検索するには、[ステータス] = [エラーで終了] フィルターを適用します。	

クラスターの詳細を表示または編集する

機能	コンソール	
インスタンスグループとインスタンスフリート内のイン	[インスタンス] タブで、インスタンスのオプション	

機能	コンソール	
スタンスを、スケーリング、プロビジョニング、サイズ変更、終了オプションと共に表示する	と詳細を表示します。[プロパティ] タブで、終了オプションを表示します。	
アプリ UI、ログ、設定を表示する (Apache Spark UI、Spark 履歴サービス、Apache Tez UI、YARN タイムラインサーバー)	[設定] タブで、クラスター設定を表示します。ライブの永続的なアプリケーション UI を起動して、[アプリケーション] タブからアプリケーションのログを確認します。	
CLI へのクラスターのエクスポート	クラスターの詳細およびリストビューのアクションメニューから「クラスターのクローン作成コマンドを表示」として使用できるオプション	

セキュリティ設定を操作する場合の相違点

機能	コンソール	
セキュリティ設定のクローン作成	✓	
Trino と Apache Ranger を使用したフェデレーテッドガバナンス	✓	
ランタイムロールを使用してクラスターに作業を送信する [1]	✓	

機能	コンソール	
EMR ファイルシステム (EMRFS) データへのアクセスの許可	Amazon S3 アクセスポイント	
AWS Lake Formation アクセス制御	ランタイムロール	

[1] ステップの送信時にロールを渡すには、クラスターで IAM アクセス権限ポリシーがアタッチされたセキュリティ設定を使用する必要があります。これにより、ユーザーは承認されたロールのみを渡すことができ、ジョブは Amazon EMR リソースにアクセスできます。詳細については、「Amazon EMR ステップのランタイムロール」を参照してください。

Amazon EMR Studio

Amazon EMR Studio は、Amazon EMR クラスターで実行される、フルマネージド型の Jupyter Notebook 用のウェブベースの統合開発環境 (IDE) です。チームが R、Python、Scala、および で記述されたアプリケーションを開発、視覚化、デバッグできるように EMR Studio を設定できます PySpark。EMR Studio は、AWS Identity and Access Management (IAM) および IAM Identity Center と統合されているため、ユーザーは企業認証情報を使用してログインできます。

EMR Studio は無料で作成できます。EMR Studio を使用する場合は、Amazon S3 ストレージと Amazon EMR クラスターに適用される料金が適用されます。製品の詳細とハイライトについては、Amazon EMR Studio のサービスページを参照してください。

EMR Studio の主な機能

Amazon EMR Studio には次の機能があります。

- AWS Identity and Access Management (IAM)、または AWS IAM Identity Center (信頼できる ID 伝達およびエンタープライズ ID プロバイダーの有無にかかわらず) でユーザーを認証します。
- Amazon EMR クラスターにオンデマンドでアクセスして起動し、Jupyter Notebook ジョブを実行します。
- Amazon EMR on EKS クラスターに接続して、ジョブ実行として作業を送信する。
- サンプルノートブックを探索して保存する。サンプルノートブックの詳細については、「EMR Studio Notebook examples GitHub repository」を参照してください。
- Python、、Spark Scala PySpark、Spark R、または SparkSQL を使用してデータを分析し、カスタムカーネルとライブラリをインストールします。
- 同じ Workspace 内の他のユーザーとリアルタイムでコラボレーションできます。詳細については、「Workspace コラボレーションを設定する」を参照してください。
- ノートブック内のデータを操作する前に、EMR Studio SQL Explorer を使用してデータカタログを参照し、SQL クエリを実行し、結果をダウンロードします。
- Apache Airflow や Amazon Managed Workflows for Apache Airflow などのオーケストレーションツールを使用して、スケジュールされたワークフローの一部としてパラメータ化されたノートブックを実行する。詳細については、「AWS Big Data Blog」の「Orchestrating analytics jobs on EMR Notebooks using MWAA」を参照してください。
- GitHub や などのコードリポジトリをリンクします BitBucket。

- Spark History Server、Tez UI、または YARN タイムラインサーバーを使用してジョブを追跡およびデバッグする。

EMR Studio も HIPAA 対応であり、HITRUST CSF および SOC 2の下で認定されています。AWS のサービスにおける HIPAA コンプライアンスの詳細については、https://aws.amazon.com/compliance/hipaa-compliance/ を参照してください。AWS のサービスにおける HITRUST CSF コンプライアンスの詳細については、https://aws.amazon.com/compliance/hitrust/ を参照してください。AWS のサービスにおけるその他のコンプライアンスプログラムの詳細については、「コンプライアンスプログラムによる AWS 対象範囲内のサービス」を参照してください。

Amazon EMR Studio の機能履歴

この表に、Amazon EMR Managed Scaling 機能の更新を示します。

リリース日	機能
2024 年 1 月 5 日	(米国東部) および AWS GovCloud （ AWS GovCloud 米国西部) での EMR Studio のサポートを追加しました。
2023 年 11 月 26 日	IAM Identity Center 認証を使用した EMR Studio の信頼できる ID 伝達のサポートが追加されました。
2023 年 10 月 26 日	インタラクティブ機能を備えた EMR Serverless アプリケーションを作成する機能が追加されました。
2023 年 2 月 28 日	EMR Serverless アプリケーションのアプリケーションログストレージの AWS KMS カスタマー管理キーサポートを追加しました。
2023 年 2 月 23 日	EMR Serverless ジョブ送信用の IAM ロールをワンクリックで作成できるようになりました。EMR Serverless アプリケーションのカスタムイメージを選択するときの ECR ルックアップを追加しました。
2023 年 1 月 27 日	ヘッドレス実行ノートブックは、`%execute_notebook` マジックで各セル実行の進行状況を追跡できます。

リリース日	機能
2023 年 1 月 23 日	永続アプリケーションは起動時間を短縮できるよう最適化されています。

Amazon EMR Studio の仕組み

Amazon EMR Studio は、ユーザーのチーム用に作成する Amazon EMR リソースです。EMR Studio は、Amazon EMR クラスターで実行される、Jupyter Notebook 用のウェブベースの自己完結型統合開発環境です。ユーザーは、企業の認証情報を使用して Studio にログインします。

作成する各 EMR Studio では、次の AWS リソースが使用されます。

- サブネットを使用する Amazon Virtual Private Cloud (VPC) - ユーザーは、指定された VPC 内の Amazon EMR および Amazon EMR on EKS クラスターで Studio カーネルとアプリケーションを実行します。EMR Studio は、Studio の作成時に指定したサブネット内の任意のクラスターに接続できます。
- IAM ロールとアクセス許可ポリシー - ユーザーアクセス許可を管理するには、ユーザーの IAM ID またはユーザーロールにアタッチする IAM アクセス許可ポリシーを作成します。EMR Studio は IAM サービスロールとセキュリティグループを使用して、他の AWS のサービスとの相互運用も行います。詳細については、アクセスコントロール および EMR Studio ネットワークトラフィックを制御するセキュリティグループを定義する を参照してください。
- セキュリティグループ - EMR Studio は、セキュリティグループを使用して Studio と EMR クラスターの間に安全なネットワークチャネルを確立します。
- Amazon S3 バックアップの場所 - EMR Studio は Amazon S3 の場所にノートブック作業を保存します。

次の手順は、EMR Studio を作成して管理する方法の概要です。

1. IAM または IAM Identity Center 認証を使用して AWS アカウントで Studio を作成します。手順については、「Amazon EMR Studio を設定する」を参照してください。
2. Studio にユーザーまたはグループを割り当てます。アクセス許可ポリシーを使用して、各ユーザーにきめ細かいアクセス許可を設定します。詳細については、「EMR Studio ユーザーの割り当てと管理」のトピックを参照してください。
3. AWS CloudTrail イベントを使用した EMR Studio アクションのモニタリングを開始します。詳細については、「Amazon EMR Studio アクションをモニタリングする」を参照してください。

4. クラスターテンプレートと Amazon EMR on EKS マネージドエンドポイントで Studio ユーザーにより多くのクラスターオプションを提供します。

認証とユーザーログイン

Amazon EMR Studio は、IAM 認証モードと IAM Identity Center 認証モードという 2 つの認証モードをサポートしています。IAM モードは AWS Identity and Access Management (IAM) を使用し、IAM Identity Center モードは AWS IAM Identity Center を使用します。EMR Studio を作成するときは、その Studio のすべてのユーザーの認証モードを選択します。

IAM 認証モード

IAM 認証モードでは、IAM 認証または IAM フェデレーションを使用できます。

IAM 認証では、IAM のユーザー、グループ、ロールなどの IAM ID を管理できます。IAM アクセス許可ポリシーおよび属性ベースのアクセスコントロール (ABAC) を使用して Studio へのアクセス権をユーザーに付与します。

IAM フェデレーションにより、サードパーティーの ID プロバイダー (IdP) と AWS との間の信頼を確立できます。これにより、IdP を通じてユーザー ID を管理できます。

IAM Identity Center 認証モード

IAM Identity Center 認証モードでは、ユーザーに EMR Studio へのフェデレーションアクセス権を付与できます。IAM Identity Center を使用して、IAM Identity Center ディレクトリ、既存の企業ディレクトリ、または外部 IdP (Azure Active Directory (AD) など) からのユーザーおよびグループを認証できます。次に、ID プロバイダー (IdP) を使用してユーザーを管理します。

EMR Studio では、IAM Identity Center の次の ID プロバイダーの使用をサポートしています。

- AWS Managed Microsoft AD およびセルフマネージド Active Directory — 詳細については、「Connect to your Microsoft AD directory」を参照してください。
- SAML ベースのプロバイダー — 詳細なリストについては、「Supported identity providers」を参照してください。
- IAM Identity Center ディレクトリ - 詳細については、「AWS IAM Identity Center ユーザーガイド」の「IAM Identity Center での ID の管理」および「アプリケーション間での信頼できる ID 伝達」を参照してください。

認証がログインとユーザーの割り当てに与える影響

EMR Studio で選択する認証モードは、ユーザーが Studio にログインする方法、ユーザーを Studio に割り当てる方法、および新しい Amazon EMR クラスターの作成などのアクションを実行できるようにするためにユーザーを許可する (アクセス許可を与える) 方法に影響します。

次の表に、認証モードによる EMR Studio のログイン方法をまとめます。

認証モードによる EMR Studio ログインオプション

認証モード	ログイン方法	説明
• IAM (認証とフェデレーション) • IAM Identity Center	EMR Studio URL	ユーザーは、Studio アクセス URL を使用して Studio にログインします。例えば、`https://xxxxxxxxxxxxxxxxxxxxxx.emrstudio-prod.us-east-1.amazonaws.com` です。 IAM 認証を使用した場合、ユーザーは IAM 認証情報を入力します。IAM フェデレーションまたは IAM Identity Center を使用する場合、EMR Studio は、認証情報を入力するための ID プロバイダーのサインイン URL にユーザーをリダイレクトします。 ID フェデレーションのコンテキストでは、このログインオプションは、サービスプロバイダー (SP) 開始サインインと呼ばれます。
• IAM (フェデレーション) • IAM Identity Center	ID プロバイダー (IdP) ポータル	ユーザーは Azure ポータルなどの ID プロバイダーのポータルにログインし、Amazon EMR コンソールを起動します。Amazon EMR コンソールを起動した後、ユーザーは、[Studios list] (Studio リスト) から Studio を開きます。 また、ユーザーが ID プロバイダーのポータルから特定の Studio にログインできるように、EMR Studio を SAML アプリケーションとして構成することもできます。手順については「IdP ポータルの SAML アプリケーションとして EMR Studio を設定するには」を参照してください。

認証モード	ログイン方法	説明
		ID フェデレーションのコンテキストでは、このログインオプションは、ID プロバイダー (IdP) 開始サインインと呼ばれます。
• IAM (認証)	AWS Management Console	ユーザーは、IAM 認証情報を使用して AWS Management Console にサインインし、Amazon EMR コンソールで [Studios list] (Studio リスト) から Studio を開きます。

次の表に、認証モード別の EMR Studio のユーザー割り当てと認可の概要を示します。

認証モード別の EMR Studio ユーザー割り当てと認可

[Authentication mode] (認証モード)	ユーザー割り当て	ユーザー認可
IAM (認証とフェデレーション)	IAM ID (ユーザー、グループ、ロール) にアタッチされた IAM アクセス許可ポリシーで `CreateStudioPresignedUrl` アクションを許可します。 フェデレーティッドユーザーの場合、フェデレーションに使用する IAM ロールに設定するアクセス許可ポリシーで IAM での `CreateStudioPresignedUrl` アクションを許可します。 属性ベースのアクセスコントロール (ABAC) を使用して、ユーザーがアクセスできる 1 つ以上の Studio を指定します。	特定の EMR Studio アクションを許可する IAM アクセス許可ポリシーを定義します。 ネイティブユーザーの場合、IAM ID (ユーザー、グループ、ロール) に IAM アクセス許可ポリシーをアタッチします。フェデレーティッドユーザーの場合、フェデレーションに使用する IAM ロールに設定するアクセス許可ポリシーで Studio アクションを許可します。 詳細については、「Amazon EC2 または Amazon EKS 用の EMR Studio ユーザーアクセス許可の設定」を参照してください。

[Authentication mode] (認証モード)	ユーザー割り当て	ユーザー認可
	手順については、「EMR Studio にユーザーまたはグループを割り当てる」を参照してください。	
IAM Identity Center	`IdCUserAssignment` を `REQUIRED` に設定して作成した Studio では、特定のセッションポリシーでユーザーを Studio にマッピングします。詳細については、「EMR Studio にユーザーまたはグループを割り当てる」を参照してください。 `IdCUserAssignment` を `OPTIONAL` に設定して作成した Studio では、Identity Center のユーザーまたはグループなら誰でも Studio にアクセスできます。	オプション: 特定の EMR Studio アクションを許可する IAM セッションポリシーを定義します。ユーザーを Studio に割り当てるときに、セッションポリシーをユーザーにマッピングします。 詳細については、「IAM Identity Center 認証モードのユーザーアクセス許可」を参照してください。

アクセスコントロール

Amazon EMR Studio では、AWS Identity and Access Management (IAM) ID ベースのポリシーを使用してユーザー認可 (アクセス許可) を設定します。これらのポリシーでは、許可するアクションとリソース、およびアクションを許可する条件を指定します。

IAM 認証モードのユーザーアクセス許可

EMR Studio で IAM 認証を使用するときにユーザーアクセス許可を設定するには、IAM アクセス許可ポリシーで `elasticmapreduce:RunJobFlow` のようなアクションを許可します。使用するアクセス許可ポリシーを 1 つ以上作成できます。例えば、ユーザーが新しい Amazon EMR クラスターを作成することを許可しない基本ポリシーと、クラスターの作成を許可する別のポリシーを作成できます。Studio のすべてのアクションのリストについては、「EMR Studio ユーザーの AWS Identity and Access Management アクセス許可」を参照してください。

IAM Identity Center 認証モードのユーザーアクセス許可

IAM Identity Center 認証を使用する場合、単一の EMR Studio ユーザーロールを作成します。ユーザーロールは、ユーザーがログインしたときに Studio が担う専用の IAM ロールです。

IAM セッションポリシーを EMR Studio ユーザーロールにアタッチします。セッションポリシーは、Studio ログインセッション中にフェデレーティッドユーザーが実行できる操作を制限する特別な種類の IAM アクセス許可ポリシーです。セッションポリシーを使用すると、EMR Studio の複数のユーザーロールを作成せずに、ユーザーまたはグループの特定のアクセス許可を設定できます。

Studio にユーザーとグループを割り当てるときに、セッションポリシーをそのユーザーまたはグループにマッピングして、きめ細かいアクセス許可を適用します。また、ユーザーまたはグループのセッションポリシーは、いつでも更新できます。Amazon EMR では、作成した各セッションポリシーマッピングが保存されます。

セッションポリシーの詳細については、「AWS Identity and Access Management ユーザーガイド」の「Policies and permissions」を参照してください。

ワークスペース

Workspace は Amazon EMR Studio の主要な構成要素です。ノートブックを整理するには、ユーザーは Studio に 1 つ以上の Workspace を作成します。詳細については、「Workspace の基本の説明」を参照してください。

JupyterLab の Workspace と同様に、Workspace はノートブックの作業の状態を保持します。ただし、Workspace ユーザーインターフェイスは、EMR クラスターの作成とアタッチ、ジョブの実行、サンプルノートブックの探索、Git リポジトリへのリンクを可能にする追加ツールでオープンソースの JupyterLab インターフェイスを拡張します。

次のリストには、EMR Studio Workspace の主な機能が含まれています。

- Workspace の可視性は Studio ベースです。ある Studio で作成した Workspace は、他の Studio では表示されません。
- デフォルトでは、Workspace は共有され、すべての Studio ユーザーが表示できます。ただし、Workspace を開いて作業できるのは一度に 1 人のユーザーのみです。他のユーザーと同時に作業する場合は、「Workspace コラボレーションを設定する」を参照してください。
- Workspace コラボレーションを有効にすると、Workspace 内の他のユーザーと同時にコラボレーションできます。詳細については、「Workspace コラボレーションを設定する」を参照してください。

- Workspace 内のノートブックは、コマンドを実行するために同じ EMR クラスターを共有します。Workspace は、Amazon EC2 で実行されている Amazon EMR クラスター、または Amazon EMR on EKS 仮想クラスターおよびマネージドエンドポイントにアタッチできます。
- Workspace は、Studio のサブネットに関連付けた別のアベイラビリティーゾーンに切り替えることができます。Workspace を停止して再起動して、フェイルオーバープロセスを促すことができます。Workspace を再起動すると、Studio が複数のアベイラビリティーゾーンにアクセスできるように設定されている場合、EMR Studio は Studio の VPC 内の別のアベイラビリティーゾーンで Workspace を起動します。Studio にアベイラビリティーゾーンが 1 つしかない場合、EMR Studio は異なるサブネットで Workspace を起動しようとします。詳細については、「Workspace の接続の問題を解決する」を参照してください。
- Workspace は、Studio に関連付けられているいずれかのサブネット内のクラスターに接続できます。

EMR Studio Workspace の作成と設定の詳細については、「Workspace の基本の説明」を参照してください。

Amazon EMR Studio でのノートブックストレージ

Workspace を使用している場合、EMR Studio は、Studio に関連付けられた Amazon S3 の場所にあるノートブックファイルにセルを定期的に自動保存します。このバックアッププロセスでは、セッション間の作業が保持されるため、Git リポジトリに変更をコミットせずに後で戻ることができます。詳細については、「Workspace コンテンツの保存」を参照してください。

Workspace からノートブックファイルを削除すると、EMR Studio によって Amazon S3 からバックアップバージョンが削除されます。ただし、最初にノートブックファイルを削除せずに Workspace を削除すると、ノートブックファイルは Amazon S3 に残り、ストレージ料金が引き続き課金されます。詳細については、「Workspace およびノートブックファイルを削除する」を参照してください。

EMR Studio に関する考慮事項

考慮事項

EMR Studio を使用する場合は、次の点を考慮してください。

- EMR Studio は、次の で使用できます AWS リージョン。
  - 米国東部 (オハイオ) (us-east-2)

- 米国東部 (バージニア北部) (us-east-1)
- 米国西部 (北カリフォルニア) (us-west-1)
- 米国西部 (オレゴン) (us-west-2)
- アフリカ (ケープタウン) (af-south-1)
- アジアパシフィック (香港) (ap-east-1)
- アジアパシフィック (ジャカルタ) (ap-southeast-3)*
- アジアパシフィック (メルボルン) (ap-southeast-4)*
- アジアパシフィック (ムンバイ) (ap-south-1)
- アジアパシフィック (大阪) (ap-northeast-3)*
- アジアパシフィック (ソウル) (ap-northeast-2)
- アジアパシフィック (シンガポール) (ap-southeast-1)
- アジアパシフィック (シドニー) (ap-southeast-2)
- アジアパシフィック (東京) (ap-northeast-1)
- カナダ (中部) (ca-central-1)
- ヨーロッパ (フランクフルト) (eu-central-1)
- 欧州 (アイルランド) (eu-west-1)
- ヨーロッパ (ロンドン) (eu-west-2)
- 欧州 (ミラノ) (eu-south-1)
- 欧州 (パリ) (eu-west-3)
- 欧州 (スペイン) (eu-south-2)
- 欧州 (ストックホルム) (eu-north-1)
- 欧州 (チューリッヒ) (eu-central-2)*
- イスラエル (テルアビブ) (il-central-1)*
- 中東 (アラブ首長国連邦) (me-central-1)*
- 南米 (サンパウロ) (sa-east-1)
- AWS GovCloud （ 米国東部) (gov-us-east-1)
- AWS GovCloud （ 米国西部) (gov-us-west-1)

* ライブ Spark UI は、これらのリージョンではサポートされていません。

ユーザーが Amazon EC2 で実行されている新しい EMR クラスターを Workspace 用にプロビジョニングできるようにするには、EMR Studio を一連のクラスターテンプレートに関連付けることが

できます。管理者は、Service Catalog でクラスターテンプレートを定義できます。また、Studio 内でユーザーまたはグループがクラスターテンプレートにアクセスできるのか、できないかを選択できます。

- Amazon S3 に保存されているノートブックファイルへのアクセス許可を定義する場合、または からシークレットを読み取る場合は AWS Secrets Manager、Amazon EMR サービスロールを使用します。セッションポリシーは、これらのアクセス許可ではサポートされません。
- 複数の EMR Studio を作成して、異なる VPC 内の EMR クラスターへのアクセスを制御できます。
- を使用して AWS CLI 、Amazon EMR on EKS クラスターをセットアップします。その後、Studio インターフェイスを使用して、マネージドエンドポイントを使用して Workspace にクラスターをアタッチして、ノートブックジョブを実行できます。
- Amazon EMR で信頼できる ID の伝達を使用する場合は、EMR Studio にも当てはまるその他の考慮事項があります。詳細については、「Amazon EMR と Identity Center の統合に関する考慮事項と制限」を参照してください。
- EMR Studio では、次の Python マジックコマンドはサポートされていません。
  - `%alias`
  - `%alias_magic`
  - `%automagic`
  - `%macro`
  - `%%js`
  - `%%javascript`
  - `%configure` を使用した `proxy_user` の変更
  - `%env` または `%set_env` を使用した `KERNEL_USERNAME` の変更
- Amazon EMR on EKS クラスターは、EMR Studio の SparkMagic コマンドをサポートしていません。
- ノートブックのセルに複数行の Scala ステートメントを記述する場合は、最後の行以外のすべての行がピリオドで終わっていることを確認してください。次の例では、複数行の Scala ステートメントで正しい構文を使用しています。

```
val df = spark.sql("SELECT * from table_name).
        filter("col1=='value'").
        limit(50)
```

- Amazon EMR で使用するオフコンソールアプリケーションのセキュリティを強化するために、アプリケーションホスティングドメインはパブリックサフィックスリスト (PSL) に登録されます。これらのホスティングドメインの例には以下が含まれます: `emrstudio-prod.us-east-1.amazonaws.com`、`emrnotebooks-prod.us-east-1.amazonaws.com`、`emrappui-prod.us-east-1.amazonaws.com`セキュリティ強化のため、デフォルトのドメイン名に機密性の高い Cookie を設定する必要がある場合は、`__Host-` プレフィックスの付いた Cookie を使用することをお勧めします。これは、クロスサイトリクエストフォージェリ (CSRF) 攻撃からドメインを防ぐ際に役立ちます。詳細については、「Mozilla 開発者ネットワーク」の「Set-Cookie」ページを参照してください。

既知の問題

- 信頼できる ID 伝達が有効になっている IAM Identity Center を使用する EMR Studio は、EMR クラスターのうち、信頼できる ID 伝達を使用するものにのみ関連付けることができます。
- Studio を作成する前に、ブラウザで FoxyProxy や SwitchyOmega などのプロキシ管理ツールを無効にしてください。アクティブなプロキシを使用している場合、[Create Studio] (Studio の作成) を選択するとエラーが発生し、[Network Failure] (ネットワーク障害) エラーメッセージが表示されることがあります。
- Amazon EMR on EKS クラスターで実行されるカーネルは、タイムアウトの問題により起動に失敗することがあります。カーネルの起動中にエラーまたは問題が発生した場合は、ノートブックファイルを閉じ、カーネルをシャットダウンしてから、ノートブックファイルを再度開きます。
- [Restart kernel] (カーネルの再起動) オペレーションは、Amazon EMR on EKS クラスターを使用している場合、期待どおりに機能しません。[Restart kernel] (カーネルの再起動) を選択した後に、Workspace を更新して再起動を有効にします。
- Workspace がクラスターにアタッチされていない場合、Studio ユーザーがノートブックファイルを開いてカーネルを選択しようとすると、エラーメッセージが表示されます。このエラーメッセージは、[OK]を選択して無視して構いません。ただし、ノートブックコードを実行するには、その前に Workspace をクラスターにアタッチし、カーネルを選択する必要があります。
- クラスターセキュリティを設定するための[セキュリティ設定]を使用して Amazon EMR 6.2.0 を使用している場合は、Workspace インターフェイスがブランクになり、期待どおりに動作しません。クラスターの EMRFS のデータ暗号化または Amazon S3 認可を設定する場合は、サポートされている別のバージョンの Amazon EMR を使用することをお勧めします。EMR Studio は、Amazon EMR バージョン 5.32.0 (Amazon EMR 5.x シリーズ) および 6.2.0 (Amazon EMR 6.x シリーズ) 以降で動作します。

- 「Amazon EC2 ジョブで実行中の Amazon EMR をデバッグする」を行うと、クラスター上の Spark UI へのリンクが機能しないか、表示されないことがあります。リンクを再生成するには、新しいノートブックセルを作成し、`%%info` コマンドを実行します。
- Jupyter Enterprise Gateway は、Amazon EMR リリースバージョン 5.32.0、5.33.0、6.2.0、6.3.0 では、クラスターのプライマリノード上のアイドル状態のカーネルをクリーンアップしません。アイドル状態のカーネルはコンピューティングリソースを消費するため、長時間稼働クラスターが失敗する原因となる可能性があります。次のサンプルスクリプトを使用して、Jupyter Enterprise Gateway のアイドル状態のカーネルのクリーンアップを設定できます。「SSH を使用してプライマリノードに接続する」やステップとしてのスクリプトの送信を行うことができます。詳細については、「Amazon EMR クラスターでコマンドとスクリプトを実行する」を参照してください。

```
#!/bin/bash
sudo tee -a /emr/notebook-env/conf/jupyter_enterprise_gateway_config.py << EOF
c.MappingKernelManager.cull_connected = True
c.MappingKernelManager.cull_idle_timeout = 10800
c.MappingKernelManager.cull_interval = 300
EOF
sudo systemctl daemon-reload
sudo systemctl restart jupyter_enterprise_gateway
```

- Amazon EMR バージョン 5.32.0、5.33.0、6.2.0、または 6.3.0 で自動終了ポリシーを使用すると、Amazon EMR はクラスターをアイドルとしてマークし、アクティブな Python3 カーネルがあっても自動的にクラスターが終了されることがあります。これは、Python3 カーネルを実行しても Spark ジョブがクラスターで送信されないためです。Python3 カーネルで自動終了を使用するには、Amazon EMR バージョン 6.4.0 以降を使用することをお勧めします。自動終了の詳細については、「自動終了ポリシーを使用する」を参照してください。
- `%%display` を使用してテーブル DataFrame に Spark を表示すると、非常に広いテーブルが切り捨てられる可能性があります。出力を右クリックして[Create New View for Output] (出力用の新しいビューを作成) を選択し、出力のスクロール可能なビューを取得できます。
- PySpark、Spark、SparkR などの Spark ベースのカーネルを起動すると、Spark セッションが開始され、ノートブックでセルを実行すると、そのセッションで Spark ジョブがキューに入れられます。実行中のセルを中断すると、Spark ジョブは引き続き実行されます。Spark ジョブを停止するには、クラスター上の Spark UI を使用する必要があります。Spark UI に接続する方法の手順については、「EMR Studio でアプリケーションとジョブをデバッグする」を参照してください。

機能の制限

Amazon EMR Studio では、次の Amazon EMR 機能はサポートされていません。

- Kerberos 認証を指定するセキュリティ構成を使用した EMR クラスターでのジョブのアタッチと実行
- 複数のプライマリノードを持つクラスター
- 6.9.0 より前の Amazon EMR 6.x リリースと 5.36.1 より前の 5.x リリースで AWS Graviton2 に基づく Amazon EC2 インスタンスを使用するクラスター

信頼できる ID 伝達を使用する Studio では、以下の機能はサポートされません。

- テンプレートなしで EMR クラスターを作成。
- EMR サーバーレスアプリケーションの使用。
- Amazon EMR on EKS クラスターの起動。
- ランタイムロールの使用。
- SQL エクスプローラーまたはワークスペースコラボレーションの有効化。

EMR Studio のサービスの制限

次の表に、EMR Studio のサービスの制限を示します。

項目	制限
EMR Studio	AWS アカウントあたり最大 100
サブネット	各 EMR Studio に関連付けることができるのは最大 5 個
IAM Identity Center グループ	各 EMR Studio に割り当てることができるのは最大 5 個
IAM Identity Center ユーザー	各 EMR Studio に割り当てることができるのは最大 100 個

VPC とサブネットのベストプラクティス

次のベストプラクティスを使用して、EMR Studio のサブネットで Amazon Virtual Private Cloud (Amazon VPC) をセットアップします。

- VPC には、最大 5 つのサブネットを指定して Studio に関連付けることができます。Workspace の可用性をサポートし、異なるアベイラビリティーゾーンのクラスターへのアクセス権を Studio ユーザーに付与するために、異なるアベイラビリティーゾーン内の複数のサブネットを提供することをお勧めします。VPC、サブネット、アベイラビリティーゾーンの操作の詳細については、「Amazon Virtual Private Cloud ユーザーガイド」の「VPC とサブネット」を参照してください。
- 指定するサブネットは、相互に通信できる必要があります。
- ユーザーがパブリックにホストされている Git リポジトリに Workspace をリンクできるようにするには、NAT を介してインターネットにアクセスできるプライベートサブネットのみを指定する必要があります。Amazon EMR に対するプライベートサブネットの設定の詳細については、「プライベートサブネット」を参照してください。
- EMR Studio で Amazon EMR on EKS を使用する場合、Studio と、仮想クラスターの登録に使用する Amazon EKS クラスターの間に少なくとも 1 つの共通のサブネットが必要です。そうなっていないと、マネージドエンドポイントは Studio Workspace のオプションとして表示されません。Amazon EKS クラスターを作成し、Studio に属するサブネットに関連付けるか、Studio を作成して EKS クラスターのサブネットを指定できます。
- EMR Studio で Amazon EMR on EKS を使用する場合は、Amazon EKS クラスターワーカーノードと同じ VPC を選択します。

Amazon EMR Studio のクラスター要件

Amazon EC2 で実行されている Amazon EMR クラスター

EMR Studio Workspace 用に作成する Amazon EC2 で実行されているすべての Amazon EMR クラスターは、次の要件を満たす必要があります。EMR Studio インターフェイスを使用して作成したクラスターは、これらの要件を自動的に満たします。

- クラスターは、Amazon EMR バージョン 5.32.0 (Amazon EMR 5.x シリーズ) または 6.2.0 (Amazon EMR 6.x シリーズ) 以降を使用する必要があります。Amazon EMR コンソール AWS Command Line Interfaceまたは SDK を使用してクラスターを作成し、EMR Studio Workspace にアタッチできます。Studio ユーザーは、Amazon EMR Workspace を作成または作業するときに、クラスターをプロビジョニングしてアタッチすることもできます。詳細については、「EMR Studio Workspace にコンピューティングをアタッチする」を参照してください。

- クラスターは Amazon Virtual Private Cloud 内に存在する必要があります。EC2-Classic プラットフォームはサポートされません。
- クラスターには Spark、Livy、および Jupyter Enterprise Gateway がインストールされている必要があります。SQL Explorer にクラスターを使用する予定がある場合は、Presto と Spark の両方をインストールする必要があります。
- SQL Explorer を使用するには、クラスターで Amazon EMR バージョン 5.34.0 以降またはバージョン 6.4.0 以降を使用し、Presto をインストールする必要があります。Presto の Hive メタストアとして AWS Glue データカタログを指定する場合は、クラスターで設定する必要があります。詳細については、「AWS Glue Data Catalog での Presto の使用」を参照してください。
- EMR Studio でパブリックにホストされた Git リポジトリを使用するには、クラスターが NAT を使用するプライベートサブネット内にある必要があります。

EMR Studio を使用する場合は、次のクラスター設定をお勧めします。

- Spark セッションのデプロイモードをクラスターモードに設定する。クラスターモードでは、アプリケーションマスタープロセスは、クラスターのプライマリノードではなく、コアノードに配置されます。そうすることで、プライマリノードでメモリ不足になる可能性が軽減されます。詳細については、Apache Spark ドキュメントで「クラスターモードの概要」を参照してください。
- 次の設定例のように、Livy タイムアウトをデフォルトの 1 時間から 6 時間に変更する。

```
{
    "classification":"livy-conf",
        "Properties":{
            "livy.server.session.timeout":"6h",
            "livy.spark.deploy-mode":"cluster"
        }
}
```

- 最大 30 のインスタンスで多様なインスタンスフリートを作成し、スポットインスタンスフリートで複数のインスタンスタイプを選択する。例えば、Spark ワークロードに対してメモリ最適化インスタンスタイプ r5.2x、r5.4x、r5.8x、r5.12x、r5.16x、r4.2x、r4.4x、r4.8x、r4.12 などを指定できます。詳細については、「インスタンスフリートを設定する」を参照してください。
- スポットインスタンスのキャパシティ最適化割り当て戦略を使用して、Amazon EMR が Amazon EC2 のリアルタイムのキャパシティインサイトに基づいて効果的にインスタンスを選択できるようにする。詳細については、「インスタンスフリートの配分戦略」を参照してください。
- クラスターでマネージドスケーリングを有効にする。最大コアノードパラメータを、使用する予定の最小永続キャパシティに設定し、スポットインスタンスで実行される分散型タスクフリートでス

ケーリングを設定してコストを節約する。詳細については、「Amazon EMR でマネージドスケーリングを使用する」を参照してください。

また、Amazon EMR ブロックパブリックアクセスを有効なままにしておき、インバウンド SSH トラフィックを信頼できるソースに制限することをお勧めします。クラスターへのインバウンドアクセスにより、ユーザーはクラスターでノートブックを実行できます。詳細については、「Amazon EMR のパブリックアクセスブロックの使用」および「セキュリティグループを使用してネットワークトラフィックを制御する」を参照してください。

Amazon EMR on EKS クラスター

Amazon EC2 で実行されている EMR クラスターに加えて、 AWS CLIを使用して EMR Studio の Amazon EMR on EKS クラスターを設定および管理できます。次のガイドラインを使用して、Amazon EMR on EKS クラスターを設定します。

- Amazon EMR on EKS クラスター用のマネージド HTTPS エンドポイントを作成します。ユーザーは Workspace をマネージドエンドポイントにアタッチします。仮想クラスターの登録に使用する Amazon Elastic Kubernetes Service (EKS) クラスターには、マネージドエンドポイントをサポートするためのプライベートサブネットが必要です。
- パブリックにホストされた Git リポジトリを使用する場合は、少なくとも 1 つのプライベートサブネットおよび NAT を持つ Amazon EKS クラスターを使用します。
- Amazon EKS 最適化 Arm Amazon Linux AMI は使用しないでください。これは、Amazon EMR on EKS マネージドエンドポイントではサポートされていません。
- サポートされていない AWS Fargateのみの Amazon EKS クラスターは使用しないでください。

Amazon EMR Studio の設定

このセクションは EMR Studio 管理者向けです。チーム用に EMR Studio を設定する方法について説明し、ユーザーとグループの割り当て、クラスターテンプレートの設定、EMR Studio 向けの Apache Spark の最適化などのタスクに関する手順について説明します。

トピック

- EMR Studio を作成および管理するための管理者のアクセス許可
- Amazon EMR Studio を設定する
- Amazon EMR Studio を管理する
- EMR Studio ワークスペースのノートブックとファイルの暗号化

- EMR Studio ネットワークトラフィックを制御するセキュリティグループを定義する
- Amazon EMR Studio 用の AWS CloudFormation テンプレートを作成する
- Git ベースのリポジトリのアクセス権とアクセス許可を設定する
- EMR Studio で Spark ジョブを最適化する

EMR Studio を作成および管理するための管理者のアクセス許可

このページで説明する IAM アクセス許可により、EMR Studio を作成および管理できます。必要な各アクセス許可の詳細については、「EMR Studio を管理するために必要なアクセス許可」を参照してください。

EMR Studio を管理するために必要なアクセス許可

次の表に、EMR Studio の作成と管理に関連するオペレーションを示します。この表には、各オペレーションに必要なアクセス許可も表示されています。

> **Note**
>
> IAM Identity Center 認証モードを使用する場合は、IAM Identity Center および Studio `SessionMapping` アクションのみが必要です。

EMR Studio を作成および管理するためのアクセス許可

操作	アクセス許可
Studio を作成する	`"elasticmapreduce:CreateStudio", "sso:CreateApplication", "sso:PutApplicationAuthenticationMethod", "sso:PutApplicationGrant", "sso:PutApplicationAccessScope", "sso:PutApplicationAssignmentConfiguration", "iam:PassRole"`
Studio について記述する	`"elasticmapreduce:DescribeStudio", "sso:GetManagedApplicationInstance"`

操作	アクセス許可
Studio をリストする	`"elasticmapreduce:ListStudios"`
Studio を削除する	`"elasticmapreduce:DeleteStudio", "sso:DeleteApplication", "sso:DeleteApplicationAuthentica tionMethod", "sso:DeleteApplicationAccessScope", "sso:DeleteApplicationGrant"`
Additional permissions required when you use IAM Identity Center mode	
Studio にユーザーまたはグループを割り当てる	`"elasticmapreduce:CreateStudioSessio nMapping", "sso:GetProfile", "sso:ListDirectoryAssociations", "sso:ListProfiles", "sso:AssociateProfile", "sso-directory:SearchUsers", "sso-directory:SearchGroups", "sso-directory:DescribeUser", "sso-directory:DescribeGroup", "sso:ListInstances", "sso:CreateApplicationAssignment", "sso:DescribeInstance", "organizations:DescribeOrga nization", "organizations:ListDelegatedAdmini strators", "sso:CreateInstance", "sso:DescribeRegisteredRegions", "sso:GetSharedSsoConfiguration", "iam:ListPolicies"`

操作	アクセス許可
特定のユーザーまたはグループの Studio 割り当ての詳細を取得する	`"sso-directory:SearchUsers", "sso-directory:SearchGroups", "sso-directory:DescribeUser", "sso-directory:DescribeGroup", "sso:DescribeApplication", "elasticmapreduce:GetStudioSessionMapping"`
Studio に割り当てられているすべてのユーザーとグループをリストする	`"elasticmapreduce:ListStudioSessionMappings"`
Studio に割り当てられているユーザーまたはグループにアタッチされたセッションポリシーを更新する	`"sso-directory:SearchUsers", "sso-directory:SearchGroups", "sso-directory:DescribeUser", "sso-directory:DescribeGroup", "sso:DescribeApplication", "sso:DescribeInstance", "elasticmapreduce:UpdateStudioSessionMapping"`
Studio からユーザーまたはグループを削除する	`"elasticmapreduce:DeleteStudioSessionMapping", "sso-directory:SearchUsers", "sso-directory:SearchGroups", "sso-directory:DescribeUser", "sso-directory:DescribeGroup", "sso:ListDirectoryAssociations", "sso:GetProfile", "sso:DescribeApplication", "sso:DescribeInstance", "sso:ListProfiles", "sso:DisassociateProfile", "sso:DeleteApplicationAssignment", "sso:ListApplicationAssignments"`

EMR Studio の管理者のアクセス許可を使用してポリシーを作成するには

1. 「IAM ポリシーの作成」の指示に従って、次の例のいずれかを使用してポリシーを作成します。必要なアクセス許可は、EMR Studio の認証モードによって異なります。

   次の項目に独自の値を挿入します。

   - *<your-resource-ARN>* を置き換え、ステートメントが自身のユースケースでカバーするオブジェクトの Amazon リソースネーム (ARN) を指定します。
   - *<region>* を、Studio を作成する予定の AWS リージョンのコードに置き換えます。
   - *<aws-account_id>* を、Studio の AWS アカウントの ID に置き換えます。
   - *<EMRStudio-Service-Role>* および *<EMRStudio-User-Role>* を、EMR Studio サービスロールおよび EMR Studio ユーザーロールの名前に置き換えます。

   Example ポリシーの例: IAM 認証モードを使用する場合の管理者のアクセス許可

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Resource": "arn:aws:elasticmapreduce:<region>:<aws-account-id>:studio/
*",
            "Action": [
                "elasticmapreduce:CreateStudio",
                "elasticmapreduce:DescribeStudio",
                "elasticmapreduce:DeleteStudio"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": "<your-resource-ARN>",
            "Action": [
                "elasticmapreduce:ListStudios"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": [
                "arn:aws:iam::<aws-account-id>:role/<EMRStudio-Service-Role>"
            ],
```

```
            "Action": "iam:PassRole"
        }
    ]
}
```

Example ポリシーの例: IAM Identity Center 認証モードを使用する場合の管理者のアクセス許可

> **Note**
>
> Identity Center および Identity Center ディレクトリ API では、IAM ポリシーステートメントの resource 要素での ARN の指定はサポートされていません。IAM Identity Center と IAM Identity Center Directory へのアクセスを許可するには、以下のアクセス許可で IAM Identity Center アクションにすべてのリソース ("Resource":"*") を指定します。詳細については、「IAM Identity Center Directory 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。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Resource": "arn:aws:elasticmapreduce:<region>:<aws-account-id>:studio/
*",
            "Action": [
                "elasticmapreduce:CreateStudio",
                "elasticmapreduce:DescribeStudio",
                "elasticmapreduce:DeleteStudio",
                "elasticmapreduce:CreateStudioSessionMapping",
                "elasticmapreduce:GetStudioSessionMapping",
                "elasticmapreduce:UpdateStudioSessionMapping",
                "elasticmapreduce:DeleteStudioSessionMapping"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": "<your-resource-ARN>",
            "Action": [
                "elasticmapreduce:ListStudios",
                "elasticmapreduce:ListStudioSessionMappings"
            ]
```

```
        },
        {
            "Effect": "Allow",
            "Resource": [
                "arn:aws:iam::<aws-account-id>:role/<EMRStudio-Service-Role>",
                "arn:aws:iam::<aws-account-id>:role/<EMRStudio-User-Role>"
            ],
            "Action": "iam:PassRole"
        },
        {
            "Effect": "Allow",
            "Resource": "*",
            "Action": [
                "sso:CreateApplication",
                "sso:PutApplicationAuthenticationMethod",
                "sso:PutApplicationGrant",
                "sso:PutApplicationAccessScope",
                "sso:PutApplicationAssignmentConfiguration",
                "sso:DescribeApplication",
                "sso:DeleteApplication",
                "sso:DeleteApplicationAuthenticationMethod",
                "sso:DeleteApplicationAccessScope",
                "sso:DeleteApplicationGrant",
                "sso:ListInstances",
                "sso:CreateApplicationAssignment",
                "sso:DeleteApplicationAssignment",
                "sso:ListApplicationAssignments",
                "sso:DescribeInstance",
                "sso:AssociateProfile",
                "sso:DisassociateProfile",
                "sso:GetProfile",
                "sso:ListDirectoryAssociations",
                "sso:ListProfiles",
                "sso-directory:SearchUsers",
                "sso-directory:SearchGroups",
                "sso-directory:DescribeUser",
                "sso-directory:DescribeGroup",
                "organizations:DescribeOrganization",
                "organizations:ListDelegatedAdministrators",
                "sso:CreateInstance",
                "sso:DescribeRegisteredRegions",
                "sso:GetSharedSsoConfiguration",
                "iam:ListPolicies"
            ]
```

```
        }
    ]
}
```

2. 次に、IAM ID (ユーザー、ロール、またはグループ) にポリシーをアタッチします。手順については、「IAM ID のアクセス許可の追加および削除」を参照してください。

Amazon EMR Studio を設定する

Amazon EMR Studio を設定するには、以下のステップを完了します。

開始する前に

> **Note**
>
> Amazon EMR on EKS で EMR Studio を使用する予定の場合は、Studio を設定する前に EMR Studio に Amazon EMR on EKS を設定することをお勧めします。

EMR Studio を設定する前に、以下のものがあることを確認してください。

- AWS アカウント。手順については、「Amazon EMR のセットアップ」を参照してください。
- EMR Studio 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、「the section called "EMR Studio を作成するための管理者のアクセス許可"」を参照してください。
- EMR Studio が Studio 内の Workspace とノートブックファイルをバックアップできる Amazon S3 バケット。手順については、「Amazon Simple Storage Service (S3) ユーザーガイド」の「バケットの作成」を参照してください。
- Amazon EMR on EC2 または Amazon EMR on EKS にアタッチする場合、または Git リポジトリを使用する場合は、Studio 用の Amazon 仮想プライベートクラウド (VPC) と、最大 5 つのサブネットが必要です。EMR Studio を EMR Serverless で使用する VPC は必要ありません。ネットワークの設定方法に関するヒントについては、「VPC とサブネットのベストプラクティス」を参照してください。

EMR Studio を設定するには

1. Amazon EMR Studio の認証モードの選択
2. 次の Studio リソースを作成します。

- EMR Studio サービスロールを作成する
- Amazon EC2 または Amazon EKS 用の EMR Studio ユーザーアクセス許可の設定
- (オプション) EMR Studio ネットワークトラフィックを制御するセキュリティグループを定義する。

3. EMR Studio の作成
4. EMR Studio にユーザーまたはグループを割り当てる

設定ステップを完了すると、「Amazon EMR Studio を使用する」を行うことができます。

Amazon EMR Studio の認証モードの選択

EMR Studio は、IAM 認証モードと IAM Identity Center 認証モードという 2 つの認証モードをサポートしています。IAM モードは AWS Identity and Access Management (IAM) を使用し、IAM Identity Center モードは AWS IAM Identity Center を使用します。EMR Studio を作成するときは、その Studio のすべてのユーザーの認証モードを選択します。各種認証モードの詳細については、「認証とユーザーログイン」を参照してください。

次の表を使用して、EMR Studio の認証モードを選択します。

状況	推奨設定
IAM 認証またはフェデレーションに精通している、または以前に設定したことがある	IAM 認証モード。以下のような利点があります。 • IAM でユーザーやグループなどの ID をすでに管理している場合、EMR Studio 用に迅速に設定できます。 • OpenID Connect (OIDC) または SAML 2.0 (Security Assertion Markup Language 2.0) と互換性のある ID プロバイダーで動作します。 • 同じ AWS アカウント で複数の ID プロバイダーを使用できます。 • 幅広い AWS リージョン で使用可能です。 • SOC 2 に準拠しています。

状況	推奨設定
AWS または Amazon EMR を初めて使用している	IAM Identity Center 認証モード。次の特長があります。 • ユーザーおよびグループを AWS リソースに簡単に割り当てることができます。 • Microsoft Active Directory および SAML 2.0 ID プロバイダーで動作します。 • マルチアカウントフェデレーションの設定が容易になり、組織内の AWS アカウント ごとにフェデレーションを個別に構成する必要がなくなります。

Amazon EMR Studio の IAM 認証モードの設定

IAM 認証モードでは、IAM 認証または IAM フェデレーションを使用できます。IAM 認証では、IAM のユーザー、グループ、ロールなどの IAM ID を管理できます。IAM アクセス許可ポリシーおよび属性ベースのアクセスコントロール (ABAC) を使用して Studio へのアクセス権をユーザーに付与します。IAM フェデレーションにより、サードパーティーの ID プロバイダー (IdP) と AWS との間の信頼を確立できます。これにより、IdP を通じてユーザー ID を管理できます。

> **Note**
>
> IAM を使用して AWS リソースへのアクセスをすでにコントロールしている場合、または IAM 用に ID プロバイダー (IdP) をすでに設定している場合は、「IAM 認証モードのユーザーアクセス許可」を参照して、EMR Studio の IAM 認証モードを使用するときにユーザーアクセス許可を設定します。

Amazon EMR Studio での IAM フェデレーションの使用

EMR Studio 用に IAM フェデレーションを使用するには、AWS アカウントと ID プロバイダー (IdP) との間に信頼関係を作成し、フェデレーションユーザーが AWS Management Consoleにアクセスできるようにします。この信頼関係を作成するために実行する手順は、ご使用の IdP のフェデレーション標準によって異なります。

一般に、外部 IdP とのフェデレーションを設定するには、次のタスクを実行します。詳細な手順については、「AWS Identity and Access Management ユーザーガイド」の「SAML 2.0 フェデレーティッドユーザーが AWS Management Consoleにアクセス可能にする」および「カスタム ID ブローカーに対する AWS Management Consoleへのアクセスの許可」を参照してください。

1. IdP から情報を収集します。これは通常、IdP からの SAML 認証リクエストを検証するためのメタデータドキュメントを生成することを意味します。
2. ID プロバイダーの IAM エンティティを作成して、IdP に関する情報を保存します。手順については、「IAM ID プロバイダーの作成」を参照してください。
3. IdP のために 1 つ以上の IAM ロールを作成します。EMR Studio は、ユーザーのログイン時にフェデレーティッドユーザーにロールを割り当てます。このロールで、IdP が AWS にアクセスするための一時的なセキュリティ認証情報をリクエストできるようにします。手順については、「サードパーティー ID プロバイダー (フェデレーション) 用のロールの作成」を参照してください。ロールに割り当てるアクセス許可ポリシーにより、フェデレーティッドユーザーが AWS および EMR Studio で実行できる操作が決定されます。詳細については、「IAM 認証モードのユーザーアクセス許可」を参照してください。
4. (SAML プロバイダーの場合) AWS に関する情報、およびフェデレーティッドユーザーに付与するロールを使用して IdP を設定して SAML 信頼を確立します。この設定プロセスにより、IdP と AWS の間に証明書利用者の信頼が作成されます。詳細については、「証明書利用者の信頼およびクレームの追加によって SAML 2.0 IdP を設定する」を参照してください。

IdP ポータルの SAML アプリケーションとして EMR Studio を設定するには

Studio へのディープリンクを使用して、特定の EMR Studio を SAML アプリケーションとして設定できます。これにより、ユーザーは、Amazon EMR コンソールでナビゲートする代わりに、IdP ポータルにログインして、特定の Studio を起動できます。

- SAML アサーション検証後のランディング URL として EMR Studio へのディープリンクを設定するには、次の形式を使用します。

```
https://console.aws.amazon.com/emr/home?region=<aws-region>#studio/<your-studio-id>/start
```

Amazon EMR Studio の IAM Identity Center 認証モードの設定

EMR Studio 用に AWS IAM Identity Center を準備するには、ID ソースを設定し、ユーザーとグループをプロビジョニングする必要があります。プロビジョニングとは、ユーザーおよびグループの情報

を IAM Identity Center および IAM Identity Center を使用するアプリケーションで使用できるようにするプロセスです。詳細については、「ユーザーおよびグループのプロビジョニング」を参照してください。

EMR Studio では、IAM Identity Center の次の ID プロバイダーの使用をサポートしています。

- AWS Managed Microsoft AD およびセルフマネージド Active Directory — 詳細については、「Connect to your Microsoft AD directory」を参照してください。
- SAML ベースのプロバイダー — 詳細なリストについては、「Supported identity providers」を参照してください。
- IAM Identity Center ディレクトリ — 詳細については、「IAM Identity Center での ID の管理」を参照してください。

EMR Studio 用に IAM Identity Center を設定するには

1. EMR Studio 用に IAM Identity Center を設定するには、次のものが必要です。
   - AWS 組織の管理アカウント (組織で複数のアカウントを使用する場合)。

     > **Note**
     >
     > 管理アカウントを使用するのは、IAM Identity Center を有効にする場合とユーザーおよびグループをプロビジョニングする場合のみにしてください。IAM Identity Center の設定後、メンバーアカウントを使用して EMR Studio を作成し、ユーザーとグループを割り当てます。AWS の用語の詳細については、「AWS Organizations の用語と概念」を参照してください。

   - 2019 年 11 月 25 日より前に IAM Identity Center を有効にした場合は、AWS 組織のアカウントに IAM Identity Center を使用するアプリケーションを有効にする必要があることがあります。詳細については、「Enable IAM Identity Center-integrated applications in AWS accounts」を参照してください。
   - 「IAM Identity Center prerequisites」ページに示されている前提条件を満たしていることを確認します。
2. 「Enable IAM Identity Center 」の手順に従って、EMR Studio を作成する AWS リージョンで IAM Identity Center を有効にします。
3. IAM Identity Center を ID プロバイダーに接続して、Studio に割り当てるユーザーとグループをプロビジョニングします。

使用するもの	手順
Microsoft AD ディレクトリ	1. 「Connect to your Microsoft AD directory」の手順に従って、AWS Directory Service を使用してセルフマネージド Active Directory または AWS Managed Microsoft AD ディレクトリを接続します。 2. IAM Identity Center のユーザーとグループをプロビジョニングするには、ソース AD の ID データを IAM Identity Center に同期します。ソース AD の ID をさまざまな方法で同期できます。1 つの方法として、AD ユーザーまたはグループを組織内の AWS アカウントに割り当てることができます。手順については、「Single sign-on」を参照してください。 同期には最大 2 時間かかることがあります。このステップを完了すると、同期されたユーザーとグループが ID ストアに表示されます。 **Note** ユーザーとグループは、ユーザーとグループの情報を同期するか、just-in-time (JIT) ユーザープロビジョニングを使用するまで、ID ストアに表示されません。詳細については、「Provisioning when users come from Active Directory」を参照してください。 3. (オプション) AD ユーザーとグループを同期した後に、前のステップで設定した AWS アカウントへのアクセス権を削除で

使用するもの	手順
	きます。手順については、「ユーザーアクセスを削除する」を参照してください。
外部 ID プロバイダー	「Connect to your external identity provider」の指示に従います。
IAM Identity Center ディレクトリ	IAM Identity Center でユーザーとグループを作成すると、プロビジョニングが自動的に行われます。詳細については、「IAM Identity Center での ID の管理」を参照してください。

これで、ID ストアのユーザーおよびグループを EMR Studio に割り当てることができるようになりました。手順については、「EMR Studio にユーザーまたはグループを割り当てる」を参照してください。

EMR Studio サービスロールを作成する

EMR Studio サービスロールについて

各 EMR Studio は、Studio が他の AWS のサービスと対話できるようにするアクセス許可を備えた IAM ロールを使用します。このサービスロールには、EMR Studio が Workspace とクラスター間のセキュアなネットワークチャネルを確立し、Amazon S3 Control にノートブックファイルを保存し、Workspace を Git リポジトリにリンクしているとき AWS Secrets Manager にアクセスできるようにするアクセス許可を含める必要があります。

(セッションポリシーの代わりに) Studio サービスロールを使用して、ノートブックファイルを保存するためのすべての Amazon S3 アクセス許可を定義し、AWS Secrets Manager アクセス許可を定義します。

Amazon EC2 または Amazon EKS で EMR Studio 用にサービスロールを作成する方法

1. 「AWS のサービスにアクセス許可を委任するロールの作成」の指示に従って、次の信頼ポリシーを使用してサービスロールを作成します。

> **Important**
>
> 次の信頼ポリシーには、EMR Studio に付与するアクセス許可をアカウント内の特定のリソースに制限するための `aws:SourceArn` および `aws:SourceAccount` グローバル条件キーが含まれています。そうすることで、「混乱した代理」問題から保護できます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "elasticmapreduce.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "<account-id>"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:elasticmapreduce:<region>:<account-id>:*"
        }
      }
    }
  ]
}
```

2. デフォルトのロールアクセス許可を削除します。次に、次のサンプル IAM アクセス許可ポリシーのアクセス許可を含めます。または、EMR Studio サービスロールのアクセス許可 を使用するカスタムポリシーを作成できます。

> **Important**
>
> - による Amazon EC2 タグベースのアクセスコントロールを EMR Studio と連携させるには、次のポリシーに示すように `ModifyNetworkInterfaceAttribute` API へのアクセスを設定する必要があります。

- EMR Studio がサービス ロールを操作するには、次のステートメント `AllowAddingEMRTagsDuringDefaultSecurityGroupCreation` および `AllowAddingTagsDuringEC2ENICreation` を変更しないでください。
- サンプルポリシーを使用するには、キー "**for-use-with-amazon-emr-managed-policies**" および値 "**true**" を使用して次のリソースにタグ付けする必要があります。
  - EMR Studio の Amazon Virtual Private Cloud (VPC)。
  - Studio で使用する各サブネット。
  - カスタム EMR Studio セキュリティグループ。EMR Studio のプレビュー期間中に作成したセキュリティグループを引き続き使用する場合は、そのセキュリティグループにタグを付ける必要があります。
  - Studio ユーザーが Git リポジトリを Workspace にリンクするために使用する、AWS Secrets Manager に保存されたシークレット。

  AWS Management Console の関連するリソース画面の [Tag] (タグ) タブを使用してリソースにタグを適用できます。

該当する場合は、以下のポリシーの `"Resource":"*"` で `*` を変更して、ご使用のユースケースでステートメントがカバーする 1 つ以上のリソースの Amazon リソースネーム (ARN) を指定します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowEMRReadOnlyActions",
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:ListInstances",
        "elasticmapreduce:DescribeCluster",
        "elasticmapreduce:ListSteps"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowEC2ENIActionsWithEMRTags",
      "Effect": "Allow",
```

```
      "Action": [
        "ec2:CreateNetworkInterfacePermission",
        "ec2:DeleteNetworkInterface"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:network-interface/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/for-use-with-amazon-emr-managed-policies": "true"
        }
      }
    },
    {
      "Sid": "AllowEC2ENIAttributeAction",
      "Effect": "Allow",
      "Action": [
        "ec2:ModifyNetworkInterfaceAttribute"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:instance/*",
        "arn:aws:ec2:*:*:network-interface/*",
        "arn:aws:ec2:*:*:security-group/*"
      ]
    },
    {
      "Sid": "AllowEC2SecurityGroupActionsWithEMRTags",
      "Effect": "Allow",
      "Action": [
        "ec2:AuthorizeSecurityGroupEgress",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:RevokeSecurityGroupEgress",
        "ec2:RevokeSecurityGroupIngress",
        "ec2:DeleteNetworkInterfacePermission"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/for-use-with-amazon-emr-managed-policies": "true"
        }
      }
    },
    {
      "Sid": "AllowDefaultEC2SecurityGroupsCreationWithEMRTags",
```

```
      "Effect": "Allow",
      "Action": [
        "ec2:CreateSecurityGroup"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:security-group/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/for-use-with-amazon-emr-managed-policies": "true"
        }
      }
    },
    {
      "Sid": "AllowDefaultEC2SecurityGroupsCreationInVPCWithEMRTags",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateSecurityGroup"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:vpc/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/for-use-with-amazon-emr-managed-policies": "true"
        }
      }
    },
    {
      "Sid": "AllowAddingEMRTagsDuringDefaultSecurityGroupCreation",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateTags"
      ],
      "Resource": "arn:aws:ec2:*:*:security-group/*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/for-use-with-amazon-emr-managed-policies": "true",
          "ec2:CreateAction": "CreateSecurityGroup"
        }
      }
    },
    {
      "Sid": "AllowEC2ENICreationWithEMRTags",
```

```
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:network-interface/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/for-use-with-amazon-emr-managed-policies": "true"
        }
      }
    },
    {
      "Sid": "AllowEC2ENICreationInSubnetAndSecurityGroupWithEMRTags",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:subnet/*",
        "arn:aws:ec2:*:*:security-group/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/for-use-with-amazon-emr-managed-policies": "true"
        }
      }
    },
    {
      "Sid": "AllowAddingTagsDuringEC2ENICreation",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateTags"
      ],
      "Resource": "arn:aws:ec2:*:*:network-interface/*",
      "Condition": {
        "StringEquals": {
          "ec2:CreateAction": "CreateNetworkInterface"
        }
      }
    },
    {
      "Sid": "AllowEC2ReadOnlyActions",
```

```
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeTags",
        "ec2:DescribeInstances",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcs"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowSecretsManagerReadOnlyActionsWithEMRTags",
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Resource": "arn:aws:secretsmanager:*:*:secret:*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/for-use-with-amazon-emr-managed-policies": "true"
        }
      }
    },
    {
      "Sid": "AllowWorkspaceCollaboration",
      "Effect": "Allow",
      "Action": [
        "iam:GetUser",
        "iam:GetRole",
        "iam:ListUsers",
        "iam:ListRoles",
        "sso:GetManagedApplicationInstance",
        "sso-directory:SearchUsers"
      ],
      "Resource": "*"
    }
  ]
}
```

3. EMR Studio の Amazon S3 の場所への読み取りおよび書き込みアクセス権をサービスロールに付与します。次の最小アクセス許可セットを使用します。詳細は、「Amazon S3: S3 バケット

のオブジェクトへの読み取りおよび書き込みアクセスをプログラムによりコンソールで許可する」の例を参照してください。

```
"s3:PutObject",
"s3:GetObject",
"s3:GetEncryptionConfiguration",
"s3:ListBucket",
"s3:DeleteObject"
```

Amazon S3 バケットを暗号化する場合は、AWS Key Management Service の以下のアクセス許可を含めます。

```
"kms:Decrypt",
"kms:GenerateDataKey",
"kms:ReEncryptFrom",
"kms:ReEncryptTo",
"kms:DescribeKey"
```

4. Git シークレットへのアクセスをユーザーレベルで制御する場合は、EMR Studio ユーザーロールポリシーで `secretsmanager:GetSecretValue` へタグベースのアクセス許可を追加し、EMR Studio サービスロールポリシーから `secretsmanager:GetSecretValue` ポリシーへのアクセス許可を削除します。きめ細かいユーザーのアクセス許可の設定方法の詳細については、「EMR Studio ユーザーのアクセス許可ポリシーの作成」を参照してください。

EMR Serverless の最小限のサービスロール

EMR Studio ノートブックを介して EMR Serverless でインタラクティブなワークロードを実行する場合は、前のセクション (Amazon EC2 または Amazon EKS で EMR Studio 用にサービスロールを作成する方法) で EMR Studio の設定に使用したのと同じ信頼ポリシーを使用します。

IAM ポリシーでは、実用最小限のポリシーには次のようなアクセス許可があります。EMR Studio と Workspace を設定するときに使用する予定のバケットの名前で *`bucket-name`* を更新します。EMR Studio はこのバケットを使用して Studio 内の Workspace とノートブックファイルをバックアップします。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ObjectActions",
```

```
            "Effect": "Allow",
            "Action": [
               "s3:PutObject",
               "s3:GetObject",
               "s3:DeleteObject"
            ],
            "Resource": ["arn:aws:s3:::bucket-name/*"]
        },
        {
            "Sid": "BucketActions",
            "Effect": "Allow",
            "Action": [
               "s3:ListBucket",
               "s3:GetEncryptionConfiguration"
            ],
            "Resource": ["arn:aws:s3:::bucket-name"]
        }
    ]
}
```

暗号化された Amazon S3 バケットを使用する場合は、ポリシーに以下のアクセス許可を追加します。

```
"kms:Decrypt",
"kms:GenerateDataKey",
"kms:ReEncryptFrom",
"kms:ReEncryptTo",
"kms:DescribeKey"
```

EMR Studio サービスロールのアクセス許可

次の表に、EMR Studio がサービスロールを使用して実行するオペレーションと、各オペレーションに必要な IAM アクションを示します。

操作	アクション

Workspace と EMR クラスターの間にセキュアネットワークチャネルを確立し、必要なクリーンアップアクションを実行する。	`"ec2:CreateNetworkInterface",`
	`"ec2:CreateNetworkInterfacePermission",`
	`"ec2:DeleteNetworkInterface",`
	`"ec2:DeleteNetworkInterfacePermission",`
	`"ec2:DescribeNetworkInterfaces",`

操作	アクション
	"ec2:ModifyNetworkInterfaceAttribute", "ec2:AuthorizeSecurityGroupEgress", "ec2:AuthorizeSecurityGroupIngress", "ec2:CreateSecurityGroup", "ec2:DescribeSecurityGroups", "ec2:RevokeSecurityGroupEgress", "ec2:DescribeTags", "ec2:DescribeInstances", "ec2:DescribeSubnets", "ec2:DescribeVpcs", "elasticmapreduce:ListInstances", "elasticmapreduce:DescribeCluster", "elasticmapreduce:ListSteps"
AWS Secrets Manager に保存されている Git 認証情報を使用して Git リポジトリを Workspace にリンクする。	"secretsmanager:GetSecretValue"
EMR Studio がセキュアネットワークチャネルの設定時に作成するネットワークインターフェイスおよびデフォルトのセキュリティグループに AWS タグを適用する。詳細については、「AWS リソースのタグ付け」を参照してください。	"ec2:CreateTags"

操作	アクション

ノートブックファイルとメタデータにアクセスする、それらを Amazon S3 にアップロードする。	`"s3:PutObject",`
	`"s3:GetObject",`
	`"s3:GetEncryptionConfiguration",`
	`"s3:ListBucket",`
	`"s3:DeleteObject"`
	
	暗号化された Amazon S3 バケットを使用する場合は、以下のアクセス許可を含めます。
	
	`"kms:Decrypt",`
	`"kms:GenerateDataKey",`
	`"kms:ReEncryptFrom",`
	`"kms:ReEncryptTo",`
	`"kms:DescribeKey"`

Workspace コラボレーションを有効にして設定します。	`"iam:GetUser",`
	`"iam:GetRole",`
	`"iam:ListUsers",`
	`"iam:ListRoles",`
	`"sso:GetManagedApplicationInstance",`
	`"sso-directory:SearchUsers"`

でカスタマーマネージドキー (CMK) を使用して EMR Studio ワークスペースのノートブックとファイルを暗号化する AWS Key Management Service	`"kms:Decrypt",`
	`"kms:GenerateDataKey",`
	`"kms:ReEncryptFrom",`
	`"kms:ReEncryptTo",`
	`"kms:DescribeKey"`

Amazon EC2 または Amazon EKS 用の EMR Studio ユーザーアクセス許可の設定

きめ細かなユーザーおよびグループのアクセス許可を設定できるように、Amazon EMR Studio のユーザーアクセス許可ポリシーを設定する必要があります。EMR Studio でのユーザーアクセス許可の仕組みについては、「Amazon EMR Studio の仕組み」の「アクセスコントロール」を参照してください。

> **Note**
>
> このセクションで説明するアクセス許可は、データアクセスコントロールを適用しません。入力データセットへのアクセスを管理するには、Studio が使用するクラスターの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、「Amazon EMR でのセキュリティ」を参照してください。

IAM Identity Center 認証モードの EMR Studio ユーザーロールの作成

IAM Identity Center 認証モードを使用する場合、EMR Studio ユーザーロールを作成する必要があります。

EMR Studio のユーザーロールを作成するには

1. ユーザーロールを作成するには、「AWS Identity and Access Management ユーザーガイド」の「AWS サービスにアクセス許可を委任するロールの作成」の手順に従います。

   ロールを作成する場合、次の信頼関係ポリシーを使用します。

```
{
  "Version": "2008-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "elasticmapreduce.amazonaws.com"
      },
      "Action": [
              "sts:AssumeRole",
              "sts:SetContext"
              ]
    }
  ]
}
```

2. デフォルトのロールアクセス許可およびポリシーを削除します。
3. ユーザーとグループを Studio に割り当てる前に、EMR Studio セッションポリシーをユーザーロールにアタッチします。セッションポリシーを作成する方法の手順については、「EMR Studio ユーザーのアクセス許可ポリシーの作成」を参照してください。

EMR Studio ユーザーのアクセス許可ポリシーの作成

EMR Studio のアクセス許可ポリシーを作成するには、次のセクションを参照してください。

トピック

- アクセス許可ポリシーを作成します。
- Workspace コラボレーションの所有権の設定
- ユーザーレベルの Git シークレットポリシーの作成
- アクセス許可ポリシーを IAM ID にアタッチする

> **Note**
>
> ノートブックファイルを保存するための Amazon S3 アクセス許可および Workspace を Git リポジトリにリンクしているときにシークレットを読み取るためのAWS Secrets Manager アクセス許可を設定するには、EMR Studio サービスロールを使用します。

アクセス許可ポリシーを作成します。

ユーザーが Studio で実行できるアクションを指定する、1 つ以上の IAM アクセス許可ポリシーを作成します。たとえば、このページのサンプルポリシーを使用して、基本、中級、上級の Studio ユーザーに対して 3 つの個別のポリシーを作成できます。

ユーザーが実行する可能性のある各 Studio 操作の内訳と、各操作を実行するために必要な最小限の IAM アクションについては、「EMR Studio ユーザーの AWS Identity and Access Management アクセス許可」を参照してください。ポリシーを作成する手順については、「IAM ユーザー ガイド」の「IAM ポリシーの作成」を参照してください。

アクセス許可ポリシーには以下のステートメントを含める必要があります。

```
{
            "Sid": "AllowAddingTagsOnSecretsWithEMRStudioPrefix",
            "Effect": "Allow",
            "Action": "secretsmanager:TagResource",
            "Resource": "arn:aws:secretsmanager:*:*:secret:emr-studio-*"
},
{
            "Sid": "AllowPassingServiceRoleForWorkspaceCreation",
            "Action": "iam:PassRole",
```

```
            "Resource": [
                "arn:aws:iam::*:role/your-emr-studio-service-role"
            ],
            "Effect": "Allow"
}
```

Workspace コラボレーションの所有権の設定

Workspace コラボレーションでは、複数のユーザーは同じワークスペースで同時に作業できます。コラボレーションは Workspace UI の [コラボレーション] パネルを使用して設定できます。[コラボレーション] パネルを表示して使用する場合、ユーザーには以下のアクセス許可が必要です。これらのアクセス許可を持つユーザーなら誰でも [コラボレーション] パネルを表示して使用できます。

```
"elasticmapreduce:UpdateEditor",
"elasticmapreduce:PutWorkspaceAccess",
"elasticmapreduce:DeleteWorkspaceAccess",
"elasticmapreduce:ListWorkspaceAccessIdentities"
```

[コラボレーション] パネルへのアクセスを制限するには、タグベースのアクセス制御を使用します。ユーザーが Workspace を作成すると、EMR Studio は、Workspace を作成したユーザーの ID を値とする `creatorUserId` のキーの付いたデフォルトタグを適用します。

> **Note**
>
> EMR Studio は、2021 年 11 月 16 日以降に作成されたワークスペースに `creatorUserId` タグを追加します。この日付より前に作成したワークスペースに対してコラボレーションを構成できるユーザーを制限するには、`creatorUserId` タグを Workspace に手動で追加してから、ユーザーアクセス許可ポリシーでタグベースのアクセス制御を使用することをお勧めします。

次のステートメント例では、ユーザーはユーザーの ID (ポリシー変数 `aws:userId` で指定) と一致する値を持つタグキー `creatorUserId` がある Workspace にコラボレーションを設定できます。つまり、このステートメントにより、ユーザーは自分で作成した Workspace にコラボレーションを設定できます。ポリシー変数の詳細については、「IAM ユーザーガイド」の「IAM ポリシーエレメント: 変数とタグ」を参照してください。

```
    {
        "Sid": "UserRolePermissionsForCollaboration",
```

```
        "Action": [
            "elasticmapreduce:UpdateEditor",
            "elasticmapreduce:PutWorkspaceAccess",
            "elasticmapreduce:DeleteWorkspaceAccess",
            "elasticmapreduce:ListWorkspaceAccessIdentities"
        ],
        "Resource": "*",
        "Effect": "Allow",
        "Condition": {
            "StringEquals": {
                "elasticmapreduce:ResourceTag/creatorUserId": "${aws:userid}"
            }
        }
    }
```

ユーザーレベルの Git シークレットポリシーの作成

トピック

- ユーザーレベルのアクセス許可を使用するには
- サービスレベルの権限からユーザーレベルの権限に移行するには
- サービスレベルのアクセス許可を使用するには

ユーザーレベルのアクセス許可を使用するには

EMR Studio は Git シークレットを作成するときに自動的に `for-use-with-amazon-emr-managed-user-policies` タグを追加します。Git シークレットへのアクセスをユーザーレベルで制御する場合は、以下の サービスレベルの権限からユーザーレベルの権限に移行するには セクションに示すように、`secretsmanager:GetSecretValue` を使用して EMR Studio ユーザーロールポリシーにタグベースのアクセス許可を追加します。

EMR Studio サービスロールポリシーに既存の `secretsmanager:GetSecretValue` のアクセス許可がある場合は、それらのアクセス許可を削除する必要があります。

サービスレベルの権限からユーザーレベルの権限に移行するには

> **Note**
>
> `for-use-with-amazon-emr-managed-user-policies` タグにより、以下の [ステップ 1] からのアクセス許可は、ワークスペースの作成者に Git シークレットへのアクセス許可を確実に付与します。ただし、2023 年 9 月 1 日より前に Git リポジトリをリンクした場合、

対応する Git シークレットには for-use-with-amazon-emr-managed-user-policies タグが適用されていないためアクセスが拒否されます。ユーザーレベルのアクセス許可を適用するには、 から古いシークレットを再作成 JupyterLab し、適切な Git リポジトリを再度リンクする必要があります。
ポリシー変数に関する詳細については、IAM ユーザーガイドの「IAM ポリシー要素: 変数およびタグ」を参照してください。

1. EMR Studio ユーザーロールポリシーに次の権限を追加します。値 "${aws:*userid*}" のある for-use-with-amazon-emr-managed-user-policies キーを使用します。

```
{
   "Sid": "AllowSecretsManagerReadOnlyActionsWithEMRTags",
    "Effect": "Allow",
    "Action": "secretsmanager:GetSecretValue",
    "Resource": "arn:aws:secretsmanager:*:*:secret:*",
    "Condition": {
        "StringEquals": {
            "secretsmanager:ResourceTag/for-use-with-amazon-emr-managed-user-
policies": "${aws:userid}"
        }
    }
}
```

2. 存在する場合は、EMR Studio サービスロールポリシーから次のアクセス許可を削除します。サービスロールポリシーは各ユーザーが定義したすべてのシークレットに適用されるため、この操作は 1 回のみで済みます。

```
{
    "Sid": "AllowSecretsManagerReadOnlyActionsWithEMRTags",
    "Effect": "Allow",
    "Action": [
        "secretsmanager:GetSecretValue"
     ],
    "Resource": "arn:aws:secretsmanager:*:*:secret:*",
    "Condition": {
        "StringEquals": {
            "aws:ResourceTag/for-use-with-amazon-emr-managed-policies": "true"
        }
    }
}
```

サービスレベルのアクセス許可を使用するには

2023 年 9 月 1 日以降、EMR Studio はユーザーレベルのアクセス制御用の `for-use-with-amazon-emr-managed-user-policies` タグを自動的に追加します。これは追加機能であるため、EMR Studio サービスロールの `GetSecretValue` アクセス許可を通じて利用できるサービスレベルのアクセスを引き続き使用できます。

2023 年 9 月 1 日より前に作成されたシークレットについては、EMR Studio は `for-use-with-amazon-emr-managed-user-policies` タグを追加しませんでした。サービスレベルのアクセス許可を引き続き使用するには、既存の EMR Studio サービスロールとユーザーロールのアクセス許可を保持するだけです。ただし、個々のシークレットにアクセスできるユーザーを制限するには、ユーザーレベルのアクセス許可を使用するには の手順に従って `for-use-with-amazon-emr-managed-user-policies` タグをシークレットに手動で追加してから、ユーザーアクセス許可ポリシーでタグベースのアクセス制御を使用することをお勧めします。

ポリシー変数に関する詳細については、IAM ユーザーガイドの「IAM ポリシー要素: 変数およびタグ」を参照してください。

アクセス許可ポリシーを IAM ID にアタッチする

次の表は、EMR Studio 認証モードに応じて、アクセス許可ポリシーをアタッチする IAM ID をまとめたものです。ポリシーをアタッチする方法については、「IAM ID のアクセス許可の追加および削除」を参照してください。

使用するもの	ポリシーのアタッチ先
IAM 認証	IAM ID (ユーザー、ユーザーのグループ、ロール)。例えば、AWS アカウントのユーザーにアクセス許可ポリシーをアタッチできます。
外部 ID プロバイダー (IdP) による IAM フェデレーション	外部 IdP 用に作成した 1 つ以上の IAM ロール。例えば、SAML 2.0 フェデレーション用の IAM などです。 EMR Studio は、Studio へのフェデレーションアクセス権を持つユーザーに対して、IAM ロールにアタッチしたアクセス許可を使用します。
IAM アイデンティティセンター	Amazon EMR Studio ユーザーロール。

ユーザーポリシーの例

次の基本ユーザーポリシーでは、ほとんどの EMR Studio アクションが許可されますが、ユーザーが新しい Amazon EMR クラスターを作成することはできません。

基本的なポリシー

> **Important**
>
> サンプルポリシーには、`CreateStudioPresignedUrl` アクセス許可が含まれていません。このアクセス許可は、IAM 認証モードを使用する場合はユーザーに許可する必要があります。詳細については、「EMR Studio にユーザーまたはグループを割り当てる」を参照してください。

サンプルポリシーには、EMR Studio のサンプルサービスロールでポリシーを使用できるように、タグベースのアクセスコントロール (TBAC) を適用する `Condition` 要素が含まれています。詳細については、「EMR Studio サービスロールを作成する」を参照してください。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowDefaultEC2SecurityGroupsCreationInVPCWithEMRTags",
         "Effect":"Allow",
         "Action":[
            "ec2:CreateSecurityGroup"
         ],
         "Resource":[
            "arn:aws:ec2:*:*:vpc/*"
         ],
         "Condition":{
            "StringEquals":{
               "aws:ResourceTag/for-use-with-amazon-emr-managed-policies":"true"
            }
         }
      },
      {
         "Sid":"AllowAddingEMRTagsDuringDefaultSecurityGroupCreation",
         "Effect":"Allow",
         "Action":[
            "ec2:CreateTags"
```

```
        ],
        "Resource":"arn:aws:ec2:*:*:security-group/*",
        "Condition":{
           "StringEquals":{
              "aws:RequestTag/for-use-with-amazon-emr-managed-policies":"true",
              "ec2:CreateAction":"CreateSecurityGroup"
           }
        }
     },
     {
        "Sid":"AllowSecretManagerListSecrets",
        "Action":[
           "secretsmanager:ListSecrets"
        ],
        "Resource":"*",
        "Effect":"Allow"
     },
     {
        "Sid":"AllowSecretCreationWithEMRTagsAndEMRStudioPrefix",
        "Effect":"Allow",
        "Action":"secretsmanager:CreateSecret",
        "Resource":"arn:aws:secretsmanager:*:*:secret:emr-studio-*",
        "Condition":{
           "StringEquals":{
              "aws:RequestTag/for-use-with-amazon-emr-managed-policies":"true"
           }
        }
     },
     {
        "Sid":"AllowAddingTagsOnSecretsWithEMRStudioPrefix",
        "Effect":"Allow",
        "Action":"secretsmanager:TagResource",
        "Resource":"arn:aws:secretsmanager:*:*:secret:emr-studio-*"
     },
     {
        "Sid":"AllowPassingServiceRoleForWorkspaceCreation",
        "Action":"iam:PassRole",
        "Resource":[
           "arn:aws:iam::*:role/<your-emr-studio-service-role>"
        ],
        "Effect":"Allow"
     },
     {
        "Sid":"AllowS3ListAndLocationPermissions",
```

```
        "Action":[
           "s3:ListAllMyBuckets",
           "s3:ListBucket",
           "s3:GetBucketLocation"
        ],
        "Resource":"arn:aws:s3:::*",
        "Effect":"Allow"
     },
     {
        "Sid":"AllowS3ReadOnlyAccessToLogs",
        "Action":[
           "s3:GetObject"
        ],
        "Resource":[
           "arn:aws:s3:::aws-logs-<aws-account-id>-<region>/elasticmapreduce/*"
        ],
        "Effect":"Allow"
     },
     {
        "Sid":"AllowConfigurationForWorkspaceCollaboration",
        "Action":[
           "elasticmapreduce:UpdateEditor",
           "elasticmapreduce:PutWorkspaceAccess",
           "elasticmapreduce:DeleteWorkspaceAccess",
           "elasticmapreduce:ListWorkspaceAccessIdentities"
        ],
        "Resource":"*",
        "Effect":"Allow",
        "Condition":{
           "StringEquals":{
              "elasticmapreduce:ResourceTag/creatorUserId":"${aws:userId}"
           }
        }
     },
     {
        "Sid":"DescribeNetwork",
        "Effect":"Allow",
        "Action":[
           "ec2:DescribeVpcs",
           "ec2:DescribeSubnets",
           "ec2:DescribeSecurityGroups"
        ],
        "Resource":"*"
     },
```

```
      {
         "Sid":"ListIAMRoles",
         "Effect":"Allow",
         "Action":[
            "iam:ListRoles"
         ],
         "Resource":"*"
      }
   ]
}
```

次の中級ユーザーポリシーでは、ほとんどの EMR Studio アクションが許可され、ユーザーはクラスターテンプレートを使用して新しい Amazon EMR クラスターを作成できます。

中級ポリシー

> **Important**
>
> サンプルポリシーには、CreateStudioPresignedUrl アクセス許可が含まれていません。このアクセス許可は、IAM 認証モードを使用する場合はユーザーに許可する必要があります。詳細については、「EMR Studio にユーザーまたはグループを割り当てる」を参照してください。

サンプルポリシーには、EMR Studio のサンプルサービスロールでポリシーを使用できるように、タグベースのアクセスコントロール (TBAC) を適用する Condition 要素が含まれています。詳細については、「EMR Studio サービスロールを作成する」を参照してください。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowEMRBasicActions",
         "Action":[
            "elasticmapreduce:CreateEditor",
            "elasticmapreduce:DescribeEditor",
            "elasticmapreduce:ListEditors",
            "elasticmapreduce:StartEditor",
            "elasticmapreduce:StopEditor",
            "elasticmapreduce:DeleteEditor",
            "elasticmapreduce:OpenEditorInConsole",
            "elasticmapreduce:AttachEditor",
```

```
            "elasticmapreduce:DetachEditor",
            "elasticmapreduce:CreateRepository",
            "elasticmapreduce:DescribeRepository",
            "elasticmapreduce:DeleteRepository",
            "elasticmapreduce:ListRepositories",
            "elasticmapreduce:LinkRepository",
            "elasticmapreduce:UnlinkRepository",
            "elasticmapreduce:DescribeCluster",
            "elasticmapreduce:ListInstanceGroups",
            "elasticmapreduce:ListBootstrapActions",
            "elasticmapreduce:ListClusters",
            "elasticmapreduce:ListSteps",
            "elasticmapreduce:CreatePersistentAppUI",
            "elasticmapreduce:DescribePersistentAppUI",
            "elasticmapreduce:GetPersistentAppUIPresignedURL",
            "elasticmapreduce:GetOnClusterAppUIPresignedURL"
         ],
         "Resource":"*",
         "Effect":"Allow"
      },
      {
         "Sid":"AllowEMRContainersBasicActions",
         "Action":[
            "emr-containers:DescribeVirtualCluster",
            "emr-containers:ListVirtualClusters",
            "emr-containers:DescribeManagedEndpoint",
            "emr-containers:ListManagedEndpoints",
            "emr-containers:DescribeJobRun",
            "emr-containers:ListJobRuns"
         ],
         "Resource":"*",
         "Effect":"Allow"
      },
      {
          "Sid": "AllowRetrievingManagedEndpointCredentials",
          "Effect": "Allow",
          "Action": [
              "emr-containers:GetManagedEndpointSessionCredentials"
          ],
          "Resource": [
              "arn:aws:emr-containers:<region>:<account-id>:/virtualclusters/<virtual-
cluster-id>/endpoints/<managed-endpoint-id>"
          ],
          "Condition": {
```

```
            "StringEquals": {
                "emr-containers:ExecutionRoleArn": [
                    "arn:aws:iam::<account-id>:role/<emr-on-eks-execution-role>"
                ]
            }
        }
    },
    {
       "Sid":"AllowSecretManagerListSecrets",
       "Action":[
          "secretsmanager:ListSecrets"
       ],
       "Resource":"*",
       "Effect":"Allow"
    },
    {
       "Sid":"AllowSecretCreationWithEMRTagsAndEMRStudioPrefix",
       "Effect":"Allow",
       "Action":"secretsmanager:CreateSecret",
       "Resource":"arn:aws:secretsmanager:*:*:secret:emr-studio-*",
       "Condition":{
          "StringEquals":{
             "aws:RequestTag/for-use-with-amazon-emr-managed-policies":"true"
          }
       }
    },
    {
       "Sid":"AllowAddingTagsOnSecretsWithEMRStudioPrefix",
       "Effect":"Allow",
       "Action":"secretsmanager:TagResource",
       "Resource":"arn:aws:secretsmanager:*:*:secret:emr-studio-*"
    },
    {
       "Sid":"AllowClusterTemplateRelatedIntermediateActions",
       "Action":[
          "servicecatalog:DescribeProduct",
          "servicecatalog:DescribeProductView",
          "servicecatalog:DescribeProvisioningParameters",
          "servicecatalog:ProvisionProduct",
          "servicecatalog:SearchProducts",
          "servicecatalog:UpdateProvisionedProduct",
          "servicecatalog:ListProvisioningArtifacts",
          "servicecatalog:ListLaunchPaths",
          "servicecatalog:DescribeRecord",
```

```
            "cloudformation:DescribeStackResources"
         ],
         "Resource":"*",
         "Effect":"Allow"
      },
      {
         "Sid":"AllowPassingServiceRoleForWorkspaceCreation",
         "Action":"iam:PassRole",
         "Resource":[
            "arn:aws:iam::*:role/<your-emr-studio-service-role>"
         ],
         "Effect":"Allow"
      },
      {
         "Sid":"AllowS3ListAndLocationPermissions",
         "Action":[
            "s3:ListAllMyBuckets",
            "s3:ListBucket",
            "s3:GetBucketLocation"
         ],
         "Resource":"arn:aws:s3:::*",
         "Effect":"Allow"
      },
      {
         "Sid":"AllowS3ReadOnlyAccessToLogs",
         "Action":[
            "s3:GetObject"
         ],
         "Resource":[
            "arn:aws:s3:::aws-logs-<aws-account-id>-<region>/elasticmapreduce/*"
         ],
         "Effect":"Allow"
      },
      {
         "Sid":"AllowConfigurationForWorkspaceCollaboration",
         "Action":[
            "elasticmapreduce:UpdateEditor",
            "elasticmapreduce:PutWorkspaceAccess",
            "elasticmapreduce:DeleteWorkspaceAccess",
            "elasticmapreduce:ListWorkspaceAccessIdentities"
         ],
         "Resource":"*",
         "Effect":"Allow",
         "Condition":{
```

```
            "StringEquals":{
               "elasticmapreduce:ResourceTag/creatorUserId":"${aws:userId}"
            }
         }
      },
      {
         "Sid":"DescribeNetwork",
         "Effect":"Allow",
         "Action":[
            "ec2:DescribeVpcs",
            "ec2:DescribeSubnets",
            "ec2:DescribeSecurityGroups"
         ],
         "Resource":"*"
      },
      {
         "Sid":"ListIAMRoles",
         "Effect":"Allow",
         "Action":[
            "iam:ListRoles"
         ],
         "Resource":"*"
      },
      {
         "Sid": "AllowServerlessActions",
         "Action": [
            "emr-serverless:CreateApplication",
            "emr-serverless:UpdateApplication",
            "emr-serverless:DeleteApplication",
            "emr-serverless:ListApplications",
            "emr-serverless:GetApplication",
            "emr-serverless:StartApplication",
            "emr-serverless:StopApplication",
            "emr-serverless:StartJobRun",
            "emr-serverless:CancelJobRun",
            "emr-serverless:ListJobRuns",
            "emr-serverless:GetJobRun",
            "emr-serverless:GetDashboardForJobRun",
            "emr-serverless:AccessInteractiveEndpoints"
         ],
         "Resource": "*",
         "Effect": "Allow"
        },
        {
```

```
            "Sid": "AllowPassingRuntimeRoleForRunningServerlessJob",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::*:role/serverless-runtime-role",
            "Effect": "Allow"
        }
    ]
}
```

次のアドバンストユーザーポリシーでは、すべての EMR Studio アクションが許可され、ユーザーはクラスターテンプレートを使用するか、クラスター設定を提供して新しい Amazon EMR クラスターを作成できます。

アドバンストポリシー

> **Important**
>
> サンプルポリシーには、CreateStudioPresignedUrl アクセス許可が含まれていません。このアクセス許可は、IAM 認証モードを使用する場合はユーザーに許可する必要があります。詳細については、「EMR Studio にユーザーまたはグループを割り当てる」を参照してください。

サンプルポリシーには、EMR Studio のサンプルサービスロールでポリシーを使用できるように、タグベースのアクセスコントロール (TBAC) を適用する Condition 要素が含まれています。詳細については、「EMR Studio サービスロールを作成する」を参照してください。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowEMRBasicActions",
         "Action":[
            "elasticmapreduce:CreateEditor",
            "elasticmapreduce:DescribeEditor",
            "elasticmapreduce:ListEditors",
            "elasticmapreduce:StartEditor",
            "elasticmapreduce:StopEditor",
            "elasticmapreduce:DeleteEditor",
            "elasticmapreduce:OpenEditorInConsole",
            "elasticmapreduce:AttachEditor",
            "elasticmapreduce:DetachEditor",
```

```
            "elasticmapreduce:CreateRepository",
            "elasticmapreduce:DescribeRepository",
            "elasticmapreduce:DeleteRepository",
            "elasticmapreduce:ListRepositories",
            "elasticmapreduce:LinkRepository",
            "elasticmapreduce:UnlinkRepository",
            "elasticmapreduce:DescribeCluster",
            "elasticmapreduce:ListInstanceGroups",
            "elasticmapreduce:ListBootstrapActions",
            "elasticmapreduce:ListClusters",
            "elasticmapreduce:ListSteps",
            "elasticmapreduce:CreatePersistentAppUI",
            "elasticmapreduce:DescribePersistentAppUI",
            "elasticmapreduce:GetPersistentAppUIPresignedURL",
            "elasticmapreduce:GetOnClusterAppUIPresignedURL"
         ],
         "Resource":"*",
         "Effect":"Allow"
      },
      {
         "Sid":"AllowEMRContainersBasicActions",
         "Action":[
            "emr-containers:DescribeVirtualCluster",
            "emr-containers:ListVirtualClusters",
            "emr-containers:DescribeManagedEndpoint",
            "emr-containers:ListManagedEndpoints",
            "emr-containers:DescribeJobRun",
            "emr-containers:ListJobRuns"
         ],
         "Resource":"*",
         "Effect":"Allow"
      },
      {
          "Sid": "AllowRetrievingManagedEndpointCredentials",
          "Effect": "Allow",
          "Action": [
              "emr-containers:GetManagedEndpointSessionCredentials"
          ],
          "Resource": [
              "arn:aws:emr-containers:<region>:<account-id>:/virtualclusters/<virtual-
cluster-id>/endpoints/<managed-endpoint-id>"
          ],
          "Condition": {
              "StringEquals": {
```

```
                "emr-containers:ExecutionRoleArn": [
                    "arn:aws:iam::<account-id>:role/<emr-on-eks-execution-role>"
                ]
            }
        }
    },
    {
       "Sid":"AllowSecretManagerListSecrets",
       "Action":[
          "secretsmanager:ListSecrets"
       ],
       "Resource":"*",
       "Effect":"Allow"
    },
    {
       "Sid":"AllowSecretCreationWithEMRTagsAndEMRStudioPrefix",
       "Effect":"Allow",
       "Action":"secretsmanager:CreateSecret",
       "Resource":"arn:aws:secretsmanager:*:*:secret:emr-studio-*",
       "Condition":{
          "StringEquals":{
             "aws:RequestTag/for-use-with-amazon-emr-managed-policies":"true"
          }
       }
    },
    {
       "Sid":"AllowAddingTagsOnSecretsWithEMRStudioPrefix",
       "Effect":"Allow",
       "Action":"secretsmanager:TagResource",
       "Resource":"arn:aws:secretsmanager:*:*:secret:emr-studio-*"
    },
    {
       "Sid":"AllowClusterTemplateRelatedIntermediateActions",
       "Action":[
          "servicecatalog:DescribeProduct",
          "servicecatalog:DescribeProductView",
          "servicecatalog:DescribeProvisioningParameters",
          "servicecatalog:ProvisionProduct",
          "servicecatalog:SearchProducts",
          "servicecatalog:UpdateProvisionedProduct",
          "servicecatalog:ListProvisioningArtifacts",
          "servicecatalog:ListLaunchPaths",
          "servicecatalog:DescribeRecord",
          "cloudformation:DescribeStackResources"
```

```
        ],
        "Resource":"*",
        "Effect":"Allow"
    },
    {
        "Sid":"AllowEMRCreateClusterAdvancedActions",
        "Action":[
            "elasticmapreduce:RunJobFlow"
        ],
        "Resource":"*",
        "Effect":"Allow"
    },
    {
        "Sid":"AllowPassingServiceRoleForWorkspaceCreation",
        "Action":"iam:PassRole",
        "Resource":[
            "arn:aws:iam::*:role/<your-emr-studio-service-role>",
            "arn:aws:iam::*:role/EMR_DefaultRole_V2",
            "arn:aws:iam::*:role/EMR_EC2_DefaultRole"
        ],
        "Effect":"Allow"
    },
    {
        "Sid":"AllowS3ListAndLocationPermissions",
        "Action":[
            "s3:ListAllMyBuckets",
            "s3:ListBucket",
            "s3:GetBucketLocation"
        ],
        "Resource":"arn:aws:s3:::*",
        "Effect":"Allow"
    },
    {
        "Sid":"AllowS3ReadOnlyAccessToLogs",
        "Action":[
            "s3:GetObject"
        ],
        "Resource":[
            "arn:aws:s3:::aws-logs-<aws-account-id>-<region>/elasticmapreduce/*"
        ],
        "Effect":"Allow"
    },
    {
        "Sid":"AllowConfigurationForWorkspaceCollaboration",
```

```
            "Action":[
               "elasticmapreduce:UpdateEditor",
               "elasticmapreduce:PutWorkspaceAccess",
               "elasticmapreduce:DeleteWorkspaceAccess",
               "elasticmapreduce:ListWorkspaceAccessIdentities"
            ],
            "Resource":"*",
            "Effect":"Allow",
            "Condition":{
               "StringEquals":{
                  "elasticmapreduce:ResourceTag/creatorUserId":"${aws:userId}"
               }
            }
         },
         {
            "Sid" : "SageMakerDataWranglerForEMRStudio",
            "Effect" : "Allow",
            "Action" : [
               "sagemaker:CreatePresignedDomainUrl",
               "sagemaker:DescribeDomain",
               "sagemaker:ListDomains",
               "sagemaker:ListUserProfiles"
            ],
            "Resource":"*"
         },
         {
            "Sid":"DescribeNetwork",
            "Effect":"Allow",
            "Action":[
               "ec2:DescribeVpcs",
               "ec2:DescribeSubnets",
               "ec2:DescribeSecurityGroups"
            ],
            "Resource":"*"
         },
         {
            "Sid":"ListIAMRoles",
            "Effect":"Allow",
            "Action":[
               "iam:ListRoles"
            ],
            "Resource":"*"
         },
         {
```

```
        "Sid": "AllowServerlessActions",
        "Action": [
            "emr-serverless:CreateApplication",
            "emr-serverless:UpdateApplication",
            "emr-serverless:DeleteApplication",
            "emr-serverless:ListApplications",
            "emr-serverless:GetApplication",
            "emr-serverless:StartApplication",
            "emr-serverless:StopApplication",
            "emr-serverless:StartJobRun",
            "emr-serverless:CancelJobRun",
            "emr-serverless:ListJobRuns",
            "emr-serverless:GetJobRun",
            "emr-serverless:GetDashboardForJobRun",
            "emr-serverless:AccessInteractiveEndpoints"
        ],
        "Resource": "*",
        "Effect": "Allow"
    },
    {
        "Sid": "AllowPassingRuntimeRoleForRunningServerlessJob",
        "Action": "iam:PassRole",
        "Resource": "arn:aws:iam::*:role/serverless-runtime-role",
        "Effect": "Allow"
    },
    {
        "Sid": "AllowCodeWhisperer",
        "Effect": "Allow",
        "Action": [ "codewhisperer:GenerateRecommendations" ],
        "Resource": "*"
    },
    {
        "Sid": "AllowAthenaSQL",
        "Action": [
            "athena:StartQueryExecution",
            "athena:StopQueryExecution",
            "athena:GetQueryExecution",
            "athena:GetQueryRuntimeStatistics",
            "athena:GetQueryResults",
            "athena:ListQueryExecutions",
            "athena:BatchGetQueryExecution",
            "athena:GetNamedQuery",
            "athena:ListNamedQueries",
            "athena:BatchGetNamedQuery",
```

```
            "athena:UpdateNamedQuery",
            "athena:DeleteNamedQuery",
            "athena:ListDataCatalogs",
            "athena:GetDataCatalog",
            "athena:ListDatabases",
            "athena:GetDatabase",
            "athena:ListTableMetadata",
            "athena:GetTableMetadata",
            "athena:ListWorkGroups",
            "athena:GetWorkGroup",
            "athena:CreateNamedQuery",
            "athena:GetPreparedStatement",
            "glue:CreateDatabase",
            "glue:DeleteDatabase",
            "glue:GetDatabase",
            "glue:GetDatabases",
            "glue:UpdateDatabase",
            "glue:CreateTable",
            "glue:DeleteTable",
            "glue:BatchDeleteTable",
            "glue:UpdateTable",
            "glue:GetTable",
            "glue:GetTables",
            "glue:BatchCreatePartition",
            "glue:CreatePartition",
            "glue:DeletePartition",
            "glue:BatchDeletePartition",
            "glue:UpdatePartition",
            "glue:GetPartition",
            "glue:GetPartitions",
            "glue:BatchGetPartition",
            "kms:ListAliases",
            "kms:ListKeys",
            "kms:DescribeKey",
            "lakeformation:GetDataAccess",
            "s3:GetBucketLocation",
            "s3:GetBucketLocation",
            "s3:GetObject",
            "s3:ListBucket",
            "s3:ListBucketMultipartUploads",
            "s3:ListMultipartUploadParts",
            "s3:AbortMultipartUpload",
            "s3:PutObject",
            "s3:PutBucketPublicAccessBlock",
```

```
            "s3:ListAllMyBuckets"
        ],
        "Resource": "*",
        "Effect": "Allow"
      }
   ]
}
```

次のユーザーポリシーには、EMR Studio Workspace で EMR Serverless インタラクティブアプリケーションを使用するために必要な最低限のユーザーアクセス許可が含まれています。

EMR Serverless インタラクティブポリシー

EMR Studio で EMR Serverless インタラクティブアプリケーションのユーザーアクセス許可を持つこのポリシー例では、 *serverless-runtime-role*と のプレースホルダーを、正しい EMR Studio サービスロールと EMR Serverless ランタイムロール に置き換え*emr-studio-service-role*ます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowServerlessActions",
            "Action": [
                "emr-serverless:CreateApplication",
                "emr-serverless:UpdateApplication",
                "emr-serverless:DeleteApplication",
                "emr-serverless:ListApplications",
                "emr-serverless:GetApplication",
                "emr-serverless:StartApplication",
                "emr-serverless:StopApplication",
                "emr-serverless:StartJobRun",
                "emr-serverless:CancelJobRun",
                "emr-serverless:ListJobRuns",
                "emr-serverless:GetJobRun",
                "emr-serverless:GetDashboardForJobRun",
                "emr-serverless:AccessInteractiveEndpoints"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Sid": "AllowEMRBasicActions",
```

```
            "Action": [
                "elasticmapreduce:CreateEditor",
                "elasticmapreduce:DescribeEditor",
                "elasticmapreduce:ListEditors",
                "elasticmapreduce:UpdateStudio",
                "elasticmapreduce:StartEditor",
                "elasticmapreduce:StopEditor",
                "elasticmapreduce:DeleteEditor",
                "elasticmapreduce:OpenEditorInConsole",
                "elasticmapreduce:AttachEditor",
                "elasticmapreduce:DetachEditor",
                "elasticmapreduce:CreateStudio",
                "elasticmapreduce:DescribeStudio",
                "elasticmapreduce:DeleteStudio",
                "elasticmapreduce:ListStudios",
                "elasticmapreduce:CreateStudioPresignedUrl"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Sid": "AllowPassingRuntimeRoleForRunningEMRServerlessJob",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::*:role/serverless-runtime-role",
            "Effect": "Allow"
        },
        {
            "Sid": "AllowPassingServiceRoleForWorkspaceCreation",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::*:role/emr-studio-service-role",
            "Effect": "Allow"
        },
        {
            "Sid": "AllowS3ListAndGetPermissions",
            "Action": [
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
                "s3:GetBucketLocation",
                "s3:GetObject"
            ],
            "Resource": "arn:aws:s3:::*",
            "Effect": "Allow"
        },
        {
```

```
            "Sid":"DescribeNetwork",
            "Effect":"Allow",
            "Action":[
                "ec2:DescribeVpcs",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups"
            ],
            "Resource":"*"
        },
        {
            "Sid": "ListIAMRoles",
            "Effect": "Allow",
            "Action": [
                "iam:ListRoles"
            ],
            "Resource": "*"
        }
    ]
}
```

EMR Studio ユーザーの AWS Identity and Access Management アクセス許可

次の表には、ユーザーが実行できる各 Amazon EMR Studio オペレーションが含まれており、そのオペレーションを実行するために必要な最小の IAM アクションのリストを示しています。これらのアクションは、EMR Studio に対して IAM アクセス許可ポリシー (IAM 認証を使用する場合) またはユーザーロールセッションポリシー (IAM Identity Center 認証を使用する場合) で許可します。

この表には、EMR Studio のアクセス許可ポリシーの各例で許可されるオペレーションも表示されます。許可ポリシーの例の詳細については、「EMR Studio ユーザーのアクセス許可ポリシーの作成」を参照してください。

[アクション]	Basic (ベーシック)	中級	アドバンスト	関連アクション
Workspace を作成および削除する	はい	はい	はい	`"elasticmapreduce:CreateEditor", "elasticmapreduce:DescribeEditor", "elasticmapreduce:ListEditors",`

[アクション]	Basic (ベーシック)	中級	アドバンスト	関連アクション
				`"elasticmapreduce:DeleteEditor"`
[コラボレーション] パネルを表示し、Workspace コラボレーションを有効にして、コラボレーターを追加します。詳細については、「Workspace コラボレーションの所有権の設定」を参照してください。	はい	はい	はい	`"elasticmapreduce:UpdateEditor", "elasticmapreduce:PutWorkspaceAccess", "elasticmapreduce:DeleteWorkspaceAccess", "elasticmapreduce:ListWorkspaceAccessIdentities"`
新しい EMR クラスターを作成するときに Studio と同じアカウントの Amazon S3 Control ストレージバケットのリストを表示する、Web UI を使用してアプリケーションをデバッグするときにコンテナログにアクセスする	はい	はい	はい	`"s3:ListAllMyBuckets", "s3:ListBucket", "s3:GetBucketLocation", "s3:GetObject"`
Workspace にアクセスする	はい	はい	はい	`"elasticmapreduce:DescribeEditor", "elasticmapreduce:ListEditors", "elasticmapreduce:StartEditor", "elasticmapreduce:StopEditor", "elasticmapreduce:OpenEditorInConsole"`

[アクション]	Basic (ベーシック)	中級	アドバンスト	関連アクション
Workspace に関連付けられている既存の Amazon EMR クラスターをアタッチまたはデタッチする	はい	はい	はい	`"elasticmapreduce:AttachEditor", "elasticmapreduce:DetachEditor", "elasticmapreduce:ListClusters", "elasticmapreduce:DescribeCluster", "elasticmapreduce:ListInstanceGroups", "elasticmapreduce:ListBootstrapActions"`
Amazon EMR on EKS クラスターをアタッチまたはデタッチする	はい	はい	はい	`"elasticmapreduce:AttachEditor", "elasticmapreduce:DetachEditor", "emr-containers:ListVirtualClusters", "emr-containers:DescribeVirtualCluster", "emr-containers:ListManagedEndpoints", "emr-containers:DescribeManagedEndpoint", "emr-containers:GetManagedEndpointSessionCredentials"`

[アクション]	Basic (ベーシック)	中級	アドバンスト	関連アクション
Workspace に関連付けられた EMR Serverless アプリケーションをアタッチまたはデタッチする	いいえ	はい	はい	`"elasticmapreduce:AttachEditor", "elasticmapreduce:DetachEditor", "emr-serverless:GetApplication", "emr-serverless:StartApplication", "emr-serverless:ListApplications", "emr-serverless:GetDashboardForJobRun", "emr-serverless:AccessInteractiveEndpoints", "iam:PassRole"` EMR Serverless ジョブのランタイムロールを渡すには、PassRole アクセス許可が必要です。詳細については、「Amazon EMR Serverless ユーザーガイド」の「Job runtime roles」を参照してください。

[アクション]	Basic (ベーシック)	中級	アドバンスト	関連アクション
永続的なアプリケーションユーザーインターフェイスで Amazon EMR on EC2 ジョブをデバッグする	はい	はい	はい	`"elasticmapreduce:CreatePersistentAppUI", "elasticmapreduce:DescribePersistentAppUI", "elasticmapreduce:GetPersistentAppUIPresignedURL", "elasticmapreduce:ListClusters", "elasticmapreduce:ListSteps", "elasticmapreduce:DescribeCluster", "s3:ListBucket", "s3:GetObject"`
クラスター上のアプリケーションユーザーインターフェイスで Amazon EMR on EC2 ジョブをデバッグする	はい	はい	はい	`"elasticmapreduce:GetOnClusterAppUIPresignedURL"`

[アクション]	Basic (ベーシック)	中級	アドバンスト	関連アクション
Spark History Server を使用して Amazon EMR on EKS ジョブ実行をデバッグする	はい	はい	はい	`"elasticmapreduce:CreatePersistentAppUI", "elasticmapreduce:DescribePersistentAppUI", "elasticmapreduce:GetPersistentAppUIPresignedURL", "emr-containers:ListVirtualClusters", "emr-containers:DescribeVirtualCluster", "emr-containers:ListJobRuns", "emr-containers:DescribeJobRun", "s3:ListBucket", "s3:GetObject"`
Git リポジトリを作成および削除する	はい	はい	はい	`"elasticmapreduce:CreateRepository", "elasticmapreduce:DeleteRepository", "elasticmapreduce:ListRepositories", "elasticmapreduce:DescribeRepository", "secretsmanager:CreateSecret", "secretsmanager:ListSecrets", "secretsmanager:TagResource"`

[アクション]	Basic (ベーシック)	中級	アドバンスト	関連アクション
Git リポジトリをリンクまたはリンク解除する	はい	はい	はい	`"elasticmapreduce:LinkRepository", "elasticmapreduce:UnlinkRepository", "elasticmapreduce:ListRepositories", "elasticmapreduce:DescribeRepository"`
定義済みのクラスターテンプレートから新しいクラスターを作成する	いいえ	はい	はい	`"servicecatalog:SearchProducts", "servicecatalog:DescribeProduct", "servicecatalog:DescribeProductView", "servicecatalog:DescribeProvisioningParameters", "servicecatalog:ProvisionProduct", "servicecatalog:UpdateProvisionedProduct", "servicecatalog:ListProvisioningArtifacts", "servicecatalog:DescribeRecord", "servicecatalog:ListLaunchPaths", "cloudformation:DescribeStackResources", "elasticmapreduce:ListClusters", "elasticmapreduce:DescribeCluster"`

[アクション]	Basic (ベーシック)	中級	アドバンスト	関連アクション
クラスター設定を指定して新しいクラスターを作成します。	いいえ	いいえ	はい	`"elasticmapreduce:RunJobFlow", "iam:PassRole", "elasticmapreduce:ListClusters", "elasticmapreduce:DescribeCluster"`
IAM 認証モードを使用するときに、Studio にユーザーを割り当てます。	いいえ	いいえ	いいえ	`"elasticmapreduce:CreateStudioPresignedUrl"`
ネットワークオブジェクト記述する。	はい	はい	はい	`{ "Version": "2012-10-17", "Statement": [ { "Sid": "DescribeNetwork", "Effect": "Allow", "Action": [ "ec2:DescribeVpcs", "ec2:DescribeSubnets", "ec2:DescribeSecurityGroups" ], "Resource": "*" } ] }`

[アクション]	Basic (ベーシック)	中級	アドバンスト	関連アクション
IAM ロールを一覧表示する。	はい	はい	はい	`{ "Version": "2012-10-17", "Statement": [ { "Sid": "ListIAMRoles", "Effect": "Allow", "Action": [ "iam:ListRoles" ], "Resource": "*" } ] }`
Amazon Studio から EMR SageMaker Studio に接続し、Data Wrangler ビジュアルインターフェイスを使用します。	いいえ	いいえ	はい	`"sagemaker:CreatePresignedDomainUrl", "sagemaker:DescribeDomain", "sagemaker:ListDomains", "sagemaker:ListUserProfiles"`
EMR Studio CodeWhisperer で Amazon を使用します。	いいえ	いいえ	はい	`"codewhisperer:GenerateRecommendations"`

[アクション]	Basic (ベーシック)	中級	アドバンスト	関連アクション
EMR Studio から Amazon Athena SQL エディターにアクセスします。このリストには、Athena のすべての機能を使用するために必要なすべてのアクセス許可が含まれていない場合があります。ほとんどの up-to-date リストについては、「Athena フルアクセスポリシー」を参照してください。	いいえ	いいえ	はい	`"athena:StartQueryExecution", "athena:StopQueryExecution", "athena:GetQueryExecution", "athena:GetQueryRuntimeStatistics", "athena:GetQueryResults", "athena:ListQueryExecutions", "athena:BatchGetQueryExecution", "athena:GetNamedQuery", "athena:ListNamedQueries", "athena:BatchGetNamedQuery", "athena:UpdateNamedQuery", "athena:DeleteNamedQuery", "athena:ListDataCatalogs", "athena:GetDataCatalog", "athena:ListDatabases", "athena:GetDatabase", "athena:ListTableMetadata", "athena:GetTableMetadata", "athena:ListWorkGroups", "athena:GetWorkGroup", "athena:CreateNamedQuery", "athena:GetPreparedStatement", "glue:CreateDatabase",`

[アクション]	Basic (ベーシック)	中級	アドバンスト	関連アクション
				"glue:DeleteDatabase", "glue:GetDatabase", "glue:GetDatabases", "glue:UpdateDatabase", "glue:CreateTable", "glue:DeleteTable", "glue:BatchDeleteTable", "glue:UpdateTable", "glue:GetTable", "glue:GetTables", "glue:BatchCreatePartition", "glue:CreatePartition", "glue:DeletePartition", "glue:BatchDeletePartition", "glue:UpdatePartition", "glue:GetPartition", "glue:GetPartitions", "glue:BatchGetPartition", "kms:ListAliases", "kms:ListKeys", "kms:DescribeKey", "lakeformation:GetDataAccess", "s3:GetBucketLocation", "s3:GetBucketLocation", "s3:GetObject", "s3:ListBucket", "s3:ListBucketMultipartUploads", "s3:ListMultipartUploadParts", "s3:AbortMultipartUpload", "s3:PutObject", "s3:PutBucketPublicAccessBlock",

[アクション]	Basic (ベーシック)	中級	アドバンスト	関連アクション
				"s3:ListAllMyBuckets"

EMR Studio の作成

Amazon EMR コンソールまたは AWS CLI を使用してチームの EMR Studio を作成できます。Studio インスタンスの作成は、Amazon EMR Studio の設定の一部です。

> **Note**
>
> Amazon EMR コンソールは、再設計され、使いやすくなりました。古いコンソールと新しいコンソールエクスペリエンスの違いについては、「Amazon EMR コンソール」を参照してください。

前提条件

Studio を作成する前に、「Amazon EMR Studio を設定する」の前のタスクを完了していることを確認してください。

AWS CLI を使用して Studio を作成するには、最新バージョンがインストールされている必要があります。詳細については、「Installing or updating the latest version of the AWS CLI」を参照してください。

> **Important**
>
> Studio を作成する前に、ブラウザ SwitchyOmega で FoxyProxy や などのプロキシ管理ツールを無効にします。アクティブなプロキシを使用している場合、[Create Studio] (Studio の作成) を選択すると、[Network Failure] (ネットワーク障害) エラーメッセージが表示されることがあります。

Amazon EMR では、Studio を簡単に作成できるコンソールエクスペリエンスが提供されるため、デフォルト設定ですぐに開始できます。 を使用して、デフォルト設定でインタラクティブなワーク

ロードまたはバッチジョブを実行できます。EMR Studio を作成すると、インタラクティブジョブ用の EMR Serverless アプリケーションも作成されます。

Studio の設定を完全に制御したい場合は、カスタム を選択できます。これにより、追加設定をすべて設定できます。

Interactive workloads

インタラクティブワークロード用の EMR Studio を作成するには

1. https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションの [EMR Studio] で、[はじめに] を選択します。[Studios] ページから新しい Studio を作成することもできます。
3. インタラクティブワークロード用に EMR Studio を作成する場合は、Amazon EMR にデフォルト設定が用意されていますが、これらの設定を編集できます。設定可能な設定には、EMR Studio の名前、Workspace の S3 の場所、使用するサービスロール、使用する Workspace (複数可) 、EMR Serverless アプリケーション名、関連するランタイムロールが含まれます。
4. Studio の作成を選択し、Workspace を起動して終了し、Studio ページに移動します。新しい Studio がリストに表示され、[Studio name] (Studio 名)、[Creation date] (作成日)、[Studio access URL] (Studio アクセス URL) などの詳細が表示されます。Workspace がブラウザの新しいタブで開きます。

Batch jobs

インタラクティブワークロード用の EMR Studio を作成するには

1. https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションの [EMR Studio] で、[はじめに] を選択します。[Studios] ページから新しい Studio を作成することもできます。
3. バッチジョブ用に EMR Studio を作成する場合は、Amazon EMR にデフォルト設定が用意されていますが、これらの設定を編集できます。設定可能な設定には、EMR Studio の名前、EMR Serverless アプリケーション名、関連するランタイムロールが含まれます。
4. Studio を作成して Workspace を起動するを選択して終了し、Studio ページに移動します。新しい Studio がリストに表示され、[Studio name] (Studio 名)、[Creation date] (作成日)、[Studio access URL] (Studio アクセス URL) などの詳細が表示されます。EMR Studio がブラウザの新しいタブで開きます。

Custom settings

カスタム設定で EMR Studio を作成するには

1. https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションの [EMR Studio] で、[はじめに] を選択します。[Studios] ページから新しい Studio を作成することもできます。
3. [Studio を作成] を選択して、[Studio を作成] ページを開きます。
4. Studio 名を入力します。
5. 新しい S3 バケットを作成するか、既存の場所を使用するかを選択します。
6. Studio に追加する Workspace を選択します。最大 3 つの Workspace を追加できます。
7. [Authentication] (認証) で、Studio の認証モードを選択し、次の表に従って情報を入力します。EMR Studio の認証の詳細については、「Amazon EMR Studio の認証モードの選択」を参照してください。

使用するもの	手順
IAM 認証またはフェデレーション	デフォルトの認証方法は AWS Identity and Access Management (IAM) です。EMR Studio にユーザーまたはグループを割り当てる で説明されているように、特定のユーザーに Studio へのアクセスを許可するタグを追加することもできます。 フェデレーティッドユーザーが ID プロバイダー (IdP) の Studio URL と認証情報を使用してログインできるようにする場合は、ドロップダウンリストから IdP を選択し、ID プロバイダー (IdP) ログイン URL とRelayStateパラメータ名を入力します。 IdP 認証 URLs「」を参照してくださいIDプロバイダーの RelayState パラメータと認証 URLs。 RelayState
IAM Identity Center 認証	EMR Studio の[Service Role] (サービスロール) および [User Role] (ユーザーロール)

使用するもの	手順
	を選択します。詳細については、「EMR Studio サービスロールを作成する と IAM Identity Center 認証モードの EMR Studio ユーザーロールの作成」を参照してください。 Studio で [IAM Identity Center (以前は AWS シングルサインオン)] 認証を使用する場合、[信頼できる ID 伝達を有効にする] オプションを使用してユーザーのサインオン操作を効率化することができます。信頼できる ID 伝達を使用すると、ユーザーは Studio を使用するときに Identity Center の認証情報を使用してログインし、その ID をダウンストリームの AWS サービスに伝達できます。 [アプリケーションアクセス] セクションでは、Identity Center 内のすべてのユーザーとグループが Studio にアクセスできるようにするか、または選択した割り当てられたユーザーとグループのみが Studio にアクセスできるようにするかを指定することもできます。 詳細については、「Amazon EMR を と統合する AWS IAM Identity Center」を参照してください。また、「AWS IAM Identity Center ユーザーガイド」の「アプリケーション間での信頼できる ID 伝達」も参照してください。

8. VPC の場合は、ドロップダウンリストから Studio の Amazon Virtual Private Cloud (VPC) を選択します。
9. [Subnets] (サブネット) の下で、Studio に関連付ける VPC 内のサブネットを最大 5 つ選択します。Studio を作成した後に、さらにサブネットを追加することもできます。

10. [Security groups] (セキュリティグループ) で、デフォルトのセキュリティグループまたはカスタムセキュリティグループのいずれかを選択します。詳細については、「EMR Studio ネットワークトラフィックを制御するセキュリティグループを定義する」を参照してください。

選択内容	手順
デフォルトの EMR Studio セキュリティグループ	Studio で Git ベースのリポジトリリンクを有効にするには、[Enable clusters/endpoints and Git repository] (クラスター/エンドポイントと Git リポジトリを有効にする) を選択します。それ以外の場合は、[Enable clusters/endpoints] (クラスタ/エンドポイントを有効にする) を選択します。
Studio のカスタムセキュリティグループ	• [Cluster/endpoint security group] (クラスタ/エンドポイントセキュリティグループ) で、設定したエンジンセキュリティグループをドロップダウンリストから選択します。Studio は、このセキュリティグループを使用して、アタッチされた Workspace からのインバウンドアクセスを許可します。 • [Workspace security group] (Workspace セキュリティグループ) で、設定した Workspace セキュリティグループをドロップダウンリストから選択します。Studio は Workspace でこのセキュリティグループを使用して、アタッチされた Amazon EMR クラスターおよびパブリックにホストされている Git リポジトリへのアウトバウンドアクセスを提供します。

11. Studio およびその他の リソースにタグを追加します。タグの詳細については、「クラスターにタグを付ける」を参照してください。

12. Studio の作成を選択し、Workspace を起動して終了し、Studio ページに移動します。新しい Studio がリストに表示され、[Studio name] (Studio 名)、[Creation date] (作成日)、[Studio access URL] (Studio アクセス URL) などの詳細が表示されます。

Studio を作成したら、「EMR Studio にユーザーまたはグループを割り当てる」の手順に従います。

CLI

> **Note**
>
> 読みやすくするために、Linux 行連続文字 (\) が含まれています。Linux コマンドでは、これらは削除することも、使用することもできます。Windows の場合、削除するか、キャレット (^) に置き換えてください。

Example - 認証に IAM を使用する EMR Studio を作成する

次の AWS CLI サンプルコマンドは、IAM 認証モードで EMR Studio を作成します。Studio で IAM 認証またはフェデレーションを使用する場合は、`--user-role` を指定しません。

フェデレーティッドユーザーが Studio URL と ID プロバイダー (IdP) の認証情報を使用してログインできるようにするには、`--idp-auth-url` と `--idp-relay-state-parameter-name` を指定します。IdP 認証 URLs「」を参照してくださいID プロバイダーの RelayState パラメータと認証 URLs。 RelayState

```
aws emr create-studio \
--name <example-studio-name> \
--auth-mode IAM \
--vpc-id <example-vpc-id> \
--subnet-ids <subnet-id-1> <subnet-id-2>... <subnet-id-5>  \
--service-role <example-studio-service-role-name> \
--user-role studio-user-role-name \
--workspace-security-group-id <example-workspace-sg-id> \
--engine-security-group-id <example-engine-sg-id> \
--default-s3-location <example-s3-location> \
--idp-auth-url <https://EXAMPLE/login/> \
--idp-relay-state-parameter-name <example-RelayState>
```

Example – 認証に Identity Center を使用する EMR Studio を作成する

次の AWS CLI サンプルコマンドは、IAM Identity Center 認証モードで EMR Studio を作成します。IAM Identity Center 認証を使用する場合は、`--user-role` を指定する必要があります。

IAM Identity Center 認証モードの詳細については、「Amazon EMR Studio の IAM Identity Center 認証モードの設定」を参照してください。

```
aws emr create-studio \
--name <example-studio-name> \
--auth-mode SSO \
--vpc-id <example-vpc-id> \
--subnet-ids <subnet-id-1> <subnet-id-2>... <subnet-id-5>  \
--service-role <example-studio-service-role-name> \
--user-role <example-studio-user-role-name> \
--workspace-security-group-id <example-workspace-sg-id> \
--engine-security-group-id <example-engine-sg-id> \
--default-s3-location <example-s3-location>
--trusted-identity-propagation-enabled \
--idc-user-assignment OPTIONAL \
--idc-instance-arn <iam-identity-center-instance-arn>
```

Example - **`aws emr create-studio`** の CLI 出力

以下に、Studio の作成後に表示される出力の例を示します。

```
{
    StudioId: "es-123XXXXXXXXX",
    Url: "https://es-123XXXXXXXXX.emrstudio-prod.us-east-1.amazonaws.com"
}
```

`create-studio` コマンドの詳細については、「AWS CLI コマンドリファレンス」を参照してください。

ID プロバイダーの RelayState パラメータと認証 URLs

IAM フェデレーションを使用し、ユーザーに Studio URL と ID プロバイダー (IdP) の認証情報を使用してログインさせたい場合は、 時に ID プロバイダー (IdP) のログイン URL とRelayStateパラメータ名を指定できますEMR Studio の作成。

次の表は、一般的な ID プロバイダーの標準認証 URL と RelayState パラメータ名を示しています。

ID プロバイダー	パラメータ	認証 URL
Auth0	RelayState	https://*<sub_domain>* .auth0.com/samlp/*<app_id>*
Google アカウント	RelayState	https://accounts.google.com/o/saml2/initsso?idpid= *<idp_id>*&spid=*<sp_id>*&forceauthn=false
Microsoft Azure	RelayState	https://myapps.microsoft.com/signin/ *<app_name>* /*<app_id>*?tenantId= *<tenant_id>*
Okta	RelayState	https://*<sub_domain>* .okta.com/app/*<app_name>* /*<app_id>*/sso/saml
PingFederate	TargetResource	https://*<host>*/idp/*<idp_id>*/startSSO.ping?PartnerSpId= *<sp_id>*
PingOne	TargetResource	https://sso.connect.pingidentity.com/sso/sp/initsso?saasid= *<app_id>*&idpid=*<idp_id>*

EMR Studio ユーザーの割り当てと管理

EMR Studio を作成したら、その Studio にユーザーとグループを割り当てることができます。ユーザーの割り当て、更新、削除に使用する方法は、Studio 認証モードによって異なります。

- IAM 認証モードを使用する場合、IAM または IAM と ID プロバイダーで EMR Studio ユーザーの割り当てとアクセス許可を設定します。
- IAM Identity Center 認証モードでは、Amazon EMR マネジメントコンソールまたは AWS CLI を使用して、ユーザーを管理します。

Amazon EMR Studio の認証の詳細については、「Amazon EMR Studio の認証モードの選択」を参照してください。

EMR Studio にユーザーまたはグループを割り当てる

IAM

「Amazon EMR Studio の IAM 認証モードの設定」を行う際に、ユーザーの IAM アクセス許可ポリシーで `CreateStudioPresignedUrl` アクションを許可し、ユーザーを特定の Studio に制限する必要があります。IAM 認証モードのユーザーアクセス許可 に `CreateStudioPresignedUrl` を含めるか、別のポリシーを使用できます。

ユーザーを特定の Studio (または Studio のセット) に制限するには、属性ベースのアクセスコントロール (ABAC) を使用するか、Studio の Amazon リソースネーム (ARN) をアクセス許可ポリシーの `Resource` 要素で指定できます。

Example Studio ARN を使用してユーザーを Studio に割り当てる

次のポリシー例では、`CreateStudioPresignedUrl` アクションを許可し、`Resource` 要素に Studio の Amazon リソースネーム (ARN) を指定することで、ユーザーに特定の EMR Studio へのアクセス権を付与しています。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCreateStudioPresignedUrl",
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:CreateStudioPresignedUrl"
            ],
            "Resource": "arn:aws:elasticmapreduce:<region>:<account-
id>:studio/<studio-id>"
        }
    ]
}
```

Example IAM 認証用に ABAC を使用して Studio にユーザーを割り当てる

Studio の属性ベースアクセスコントロール (ABAC) を設定する方法は複数あります。例えば、EMR Studio に 1 つ以上のタグをアタッチし、`CreateStudioPresignedUrl` アクションを、それらのタグを持つ特定の Studio または Studio のセットに制限する IAM ポリシーを作成できます。

Studio の作成中または作成後にタグを追加できます。タグを既存の Studio に追加するには、AWS CLI `emr add-tags` コマンドを使用します。次の例では、キーと値のペア `Team = Data Analytics` を持つタグを EMR Studio に追加します。

```
aws emr add-tags --resource-id <example-studio-id> --tags Team="Data Analytics"
```

次のアクセス許可ポリシー例では、タグのキーと値のペア `Team = DataAnalytics` を持つ EMR Studio に対して `CreateStudioPresignedUrl` アクションを許可しています。タグを使用したアクセス制御の詳細については、「タグを使用したユーザーおよびロールへのアクセスとそのユーザーおよびロールのアクセスの制御」または「タグを使用した AWS リソースへのアクセスの制御」を参照してください。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCreateStudioPresignedUrl",
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:CreateStudioPresignedUrl"
            ],
            "Resource": "arn:aws:elasticmapreduce:<region>:<account-id>:studio/*",
            "Condition": {
                "StringEquals": {
                    "elasticmapreduce:ResourceTag/Team": "Data Analytics"
                }
            }
        }
    ]
}
```

Example aws:SourceIdentity global 条件キーを使用して Studio にユーザーを割り当てる

IAM フェデレーションを使用する場合、アクセス許可ポリシーでグローバル条件キー `aws:SourceIdentity` を使用して、フェデレーションの IAM ロールを引き受けるときに Studio へのアクセス権をユーザーに付与できます。

まず、ユーザーが認証してフェデレーションの IAM ロールを引き受けるときに、E メールアドレスやユーザー名などの識別文字列を返すように ID プロバイダー (IdP) を設定する必要があります。IAM は、グローバル条件キー `aws:SourceIdentity` を IdP から返された識別文字列に設定します

詳細については、 AWS セキュリティブログの「IAM ロールアクティビティを企業 ID に関連付ける方法」ブログ記事と、 グローバル条件キーリファレンスの「aws:SourceIdentity エントリ」を参照してください。

次のポリシー例では、 CreateStudioPresignedUrlアクションを許可し、*#example-source-identity#* で指定された EMR Studio への <*example-studio-arn#* アクセスaws:SourceIdentityと一致する を持つユーザーに付与します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "elasticmapreduce:CreateStudioPresignedUrl",
      "Resource": "<example-studio-arn>",
      "Condition": {
        "StringLike": {
          "aws:SourceIdentity": "<example-source-identity>"
        }
      }
    }
  ]
}
```

IAM Identity Center

ユーザーまたはグループを EMR Studio に割り当てるときは、そのユーザーまたはグループに対して、新しい EMR クラスターを作成する機能など、きめ細かいアクセス許可を定義するセッションポリシーを指定します。Amazon EMR では、これらのセッションポリシーマッピングが保存されます。ユーザーまたはグループのセッションポリシーは、割り当て後に更新できます。

> **Note**
>
> ユーザーまたはグループの最終的なアクセス許可は、EMR Studio ユーザーロールで定義されたアクセス許可と、そのユーザーまたはグループのセッションポリシーで定義されているアクセス許可の共通部分です。ユーザーが Studio に割り当てられた複数のグループに属している場合、EMR Studio はそのユーザーに対してアクセス許可の和集合を使用します。

Amazon EMR コンソールを使用して EMR Studio にユーザーまたはグループを割り当てるには

1. 新しい Amazon EMR コンソールに移動し、サイドナビゲーションから [古いコンソールに切り替え] を選択します。古いコンソールに切り替えたときの動作の詳細については、「Using the old console」を参照してください。
2. 左のナビゲーションから [EMR Studio] を選択します。
3. Studio 名を [Studios] (Studio) リストから選択するか、Studio を選択して[View details] (詳細を表示) を選択して、Studio の詳細ページを開きます。
4. [Add Users] (ユーザーの追加) を選択して、[Users] (ユーザー) および [Groups] (グループ) 検索テーブルを表示します。
5. [Users] (ユーザー) タブまたは [Groups] (グループ) タブを選択し、検索バーに検索語を入力して、ユーザーまたはグループを検索します。
6. 検索結果リストから 1 つ以上のユーザーまたはグループを選択します。[Users] (ユーザー) タブと [Groups] (グループ) タブを切り替えることができます。
7. Studio に追加するユーザーおよびグループを選択したら、[Add] (追加) を選択します。ユーザーとグループが [Studio users] (Studio ユーザー) リストに表示されます。リストが更新されるまでに数秒かかることがあります。
8. 「Studio に割り当てられたユーザーまたはグループのアクセス許可を更新する」の指示に従って、ユーザーまたはグループの Studio アクセス許可を絞り込みます。

AWS CLI を使用して EMR Studio にユーザーまたはグループを割り当てるには

次の `create-studio-session-mapping` 引数に独自の値を挿入します。`create-studio-session-mapping` コマンドの詳細については、「AWS CLI Command Reference」を参照してください。

- **`--studio-id`** - ユーザーまたはグループを割り当てる Studio の ID。Studio ID を取得する方法の手順については、「Studio の詳細の表示」を参照してください。
- **`--identity-name`** - ID ストアからのユーザーまたはグループの名前。詳細については、UserName「 Identity Store API リファレンス」の「 ユーザーの場合は」を、グループDisplayNameの場合は「」を参照してください。
- **`--identity-type`** - `USER` または `GROUP` のいずれかを使用して、ID タイプを指定します。
- **`--session-policy-arn`** - ユーザーまたはグループに関連付けるセッションポリシーの Amazon リソースネーム (ARN)。例えば、**`arn:aws:iam::<aws-account-`**

id:policy/*EMRStudio_Advanced_User_Policy* です。詳細については、「EMR Studio ユーザーのアクセス許可ポリシーの作成」を参照してください。

```
aws emr create-studio-session-mapping \
 --studio-id <example-studio-id> \
 --identity-name <example-identity-name> \
 --identity-type <USER-or-GROUP> \
 --session-policy-arn <example-session-policy-arn>
```

> **Note**
>
> 読みやすくするために、Linux 行連続文字 (\) が含まれています。Linux コマンドでは、これらは削除することも、使用することもできます。Windows の場合、削除するか、キャレット (^) に置き換えてください。

get-studio-session-mapping コマンドを使用して、新しい割り当てを確認します。*# example-identity-name#* を、更新したユーザーまたはグループの IAM Identity Center 名に置き換えます。

```
aws emr get-studio-session-mapping \
 --studio-id <example-studio-id> \
 --identity-type <USER-or-GROUP> \
 --identity-name <user-or-group-name> \
```

Studio に割り当てられたユーザーまたはグループのアクセス許可を更新する

IAM

IAM 認証モードを使用するときにユーザーまたはグループのアクセス許可を更新するには、IAM を使用して IAM ID (ユーザー、グループ、またはロール) にアタッチされた IAM アクセス許可ポリシーを変更します。

詳細については、「IAM 認証モードのユーザーアクセス許可」を参照してください。

IAM Identity Center

コンソールを使用してユーザーまたはグループの EMR Studio アクセス許可を更新するには

1. 新しい Amazon EMR コンソールに移動し、サイドナビゲーションから [古いコンソールに切り替え] を選択します。古いコンソールに切り替えたときの動作の詳細については、「Using the old console」を参照してください。
2. 左のナビゲーションから [EMR Studio] を選択します。
3. Studio 名を [Studios] (Studio) リストから選択するか、Studio を選択して[View details] (詳細を表示) を選択して、Studio の詳細ページを開きます。
4. Studio の詳細ページの [Studio users] (Studio ユーザー) リストで、更新するユーザーまたはグループを検索します。名前または ID タイプで検索できます。
5. 更新対象のユーザーまたはグループを選択し、[Assign policy] (ポリシーの割り当て) を選択して [Session policy] (セッションポリシー) ダイアログボックスを開きます。
6. 手順 5 で選択したユーザーまたはグループに適用するポリシーを選択し、[Apply policy] (ポリシーを適用) を選択します。[Studio users] (Studio ユーザー) リストでは、更新したユーザーまたはグループの [Session policy] (セッションポリシー) 列にポリシー名が表示されます。

AWS CLI を使用してユーザーまたはグループの EMR Studio アクセス許可を更新するには

次の `update-studio-session-mappings` 引数に独自の値を挿入します。`update-studio-session-mappings` コマンドの詳細については、「AWS CLI Command Reference」を参照してください。

```
aws emr update-studio-session-mapping \
 --studio-id <example-studio-id> \
 --identity-name <name-of-user-or-group-to-update> \
 --session-policy-arn <new-session-policy-arn-to-apply> \
 --identity-type <USER-or-GROUP> \
```

`get-studio-session-mapping` コマンドを使用して、新しいセッションポリシーの割り当てを確認します。*`#example-identity-name#`* を、更新したユーザーまたはグループの IAM Identity Center 名に置き換えます。

```
aws emr get-studio-session-mapping \
 --studio-id <example-studio-id> \
```

```
  --identity-type <USER-or-GROUP> \
  --identity-name <user-or-group-name> \
```

Studio からユーザーまたはグループを削除する

IAM

IAM 認証モードを使用するときに EMR Studio からユーザーまたはグループを削除するには、ユーザーの IAM アクセス許可ポリシーを再構成して、Studio へのユーザーのアクセスを取り消す必要があります。

次のポリシーの例では、タグのキーと値のペア `Team = Quality Assurance` を持つ EMR Studio があると仮定します。ポリシーに従って、ユーザーは、値が `Data Analytics` または `Quality Assurance` のいずれかに等しい `Team` キーでタグ付けされた Studio にアクセスできます。`Team = Quality Assurance` でタグ付けされた Studio からユーザーを削除するには、タグ値のリストから `Quality Assurance` を削除します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCreateStudioPresignedUrl",
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:CreateStudioPresignedUrl"
            ],
            "Resource": "arn:aws:elasticmapreduce:<region>:<account-id>:studio/*",
            "Condition": {
                "StringEquals": {
                    "emr:ResourceTag/Team": [
                        "Data Analytics",
                        "Quality Assurance"
                  ]
                }
            }
        }
    ]
}
```

IAM Identity Center

コンソールを使用して EMR Studio からユーザーまたはグループを削除するには

1. 新しい Amazon EMR コンソールに移動し、サイドナビゲーションから [古いコンソールに切り替え] を選択します。古いコンソールに切り替えたときの動作の詳細については、「Using the old console」を参照してください。
2. 左のナビゲーションから [EMR Studio] を選択します。
3. Studio 名を [Studios] (Studio) リストから選択するか、Studio を選択して[View details] (詳細を表示) を選択して、Studio の詳細ページを開きます。
4. Studio の詳細ページの [Studio users] (Studio ユーザー) リストで、Studio から削除するユーザーまたはグループを検索します。名前または ID タイプで検索できます。
5. 削除するユーザーまたはグループを選択し、[Delete] (削除) を選択して確認します。削除したユーザーまたはグループが、[Studio users] (Studio ユーザー) リストに表示されなくなります。

AWS CLI を使用して EMR Studio からユーザーまたはグループを削除するには

次の `delete-studio-session-mapping` 引数に独自の値を挿入します。`delete-studio-session-mapping` コマンドの詳細については、「AWS CLI Command Reference」を参照してください。

```
aws emr delete-studio-session-mapping \
 --studio-id <example-studio-id> \
 --identity-type <USER-or-GROUP> \
 --identity-name <name-of-user-or-group-to-delete> \
```

Amazon EMR Studio を管理する

このセクションでは、EMR Studio リソースのモニタリング、更新、または削除に役立つ手順を示します。ユーザーの割り当てやユーザーアクセス許可の更新については、「EMR Studio ユーザーの割り当てと管理」を参照してください。

Studio の詳細の表示

New console

新しいコンソールを使用して EMR Studio の詳細を表示するには

1. https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションの [EMR Studio] で、[Studios] を選択します。
3. [Studios] (Studio) リストから Studio を選択し、Studio の詳細ページを開きます。Studio の詳細ページには、Studio の [Description] (説明)、[VPC]、[Subnets] (サブネット) などの [Studio setting] (Studio 設定) 情報が含まれています。

Old console

古いコンソールを使用して EMR Studio の詳細を表示するには

1. https://console.aws.amazon.com/elasticmapreduce/home で Amazon EMR コンソールを開きます。
2. 左のナビゲーションから [EMR Studio] を選択します。
3. [Studios] (Studio) リストから Studio を選択し、Studio の詳細ページを開きます。Studio の詳細ページには、Studio の [Description] (説明)、[VPC]、[Subnets] (サブネット) などの [Studio setting] (Studio 設定) 情報が含まれています。

CLI

AWS CLI を使用して Studio ID で EMR Studio の詳細を取得するには

`describe-studio` AWS CLI コマンドを使用して、特定の EMR Studio に関する詳細情報を取得します。詳細については、「AWS CLI Command Reference」を参照してください。

```
aws emr describe-studio \
 --studio-id <id-of-studio-to-describe> \
```

AWS CLI を使用して EMR Studio のリストを取得するには

次の `list-studios` AWS CLI コマンドを使用します。詳細については、「AWS CLI Command Reference」を参照してください。

```
aws emr list-studios
```

以下に、`list-studios` コマンドの JSON 形式の戻り値の例を示します。

```
{
    "Studios": [
        {
            "AuthMode": "IAM",
            "VpcId": "vpc-b21XXXXX",
            "Name": "example-studio-name",
            "Url": "https://es-7HWP74SNGDXXXXXXXXXXXXXX.emrstudio-prod.us-
east-1.amazonaws.com",
            "CreationTime": 1605672582.781,
            "StudioId": "es-7HWP74SNGDXXXXXXXXXXXXXX",
            "Description": "example studio description"
        }
    ]
}
```

Amazon EMR Studio アクションをモニタリングする

EMR Studio と API アクティビティを表示する

EMR Studio は、EMR Studio でユーザー、IAM ロール、または別の AWS サービスが実行したアクションを記録するためのサービスである AWS CloudTrail と統合されています。 CloudTrail は、EMR Studio の API コールをイベントとしてキャプチャします。イベントは、https://console.aws.amazon.com/cloudtrail/ で CloudTrailコンソールを使用して表示できます。

EMR Studio イベントは、リクエストを行った Studio または IAM ユーザーやリクエストの種類などの情報を提供します。

> **Note**
>
> ノートブックジョブの実行などのクラスター上のアクションについては AWS CloudTrail で記録は行われません。

Amazon S3 バケットに EMR Studio CloudTrail イベントを継続的に配信するための証跡を作成することもできます。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。

CloudTrail イベントの例: ユーザーが API を DescribeStudio呼び出す

以下は、ユーザー が DescribeStudio API をadmin呼び出したときに作成されるAWS CloudTrailイベントの例です。 は、ユーザー名を として CloudTrail 記録しますadmin。

> **Note**
>
> Studio の詳細を保護するために、 の EMR Studio API イベントは の値 DescribeStudio を除外しますresponseElements。

```
{
   "eventVersion":"1.08",
   "userIdentity":{
      "type":"IAMUser",
      "principalId":"AIDXXXXXXXXXXXXXXXXX",
      "arn":"arn:aws:iam::653XXXXXXXX:user/admin",
      "accountId":"653XXXXXXXX",
      "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
      "userName":"admin"
   },
   "eventTime":"2021-01-07T19:13:58Z",
   "eventSource":"elasticmapreduce.amazonaws.com",
   "eventName":"DescribeStudio",
   "awsRegion":"us-east-1",
   "sourceIPAddress":"72.XX.XXX.XX",
   "userAgent":"aws-cli/1.18.188 Python/3.8.5 Darwin/18.7.0 botocore/1.19.28",
   "requestParameters":{
      "studioId":"es-9O5XXXXXXXXXXXXXXXXXXX"
   },
   "responseElements":null,
   "requestID":"0fxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
   "eventID":"b0xxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
   "readOnly":true,
   "eventType":"AwsApiCall",
   "managementEvent":true,
   "eventCategory":"Management",
   "recipientAccountId":"653XXXXXXXX"
}
```

Spark ユーザーおよびジョブアクティビティの表示

Amazon EMR Studio ユーザーによる Spark ジョブアクティビティを表示するには、クラスターでユーザー偽装を設定できます。ユーザー偽装により、Workspace から送信された各 Spark ジョブは、コードを実行した Studio ユーザーに関連付けられます。

ユーザー偽装が有効になっている場合、Amazon EMR は、Workspace でコードを実行するユーザーごとに、クラスターのプライマリノード上に HDFS ユーザーディレクトリを作成します。例えば、ユーザー `studio-user-1@example.com` がコードを実行した場合、プライマリノードに接続して、`hadoop fs -ls /user` に `studio-user-1@example.com` のディレクトリがあることを確認できます。

Spark ユーザー偽装を設定するには、次の設定分類で次のプロパティを設定します。

- `core-site`
- `livy-conf`

```
[
    {
        "Classification": "core-site",
        "Properties": {
          "hadoop.proxyuser.livy.groups": "*",
          "hadoop.proxyuser.livy.hosts": "*"
        }
    },
    {
        "Classification": "livy-conf",
        "Properties": {
          "livy.impersonation.enabled": "true"
        }
    }
]
```

履歴サーバページを表示するには、「EMR Studio でアプリケーションとジョブをデバッグする」を参照してください。SSH を使用してクラスターのプライマリノードに接続して、アプリケーションウェブインターフェイスを表示することもできます。詳細については、「Amazon EMR クラスターでホストされているウェブインターフェイスを表示する」を参照してください。

Amazon EMR Studio を更新する

EMR Studio を作成したら、AWS CLI を使用して以下の属性を更新できます。

- 名前
- 説明
- S3 のデフォルトの場所
- サブネット

AWS CLI を使用して EMR Studio を更新するには

`update-studio` AWS CLI コマンドを使用して、EMR Studio を更新します。詳細については、「AWS CLI Command Reference」を参照してください。

> **Note**
>
> Studio を最大 5 つのサブネットに関連付けることができます。これらのサブネットは Studio と同じ VPC に属している必要があります。`update-studio` コマンドに送信するサブネット ID のリストには、新しいサブネット ID を含めることができますが、すでに Studio に関連付けられているすべてのサブネット ID も含める必要があります。Studio からサブネットを削除することはできません。

```
aws emr update-studio \
 --studio-id <example-studio-id-to-update> \
 --name <example-new-studio-name> \
 --subnet-ids <old-subnet-id-1 old-subnet-id-2 old-subnet-id-3 new-subnet-id> \
```

変更を確認するには、`describe-studio` AWS CLI コマンドを使用し、Studio ID を指定します。詳細については、「AWS CLI Command Reference」を参照してください。

```
aws emr describe-studio \
 --studio-id <id-of-updated-studio> \
```

Amazon EMR Studio と Workspace を削除する

Studio を削除すると、EMR Studio によって、Studio に関連付けられている IAM Identity Center ユーザーおよびグループの割り当てがすべて削除されます。

> **Note**
>
> Studio を削除しても、Amazon EMR によって、その Studio に関連付けられている Workspace が削除されることはありません。Studio で Workspace を個別に削除する必要があります。

Workspace の削除

Console

各 EMR Studio Workspace は EMR ノートブックインスタンスであるため、Amazon EMR マネジメントコンソールを使用して Workspace を削除できます。Studio を削除する前または後に Amazon EMR コンソールを使用して Workspace を削除できます。

Amazon EMR コンソールを使用して Workspace を削除するには

1. 新しい Amazon EMR コンソールに移動し、サイドナビゲーションから [古いコンソールに切り替え] を選択します。古いコンソールに切り替えたときの動作の詳細については、「Using the old console」を参照してください。
2. [Notebooks] (ノートブック) を選択します。
3. 削除する Workspace を選択します。
4. [Delete] (削除) を選択し、[Delete] (削除) を再度選択して確認します。
5. 「Amazon Simple Storage Service コンソールユーザーガイド」の「オブジェクトの削除」の指示に従って、削除した Workspace に関連付けられているノートブックファイルを Amazon S3 から削除します。

EMR Studio UI

From the Workspace UI

EMR Studio から Workspace とその関連バックアップファイルを削除する

1. Studio アクセス URL を使用して EMR Studio にログインし、左のナビゲーションから [Workspaces] (Workspace) を選択します。
2. リストから Workspace を見つけ、その名前の横にあるチェックボックスを選択します。複数の Workspace を選択して、同時に削除することができます。

3. [Workspaces] (Workspace) リストの右上にある [Delete] (削除) を選択して、選択した Workspace を削除することを確認します。[Delete] を選択して確定します。
4. Amazon S3 から削除された Workspace に関連付けられたノートブックファイルを削除する場合は、「Amazon Simple Storage Service コンソールユーザーガイド」の「オブジェクトの削除」の手順に従ってください。Studio を作成していない場合は、Studio 管理者に問い合わせて、削除した Workspace の Amazon S3 バックアップの場所を確認してください。

From the Workspaces list

[Workspace] リストから Workspace とその関連バックアップファイルを削除する

1. コンソールの [Workspace] リストに移動します。
2. リストから削除する Workspace を選択してから、[アクション] を選択します。
3. [削除] を選択します。
4. Amazon S3 から削除された Workspace に関連付けられたノートブックファイルを削除する場合は、「Amazon Simple Storage Service コンソールユーザーガイド」の「オブジェクトの削除」の手順に従ってください。Studio を作成していない場合は、Studio 管理者に問い合わせて、削除した Workspace の Amazon S3 バックアップの場所を確認してください。

EMR Studio の削除

New console

新しいコンソールを使用して EMR Studio を削除するには

1. https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションの [EMR Studio] で、[Studios] を選択します。
3. Studio 名の左側にあるトグルを使用して、[Studios] リストから Studio を選択します。[削除] を選択します。

Old console

古いコンソールを使用して EMR Studio を削除するには

1. https://console.aws.amazon.com/elasticmapreduce/home で Amazon EMR コンソールを開きます。
2. 左のナビゲーションから [EMR Studio] を選択します。
3. [Studios] (Studio) リストから Studio を選択し、[Delete] (削除) を選択します。

CLI

AWS CLI を使用して EMR Studio を削除するには

`delete-studio` AWS CLI コマンドを使用して、EMR Studio を削除します。詳細については、『AWS CLI コマンドリファレンス』を参照してください。

```
aws emr delete-studio --studio-id <id-of-studio-to-delete>
```

EMR Studio ワークスペースのノートブックとファイルの暗号化

EMR Studio では、ノートブックを整理して実行するためのさまざまなワークスペースを作成して設定できます。これらのワークスペースは、指定した Amazon S3 バケットにノートブックと関連ファイルを保存します。デフォルトでは、これらのファイルは Amazon S3-managedキー (SSE-S3) で暗号化され、基本レベルの暗号化としてサーバー側の暗号化が使用されます。カスタマーマネージド KMS キー (SSE-KMS) を使用してファイルを暗号化することもできます。そのためには、Amazon EMR マネジメントコンソールを使用するか、EMR Studio の作成時に AWS CLIと AWS SDK を使用します。

EMR Studio ワークスペースストレージの暗号化は、EMR Studio が利用可能なすべてのリージョンで使用できます。

前提条件

EMR Studio ワークスペースのノートブックとファイルを暗号化する前に、 AWS Key Management Serviceを使用して EMR Studio と同じ AWS アカウントとリージョンに対称カスタマーマネージャーキー (CMK) を作成する必要があります。

のリソースポリシーには、EMR Studio のサービスロールに必要なアクセス許可AWS KMSが必要です。以下は、EMR Studio Workspace ストレージ暗号化の最小アクセス許可を付与するサンプル IAM ポリシーです。

```
{
    "Sid": "AllowEMRStudioServiceRoleAccess",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::<ACCOUNT_ID>:role/<ROLE_NAME>"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey",
        "kms:ReEncryptFrom",
        "kms:ReEncryptTo",
        "kms:DescribeKey"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "kms:CallerAccount": "<ACCOUNT_ID>",
            "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::<S3_BUCKET_NAME>",
            "kms:ViaService": "s3.<AWS_REGION>.amazonaws.com"
        }
    }
}
```

EMR Studio サービスロールには、 AWS KMSキーを使用するためのアクセス許可も必要です。以下は、EMR Studio Workspace ストレージ暗号化の最小アクセス許可を付与するサンプル IAM ポリシーです。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEMRStudioWorkspaceStorageEncryptionAccess",
            "Effect": "Allow",
            "Action": [
               "kms:Decrypt",
               "kms:GenerateDataKey",
               "kms:ReEncryptFrom",
               "kms:ReEncryptTo",
```

```
                "kms:DescribeKey"
            ],
            "Resource": ["arn:aws:kms:<REGION>:<ACCOUNT_ID>:key/<KEY_IDENTIFIER>"]
        }
    ]
}
```

セットアップ

ワークスペースストレージの暗号化を使用する新しい EMR Studio を作成するには、次の手順に従います。

1. Amazon EMR コンソール (https://console.aws.amazon.com/elasticmapreduce/) を開きます。
2. Studio を選択し、Studio の作成 を選択します。
3. ストレージ の S3 の場所には、Amazon S3 パスを入力または選択します。これは、Amazon EMR がワークスペースノートブックとファイルを保存する Amazon S3 の場所です。
4. サービスロール で、IAM ロールを入力または選択します。これは、Amazon EMR が引き受ける IAM ロールです。
5. 独自のAWS KMSキー で Workspace ファイルを暗号化を選択します。
6. Amazon S3 のワークスペースノートブックとファイルの暗号化に使用する AWS KMSキーを入力または選択します。
7. Studio の作成 または Studio を作成して Workspace を起動 を選択します。
8. 独自のAWS KMSキー で Workspace ファイルを暗号化を選択します。
9. Amazon S3 のワークスペースノートブックとファイルの暗号化AWS KMSに使用する を入力または選択します。
10. [変更を保存] を選択します。

次の手順は、EMR Studio を更新し、ワークスペースストレージの暗号化を設定する方法を示しています。

1. Amazon EMR コンソール (https://console.aws.amazon.com/elasticmapreduce/) を開きます。
2. リストから既存の EMR Studio を選択し、編集を選択します。
3. 独自のAWS KMSキー で Workspace ファイルを暗号化を選択します。
4. Amazon S3 のワークスペースノートブックとファイルの暗号化AWS KMSに使用する を入力または選択します。
5. [変更を保存] を選択します。

EMR Studio ネットワークトラフィックを制御するセキュリティグループを定義する

EMR Studio セキュリティグループについて

Amazon EMR Studio は、次の 2 つのセキュリティグループを使用して、Studio 内の Workspace と Amazon EC2 で実行されているアタッチされた Amazon EMR クラスター間のネットワークトラフィックを制御します。

- エンジンセキュリティグループ。これは、ポート 18888 を使用して、Amazon EC2 で実行されているアタッチされた Amazon EMR クラスターと通信します。
- Workspace セキュリティグループ。これは、Studio の Workspace に関連付けられています。このセキュリティグループには、Workspace がトラフィックをインターネットにルーティングできるようにするアウトバウンド HTTPS ルールが含まれており、Git リポジトリを Workspace にリンクできるようにするために、ポート 443 でインターネットへのアウトバウンドトラフィックを許可する必要があります。

EMR Studio は、Workspace にアタッチされた EMR クラスターに関連付けられているセキュリティグループに加えて、これらのセキュリティグループを使用します。

AWS CLI を使用して Studio を作成する際にこれらのセキュリティグループを作成する必要があります。

> **Note**
>
> ご使用の環境に合わせたルールで EMR Studio のセキュリティグループをカスタマイズできますが、このページに記載されているルールを含める必要があります。Workspace セキュリティグループはインバウンドトラフィックを許可できず、エンジンセキュリティグループは Workspace セキュリティグループからのインバウンドトラフィックを許可する必要があります。

デフォルトの EMR Studio セキュリティグループの使用

Amazon EMR コンソールを使用する場合、次のデフォルトのセキュリティグループを選択できます。デフォルトのセキュリティグループはユーザーに代わって EMR Studio によって作成され、EMR Studio のワークスペースに必要な最小限のインバウンドおよびアウトバウンドのルールが含まれています。

- `DefaultEngineSecurityGroup`
- `DefaultWorkspaceSecurityGroupGit` または `DefaultWorkspaceSecurityGroupWithoutGit`

前提条件

EMR Studio のセキュリティグループを作成するには、Studio 用の Amazon Virtual Private Cloud (VPC) が必要です。この VPC は、セキュリティグループを作成するときに選択します。これは、Studio の作成時に指定した VPC と同じである必要があります。EMR Studio で Amazon EMR on EKS を使用する場合は、Amazon EKS クラスターワーカーノード用の VPC を選択します。

手順

「Linux インスタンス用 Amazon EC2 ユーザーガイド」の「セキュリティグループの作成」の指示に従って、VPC 内にエンジンセキュリティグループと Workspace セキュリティグループを作成します。セキュリティグループには、次の表に要約されているルールを含める必要があります。

EMR Studio のセキュリティグループを作成するときは、両方の ID を書き留めます。Studio を作成するときに、各セキュリティグループを ID で指定します。

エンジンセキュリティグループ

EMR Studio は、ポート 18888 を使用して、アタッチされたクラスターと通信します。

インバウンドルール

タイプ	プロトコル	ポート	デスティネーション	説明
TCP	TCP	18888	EMR Studio Workspace セキュリティグループ。	EMR Studio の Workspace セキュリティグループですべてのリソースからのトラフィックを許可します。

Workspace セキュリティグループ

このセキュリティグループは、EMR Studio の Workspace に関連付けられています。

アウトバウンドルール

タイプ	プロトコル	ポート	デスティネーション	説明
TCP	TCP	18888	EMR Studio エンジンセキュリティグループ。	EMR Studio のエンジンセキュリティグループですべてのリソースへのトラフィックを許可します。
HTTPS	TCP	443	0.0.0.0/0	パブリックにホストされている Git リポジトリを Workspace にリンクするために、インターネットへのトラフィックを許可します。

Amazon EMR Studio 用の AWS CloudFormation テンプレートを作成する

EMR Studio クラスターテンプレートについて

EMR Studio ユーザーが Workspace で新しい Amazon EMR クラスターを起動するのに役立つAWS CloudFormationテンプレートを作成できます。 CloudFormation テンプレートは JSON または YAML 形式のテキストファイルです。テンプレートでは、AWSリソースのスタックを記述し、それらのリソースをプロビジョニング CloudFormation する方法を に伝えます。EMR Studio では、Amazon EMR クラスターを記述する 1 つ以上のテンプレートを作成できます。

テンプレートは AWS Service Catalog で整理します。AWS Service Catalog により、AWS で製品と呼ばれる、一般的にデプロイされる IT サービスを作成し、管理できます。EMR Studio ユーザーと共有するポートフォリオで製品としてテンプレートを収集します。クラスターテンプレートを作成した後に、Studio ユーザーは、いずれかのテンプレートを使用して Workspace の新しいクラスターを起動できます。ユーザーは、テンプレートから新しいクラスターを作成するアクセス許可が必要です。ユーザーアクセス許可は、EMR Studio アクセス許可ポリシーで設定できます。

CloudFormation テンプレートの詳細については、「 AWS CloudFormationユーザーガイド」の「テンプレート」を参照してください。AWS Service Catalogの詳細については、とはAWS Service Catalogを参照してください。

次の動画では、EMR Studio 用に AWS Service Catalog でクラスターテンプレートを設定する方法が示されています。詳細については、ブログ投稿「Build a self-service environment for each line of business using Amazon EMR and Service Catalog」でも確認できます。

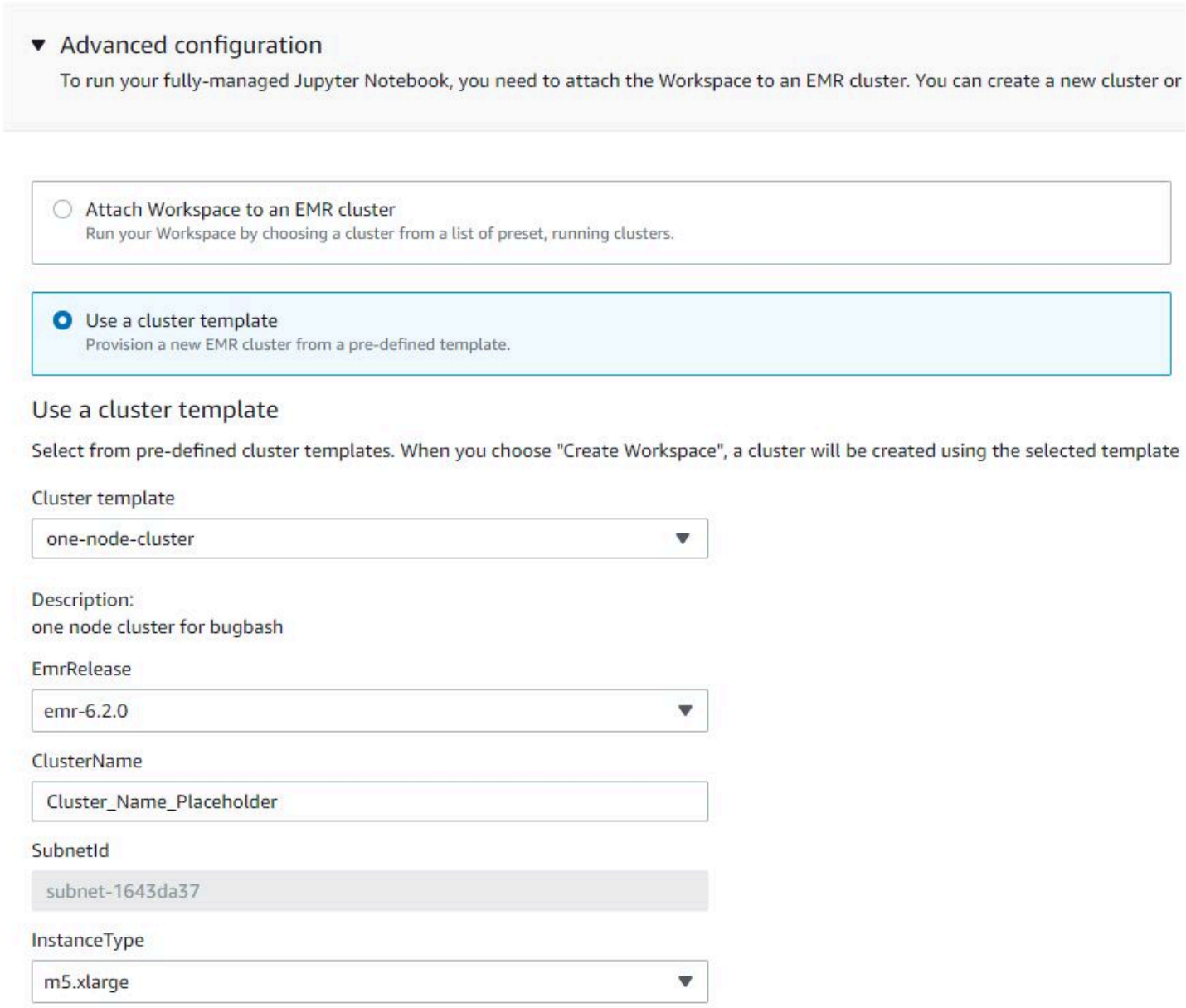
オプションのテンプレートパラメータ

テンプレートの `Parameters` セクションに追加のオプションを含めることができます。Parameters により、Studio ユーザーがクラスターのカスタム値を入力または選択できます。例えば、ユーザーが特定の Amazon EMR リリースを選択できるようにするパラメータを追加できます。詳細については、「AWS CloudFormation ユーザーガイド」の「パラメータ」を参照してください。

次のサンプル `Parameters` セクションでは、`ClusterName`、`EmrRelease` バージョン、`ClusterInstanceType` などの追加の入力パラメータを定義しています。

```
Parameters:
  ClusterName:
    Type: "String"
    Default: "Cluster_Name_Placeholder"
  EmrRelease:
    Type: "String"
    Default: "emr-6.2.0"
    AllowedValues:
    - "emr-6.2.0"
    - "emr-5.32.0"
  ClusterInstanceType:
    Type: "String"
    Default: "m5.xlarge"
    AllowedValues:
    - "m5.xlarge"
    - "m5.2xlarge"
```

パラメータを追加すると、クラスタテンプレートを選択した後に、Studio ユーザーに追加のフォームオプションが表示されます。次の図は、EmrReleaseバージョン、、ClusterNameおよび の追加フォームオプションを示していますInstanceType。



前提条件

クラスターテンプレートを作成する前に、Service Catalog 管理者コンソールビューにアクセスするための IAM アクセス許可があることを確認してください。また、Service Catalog 管理タスク実行するための IAM アクセス許可も必要です。詳細については、「Grant permissions to Service Catalog administrators」を参照してください。

手順

Service Catalog を使用して EMR クラスターテンプレートを作成するには

1. 1 つ以上の CloudFormation テンプレートを作成します。テンプレートを保存する場所はユーザーが決定します。テンプレートはフォーマット設定されたテキストファイルであるため、Amazon S3 にアップロードするか、ローカルファイルシステムに保存することができます。 CloudFormation テンプレートの詳細については、「 AWS CloudFormationユーザーガイド」の「テンプレート」を参照してください。

   次のルールを使用してテンプレートに名前を付けるか、パターン `[a-zA-Z0-9][a-zA-Z0-9._-]*` に照らして名前をチェックします。

   - テンプレート名の先頭は、英字または数字である必要があります。
   - テンプレート名には、英字、数字、ピリオド (.)、アンダースコア (_)、ハイフン (-) のみを使用できます。

   作成する各クラスターテンプレートには、以下のオプションが含まれている必要があります。

   入力パラメータ

   - ClusterName – プロビジョニング後にユーザーがクラスターを識別しやすくするクラスターの名前。

   出力

   - `ClusterId` - 新しくプロビジョニングされた EMR クラスターの ID。

   以下に、2 つのノードがあるクラスターの YAML 形式のサンプル AWS CloudFormation テンプレートを示します。サンプルテンプレートには、必要なテンプレートオプションが含まれており、`EmrRelease` および `ClusterInstanceType` の追加の入力パラメータが定義されています。

```
awsTemplateFormatVersion: 2010-09-09

Parameters:
  ClusterName:
    Type: "String"
```

```
    Default: "Example_Two_Node_Cluster"
  EmrRelease:
    Type: "String"
    Default: "emr-6.2.0"
    AllowedValues:
    - "emr-6.2.0"
    - "emr-5.32.0"
  ClusterInstanceType:
    Type: "String"
    Default: "m5.xlarge"
    AllowedValues:
    - "m5.xlarge"
    - "m5.2xlarge"

Resources:
  EmrCluster:
    Type: AWS::EMR::Cluster
    Properties:
      Applications:
      - Name: Spark
      - Name: Livy
      - Name: JupyterEnterpriseGateway
      - Name: Hive
      EbsRootVolumeSize: '10'
      Name: !Ref ClusterName
      JobFlowRole: EMR_EC2_DefaultRole
      ServiceRole: EMR_DefaultRole_V2
      ReleaseLabel: !Ref EmrRelease
      VisibleToAllUsers: true
      LogUri:
        Fn::Sub: 's3://aws-logs-${AWS::AccountId}-${AWS::Region}/elasticmapreduce/'
      Instances:
        TerminationProtected: false
        Ec2SubnetId: 'subnet-ab12345c'
        MasterInstanceGroup:
          InstanceCount: 1
          InstanceType: !Ref ClusterInstanceType
        CoreInstanceGroup:
          InstanceCount: 1
          InstanceType: !Ref ClusterInstanceType
          Market: ON_DEMAND
          Name: Core

Outputs:
```

```
  ClusterId:
    Value:
      Ref: EmrCluster
    Description: The ID of the  EMR cluster
```

2. Studio と同じ AWS アカウントでクラスターテンプレートのポートフォリオを作成します。

   a. AWS Service Catalog コンソール (https://console.aws.amazon.com/servicecatalog/) を開きます。

   b. 左のナビゲーションメニューから [Portfolios] (ポートフォリオ) を選択します。

   c. [Create portfolio] (ポートフォリオの作成) ページで、必要情報を入力します。

   d. [Create] (作成) を選択します。AWS Service Catalog によってポートフォリオが作成され、ポートフォリオの詳細が表示されます。

3. 次のステップを使用して、クラスターテンプレートを AWS Service Catalog 製品として追加します。

   a. AWS Service Catalog マネジメントコンソールで [Administration] (管理) の [Products] (製品) ページに移動します。

   b. [Upload new product] (新しい製品のアップロード) を選択します。

   c. [Product name] (製品名) および [Owner] (所有者) を入力します。

   d. [Version details] (バージョンの詳細) でテンプレートファイルを指定します。

   e. [Review] (確認) を選択して製品設定を確認してから、[Create product] (製品の作成) を選択します。

4. ポートフォリオに製品を追加するには、以下のステップを完了します。

   a. AWS Service Catalog マネジメントコンソールで [Products] (製品) ページに移動します。

   b. 製品を選択し、[Actions] (アクション)、[Add product to portfolio]] (ポートフォリオへの製品の追加) の順に選択します。

   c. ポートフォリオを選択し、[Add product to portfolio] (ポートフォリオへの製品の追加) を選択します。

5. 製品の起動制約を作成します。起動制約は、製品を起動するためのユーザーアクセス許可を指定する IAM ロールです。起動制約は調整できますが、 CloudFormation、Amazon EMR、およびを使用するアクセス許可を許可する必要がありますAWS Service Catalog。詳細な情報と手順については、「Service Catalog launch constraints」を参照してください。

6. 起動制約をポートフォリオ内の各製品に適用します。起動制約は、各製品に個別に適用する必要があります。

a. AWS Service Catalog マネジメントコンソールの [Portfolios] (ポートフォリオ) ページからポートフォリオを選択します。

b. [制約] タブを選択して、[制約の作成] を選択します。

c. 製品を選択し、[Constraint type] (制約タイプ) で [Launch] (起動) を選択します。[続行] を選択します。

d. [Launch constraint] (起動制約) セクションで起動制約ロールを選択し、[Create] (作成) を選択します。

7. ポートフォリオへのアクセス権を付与します。

a. AWS Service Catalog マネジメントコンソールの [Portfolios] (ポートフォリオ) ページからポートフォリオを選択します。

b. [Groups, roles, and users] (グループ、ロール、およびユーザー) タブをを展開し、[Add groups, roles, users] (グループ、ロール、またはユーザーの追加) を選択します。

c. [Roles] (ロール) タブで EMR Studio IAM ロールを検索し、ロールを選択し、[Add access] (アクセス権の追加) を選択します。

使用するもの	アクセス権を付与する対象
IAM 認証	ネイティブユーザー
IAM フェデレーション	フェデレーションの IAM ロール
IAM Identity Center フェデレーション	EMR Studio ユーザーロール

Git ベースのリポジトリのアクセス権とアクセス許可を設定する

EMR Studio では、以下の Git ベースのサービスがサポートされています。

- AWS CodeCommit
- GitHub
- Bitbucket
- GitLab

EMR Studio ユーザーが Git リポジトリを Workspace に関連付けられるようにするには、次のアクセス権およびアクセス許可の要件を設定します。「EMR Studio 用にプライベートにホストされた Git リポジトリを設定する」の手順に従って、プライベートネットワークでホストしている Git ベースのリポジトリを設定することもできます。

クラスターインターネットアクセス権

Amazon EC2 で実行されている Amazon EMR クラスターと、Studio Workspace にアタッチされた Amazon EMR on EKS クラスターの両方が、NAT ゲートウェイを使用するプライベートサブネット内にあるか、仮想プライベートゲートウェイを介してインターネットにアクセスできる必要があります。詳細については、「Amazon VPC オプション」を参照してください。

EMR Studio で使用するセキュリティグループには、アタッチされた EMR クラスターからインターネットへのトラフィックの Workspace によるルーティングを許可するアウトバウンドルールも含まれている必要があります。詳細については、「EMR Studio ネットワークトラフィックを制御するセキュリティグループを定義する」を参照してください。

> **Important**
>
> ネットワークインターフェイスがパブリックサブネット内にある場合、インターネットゲートウェイ (IGW) を介してインターネットと通信できなくなります。

AWS Secrets Manager のアクセス権限

EMR Studio ユーザーが AWS Secrets Manager に保存されているシークレットを使用して Git リポジトリにアクセスできるようにするには、`secretsmanager:GetSecretValue` オペレーションを許可するアクセス許可ポリシーを EMR Studio のサービスロールに追加します。

Git ベースのリポジトリを Workspace にリンクする方法については、「GIT ベースのリポジトリを EMR Studio Workspace にリンクする」を参照してください。

EMR Studio 用にプライベートにホストされた Git リポジトリを設定する

次の手順を使用して、Amazon EMR Studio 用にプライベートにホストされたリポジトリを設定します。DNS サーバーおよび Git サーバーに関する情報が含まれた設定ファイルを用意します。EMR Studio は、この情報を使用して、トラフィックをセルフマネージド型リポジトリにルーティングできる Workspace を設定します。

> **Note**
>
> DnsServerIpV4 を設定すると、EMR Studio は DNS サーバーを使用して、GitServerDnsName と elasticmapreduce.us-east-1.amazonaws.com などの Amazon EMR エンドポイントの両方を解決します。Amazon EMR のエンドポイントを設定するには、Studio で使用している VPC を介してエンドポイントに接続します。これにより、Amazon EMR エンドポイントはプライベート IP に解決されます。詳細については、「インターフェイス VPC エンドポイントを使用して Amazon EMR に接続する」を参照してください。

前提条件

EMR Studio 用にプライベートにホストされた Git リポジトリを設定する前に、EMR Studio が Studio 内の Workspace とノートブックファイルをバックアップできる Amazon S3 ストレージの場所が必要です。Studio の作成時に指定した S3 バケットと同じ S3 バケットを使用します。

EMR Studio 用にプライベートにホストされた 1 つ以上の Git リポジトリを設定するには

1. 次のテンプレートを使用して、設定ファイルを作成します。設定で指定する Git サーバーごとに次の値を含めます。

   - **DnsServerIpV4** - DNS サーバーの IPv4 アドレス。DnsServerIpV4 と GitServerIpV4List の両方に値を指定した場合、DnsServerIpV4 の値が優先され、EMR Studio は DnsServerIpV4 を使用して GitServerDnsName を解決します。

     > **Note**
     >
     > プライベートにホストされた Git リポジトリを使用するには、DNS サーバーで EMR Studio からのインバウンドアクセスを許可する必要があります。DNS サーバーを他の不正アクセスから保護することをお勧めします。

   - **GitServerDnsName** - Git サーバーの DNS 名。例えば、"git.example.com" です。
   - **GitServerIpV4List** - Git サーバーに属する IPv4 アドレスのリスト。

```
[
    {
        "Type": "PrivatelyHostedGitConfig",
```

```
        "Value": [
            {
                "DnsServerIpV4": "<10.24.34.xxx>",
                "GitServerDnsName": "<enterprise.git.com>",
                "GitServerIpV4List": [
                    "<xxx.xxx.xxx.xxx>",
                    "<xxx.xxx.xxx.xxx>"
                ]
            },
            {
                "DnsServerIpV4": "<10.24.34.xxx>",
                "GitServerDnsName": "<git.example.com>",
                "GitServerIpV4List": [
                    "<xxx.xxx.xxx.xxx>",
                    "<xxx.xxx.xxx.xxx>"
                ]
            }
        ]
    }
]
```

2. configuration.json という名前で設定ファイルを保存します。
3. 設定ファイルを Amazon S3 ストレージの場所にある life-cycle-configuration というフォルダーにアップロードします。例えば、デフォルトの S3 の場所が s3://*DOC-EXAMPLE-BUCKET*/studios の場合、設定ファイルは s3://*DOC-EXAMPLE-BUCKET*/studios/**life-cycle-configuration/configuration.json** に配置します。

> **Important**
>
> life-cycle-configuration フォルダへのアクセスを Studio 管理者と EMR Studio サービスロールに制限し、configuration.json を不正アクセスから保護することをお勧めします。手順については、「ユーザーポリシーを使用したバケットへのアクセスの制御」または「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。

アップロードの手順については、「Amazon Simple Storage Service ユーザーガイド」の「フォルダの作成」と「オブジェクトのアップロード」を参照してください。設定を既存の Workspace に適用するには、設定ファイルを Amazon S3 にアップロードした後、Workspace を閉じて再起動します。

EMR Studio で Spark ジョブを最適化する

EMR Studio を使用して Spark ジョブを実行する場合、Amazon EMR クラスターリソースを最適化できるようにするための手順がいくつかあります。

Livy セッションを延長する

Apache Livy を Amazon EMR クラスターで Spark とともに使用する場合は、次のいずれかを実行して Livy セッションのタイムアウトを増やすことをお勧めします。

- Amazon EMR クラスターを作成する際に、[Enter Configuration] (設定の入力) フィールドでこの設定分類を設定します。

```
[
    {
        "Classification": "livy-conf",
        "Properties": {
          "livy.server.session.timeout": "8h"
        }
    }
]
```

- すでに実行されている EMR クラスターの場合は、ssh を使用してクラスターに接続し、/etc/livy/conf/livy.conf で livy-conf 設定分類を設定します。

```
[
    {
        "Classification": "livy-conf",
        "Properties": {
          "livy.server.session.timeout": "8h"
        }
    }
]
```

  設定を変更した後、Livy を再起動する必要がある場合があります。

- Livy セッションが一切タイムアウトしないようにする場合は、/etc/livy/conf/livy.conf でプロパティ livy.server.session.timeout-check を false に設定します。

Spark をクラスターモードで実行する

クラスターモードでは、Spark ドライバーはプライマリノードではなくコアノードで実行され、プライマリノードのリソース使用率が向上します。

Spark アプリケーションをデフォルトのクライアントモードではなくクラスターモードで実行するには、新しい Amazon EMR クラスターで Spark ステップを設定するときに、[Deploy mode] (デプロイモード) を設定する際に [Cluster] (クラスター) モードを選択します。詳細については、Apache Spark ドキュメントで「クラスターモードの概要」を参照してください。

Spark ドライバーのメモリを増やす

Spark ドライバーのメモリを増やすには、次の例のように、EMR ノートブックで `%%configure` マジックコマンドを使用して Spark セッションを設定します。

```
%%configure -f
{"driverMemory": "6000M"}
```

Amazon EMR Studio を使用する

このセクションでは、Amazon EMR Studio の設定と操作に役立つトピックを案内します。

次のビデオでは、新しい Workspace を作成する方法、クラスターテンプレートを使用して新しい Amazon EMR クラスターを起動する方法などの実用的な情報について説明しています。このビデオでは、サンプルノートブックについても紹介しています。

このセクションには、EMR Studio での作業に役立つ次のトピックが含まれています。

- Workspace の基本の説明
- Workspace コラボレーションを設定する
- ランタイムロールを使用して EMR Studio Workspace を実行する
- Workspace ノートブックをプログラムで実行する
- SQL Explorer でデータをブラウズする
- EMR Studio Workspace にコンピューティングをアタッチする
- GIT ベースのリポジトリを EMR Studio Workspace にリンクする
- EMR Studio での Amazon Athena SQL エディターの使用
- Amazon と EMR Studio Workspace CodeWhisperer の統合

- EMR Studio でアプリケーションとジョブをデバッグする
- EMR Studio Workspace にカーネルとライブラリをインストールする
- magic コマンドでカーネルを強化する
- Spark カーネルで多言語ノートブックを使用する

Workspace の基本の説明

EMR Studio を使用すると、さまざまな Workspace を作成して設定し、ノートブックを整理して実行できます。このセクションでは、Workspace の作成と操作について説明します。概念の概要については、「Amazon EMR Studio の仕組み」ページの「ワークスペース」を参照してください。

このセクションでは、EMR Studio Workspace の使用に役立つ次のトピックについて説明します。

- EMR Studio Workspace の作成
- Workspace を起動する
- Workspace ユーザーインターフェイスを理解する
- ノートブックサンプルの探索
- Workspace コンテンツの保存
- Workspace およびノートブックファイルを削除する
- Workspace ステータスを理解する
- Workspace の接続の問題を解決する

EMR Studio Workspace の作成

EMR Studio インターフェイスを使用してノートブックコードを実行するために EMR Studio Workspace を作成できます。

EMR Studio で Workspace を作成するには

1. EMR Studio にログインします。
2. [Workspace の作成] を選択します。
3. [Workspace name] (Workspace 名) と [Description] (説明) を入力します。Workspace に名前を付けると、[Workspaces] (Workspace) ページでその Workspace を識別するのに役立ちます。
4. この Workspace で他の Studio ユーザーとリアルタイムで作業する場合は、Workspace のコラボレーションを有効にします。共同作業者は Workspace の起動後に設定できます。

5. クラスターを Workspace にアタッチする場合は、[高度な設定] セクションを展開します。クラスターは後でアタッチすることもできます。詳細については、「EMR Studio Workspace にコンピューティングをアタッチする」を参照してください。

   > **Note**
   >
   > 新しいクラスターをプロビジョニングするには、管理者からのアクセス許可が必要です。

   Workspace のクラスターオプションの 1 つを選択し、クラスターをアタッチします。Workspace の作成時におけるクラスターのプロビジョニングの詳細については、「新しい EMR クラスターを作成して EMR Studio Workspace にアタッチする」を参照してください。
6. ページの右下の [Workspace の作成] を選択します。

Workspace を作成すると、EMR Studio で [Workspaces] (Workspace) ページが開きます。ページの上部に緑色の成功バナーが表示され、新しく作成された Workspace がリストに表示されます。

デフォルトでは、Workspace は共有され、すべての Studio ユーザーが表示できます。ただし、Workspace を開いて作業できるのは一度に 1 人のユーザーのみです。他のユーザーと同時に作業する場合は、「Workspace コラボレーションを設定する」を参照してください。

Workspace を起動する

ノートブックファイルの操作を開始するには、Workspace を起動してノートブックエディタにアクセスします。Studio の [Workspaces] ページには、アクセスできるすべての Workspace が、[名前]、[ステータス]、[作成時刻]、[最終更新日時] などの詳細と共にリストされます。

> **Note**
>
> 古い Amazon EMR コンソールに EMR ノートブックがある場合、そのノートブックは新しいコンソールで EMR Studio Workspace として表示されます。EMR Notebooks ユーザーが Workspace にアクセスしたり、Workspace を作成したりするには、追加の IAM ロールのアクセス許可が必要です。古いコンソールで最近ノートブックを作成した場合、そのノートブックを新しいコンソールに表示するには Workspace リストを更新する必要がある場合があります。移行の詳細については、「Amazon EMR Notebooks がコンソールで Amazon

EMR Studio Workspace として利用可能に」と「Amazon EMR コンソール」を参照してください。

ノートブックを編集および実行するために Workspace を起動するには

1. Studo の [Workspaces] (Workspace) ページで、Workspace を見つけます。キーワードまたは列の値でリストをフィルタリングできます。
2. Workspace 名を選択して、新しいブラウザタブで Workspace を起動します。[Idle] (アイドル) 状態の Workspace は開くまでに数分かかることがあります。または、Workspace の行を選択し、[Workspace を起動] を選択します。次の起動オプションを選択できます。
   - クイック起動 - デフォルトのオプションを使用して Workspace をすばやく起動します。で Workspace にクラスターをアタッチする場合は、クイック起動を選択します JupyterLab。
   - オプションを使用して起動する - カスタムオプションを使用して Workspace を起動します。Jupyter または のいずれかで起動し JupyterLab、Workspace を EMR クラスターにアタッチして、セキュリティグループを選択できます。

> **Note**
>
> Workspace を開いて作業できるのは一度に 1 人のユーザーのみです。すでに使用中の Workspace を選択すると、開こうとしたときに EMR Studio で通知が表示されます。[Workspaces] (Workspace) ページの [User] (ユーザー) 列には、Workspace で作業しているユーザーが表示されます。

Workspace ユーザーインターフェイスを理解する

EMR Studio Workspace ユーザーインターフェイスは、左側のサイドバーにアイコンで示されたタブがあるJupyterLabインターフェイスに基づいています。アイコンの上にカーソルを置くと、タブの名前を示すツールチップが表示されます。左サイドバーからタブを選択して、次のパネルにアクセスします。

- [File Browser] (ファイルブラウザ) - Workspace 内のファイルとディレクトリ、リンクされた Git リポジトリのファイルとディレクトリが表示されます。

- [Running Kernels and Terminals] (カーネルとターミナルの実行) - Workspace で実行されているすべてのカーネルとターミナルがリストされます。詳細については、公式 JupyterLab ドキュメントの「カーネルとターミナルの管理」を参照してください。
- [Git] - Workspace にアタッチされた Git リポジトリでコマンドを実行するためのグラフィカルユーザーインターフェイスが示されます。このパネルは jupyterlab-git と呼ばれる JupyterLab 拡張機能です。詳細については、「jupyterlab-git」を参照してください。
- EMR クラスター - ノートブックコードを実行するために Workspace にクラスターをアタッチしたり、Workspace からクラスターをデタッチしたりできます。EMR クラスタ設定パネルには、新しいクラスターを Workspace に作成してアタッチできるようにする高度な設定オプションも用意されています。詳細については、「新しい EMR クラスターを作成して EMR Studio Workspace にアタッチする」を参照してください。
- Amazon EMR Git リポジトリ - Workspace を最大 3 つの Git リポジトリにリンクできます。詳細および手順については、「GIT ベースのリポジトリを EMR Studio Workspace にリンクする」を参照してください。
- [Notebook Examples] (ノートブックサンプル) – Workspace に保存できるノートブックの例のリストが示されます。また、Workspace の [Launcher] (ランチャー) ページで [Notebook Examples] (ノートブックサンプル) を選択してサンプルにアクセスすることもできます。
- コマンド — JupyterLab コマンドを検索して実行するキーボード駆動型の方法を提供します。詳細については、 JupyterLab ドキュメントの「コマンドパレット」ページを参照してください。
- [Notebook Tools] (ノートブックツール) - セルのスライドタイプやメタデータなどのオプションを選択および設定できます。[Notebook Tools] (ノートブックツール) オプションは、ノートブックファイルを開いた後に左サイドバーに表示されます。
- [Open Tabs] (開いているタブ) - 開いているタブにジャンプできるように、メイン作業領域で開いているドキュメントとアクティビティがリストされます。詳細については、 JupyterLab ドキュメントの「Tabs and single-document mode」ページを参照してください。
- コラボレーション - Workspace コラボレーションを有効または無効にしたり、共同作業者を管理したりできます。[コラボレーション] パネルを表示するには、必要なアクセス許可を持っている必要があります。詳細については、「Workspace コラボレーションの所有権の設定」を参照してください。

ノートブックサンプルの探索

すべての EMR Studio Workspace には、EMR Studio の機能を調べるために使用できるノートブックサンプルのセットが含まれています。ノートブックサンプルを編集または実行するには、Workspace に保存します。

ノートブックサンプルを Workspace に保存するには

1. 左サイドバーで、[Notebook Examples] (ノートブックサンプル) タブを選択して、[Notebook Examples] (ノートブックサンプル) パネルを開きます。また、Workspace の [Launcher] (ランチャー) ページで [Notebook Examples] (ノートブックサンプル) を選択してサンプルにアクセスすることもできます。
2. ノートブックサンプルを選択して、メイン作業領域でプレビューします。サンプルは読み取り専用です。
3. ノートブックサンプルを Workspace に保存するには、[Save to Workspace] (Workspace に保存) を選択します。EMR Studio は、ホームディレクトリにそのサンプルを保存します。ノートブックサンプルを Workspace に保存した後に、サンプルの名前変更、編集、および実行を行うことができます。

ノートブックの例の詳細については、「EMR Studio Notebook examples GitHub repository」を参照してください。

Workspace コンテンツの保存

Workspace のノートブック エディタで作業する場合、EMR Studio はノートブックのセルと出力のコンテンツを Studio に関連付けられた Amazon S3 の場所に保存します。このバックアッププロセスでは、セッション間の作業が保持されます。

開いているノートブックのタブで CTRL+S を押すか、[File] (ファイル) で保存オプションのいずれかを使用することで、ノートブックを保存することもできます。

Workspace 内のノートブックファイルをバックアップするもう 1 つの方法として、Workspace を Git ベースのリポジトリに関連付けて、変更をリモートリポジトリに同期できます。そうすることで、ノートブックを保存して、別の Workspace または Studio を使用しているチームメンバーに共有することもできます。手順については、「GIT ベースのリポジトリを EMR Studio Workspace にリンクする」を参照してください。

Workspace およびノートブックファイルを削除する

EMR Studio Workspace からノートブックファイルを削除すると、そのファイルは [File browser] (ファイルブラウザ) から削除され、EMR Studio によって Amazon S3 のバックアップコピーが削除されます。Workspace からファイルを削除するときに、ストレージ料金を回避するために、これ以上の手順を実行する必要はありません。

Workspace 全体を削除しても、そのノートブックファイルとフォルダは Amazon S3 の保存場所に残ります。ファイルにはストレージ料金が発生し続けます。ストレージ料金を回避するには、削除した Workspace に関連付けられているすべてのバックアップファイルとフォルダを Amazon S3 から削除します。

EMR Studio Workspace からノートブックファイルを削除するには

1. Workspace の左サイドバーにある [File browser] (ファイルブラウザ) パネルを選択します。
2. 削除するファイルまたはフォルダを選択します。選択項目を右クリックし、[Delete] (削除) を選択します。ファイルがリストから消えます。EMR Studio によって Amazon S3 からファイルまたはフォルダが削除されます。

From the Workspace UI

EMR Studio から Workspace とその関連バックアップファイルを削除する

1. Studio アクセス URL を使用して EMR Studio にログインし、左のナビゲーションから [Workspaces] (Workspace) を選択します。
2. リストから Workspace を見つけ、その名前の横にあるチェックボックスを選択します。複数の Workspace を選択して、同時に削除することができます。
3. [Workspaces] (Workspace) リストの右上にある [Delete] (削除) を選択して、選択した Workspace を削除することを確認します。[Delete] を選択して確定します。
4. Amazon S3 から削除された Workspace に関連付けられたノートブックファイルを削除する場合は、「Amazon Simple Storage Service コンソールユーザーガイド」の「オブジェクトの削除」の手順に従ってください。Studio を作成していない場合は、Studio 管理者に問い合わせて、削除した Workspace の Amazon S3 バックアップの場所を確認してください。

From the Workspaces list

[Workspace] リストから Workspace とその関連バックアップファイルを削除する

1. コンソールの [Workspace] リストに移動します。
2. リストから削除する Workspace を選択してから、[アクション] を選択します。
3. [削除] を選択します。
4. Amazon S3 から削除された Workspace に関連付けられたノートブックファイルを削除する場合は、「Amazon Simple Storage Service コンソールユーザーガイド」の「オブジェクト

の削除」の手順に従ってください。Studio を作成していない場合は、Studio 管理者に問い合わせて、削除した Workspace の Amazon S3 バックアップの場所を確認してください。

Workspace ステータスを理解する

EMR Studio Workspace を作成すると、その Workspace は、Studio内の [Workspaces] (Workspace) リストで行としてその名前、ステータス、作成時刻、最終更新タイムスタンプとともに表示されます。次の表で Workspace のステータスについて説明します。

ステータス	説明
スタート	Workspace は準備中ですが、まだ使用できる状態ではありません。ステータスが [Starting] (開始中) の場合、Workspace を開くことはできません。
準備完了	Workspace を開いてノートブックエディタを使用できますが、ノートブックコードを実行する前に、Workspace を EMR クラスターにアタッチする必要があります。
[Attaching] (アタッチ中)	Workspace は、クラスターにアタッチされているところです。
アタッチ済み	Workspace は EMR クラスターにアタッチされ、ノートブックコードを記述して実行できる状態になっています。Workspaceのステータスが [Attached] (アタッチ済み) でない場合は、ノートブックコードを実行する前に、Workspace をクラスターにアタッチする必要があります。
アイドル状態	Workspace が停止しました。アイドル状態の Workspace を再度アクティブ化するには、Workspace リストから選択します。Workspace を選択すると、ステータ

ステータス	説明
	スが [Idle] (アイドル) から [Starting] (開始中)、[Ready] (準備完了) に変化します。
停止中	Workspace はシャットダウン中で、その後 [アイドル状態] に設定されます。Workspace を停止すると、対応するノートブックカーネルはすべて終了します。EMR Studio は、長時間非アクティブになっているノートブックを停止します。
[Deleting] (削除中)	Workspace を削除すると、EMR Studio はその Workspace を削除対象としてマークし、削除プロセスを開始します。削除プロセスが完了すると、Workspace はリストで表示されなくなります。Workspace を削除しても、そのノートブックファイルは Amazon S3 の保存場所に残ります。

Workspace の接続の問題を解決する

Workspace の接続の問題を解決するには、Workspace を停止して再起動します。Workspace を再起動すると、EMR Studio は、Studio に関連付けられている別のアベイラビリティーゾーンまたは別のサブネットで Workspace を起動します。

EMR Studio Workspace を停止して再起動するには

1. ブラウザで Workspace を閉じます。
2. コンソールで [Workspace] リストに移動します。
3. リストから Workspace を選択し、[Actions] (アクション) を選択します。
4. [Stop] (停止) を選択し、Workspace のステータスが [Stopping] (停止中) から [Idle] (アイドル) に変わるまで待ちます。
5. [Actions] (アクション) を再度選択してから、[Start] (開始) を選択して Workspace を再起動します。
6. Workspaceのステータスが [Starting] (開始中) から [Ready] (準備完了) に変わるまで待ってから、Workspace 名を選択して、新しいブラウザタブで Workspace を再度開きます。

Workspace コラボレーションを設定する

Workspace コラボレーションでは、チームの他のメンバーと同時にノートブックコードを記述して実行できます。同じノートブックファイルで作業をしている場合、共同作業者が行った変更が表示されます。Workspace を作成するときにコラボレーションを有効にするか、既存の Workspace でコラボレーションを有効または無効に切り替えることができます。

> **Note**
>
> EMR Studio Workspace コラボレーションは、EMR Serverless インタラクティブアプリケーションではサポートされず、信頼できる ID 伝達が有効になっている場合もサポートされません。

前提条件

Workspace のコラボレーションを設定する前に、以下のタスクを完了してください。

- EMR Studio 管理者から必要なアクセス許可が与えられていることを確認してください。例えば、次のステートメントでは、タグキー `creatorUserId` の値がユーザーの ID (ポリシー変数 `aws:userId` で示されている) と一致する Workspace に対してコラボレーションを設定できます。

```
{
    "Sid": "UserRolePermissionsForCollaboration",
    "Action": [
        "elasticmapreduce:UpdateEditor",
        "elasticmapreduce:PutWorkspaceAccess",
        "elasticmapreduce:DeleteWorkspaceAccess",
        "elasticmapreduce:ListWorkspaceAccessIdentities"
    ],
    "Resource": "*",
    "Effect": "Allow",
    "Condition": {
        "StringEquals": {
            "elasticmapreduce:ResourceTag/creatorUserId": "${aws:userid}"
        }
    }
}
```

- EMR Studio に関連付けられているサービスロールに、次のステートメントの例のように、Workspace コラボレーションを有効にして設定するために必要なアクセス許可があることを確認してください。

```
{
    "Sid": "AllowWorkspaceCollaboration",
    "Effect": "Allow",
    "Action": [
        "iam:GetUser",
        "iam:GetRole",
        "iam:ListUsers",
        "iam:ListRoles",
        "sso:GetManagedApplicationInstance",
        "sso-directory:SearchUsers"
    ],
    "Resource": "*"
}
```

詳細については、「EMR Studio サービスロールを作成する」を参照してください。

Workspace コラボレーションを有効にして共同作業者を追加するには

1. Workspace で、ランチャー画面または左側のパネルの下部にある [コラボレーション] アイコンを選択します。

> **Note**
>
> [コラボレーション] パネルは、Studio 管理者が Workspace のコラボレーションを設定するためのアクセス許可を与えていない限り、表示されません。詳細については、「Workspace コラボレーションの所有権の設定」を参照してください。

2. [Workspace コラボレーションを許可] トグルがオンになっていることを確認します。コラボレーションを有効にすると、自分と追加した共同作業者のみが Studio の [Workspaces] ページのリストで Workspace を確認できます。
3. [共同作業者名] を入力します。Workspace には、自分を含めて最大 5 人の共同作業者を含めることができます。共同作業者は、EMR Studio へのアクセス権を持つユーザーであれば誰でも指定できます。共同作業者を入力しない場合、Workspace は自分だけがアクセスできるプライベートな Workspace になります。

以下の表は、所有者の ID タイプに基づいて入力する共同作業者の値を示しています。

> **Note**
>
> 所有者は同じ ID タイプの共同作業者のみを招待できます。例えば、ユーザーは他のユーザーのみを追加でき、IAM Identity Center ユーザーは他の IAM Identity Center ユーザーのみを追加できます。

[Authentication mode] (認証モード)	[共同作業者名] に入力する値
IAM 認証	ユーザー名。これは AWS Management Consoleにログインしたときにユーザーに表示される名前です。
IAM フェデレーション	IAM ロールの名前とオプションのセッション名。 同じ IAM ロールを継承するすべてのフェデレーションユーザーを追加するには、フェデレーション用の IAM ロールの名前を指定します。 1 人のユーザーを共同作業者として追加する場合は、ロールとセッション名を指定します。例えば `MyRoleName:MySessionName` です。
SSO	IAM Identity Center ユーザー名 (例: `user@example.com`。)

4. [追加] を選択します。これで、共同作業者の EMR Studio の [Workspaces] ページに Workspace が表示され、Workspace を起動して一緒にリアルタイムで使用できるようになります。

> **Note**
>
> Workspace コラボレーションを無効にすると Workspace は共有状態に戻り、すべての Studio ユーザーに表示されるようになります。共有状態では、Workspace を開いて作業できるのは一度に 1 人の Studio ユーザーのみです。

ランタイムロールを使用して EMR Studio Workspace を実行する

> **Note**
>
> このページで説明されているランタイムロール機能は、Amazon EC2 で実行される Amazon EMR にのみ適用され、EMR Serverless インタラクティブアプリケーションのランタイムロール機能を指すものではありません。EMR Serverless でランタイムロールを使用する方法の詳細については、「Amazon EMR Serverless ユーザーガイド」の「Job runtime roles」を参照してください。

ランタイムロールは、Amazon EMR クラスターにジョブまたはクエリを送信するときに指定できる AWS Identity and Access Management (IAM) ロールです。EMR クラスターに送信するジョブまたはクエリは、ランタイムロールを使用して Amazon S3 のオブジェクトなどの AWS リソースにアクセスします。

Amazon EMR 6.11 以降を使用する EMR クラスターに EMR Studio Workspace をアタッチすると、AWS リソースにアクセスするときに使用するジョブまたはクエリのランタイムロールを選択できます。ただし、EMR クラスターがランタイムロールをサポートしていない場合、EMR クラスターは AWS リソースにアクセスするときにロールを引き受けません。

Amazon EMR Studio Workspace でランタイムロールを使用する前に、管理者は Studio ユーザーがランタイムロールで `elasticmapreduce:GetClusterSessionCredentials` API を呼び出せるようにユーザーアクセス許可を設定する必要があります。その後、Amazon EMR Studio Workspace で使用できるランタイムロールを使用して新しいクラスターを起動します。

このページの内容

- 考慮事項

ランタイムロールのユーザーアクセス許可を設定する

ユーザーアクセス許可を設定して、Studio ユーザーが使用するランタイムロールで `elasticmapreduce:GetClusterSessionCredentials` API を呼び出せるようにします。また、ユーザーが Studio を使い始める前に「the section called “Studio ユーザーアクセス許可 (EC2、EKS)”」を実施する必要があります。

> **Warning**
>
> このアクセス許可を付与するには、`GetClusterSessionCredentials` API を呼び出すためのアクセス許可を呼び出し元に付与するときに、`elasticmapreduce:ExecutionRoleArn` コンテキストキーに基づく条件を作成します。次の例は、その方法を示しています。

```
{
      "Sid": "AllowSpecificExecRoleArn",
      "Effect": "Allow",
      "Action": [
          "elasticmapreduce:GetClusterSessionCredentials"
      ],
      "Resource": "*",
      "Condition": {
          "StringEquals": {
              "elasticmapreduce:ExecutionRoleArn": [
                  "arn:aws:iam::111122223333:role/test-emr-demo1",
                  "arn:aws:iam::111122223333:role/test-emr-demo2"
              ]
          }
      }
  }
```

次の例は、IAM プリンシパルが `test-emr-demo3` という IAM ロールをランタイムロールとして使用できるようにする方法を示しています。また、ポリシー所有者はクラスター ID が `j-123456789` の Amazon EMR クラスターにのみアクセスできます。

```
{
```

```
    "Sid":"AllowSpecificExecRoleArn",
    "Effect":"Allow",
    "Action":[
        "elasticmapreduce:GetClusterSessionCredentials"
    ],
    "Resource": [
          "arn:aws:elasticmapreduce:<region>:111122223333:cluster/j-123456789"
     ],
    "Condition":{
        "StringEquals":{
            "elasticmapreduce:ExecutionRoleArn":[
                "arn:aws:iam::111122223333:role/test-emr-demo3"
            ]
        }
    }
}
```

次の例は、IAM プリンシパルが test-emr-demo4 という文字列で始まる名前を持つ任意の IAM ロールをランタイムロールとして使用できるようにします。また、ポリシー所有者は tagKey:tagValue というキーと値のペアでタグ付けされた Amazon EMR クラスターにのみアクセスできます。

```
{
    "Sid":"AllowSpecificExecRoleArn",
    "Effect":"Allow",
    "Action":[
        "elasticmapreduce:GetClusterSessionCredentials"
    ],
    "Resource": "*",
    "Condition":{
        "StringEquals":{
             "elasticmapreduce:ResourceTag/tagKey": "tagValue"
        },
        "StringLike":{
            "elasticmapreduce:ExecutionRoleArn":[
                "arn:aws:iam::111122223333:role/test-emr-demo4*"
            ]
        }
    }
}
```

ランタイムロールを使用して新しいクラスターを起動する

必要なアクセス許可を取得したので、Amazon EMR Studio Workspace で使用できるランタイムロールを使用して新しいクラスターを起動します。

ランタイムロールを使用して新しいクラスターを既に起動している場合は、「the section called "Workspace でクラスターを使用する"」セクションに進んでください。

1. まず、「Amazon EMR ステップのランタイムロール」セクションの前提条件を満たします。
2. 次に、以下の設定でクラスターを起動し、Amazon EMR Studio Workspace でランタイムロールを使用します。クラスターを起動する手順については、「クラスターのセキュリティ設定を指定する」を参照してください。
   - emr-6.11.0 以降のリリースラベルを選択します。
   - クラスターアプリケーションとして Spark、Livy、Jupyter Enterprise Gateway を選択します。
   - 前のステップで作成したセキュリティ設定を使用します。
   - オプションで、EMR クラスターの Lake Formation を有効にできます。詳細については、「Amazon EMR での Lake Formation の有効化」を参照してください。

クラスターを起動すると、EMR Studio Workspace でランタイムロールが有効なクラスターを使用する準備が整います。

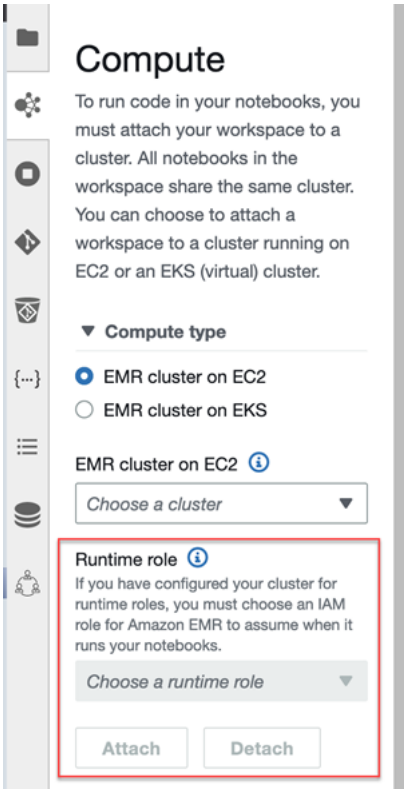
> **Note**
>
> ExecutionRoleArn 値が の場合、 `ExecutionEngineConfig.Type`値は StartNotebookExecution API オペレーションでは現在サポートされていませんEMR。

Workspace でランタイムロールを使用した EMR クラスターを使用する

クラスターをセットアップして起動したら、ランタイムロールが有効なクラスターを EMR Studio Workspace で使用できます。

1. 新しい Workspace を作成するか、既存の Workspace を起動します。詳細については、「EMR Studio Workspace の作成」を参照してください。

2. 開いている Workspace の左側のサイドバーで [EMR クラスター] タブを選択します。[コンピューティングタイプ] セクションを展開して、[EC2 上の EMR クラスター] メニューからクラスターを選択し、[ランタイムロール] メニューからランタイムロールを選択します。



3. [アタッチ] を選択して、ランタイムロールを使用するクラスターを Workspace にアタッチします。

考慮事項

Amazon EMR Studio Workspace でランタイムロールが有効なクラスターを使用するときは、以下の考慮事項に注意してください。

- Amazon EMR リリース 6.11 以降を使用する EMR クラスターに EMR Studio Workspace をアタッチする場合にのみ、ランタイムロールを選択できます。
- このページで説明されているランタイムロール機能は、Amazon EC2 で実行されている Amazon EMR でのみサポートされ、EMR Serverless インタラクティブアプリケーションではサポートされません。EMR Serverless のランタイムロールの詳細については、「Amazon EMR Serverless ユーザーガイド」の「Job runtime roles」を参照してください。
- クラスターにジョブを送信するとき、ランタイムロールを指定する前に追加のアクセス許可を設定する必要がありますが、EMR Studio Workspace によって生成されたファイルにアクセスするため

の追加の許可は必要ありません。これらのファイルのアクセス許可は、ランタイムロールを使用しないクラスターから生成されたファイルと同じです。

- ランタイムロールを持つクラスターを使用している EMR Studio Workspace では、SQL Explorer は使用できません。Amazon EMR では、Workspace がランタイムロールが有効な EMR クラスターにアタッチされると、UI の SQL Explorer が無効になります。
- ランタイムロールを持つクラスターを使用する EMR Studio Workspace では、コラボレーションモードは使用できません。Amazon EMR では、Workspace がランタイムロールが有効な EMR クラスターにアタッチされると、Workspace のコラボレーション機能が無効になります。Workspace には、Workspace をアタッチしたユーザーのみが引き続きアクセスできます。
- IAM Identity Center の信頼できる ID 伝達が有効になっている Studio では、ランタイムロールを使用できません。
- ランタイムロールが有効なクラスターに対して Spark UI から [Page may not be safe!] という警告が 表示される場合があります。この場合は、アラートを無視して続行し、Spark UI を表示してください。

Workspace ノートブックをプログラムで実行する

> **Note**
>
> Notebooks のプログラムによる実行は、Amazon EMR Serverless インタラクティブアプリケーションではサポートされていません。

Amazon EMR Studio Workspace ノートブックは、スクリプトを使用してプログラムで実行することも、 AWS CLIで実行することもできます。ノートブックをプログラムで実行する方法については、「EMR Notebooks をプログラムで実行するサンプルコマンド」を参照してください。

SQL Explorer でデータをブラウズする

> **Note**
>
> EMR Studio の SQL Explorer は、Amazon EMR Serverless 対話型アプリケーション、または IAM Identity Center の信頼できる ID 伝達が有効になっている Studio ではサポートされていません。

このトピックでは、Amazon EMR Studio で SQL Explorer の使用を開始する際に役立つ情報を紹介します。SQL Explorer は Workspace の単一ページのツールで、EMR クラスターのデータカタログのデータソースについて理解するのに役立ちます。SQL Explorer を使用してデータを参照したり、SQL クエリを実行してデータを取得したり、クエリ結果をダウンロードしたりできます。

SQL Explorer は Presto をサポートしています。SQL Explorer を使用する前に、Amazon EMR バージョン 5.34.0 以降またはバージョン 6.4.0 以降を使用するクラスターに Presto がインストールされていることを確認してください。Amazon EMR Studio SQL Explorer は、転送時の暗号化を設定している Presto クラスターをサポートしていません。これは、これらのクラスターでは Presto が TLS モードで実行されるためです。

クラスターのデータカタログを参照する

SQL Explorer にはカタログブラウザインターフェイスが用意されており、これを使用してデータがどのように構成されているかを調べて理解することができます。例えば、SQL クエリを記述する前に、データカタログブラウザを使用してテーブル名や列名を確認できます。

データカタログを参照するには

1. Workspace で SQL Explorer を開きます。
2. Workspace が、Amazon EMR バージョン 6.4.0 以降を使用し、Presto がインストール済みの EC2 で実行されている EMR クラスターにアタッチされていることを確認します。新しいクラスターを作成することも、既存のクラスターを選択することもできます。詳細については、「EMR Studio Workspace にコンピューティングをアタッチする」を参照してください。
3. ドロップダウンリストから [データベース] を選択して参照します。
4. データベースのテーブルを展開すると、そのテーブルの列名が表示されます。また、検索バーにキーワードを入力して、テーブルの結果をフィルタリングすることもできます。

SQL クエリを実行してデータを取得する

SQL クエリを使用してデータを取得し、結果をダウンロードするには

1. Workspace で SQL Explorer を開きます。
2. Workspace が、Presto と Spark がインストール済みの EC2 で実行されている EMR クラスターにアタッチされていることを確認します。新しいクラスターを作成することも、既存のクラスターを選択することもできます。詳細については、「EMR Studio Workspace にコンピューティングをアタッチする」を参照してください。

3. [エディタを開く] を選択して、Workspace で新しいエディタタブを開きます。
4. エディタタブで SQL クエリを作成します。
5. [実行] を選択します。
6. [結果のプレビュー] でクエリの結果を表示します。SQL Explorer は、デフォルトで最初の 100 件の結果を表示します。[最初の 100 件のクエリ結果をプレビュー] ドロップダウンを使用すると、表示する結果の数を選択して変更できます (最大 1000 件)。
7. 結果を CSV 形式でダウンロードするには、[結果をダウンロード] を選択します。最大 1000 行の結果をダウンロードできます。

EMR Studio Workspace にコンピューティングをアタッチする

Amazon EMR Studio は、EMR クラスターのカーネルを使用してノートブックコマンドを実行します。カーネルを選択する前に、Amazon EC2 インスタンスを使用するクラスター、Amazon EMR on EKS クラスター、または EMR Serverless アプリケーションに Workspace をアタッチする必要があります。EMR Studio では、Workspace を新規または既存のクラスターにアタッチでき、Workspace を閉じずにクラスターを柔軟に変更できます。

このセクションでは、EMR Studio のクラスターの操作およびプロビジョニングに役立つ次のトピックについて説明します。

- Amazon EC2 クラスターを EMR Studio Workspace にアタッチする
- Amazon EMR on EKS クラスターを EMR Studio Workspace にアタッチする
- Amazon EMR Serverless アプリケーションを EMR Studio Workspace にアタッチする
- 新しい EMR クラスターを作成して EMR Studio Workspace にアタッチする
- EMR Studio Workspace からコンピューティングをデタッチする

Amazon EC2 クラスターを EMR Studio Workspace にアタッチする

Workspace の作成時に Amazon EC2 で実行されている EMR クラスターを Workspace にアタッチするか、既存の Workspace にクラスターをアタッチできます。新しいクラスターを作成してアタッチする場合は、「新しい EMR クラスターを作成して EMR Studio Workspace にアタッチする」を参照してください。

> **Note**
>
> IAM Identity Center の信頼できる ID 伝達が有効になっている Studio のワークスペースは、Identity Center が有効になっているセキュリティ設定の EMR クラスターにのみ接続できます。

On create

Workspace の作成時に Amazon EMR コンピューティングクラスターをアタッチする

1. [Create a Workspace] (Workspace の作成) ダイアログボックスで、新しい Workspace のサブネットが既に選択されていることを確認します。[Advanced configuration] (高度な設定) セクションを展開します。
2. [Attach Workspace to an EMR cluster] (EMR クラスターへの Workspace のアタッチ) を選択します。
3. [EMR クラスター] ドロップダウンリストで、既存の EMR クラスターを選択して Workspace にアタッチします。

クラスターをアタッチしたら、Workspace の作成を完了します。新しい Workspace を初めて開き、[EMR クラスター] パネルを選択すると、選択したクラスターがアタッチされていることがわかります。

On launch

Workspace の起動時に Amazon EMR コンピューティングクラスターにアタッチする

1. Workspace のリストに移動し、起動する Workspace の行を選択します。次に、[Workspace を起動] > [オプションを使用して起動する] を選択します。
2. Workspace にアタッチする EMR クラスターを選択します。

クラスターをアタッチしたら、Workspace の作成を完了します。新しい Workspace を初めて開き、[EMR クラスター] パネルを選択すると、選択したクラスターがアタッチされていることがわかります。

In JupyterLab

の Amazon EMR コンピューティングクラスターに Workspace をアタッチする JupyterLab

1. Workspace を選択し、[Workspace を起動] > [クイック起動] を選択します。
2. 内で JupyterLab、左側のサイドバーのクラスタータブを開きます。
3. [EMR on EC2 クラスター] ドロップダウンを選択するか、Amazon EMR on EKS クラスターを選択します。
4. [アタッチ] を選択して、クラスターを Workspace にアタッチします。

クラスターをアタッチしたら、Workspace の作成を完了します。新しい Workspace を初めて開き、[EMR クラスター] パネルを選択すると、選択したクラスターがアタッチされていることがわかります。

In the Workspace UI

Workspace ユーザーインターフェイスから Workspace を Amazon EMR コンピューティングクラスターにアタッチする

1. クラスターにアタッチする Workspace で、左側のサイドバーから [EMR クラスター] アイコンを選択して、[クラスター] パネルを開きます。
2. [クラスターのタイプ] でドロップダウンを展開し、[EC2 上の EMR クラスター] を選択します。
3. ドロップダウンリストからクラスターを選択します。クラスター選択ドロップダウンリストを有効にするには、まず既存のクラスターをデタッチする必要がある場合があります。
4. 添付を選択します。クラスターがアタッチされると、成功メッセージが表示されます。

Amazon EMR on EKS クラスターを EMR Studio Workspace にアタッチする

Amazon EC2 で実行されている Amazon EMR クラスターを使用するだけでなく、Workspace を Amazon EMR on EKS クラスターにアタッチしてノートブックコードを実行することもできます。Amazon EMR on EKS の詳細については、「Amazon EMR on EKS とは」を参照してください。

Workspace を Amazon EMR on EKS クラスターに接続する前に、Studio 管理者からアクセス許可を付与してもらう必要があります。

> **Note**
>
> IAM Identity Center の信頼できる ID 伝達を使用する EMR Studio では、EKS クラスターで Amazon EMR を起動することはできません。

On create

Workspace の作成時に Amazon EMR on EKS クラスターをアタッチするには

1. [Create a Workspace] (Workspace の作成) ダイアログボックスで、[Advanced configuration] (高度な設定) セクションを展開します。
2. [Amazon EMR on EKS クラスターへの Workspace のアタッチ] を選択します。
3. [Amazon EMR on EKS クラスター] で、ドロップダウンリストからクラスターを選択します。
4. [Select an endpoint] (エンドポイントの選択) で、Workspace にアタッチするマネージドエンドポイントを選択します。マネージドエンドポイントは、選択したクラスターと EMR Studio が通信できるようにするゲートウェイです。
5. [Workspace の作成] を選択して、Workspace の作成プロセスを完了し、選択したクラスターをアタッチします。

クラスターをアタッチしたら、Workspace の作成プロセスを完了できます。新しい Workspace を初めて開き、[EMR クラスター] パネルを選択すると、選択したクラスターがアタッチされていることがわかります。

In the Workspace UI

Workspace ユーザーインターフェイスから Amazon EMR on EKS クラスターをアタッチするには

1. クラスターにアタッチする Workspace で、左側のサイドバーから [EMR クラスター] アイコンを選択して、[クラスター] パネルを開きます。
2. [クラスターのタイプ] ドロップダウンを展開し、[EKS 上の EMR クラスター] を選択します。
3. [EKS 上の EMR クラスター] で、ドロップダウンリストからクラスターを選択します。

4. [Endpoint] (エンドポイント) で、Workspace にアタッチするマネージドエンドポイントを選択します。マネージドエンドポイントは、選択したクラスターと EMR Studio が通信できるようにするゲートウェイです。
5. 添付を選択します。クラスターがアタッチされると、成功メッセージが表示されます。

Amazon EMR Serverless アプリケーションを EMR Studio Workspace にアタッチする

Workspace を EMR Serverless アプリケーションにアタッチして、インタラクティブなワークロードを実行できます。詳細については、「Using notebooks to run interactive workloads with EMR Serverless through EMR Studio」を参照してください。

> **Note**
>
> IAM Identity Center の信頼できる ID 伝達を使用する EMR Studio に、EMR Serverless アプリケーションをアタッチすることはできません。

Example で Workspace を EMR Serverless アプリケーションにアタッチする JupyterLab

Workspace を EMR Serverless アプリケーションに接続する前に、アカウント管理者に「Required permissions for interactive workloads」で説明しているアクセス許可を付与してもらう必要があります。

1. EMR Studio に移動して Workspace を選択し、[Workspace を起動] > [クイック起動] を選択します。
2. 内で JupyterLab、左側のサイドバーのクラスタータブを開きます。
3. コンピューティングオプションとして [EMR Serverless] を選択し、EMR Serverless アプリケーションとランタイムロールを選択します。
4. クラスターを Workspace にアタッチするには、[アタッチ] を選択します。

対象の Workspace を開くと、選択したアプリケーションがアタッチされていることが確認できます。

新しい EMR クラスターを作成して EMR Studio Workspace にアタッチする

上級 EMR Studio ユーザーは、Amazon EC2 で実行されている新しい EMR クラスターを Workspace で使用するためにプロビジョニングできます。新しいクラスターには、EMR Studio に必要なすべてのビッグデータアプリケーションがデフォルトでインストールされています。

クラスターを作成するには、Studio 管理者から、セッションポリシーを使用してアクセス許可を付与してもらう必要があります。詳細については、「EMR Studio ユーザーのアクセス許可ポリシーの作成」を参照してください。

[Create a Workspace] (Workspace の作成) ダイアログボックスまたは [Cluster] (クラスター) パネル (Workspace UI) で、新しいクラスターを作成できます。いずれの方法でも、次の 2 つのクラスター作成オプションがあります。

1. [Create an EMR cluster] (EMR クラスターを作成) - Amazon EC2 インスタンスタイプとカウントを選択して EMR クラスターを作成します。
2. [Use a cluster template] (クラスターテンプレートの使用) - 事前定義されたクラスタテンプレートを選択して、クラスターをプロビジョニングします。このオプションは、クラスターテンプレートを使用するアクセス許可がある場合に表示されます。

> **Note**
>
> Studio 用の IAM Identity Center で信頼できる ID 伝達を有効にした場合は、テンプレートを使用してクラスターを作成する必要があります。

クラスター設定を提供して EMR クラスターを作成するには

1. 開始点を選択します。

実行方法	手順
[Create a Workspace] (Workspace の作成) ダイアログボックスを使用して Workspace を作成するときにクラスターを作成する。	[Create a Workspace] (Workspace の作成) ダイアログボックスで [Advanced configuration] (高度な設定) セクションを展開し、[Create an EMR cluster] (EMR クラスターを作成) を選択します。

実行方法	手順
Workspace を作成した後に Workspace UI の [EMR クラスター] パネルからクラスターを作成する。	開いている Workspace の左側のサイドバーにある [EMR クラスター] タブで、[高度な設定] セクションを展開し、[クラスターの作成] を選択します。

2. [Cluster name] (クラスター名) を入力します。クラスターに名前を付けると、後から EMR Studio クラスターリストでそのクラスターを見つけるのに役立ちます。
3. [Amazon EMR リリース] で、クラスターの Amazon EMR リリースバージョンを選択します。
4. [Instance] (インスタンス) で、クラスターの Amazon EC2 インスタンスのタイプと数を選択します。インスタンスタイプの選択の詳細については、「Amazon EC2 インスタンスを設定する」を参照してください。1 つのインスタンスがプライマリノードとして使用されます。
5. EMR Studio が新しいクラスターを起動できる [Subnet] (サブネット) を選択します。各サブネットオプションは Studio 管理者によって事前承認され、Workspace はリストされたサブネット内のクラスターに接続できる必要があります。
6. [S3 URI for log storage] (ログストレージの S3 URI) を選択します。
7. [Create EMR cluster] (EMR クラスターの作成) を選択して、クラスターをプロビジョニングします。[Workspace の作成] ダイアログボックスを使用する場合、[Workspace の作成] を選択して、Workspace を作成し、クラスターをプロビジョニングします。EMR Studio は新しいクラスターをプロビジョニングした後に、クラスターを Workspace にアタッチします。

クラスターテンプレートを使用してクラスターを作成するには

1. 開始点を選択します。

実行方法	手順
[Create a Workspace] (Workspace の作成) ダイアログボックスを使用して Workspace を作成するときにクラスターを作成する。	[Create a Workspace] (Workspace の作成) ダイアログボックスで [Advanced configuration] (高度な設定) セクションを展開し、[Use a cluster template] (クラスターテンプレートの使用) を選択します。
Workspace UI の [EMR クラスター] パネルからクラスターを作成する。	開いている Workspace の左側のサイドバーにある [EMR クラスター] タブで、[高度な設

実行方法	手順
	定] セクションを展開し、[クラスターテンプレート] を選択します。

2. ドロップダウンリストからクラスターテンプレートを選択します。使用可能な各クラスタテンプレートには、選択に役立つ簡単な説明が含まれています。
3. 選択したクラスターテンプレートには、Amazon EMR リリースバージョンやクラスター名などの追加のパラメーターが含まれる場合があります。値を選択または挿入するか、管理者が選択したデフォルト値を使用できます。
4. EMR Studio が新しいクラスターを起動できる [Subnet] (サブネット) を選択します。各サブネットオプションは Studio 管理者によって事前承認され、Workspace はサブネット内のクラスターに接続できる必要があります。
5. [Use cluster template] (クラスターテンプレートの使用) を選択して、クラスターをプロビジョニングし、Workspace にアタッチします。EMR Studio がクラスターを作成するまでに数分かかります。[Workspace の作成] ダイアログボックスを使用する場合、[Workspace の作成] を選択して、Workspace を作成し、クラスターをプロビジョニングします。EMR Studio は新しいクラスターをプロビジョニングした後に、クラスターを Workspace にアタッチします。

EMR Studio Workspace からコンピューティングをデタッチする

Workspace にアタッチされているクラスターを交換するには、Workspace UI からクラスターをデタッチします。

Workspace からクラスターをデタッチするには

1. クラスターからデタッチする Workspace で、左側のサイドバーから [EMR クラスター] アイコンを選択して、[クラスター] パネルを開きます。
2. [Select cluster] (クラスターの選択) で [Detach] (デタッチ) を選択し、EMR Studio によってクラスターがデタッチされるまで待ちます。クラスターがデタッチされると、成功メッセージが表示されます。

EMR Serverless アプリケーションを EMR Studio Workspace からデタッチするには

Workspace にアタッチされているコンピューティングを交換するには、Workspace UI からアプリケーションをデタッチします。

1. クラスターからデタッチする Workspace で、左側のサイドバーから [Amazon EMR コンピューティング] アイコンを選択して、[コンピューティング] パネルを開きます。
2. [コンピューティングの選択] で [デタッチ] を選択し、EMR Studio によってアプリケーションがデタッチされるまで待ちます。アプリケーションがデタッチされると、成功メッセージが表示されます。

GIT ベースのリポジトリを EMR Studio Workspace にリンクする

EMR Studio の Git リポジトリについて

EMR Studio Workspace には、最大 3 つの Git リポジトリを関連付けることができます。デフォルトでは、各 Workspace では、Studio と同じ AWS アカウントに関連付けられている Git リポジトリのリストから選択できます。新しい Git リポジトリを Workspace のリソースとして作成することもできます。

クラスターのプライマリノードに接続されているときに、ターミナルコマンドを使用して Git コマンドを次のように実行できます。

```
!git pull origin <branch-name>
```

jupyterlab-git 拡張機能を使用することもできます。左サイドバーから、[Git] アイコンを選択して開きます。の jupyterlab-git 拡張機能の詳細については JupyterLab、「jupyterlab-git」を参照してください。

前提条件

- Git リポジトリを Workspace に関連付けるには、Git リポジトリのリンクを許可するように Studio を設定する必要があります。Studio 管理者は、「Git ベースのリポジトリのアクセス権とアクセス許可を設定する」の手順を実行する必要があります。
- CodeCommit リポジトリを使用する場合は、Git 認証情報と HTTPS を使用する必要があります。AWS Command Line Interface 認証情報ヘルパーを使用した SSH キーと HTTPS はサポートされていません。 は、個人用アクセストークン (PATs CodeCommit もサポートしていません。詳細については、「IAM ユーザーガイド」の「 CodeCommitでの IAM の使用」および「 ユーザーガイド」の「Git 認証情報を使用した HTTPS AWS CodeCommit ユーザーのセットアップ」を参照してください。

手順

関連付けられた Git リポジトリを Workspace にリンクするには

1. リポジトリにリンクする Workspace を Studio の [Workspaces] (Workspace) リストから開きます。
2. 左側のサイドバーで、[Amazon EMR Git リポジトリ] アイコンを選択して、[Git リポジトリ] ツールパネルを開きます。
3. [Git repositories] (Git リポジトリ) で、ドロップダウンリストを展開し、Workspace にリンクするリポジトリを最大 3 つ選択します。EMR Studio は選択項目を登録し、各リポジトリのリンクを開始します。

リンクプロセスが完了するまでにしばらく時間がかかることがあります。[Git repository] (Git リポジトリ) ツールパネルで選択した各リポジトリのステータスを確認できます。EMR Studio がリポジトリを Workspace にリンクすると、そのリポジトリに属するファイルが [File browser] (ファイルブラウザ) パネルに表示されます。

新しい Git リポジトリをリソースとして Workspace に追加するには

1. リポジトリにリンクする Workspace を Studio の [Workspaces] (Workspace) リストから開きます。
2. 左側のサイドバーで、[Amazon EMR Git リポジトリ] アイコンを選択して、[Git リポジトリ] ツールパネルを開きます。
3. [Add new Git repository] (新しい Git リポジトリを追加する) を選択します。
4. [Repository name] (リポジトリ名) に、EMR Studio でのリポジトリのわかりやすい名前を入力します。名前には、英数字、ハイフン、およびアンダースコアのみを含めることができます。
5. [Git repository URL (Git リポジトリ URL)] に、リポジトリの URL を入力します。 CodeCommit リポジトリを使用する場合、これは URL のクローンを選択し、HTTPS のクローンを作成するときにコピーされる URL です。 例えば `https://git-codecommit.us-west-2.amazonaws.com/v1/repos/[MyCodeCommitRepoName]` です。
6. [Branch] (ブランチ) で、チェックアウトする既存のブランチの名前を入力します。
7. [Git credentials] (Git 認証情報) で、以下のガイドラインに従ってオプションを選択します。EMR Studio は、Secrets Manager に保存されたシークレットを使用して Git 認証情報にアクセスします。

> **Note**
>
> GitHub リポジトリを使用する場合は、個人アクセストークン (PAT) を使用して認証することをお勧めします。2021 年 8 月 13 日以降、 はトークンベースの認証 GitHub を必要とし、Git オペレーションの認証時にパスワードを受け入れなくなります。詳細については、 ブログの「Git オペレーションのトークン認証要件」の投稿を参照してください。 GitHub

オプション	説明
新しいシークレットを作成する	既存の Git 認証情報を、 AWS Secrets Manager で作成される新しいシークレットに関連付けるには、このオプションを選択します。リポジトリに使用する Git 認証情報に基づいて、以下のいずれかの操作を行います。 Git のユーザー名とパスワードを使用してリポジトリにアクセスする場合は、[Username and password] (ユーザー名とパスワード) を選択し、Secrets Manager で使用する [Secret name] (シークレット名) を入力してから、シークレットに関連付ける [Username] (ユーザー名) および [Password] (パスワード) を入力します。 または 個人用アクセストークンを使用してリポジトリにアクセスする場合は、[Personal access token (PAT)] (Personal access token (PAT)) を選択し、Secrets Manager で使用する [Secret name] (シークレット名) を入力してから、[personal access token] (個人用アクセストークン) を入力します。詳細については、「 のコマンドライン用の個人用アクセストークンの作成 GitHub」および「Bitbucket .repositories 用の個人用アクセストークンはこのオプションをサポートしていません」を参照してください。CodeCommit
認証情報なしでパブリックリポジトリを使用する	パブリックリポジトリにアクセスするには、このオプションを選択します。

オプション	説明
既存の AWS シークレットを使用する	認証情報を Secrets Manager でシークレットとしてすでに保存している場合は、このオプションを選択し、リストからシークレット名を選択します。 Git ユーザー名とパスワードに関連付けられたシークレットを選択する場合、シークレットは `{"gitUsername": "` *`MyUserName`* `", "gitPassword": "` *`MyPassword`* `"}` の形式にする必要があります。

8. [Add repository] (リポジトリの追加) を選択して、新しいリポジトリを作成します。EMR Studio が新しいリポジトリを作成すると、成功メッセージが表示されます。[Git repositories] (Git リポジトリ) のドロップダウンリストに新しいリポジトリが表示されます。。
9. 新しいリポジトリを Workspace にリンクするには、[Git repositories] (Git リポジトリ) のドロップダウンリストから選択します。

リンクプロセスが完了するまでにしばらく時間がかかることがあります。EMR Studio が新しいリポジトリを Workspace にリンクすると、リポジトリと同じ名前の新しいフォルダが [File Browser] (ファイルブラウザ) パネルに表示されます。

別のリンクされたリポジトリを開くには、[File browser] (ファイルブラウザ) でそのフォルダに移動します。

EMR Studio での Amazon Athena SQL エディターの使用

概要

Amazon EMR Studio を使用して、Amazon Athena でインタラクティブなクエリを開発して実行できます。これは、Spark、Scala、その他のワークロードの実行に使用しているのと同じ EMR Studio インターフェイスから Athena で SQL 分析を実行できることを意味します。この統合により、自動補完を使用して、クエリを迅速に開発したり、 AWS Glue Data Catalog 内のデータを参照したり、保存されたクエリを作成したり、クエリ履歴を表示したりできます。

Amazon Athena の使用の詳細については、「Amazon Athena ユーザーガイド」の「Athena SQL の使用」を参照してください。

EMR Studio で Athena SQL エディタを使用する

次のステップを使用して、EMR Studio から Amazon Athena でインタラクティブクエリを開発して実行します。

1. この Studio で Workspace にアクセスするユーザーのユーザーロールに必要なアクセス許可を追加します。アクセス許可は、EMR Studio ユーザーの AWS Identity and Access Management アクセス許可 表の [EMR Studio から Amazon Athena SQL エディタにアクセスする] 列に記載されています。または、ユーザーポリシーの例 から [詳細] ポリシーの内容をコピーして、これを含む EMR Studio の機能に対するすべての権限をユーザーに付与することもできます。
2. EMR Studio をセットアップして作成します。
3. Studio に移動し、サイドバーから [クエリエディター] を選択します。

おなじみの Athena エディター UI が表示されます。開始方法と Athena SQL を使用してインタラクティブクエリを実行する方法については、「Amazon Athena ユーザーガイド」の「はじめに」と「Athena SQL の使用」を参照してください。

> **Note**
>
> EMR Studio の IAM Identity Center による信頼できる ID 伝達を有効にしている場合は、Athena ワークグループを使用してクエリアクセスを制御する必要があります。また、使用するワークグループも信頼できる ID 伝達を使用する必要があります。Identity Center をセットアップし、ワークグループで信頼できる ID 伝達を有効にする手順については、「Amazon Athena ユーザーガイド」の「IAM Identity Center 対応の Athena ワークグループの使用」を参照してください。

EMR Studio で Athena SQL エディターを使用する際の考慮事項

- Athena との統合は、EMR Studio および Athena が利用できるすべての商用リージョンで利用できます。
- 次の Athena 機能は EMR Studio では利用できません。
  - Athena ワークグループ、データソース、キャパシティ予約の作成または更新などの管理者機能
  - Athena for Spark または Spark ノートブック
  - Amazon DataZone 統合
  - コストベースオプティマイザー (CBO)

- ステップ関数

Amazon と EMR Studio Workspace CodeWhisperer の統合

概要

Amazon CodeWhisperer EMR Studio で Amazon を使用すると、 でコードを記述するときにリアルタイムのレコメンデーションを取得できます JupyterLab。 CodeWhisperer は、コメントを完了し、1 行のコードを完了し、 line-by-line レコメンデーションを作成し、完全な形式の関数を生成できます。

> **Note**
>
> Amazon EMR Studio を使用する場合、サービス向上の目的で、使用状況とコンテンツに関するデータを保存する AWS 場合があります。データ共有をオプトアウトする方法の詳細と手順については、「Amazon CodeWhisperer ユーザーガイド」の「 とのデータの共有 AWS」を参照してください。

Workspace で を使用する際 CodeWhispererの考慮事項

- CodeWhisperer 統合は、 AWS リージョン 「EMR Studio の考慮事項」に記載されているように、EMR Studio が利用可能な と同じ で使用できます。
- Amazon EMR Studio は、スタジオがあるリージョンに関係なく、米国東部 (バージニア北部) (us-east-1) の CodeWhisperer エンドポイントをレコメンデーションに自動的に使用します。
- CodeWhisperer は、EMR Studio の Spark ジョブの ETL スクリプトをコーディングするための Python 言語のみをサポートします。
- クライアント側のテレメトリオプションは、 の使用を定量化します CodeWhisperer。この機能は EMR Studio ではサポートされていません。

に必要なアクセス許可 CodeWhisperer

を使用するには CodeWhisperer、Amazon EMR Studio の IAM ユーザーロールに次のポリシーをアタッチする必要があります。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
          "Sid": "CodeWhispererPermissions",
          "Effect": "Allow",
          "Action": [ "codewhisperer:GenerateRecommendations" ],
          "Resource": "*"
        }
     ]
}
```

Workspace CodeWhisperer で を使用する

で CodeWhisperer 参照ログを表示するには JupyterLab、 JupyterLab ウィンドウの下部にあるCodeWhispererパネルを開き、Open Code Reference Log を選択します。

次のリストには、提案を操作する CodeWhispererために使用できるショートカットが含まれています。

- レコメンデーションの一時停止 – CodeWhisperer 設定から自動サジェストの一時停止を使用します。
- 提案を承認する - キーボードの [Tab] キーを押します。
- 提案を拒否する - キーボードの [Esc] キーを押します。
- 提案のナビゲーション - キーボードの [上矢印] と [下矢印] を使います。
- 手動呼び出し - キーボードの [Alt] キーと [C] キーを押します。Mac を使用している場合は、[Cmd] キーと [C] キーを押します。

を使用して CodeWhisperer 、ログレベルなどの設定を変更したり、コード参照の候補を取得したりすることもできます。詳細については、「Amazon ユーザーガイド CodeWhisperer 」の「 JupyterLabと の機能のセットアップ」を参照してください。 CodeWhisperer

EMR Studio でアプリケーションとジョブをデバッグする

Amazon EMR Studio では、データアプリケーションインターフェイスを起動して、ブラウザでアプリケーションとジョブの実行を分析できます。

Amazon EMR コンソールから、EC2 クラスターで実行されている Amazon EMR の永続的なクラスター外のユーザーインターフェイスを起動することもできます。詳細については、「永続アプリケーションユーザーインターフェイスの表示」を参照してください。

> **Note**
>
> ブラウザの設定によっては、アプリケーション UI のポップアップを開くことができるようにする必要がある場合があります。

アプリケーションインターフェイスの設定と使用の詳細については、「The YARN Timeline Server」、「Monitoring and instrumentation、または「Tez UI overview」を参照してください。

Amazon EC2 ジョブで実行中の Amazon EMR をデバッグする

Workspace UI

ノートブックファイルからクラスタ上の UI を起動する

Amazon EMR リリースバージョン 5.33.0 以降を使用している場合、Workspace のノートブックから Spark ウェブユーザーインターフェイス (Spark UI または Spark History Server) を起動できます。

クラスター上の UIs、 PySpark、Spark、または SparkR カーネルで動作します。Spark イベントログまたはコンテナログの最大表示可能ファイルサイズは 10 MB です。ログファイルが 10 MB を超える場合は、ジョブをデバッグするためにクラスタ上の Spark UI ではなく、永続的な Spark History Server を使用することをお勧めします。

> **Important**
>
> EMR Studio が Workspace からクラスター上のアプリケーションユーザーインターフェイスを起動するには、クラスターが Amazon API Gateway と通信できる必要があります。Amazon API Gateway への発信ネットワークトラフィックを許可するように EMR クラスターを設定し、Amazon API Gateway がクラスターから到達可能であることを確認する必要があります。
> Spark UI は、ホスト名を解決してコンテナログにアクセスします。カスタムドメイン名を使用する場合は、クラスターノードのホスト名が Amazon DNS または指定した DNS サーバーによって解決できることを確認する必要があります。これを行うには、クラスターに関連付けられている Amazon Virtual Private Cloud (VPC) の Dynamic Host Configuration Protocol (DHCP) オプションを設定します。DHCP オプションの詳細については、「Amazon Virtual Private Cloud ユーザーガイド」の「DHCP オプションセット」を参照してください。

1. EMR Studio で、使用する Workspace を開き、EC2 で実行されている Amazon EMR クラスターにアタッチされていることを確認します。手順については、「EMR Studio Workspace にコンピューティングをアタッチする」を参照してください。
2. ノートブックファイルを開き PySpark、、Spark、または SparkR カーネルを使用します。カーネルを選択するには、ノートブックツールバーの右上にあるカーネル名を選択し、[Select Kernel] (カーネルの選択) ダイアログボックスを開きます。カーネルが選択されていない場合、名前は [No Kernel!] (カーネルなし) として表示されます。
3. ノートブックコードを実行します。Spark コンテキストを開始すると、ノートブックに次のような出力が表示されます。表示されるまでに数秒かかることがあります。Spark コンテキストを開始した場合は、`%%info` コマンドを実行して、Spark UI へのリンクにいつでもアクセスできます。

> **Note**
>
> Spark UI リンクが機能しない、または数秒経っても表示されない場合は、新しいノートブックセルを作成し、`%%info` コマンドを実行してリンクを再生成します。

4. Spark UI を起動するには、[Spark UI] の [Link] (リンク) を選択します。Spark アプリケーションが実行されている場合、Spark UI が新しいタブで開きます。アプリケーションが完了している場合、代わりに Spark History Server が開きます。

   Spark UI を起動したら、ブラウザの URL を変更して YARN ResourceManager または Yarn Timeline Server を開くことができます。`amazonaws.com` の後に次のパスのいずれかを追加します。

Web UI	パス	変更後の URL の例
YARN ResourceManager	/rm	https://*j-examplebby5ij* .emrappui-prod.*eu-west-1* .amazonaws.com/*rm*
Yarn Timeline Server	/yts	https://*j-examplebby5ij* .emrappui-prod.*eu-west-1* .amazonaws.com/*yts*
Spark History Server	/shs	https://*j-examplebby5ij* .emrappui-prod.*eu-west-1* .amazonaws.com/*shs*

Studio UI

EMR Studio UI から永続的な YARN Timeline Server、Spark History Server、または Tez UI を起動する

1. EMR Studio で、ページの左側にある [Amazon EMR on EC2] を選択して、[Amazon EMR on EC2] クラスターリストを開きます。
2. 検索ボックスに値を入力して、名前、状態、または ID でクラスターのリストをフィルタリングします。作成の時間範囲で検索することもできます。
3. クラスターを選択し、[Launch application UIs] (アプリケーション UI を起動する) を選択して、アプリケーションユーザーインターフェイスを選択します。アプリケーション UI が新しいブラウザタブで開き、ロードにしばらく時間がかかる場合があります。

EMR Serverless で実行されている EMR Studio をデバッグする

Amazon EC2 で実行される Amazon EMR と同様に、Workspace ユーザーインターフェイスを使用して EMR Serverless アプリケーションを分析できます。Workspace UI では、Amazon EMR リリースバージョン 6.14.0 以降を使用している場合、Workspace のノートブックから Spark ウェブユーザーインターフェイス (Spark UI または Spark History Server) を起動できます。また、Spark ドライバーログにすばやくアクセスできるように、ドライバーログへのリンクも用意されています。

Spark History Server を使用して Amazon EMR on EKS ジョブ実行をデバッグする

ジョブ実行を Amazon EMR on EKS クラスターに送信すると、Spark History Server を使用してそのジョブ実行のログにアクセスできます。Spark History Server は、スケジューラのステージとタスクのリスト、RDD サイズとメモリ使用量のサマリー、環境情報など、Spark アプリケーションをモニタリングするためのツールを提供します。Amazon EMR on EKS ジョブ実行用に Spark History Server を起動するには、次の方法があります。

- Amazon EMR on EKS マネジメントエンドポイントで EMR Studio を使用してジョブ実行を送信すると、Workspace のノートブックファイルから Spark History Server を起動できます。
- AWS CLI または AWS SDK for Amazon EMR on EKS を使用してジョブ実行を送信すると、EMR Studio UI から Spark History Server を起動できます。

Spark History Server を使用する方法については、Apache Spark ドキュメントの「Monitoring and Instrumentation」を参照してください。ジョブ実行の詳細については、「Amazon EMR on EKS 開発ガイド」の「Concepts and components」を参照してください。

EMR Studio Workspace のノートブックファイルから Spark History Server を起動するには

1. Amazon EMR on EKS クラスターに接続されている Workspace を開きます。
2. Workspace でノートブックファイルを選択して開きます。
3. ノートブックファイルの上部で [Spark UI] を選択して、永続的な Spark History Server を新しいタブで開きます。

EMR Studio UI から Spark History Server を起動するには

> **Note**
>
> EMR Studio UI のジョブリストには、 AWS CLI または AWS SDK for Amazon EMR on EKS を使用して送信したジョブ実行のみが表示されます。

1. EMR Studio のページの左側で [Amazon EMR on EKS] を選択します。
2. ジョブ実行の送信に使用した Amazon EMR on EKS 仮想クラスターを検索します。検索ボックスに値を入力して、状態または ID でクラスターのリストをフィルタリングできます。

3. クラスターを選択し、詳細ページを開きます。詳細ページには、ID、名前空間、ステータスなど、クラスターに関する情報が表示されます。このページには、そのクラスターに送信されたすべてのジョブ実行のリストも表示されます。
4. クラスターの詳細ページで、デバッグするジョブ実行を選択します。
5. [Jobs] (ジョブ) リストの右上で、[Launch Spark History Server] (Launch Spark History Server の起動) を選択して、新しいブラウザタブでアプリケーションインターフェイスを開きます。

EMR Studio Workspace にカーネルとライブラリをインストールする

各 Amazon EMR Studio Workspace には、プリインストールされたライブラリとカーネルのセットが付属しています。

Amazon EC2 で実行されるクラスターのカーネルとライブラリ

Amazon EC2 で実行されている EMR クラスターを使用する場合は、次の方法で EMR Studio の環境をカスタマイズすることもできます。

- クラスタープライマリノードに Jupyter Notebook カーネルと Python ライブラリをインストールする - このオプションを使用してライブラリをインストールすると、同じクラスターにアタッチされたすべての Workspace がそれらのライブラリを共有します。カーネルまたはライブラリは、ノートブックセル内から、または SSH を使用してクラスターのプライマリノードに接続されているときにインストールできます。
- ノートブックのスコープのライブラリを使用する - Workspace ユーザーがノートブックセル内からライブラリをインストールして使用する場合、それらのライブラリはそのノートブックのみで使用できます。このオプションを使用すると、ライブラリバージョンの競合を心配することなく、同じクラスターを使用するさまざまなノートブックを動作させることができます。

EMR Studio Workspace は、EMR Notebooks と同じ基盤アーキテクチャを持っています。EMR Notebooks の場合と同じ方法で、EMR Studio で Jupyter Notebook カーネルと Python ライブラリをインストールして使用できます。手順については、「カーネルとライブラリのインストールと使用」を参照してください。

Amazon EMR on EKS クラスター上のカーネルとライブラリ

Amazon EMR on EKS クラスターには、 PySpark および Python 3.7 カーネルと一連のプリインストールされたライブラリが含まれています。Amazon EMR on EKS は、追加のライブラリやクラスターのインストールをサポートしていません。

各 Amazon EMR on EKS クラスターには、次の Python と PySpark ライブラリがインストールされています。

- Python – boto3, cffi, future, ggplot, jupyter, kubernetes, matplotlib, numpy, pandas, plotly, pycryptodomex, py4j, requests, scikit-learn, scipy, seaborn
- PySpark – ggplot, jupyter, matplotlib, numpy, pandas, plotly, pycryptodomex, py4j, requests, scikit-learn, scipy, seaborn

EMR Serverless アプリケーションのカーネルとライブラリ

各 EMR Serverless アプリケーションには、次の Python と PySpark ライブラリがインストールされています。

- Python – ggplot, matplotlib, numpy, pandas, plotly, bokeh, scikit-learn, scipy, seaborn
- PySpark – ggplot, matplotlib,numpy, pandas, plotly, bokeh, scikit-learn, scipy, seaborn

magic コマンドでカーネルを強化する

概要

EMR Studio および EMR Notebooks は magic コマンドをサポートしています。Magic コマンド (magic) はデータの実行と分析に役立つように IPython カーネルによって提供されている拡張機能です。IPython は Python で構築されたインタラクティブなシェル環境です。

Amazon EMR はSparkmagic、Spark 関連のカーネル (PySpark、SparkR、Scala カーネル) に特定の magicコマンドを提供し、クラスターで Livy を使用して Spark ジョブを送信するパッケージであるもサポートしています。

EMR Notebooks に Python カーネルがある限り、magic コマンドを使用できます。同様に、Spark 関連のカーネルは Sparkmagic コマンドをサポートしています。

Magic コマンド (magic とも呼ばれる) には、次の 2 種類があります。

- ライン magic - これらの magic コマンドは単一の % プレフィックスで示され、1 行のコードで動作します。
- セル magic - これらの magic コマンドは二重の %% プレフィックスで示され、複数行のコードで動作します。

すべての利用可能な magic については「magic コマンドと Sparkmagic コマンドをリストする」を参照してください。

考慮事項と制約事項

- EMR Serverless は %%sh による spark-submit の実行をサポートしていません EMR Notebooks の magic はサポートしていません。
- Amazon EMR on EKS クラスターは、EMR Studio で Sparkmagic コマンドをサポートしていません。これは、マネージドエンドポイントで使用される Spark カーネルは Kubernetes に組み込まれており、Sparkmagic と Livy ではサポートされていないためです。次の例に示すように、回避策として Spark 設定を SparkContext オブジェクトに直接設定できます。

```
spark.conf.set("spark.driver.maxResultSize", '6g')
```

- 以下のmagicコマンドとアクションは、 では禁止されています AWS。
  - %alias
  - %alias_magic
  - %automagic
  - %macro
  - proxy_user を使用した %configure の変更
  - KERNEL_USERNAME または %env を使用した %set_env の変更

magic コマンドと Sparkmagic コマンドをリストする

使用可能な magic コマンドを一覧表示するには、次のコマンドを使用します。

- %lsmagic は、現在使用可能なすべての magic 関数をリストします。
- %%help は、Sparkmagic パッケージが提供する現在使用可能な Spark 関連の magic 関数をリストします。

%%configure を使用して Spark を設定する

Sparkmagic コマンドの中でも特に便利なコマンドの 1 つは、%%configure コマンドです。このコマンドは、セッション作成パラメータを設定します。conf 設定を使用すると、Apache Spark の設定ドキュメントで説明されている任意の Spark 設定を構成できます。

Example Maven リポジトリまたは Amazon S3 から EMR Notebooks に外部 JAR ファイルを追加する

次のアプローチを使用して、Sparkmagic でサポートされている任意の Spark 関連のカーネルに外部 JAR ファイル依存関係を追加できます。

```
%%configure -f
{"conf": {
    "spark.jars.packages": "com.jsuereth:scala-arm_2.11:2.0,ml.combust.bundle:bundle-
ml_2.11:0.13.0,com.databricks:dbutils-api_2.11:0.0.3",
    "spark.jars": "s3://DOC-EXAMPLE-BUCKET/my-jar.jar"
    }
}
```

Example : Hudi を設定する

ノートブック エディタで、Hudi を使用するように EMR Notebooks を設定できます。

```
%%configure
{ "conf": {
     "spark.jars": "hdfs://apps/hudi/lib/hudi-spark-bundle.jar,hdfs:///apps/hudi/lib/
spark-spark-avro.jar",
     "spark.serializer": "org.apache.spark.serializer.KryoSerializer",
     "spark.sql.hive.convertMetastoreParquet":"false"
     }
}
```

%%sh を使用して spark-submit を実行する

%%sh magic は、アタッチされたクラスターのインスタンスのサブプロセスでシェルコマンドを実行します。通常、Spark 関連のカーネルのいずれかを使用して、アタッチされたクラスターで Spark アプリケーションを実行します。ただし、Python カーネルを使用して Spark アプリケーションを送信する場合は、バケット名を小文字のバケット名に置き換えて、次の magic を使用できます。

```
%%sh
spark-submit --master yarn --deploy-mode cluster s3://DOC-EXAMPLE-BUCKET/test.py
```

この例では、クラスターは、s3://*DOC-EXAMPLE-BUCKET*/test.py の場所にアクセスできる必要があります。アクセスできないと、コマンドは失敗します。

%%sh magic を使用して任意の Linux コマンドを使用できます。Spark コマンドまたは YARN コマンドを実行する場合は、次のいずれかのオプションを使用して、emr-notebook Hadoop ユーザーを作成し、コマンドを実行するアクセス許可をユーザーに付与します。

- 次のコマンドを実行して、新しいユーザーを明示的に作成できます。

```
hadoop fs -mkdir /user/emr-notebook
hadoop fs -chown emr-notebook /user/emr-notebook
```

- Livy でユーザー偽装を有効にすることができます。これにより、ユーザーが自動的に作成されます。詳細については、「Spark ユーザーおよびジョブのアクティビティをモニタリングするためのユーザー偽装の有効化」を参照してください。

%%display を使用して Spark データフレームを視覚化する

%%display magic を使用して Spark データフレームを視覚化できます。この magic を使用するには、次のコマンドを実行します。

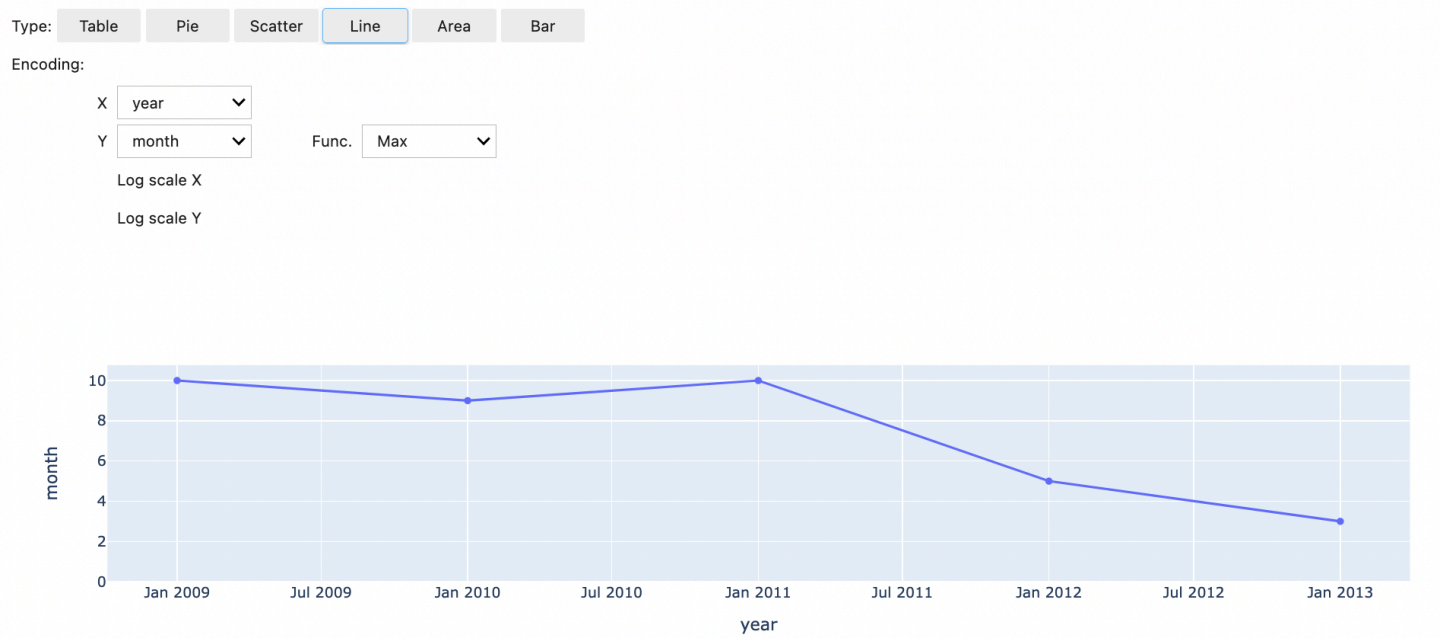
```
%%display df
```

以下の図のように、結果を表形式で表示することを選択します。

Type: Table Pie Scatter Line Area Bar

year	month	total_passengers	total_trips
2012-01-01	3	26866837	16146923
2011-01-01	3	26091246	16066350
2013-01-01	3	26965079	15749228
2011-01-01	10	26287953	15707756
2009-01-01	10	26202049	15604551
2012-01-01	5	26278817	15567525
2011-01-01	5	25508952	15554868
2010-01-01	9	25533166	15540209
2010-01-01	5	26002858	15481351
2012-01-01	4	25900645	15477914

5 種類のグラフでデータを視覚化することもできます。オプションには、円グラフ、散布図、折れ線グラフ、面グラフ、棒グラフがあります。



EMR Notebooks magic を使用する

Amazon EMR は、Python3 および Spark ベースのカーネルで使用できる次の EMR Notebooks magic を提供しています。

- `%mount_workspace_dir` – Workspace 内の他のファイルからコードをインポートして実行できるように、Workspace ディレクトリをクラスターにマウントします。

> **Note**
>
> `%mount_workspace_dir` では、Python 3 カーネルのみがローカルファイルシステムにアクセスできます。Spark エグゼキューターは、このカーネルではマウントされたディレクトリにアクセスできません。

- `%umount_workspace_dir` – Workspace ディレクトリをクラスターからアンマウントします
- `%generate_s3_download_url` - Amazon S3 オブジェクトのノートブック出力に一時的なダウンロードリンクを生成します。

前提条件

EMR Notebooks magic をインストールする前に、次のタスクを完了します。

- クラスター EC2 インスタンスのサービスロール (EC2 インスタンスプロファイル) が Amazon S3 の読み取りアクセス権を備えていることを確認してください。`EMR_EC2_DefaultRole` と `AmazonElasticMapReduceforEC2Role` マネージドポリシーを使用すると、この要件が満たされます。カスタムロールまたはポリシーを使用する場合は、必要な S3 アクセス許可があることを確認してください。

> **Note**
>
> EMR Notebooks magic は、ノートブックユーザーとしてクラスター上で実行され、EC2 インスタンスプロファイルを使用して Amazon S3 と対話します。EMR クラスターに Workspace ディレクトリをマウントすると、そのクラスターにアタッチするアクセス許可を持つすべての Workspace および EMR Notebooks が、マウントされたディレクトリにアクセスできます。
> デフォルトでは、ディレクトリは読み取り専用としてマウントされます。`s3fs-fuse` および `goofys` は読み取り/書き込みマウントを許可しますが、読み取り/書き込みモードでディレクトリをマウントするようにマウントパラメータを変更しないことを強くお勧めし

ます。書き込みアクセスを許可すると、ディレクトリに加えられたすべての変更が S3 バケットに書き込まれます。誤って削除したり上書きしたりしないように、S3 バケットのバージョニングを有効にできます。詳細については、「S3 バケットでのバージョニングの使用」を参照してください。

- クラスターで次のいずれかのスクリプトを実行して、EMR Notebooks magic の依存関係をインストールします。スクリプトを実行するには、「カスタムブートストラップアクションの使用」を行うか、すでに実行中のクラスターがある場合は「Amazon EMR クラスターでのコマンドとスクリプトの実行」の手順に従うことができます。

  インストールする依存関係を選択できます。s3fs-Fuse と goofys はともに、Amazon S3 バケットをクラスター上のローカルファイルシステムとしてマウントできるようにする FUSE (Filesystem in Userspace) ツールです。`s3fs` ツールは POSIX に似たエクスペリエンスを提供します。`goofys` ツールは、POSIX 準拠のファイルシステムよりもパフォーマンスを優先する場合に適しています。

  Amazon EMR 7.x シリーズでは、EPEL リポジトリをサポートしていない Amazon Linux 2023 を使用します。Amazon EMR 7.x を実行している場合は、s3fs-fuse GitHub の指示に従って をインストールします`s3fs-fuse`。5.x または 6.x シリーズを使用する場合は、次のコマンドを使用して をインストールします`s3fs-fuse`。

```
#!/bin/sh

# Install the s3fs dependency for EMR Notebooks magics
sudo amazon-linux-extras install epel -y
sudo yum install s3fs-fuse -y
```

  または

```
#!/bin/sh

# Install the goofys dependency for EMR Notebooks magics
sudo wget https://github.com/kahing/goofys/releases/latest/download/goofys -P /usr/
bin/
sudo chmod ugo+x /usr/bin/goofys
```

EMR Notebooks magic をインストールする

> **Note**
>
> Amazon EMR リリース 6.0 から 6.9.0、および 5.0 から 5.36.0 では、`emr-notebooks-magics` パッケージのバージョン 0.2.0 以降のみが `%mount_workspace_dir` magic をサポートしています。

EMR Notebooks magic をインストールするには、次の手順を実行します。

1. ノートブックで以下のコマンドを実行して `emr-notebooks-magics` パッケージをインストールします。

```
%pip install boto3 --upgrade
%pip install botocore --upgrade
%pip install emr-notebooks-magics --upgrade
```

2. カーネルを再起動して EMR Notebooks の magic をロードします。
3. 次のコマンドを使用してインストールを検証します。これにより、`%mount_workspace_dir` の出力ヘルプテキストが表示されます。

```
%mount_workspace_dir?
```

`%mount_workspace_dir` を使用してWorkspace ディレクトリをマウントする

`%mount_workspace_dir` magic を使用すると、Workspace ディレクトリを EMR クラスターにマウントして、ディレクトリに保存されている他のファイル、モジュール、またはパッケージをインポートして実行できます。

次の例では、Workspace ディレクトリ全体をクラスターにマウントし、オプションの *`<--fuse-type>`* 引数を指定してディレクトリのマウントに goofys を使用しています。

```
%mount_workspace_dir . <--fuse-type goofys>
```

Workspace ディレクトリがマウントされていることを確認するには、次の例のように、`ls` コマンドを使用して現在の作業ディレクトリを表示します。出力には、Workspace 内のすべてのファイルが表示されます。

```
%%sh
ls
```

Workspace での変更が完了したら、次のコマンドを使用して Workspace ディレクトリをアンマウントします。

> **Note**
>
> Workspace が停止またはデタッチされた場合でも、Workspace ディレクトリはクラスターにマウントされたままです。Workspace ディレクトリを明示的にアンマウントする必要があります。

```
%umount_workspace_dir
```

`%generate_s3_download_url` を使用して Amazon S3 オブジェクトをダウンロードする

`generate_s3_download_url` コマンドは、Amazon S3 に保存されたオブジェクトの署名済み URL を作成します。署名済み URL を使用して、オブジェクトをローカルマシンにダウンロードできます。例えば、`generate_s3_download_url` を実行して、コードが Amazon S3 に書き込む SQL クエリの結果をダウンロードできます。

署名済み URL は、デフォルトで 60 分間有効です。有効期限は、`--expires-in` フラグに秒数を指定することで変更できます。例えば、`--expires-in 1800` は、30 分間有効な URL を作成します。

次の例では、完全な Amazon S3 パス *`s3://EXAMPLE-DOC-BUCKET/path/to/my/object`* を指定して、オブジェクトのダウンロードリンクを生成しています。

```
%generate_s3_download_url s3://EXAMPLE-DOC-BUCKET/path/to/my/object
```

`generate_s3_download_url` の使用の詳細については、次のコマンドを実行してヘルプテキストを表示してください。

```
%generate_s3_download_url?
```

%execute_notebook を使用してノートブックをヘッドレスモードで実行する

%execute_notebook magic を使用すると、別のノートブックをヘッドレスモードで実行して、実行した各セルの出力を表示できます。この magic を使用するには、Amazon EMR と Amazon EC2 が共有するインスタンスロールに対する追加のアクセス許可が必要です。追加のアクセス許可を付与する方法の詳細については、%execute_notebook? コマンドを実行してください。

長時間かかるジョブの実行中、操作がないためにシステムがスリープ状態になったり、インターネット接続が一時的に切断されたりすることがあります。その結果、ブラウザと Jupyter Server 間の接続が中断される可能性があります。この場合、Jupyter Server から実行して送信したセルの出力が失われる可能性があります。

%execute_notebook magic を使用してノートブックをヘッドレスモードで実行すると、ローカルネットワークが中断した場合でも、EMR Notebooks は実行したセルからの出力をキャプチャします。EMR Notebooks は、実行したノートブックと同じ名前の新しいノートブックに出力を増分保存します。その後、EMR Notebooks はノートブックを Workspace 内の新しいフォルダに配置します。ヘッドレス実行は同じクラスターで行われ、EMR_Notebook_DefaultRole サービスロールを使用しますが、引数を追加することでデフォルト値を変更できます。

ノートブックをヘッドレスモードで実行するには、以下のコマンドを使用します。

```
%execute_notebook <relative-file-path>
```

ヘッドレス実行のクラスター ID とサービスロールを指定するには、以下のコマンドを使用します。

```
%execute_notebook <notebook_name>.ipynb --cluster-id <emr-cluster-id> --service-role
 <emr-notebook-service-role>
```

Amazon EMR と Amazon EC2 がインスタンスロールを共有する場合、ロールには以下の追加のアクセス許可が必要です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:StartNotebookExecution",
                "elasticmapreduce:DescribeNotebookExecution",
```

```
                "ec2:DescribeInstances"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::<AccoundId>:role/EMR_Notebooks_DefaultRole"
        }
    ]
}
```

> **Note**
>
> `%execute_notebook` magic を使用するには、バージョン 0.2.3 以降の `emr-notebooks-magics` パッケージをインストールします。

Spark カーネルで多言語ノートブックを使用する

Jupyter Notebook カーネルにはそれぞれデフォルト言語があります。例えば、Spark カーネルのデフォルト言語は Scala で、 PySpark カーネルのデフォルト言語は Python です。Amazon EMR 6.4.0 以降では、EMR Studio は多言語ノートブックをサポートしています。つまり、EMR Studio の各カーネルは、デフォルト言語に加えて、Python、Spark、R、および Spark SQL の各言語をサポートできます。

この機能を有効にするには、セルの先頭に次のいずれかの magic コマンドを指定します。

[言語]	Command
Python	`%%pyspark`
Scala	`%%scalaspark`
R	`%%rspark` EMR Serverless を使用したインタラクティブワークロードではサポートされません。

[言語]	Command
Spark SQL	`%%sql`

これらのコマンドを呼び出すと、対応する言語のインタプリタを使用して、同じ Spark セッション内のセル全体が実行されます。

`%%pyspark` セルmagicを使用すると、ユーザーはすべての Spark カーネルに PySpark コードを記述できます。

```
%%pyspark
a = 1
```

`%%sql` セル magic を使用すると、ユーザーはすべての Spark カーネルで Spark-SQL コードを実行できます。

```
%%sql
SHOW TABLES
```

`%%rspark` セル magic を使用すると、ユーザーはすべての Spark カーネルで SparkR コードを実行できます。

```
%%rspark
a <- 1
```

`%%scalaspark` セル magic を使用すると、ユーザーはすべての Spark カーネルで Spark Scala コードを実行できます。

```
%%scalaspark
val a = 1
```

一時テーブルを使用して言語インタプリタ間でデータを共有する

一時テーブルを使用して、言語インタプリタ間でデータを共有することもできます。次の例では、1 つのセルで `%%pyspark` を使用して Python で一時テーブルを作成し、次のセルで `%%scalaspark` を使用して Scala でそのテーブルからデータを読み込んでいます。

```
%%pyspark
```

```
df=spark.sql("SELECT * from nyc_top_trips_report LIMIT 20")
# create a temporary table called nyc_top_trips_report_view in python
df.createOrReplaceTempView("nyc_top_trips_report_view")
```

```
%%scalaspark
// read the temp table in scala
val df=spark.sql("SELECT * from nyc_top_trips_report_view")
df.show(5)
```

Amazon EMR Notebooks の概要

> **Note**
>
> EMR Notebooks は、コンソールで EMR Studio Workspace として使用できます。コンソールの「ワークスペースの作成」ボタンを使用すると、新しいノートブックを作成できます。EMR Notebooks ユーザーが Workspace にアクセスしたり作成したりするには、追加の IAM ロール権限が必要です。詳細については、「Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console」および「Amazon EMR console」を参照してください。

Amazon EMR Notebooks と Apache Spark を実行している Amazon EMR クラスターを使用して、Amazon EMR コンソール内で Jupyter Notebook と JupyterLabインターフェイスを作成して開くことができます。EMR Notebooks は、クエリとコードを実行するために使用できる「サーバーレス」のノートブックです。従来のノートブックとは異なり、EMR Notebooks の内容 (ノートブックセル内の方程式、クエリ、モデル、コード、説明テキスト) はクライアントで実行されます。コマンドは EMR クラスター上のカーネルを使用して実行されます。ノートブックの内容は、耐久性と柔軟な再利用のために、クラスターのデータとは別に Amazon S3 にも保存されます。

クラスターを起動して EMR notebooks を分析のためにアタッチしたら、クラスターを終了することができます。また、実行中のクラスターにアタッチされているノートブックを閉じて、別のものに切り替えることもできます。複数のユーザーがノートブックを同じクラスターに同時にアタッチして、Amazon S3 でノートブックファイルを互いに共有することができます。これらの機能を使用すると、クラスターをオンデマンドで実行してコストを削減し、別のクラスターやデータセットにノートブックを再設定する時間を節約できます。

また、Amazon EMR コンソールを使用することなく、Amazon EMR API を使用してプログラムで EMR Notebooks を実行することもできます (「ヘッドレス実行」)。EMR notebooks に、parameters タグを持つセルを含める必要があります。このセルにより、スクリプトで新しい入力値をノートブックに渡すことが可能になります。パラメータ化されたノートブックは、異なる入力値セットで再利用できます。新しい入力値で編集して実行するために、同じノートブックのコピーを作成する必要はありません。Amazon EMR は、パラメータ化されたノートブックの実行ごとに、S3 に出力ノートブックを作成して保存します。EMR notebooks の API コードの例については、「EMR Notebooks をプログラムで実行するサンプルコマンド」を参照してください。

> **Important**
>
> EMR Notebooks 機能は、Amazon EMR リリース 5.18.0 以降を使用するクラスターをサポートします。Amazon EMR の最新バージョン、または 5.30.0、5.32.0、または 6.2.0 以上を使用するクラスターでは EMR Notebooks を使用することをお勧めします。これらのリリースでは、Jupyter カーネルが Jupyter インスタンスではなくアタッチされたクラスターで実行されます。これにより、パフォーマンスが向上し、カーネルとライブラリのカスタマイズ機能が強化されています。詳細については、「クラスターのリリースバージョンによる機能の違い」を参照してください。

Amazon S3 ストレージの対象料金と Amazon EMR クラスターの対象料金が適用されます。

Amazon EMR Notebooks がコンソールで Amazon EMR Studio Workspace として利用可能に

EMR Notebooks から Workspace への移行

新しい Amazon EMR コンソールでは、EMR Notebooks と Amazon EMR Studio Workspace が統合され、1 つのエクスペリエンスになりました。EMR Studio を使用すると、さまざまな Workspace を作成して設定し、ノートブックを整理して実行できます。Amazon EMR Notebooks を古いコンソールで使用していた場合、新しいコンソールでは EMR Studio Workspace として使用できます。

Amazon EMR は、これらの新しい EMR Studio Workspace をお客様のために作成しました。作成した Studio の数は、EMR Notebooks から使用する個別の VPC の数に対応しています。例えば、EMR Notebooks から 2 つの異なる VPC の EMR クラスターに接続する場合、新しい EMR Studio が 2 つ作成されました。ノートブックは新しい Studio に分散されます。

> **Important**
>
> 古い Amazon EMR コンソールで新しいノートブックを作成するオプションをオフにしました。代わりに、新しい Amazon EMR コンソールの [ワークスペースの作成] を使用します。

Amazon EMR Studio Workspace の詳細については、「Workspace の基本の説明」を参照してください。EMR Studio の概念的な概要については、「Amazon EMR Studio の仕組み」ページの「ワークスペース」を参照してください。

目的

古いコンソールでも既存のノートブックを使用できますが、代わりに Amazon EMR Studio Workspace を新しいコンソールで使用することをお勧めします。EMR Notebooks では利用できない EMR Studio の機能を有効にするには、追加のロール権限を設定する必要があります。

> **Note**
>
> 少なくとも、既存の EMR Notebooks を EMR Studio Workspace として表示したり、新しい Workspace を作成したりするには、ユーザーが自分のロールに対する `elasticmapreduce:ListStudios` および `elasticmapreduce:CreateStudioPresignedUrl` 権限を持っている必要があります。EMR Studio のすべての機能にアクセスするには、「EMR Notebooks ユーザー向けの EMR Studio 機能の有効化」を参照して、EMR Notebooks ユーザーが必要とする追加権限の完全なリストをご覧ください。

EMR Notebooks を超えた EMR Studio の機能強化

Amazon EMR Studio を使用すると、EMR Notebooks では利用できない以下の機能をセットアップして使用できます。

- Jupyterlab 内から EMR クラスターを参照してアタッチする
- Jupyterlab 内から EMR Notebooks の仮想クラスターを参照してアタッチする
- Jupyterlab 内から Git リポジトリに接続する
- チームの他のメンバーと協力してノートブックコードを記述して実行する
- SQL Explorer でデータをブラウズする
- Service Catalog で EMR クラスターをプロビジョニングする

Amazon EMR Studio の機能の完全なリストについては、「EMR Studio の主な機能」を参照してください。

EMR Notebooks ユーザー向けの EMR Studio 機能の有効化

この統合の一環として作成する新しい EMR Studio では、既存の `EMR_Notebooks_DefaultRole` IAM ロールを EMR Studio サービスロールとして使用します。

EMR Notebooks から EMR Studio に移行し、EMR Studio の追加機能を使用するユーザーには、いくつかの新しいロール権限が必要です。EMR Studio を使用する予定の EMR Notebooks ユーザーのロールに以下の権限を追加します。

> **Note**
>
> 少なくとも、既存の EMR Notebooks を EMR Studio Workspace として表示したり、新しい Workspace を作成したりするには、ユーザーが自分のロールに対する `elasticmapreduce:ListStudios` および `elasticmapreduce:CreateStudioPresignedUrl` 権限を持っている必要があります。EMR Studio のすべての機能を使用するには、以下に示す権限をすべて追加します。管理者ユーザーには、EMR Studio を作成および管理する権限も必要です。詳細については、「EMR Studio を作成および管理するための管理者のアクセス許可」を参照してください。

```
"elasticmapreduce:DescribeStudio",
"elasticmapreduce:ListStudios",
"elasticmapreduce:CreateStudioPresignedUrl",
"elasticmapreduce:UpdateEditor",
"elasticmapreduce:PutWorkspaceAccess",
"elasticmapreduce:DeleteWorkspaceAccess",
"elasticmapreduce:ListWorkspaceAccessIdentities",
"emr-containers:ListVirtualClusters",
"emr-containers:DescribeVirtualCluster",
"emr-containers:ListManagedEndpoints",
"emr-containers:DescribeManagedEndpoint",
"emr-containers:CreateAccessTokenForManagedEndpoint",
"emr-containers:ListJobRuns",
"emr-containers:DescribeJobRun",
"servicecatalog:SearchProducts",
"servicecatalog:DescribeProduct",
"servicecatalog:DescribeProductView",
"servicecatalog:DescribeProvisioningParameters",
"servicecatalog:ProvisionProduct",
"servicecatalog:UpdateProvisionedProduct",
"servicecatalog:ListProvisioningArtifacts",
"servicecatalog:DescribeRecord",
"servicecatalog:ListLaunchPaths",
"cloudformation:DescribeStackResources"
```

EMR Studio のコラボレーション機能を使用するには以下の権限も必要ですが、EMR Notebooks では必要ありませんでした。

```
"sso-directory:SearchUsers",
"iam:GetUser",
"iam:GetRole",
"iam:ListUsers",
"iam:ListRoles",
"sso:GetManagedApplicationInstance"
```

EMR Notebooks を使用するときの考慮事項

> **Note**
>
> EMR Notebooks は、コンソールで EMR Studio Workspace として使用できます。コンソールの「ワークスペースの作成」ボタンを使用すると、新しいノートブックを作成できます。EMR Notebooks ユーザーが Workspace にアクセスしたり作成したりするには、追加の IAM ロール権限が必要です。詳細については、「Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console」および「Amazon EMR console」を参照してください。

EMR notebooks を使用してクラスターの作成およびソリューションの開発を行う場合は、次の要件を考慮してください。

クラスターの要件

- Amazon EMR のパブリックアクセスのブロックの有効化 - クラスターへのインバウンドアクセスを有効にすると、クラスターのユーザーがノートブックのカーネルを実行できてしまいます。許可されたユーザーのみがクラスターにアクセスできるようにしてください。パブリックアクセスのブロックを有効にし、インバウンドの SSH トラフィックを信頼できるソースのみに制限することを強くお勧めします。詳細については、「Amazon EMR のパブリックアクセスブロックの使用」および「セキュリティグループを使用してネットワークトラフィックを制御する」を参照してください。
- 互換性のあるクラスターの使用 - ノートブックにアタッチするクラスターは、以下の要件を満たしている必要があります。

- Amazon EMR を使用して作成されたクラスターのみがサポートされています。Amazon EMR でクラスターを個別に作成して EMR notebooks をアタッチするか、EMR notebooks の作成時に互換性のあるクラスターを作成することができます。
- Amazon EMR リリースバージョン 5.18.0 以降を使用して作成されたクラスターのみがサポートされています。the section called “クラスターのリリースバージョンによる機能の違い” を参照してください。
- AMD EPYC プロセッサ (例えば、m5a.* インスタンスタイプや r5a.* インスタンスタイプ) で Amazon EC2 インスタンスを使用して作成されたクラスターはサポートされていません。
- EMR Notebooks は、`VisibleToAllUsers` を `true` に設定して作成されたクラスターでのみ機能します。デフォルトでは、`VisibleToAllUsers` は `true` です。
- クラスターは EC2-VPC 内で起動する必要があります。パブリックサブネットとプライベートサブネットがサポートされています。EC2-Classic プラットフォームはサポートされません。
- クラスターは Hadoop、Spark、および Livy がインストールされている状態で起動する必要があります。その他のアプリケーションがインストールされる場合がありますが、EMR Notebooks では現在 Spark クラスターのみをサポートしています。

  > **Important**
  >
  > Amazon EMR リリースバージョン 5.32.0 以降、または 6.2.0 以降では、EMR Notebooks を使用するためには、クラスターで Jupyter Enterprise Gateway アプリケーションも実行されている必要があります。

- Kerberos 認証を使用するクラスターはサポートされていません。
- と統合されたクラスターは、ノートブックスコープのライブラリのインストールのみ AWS Lake Formation をサポートします。クラスターへのカーネルとライブラリのインストールはサポートされていません。
- 複数のプライマリノードを持つクラスターはサポートされていません。
- Graviton2 に基づく Amazon EC2 インスタンスを使用するクラスターはサポートされていません。 AWS Graviton2

クラスターのリリースバージョンによる機能の違い

EMR Notebooks は、Amazon EMR リリースバージョン 5.30.0、5.32.0 以降、または 6.2.0 以降を使用して作成されたクラスターで使用することを強くお勧めします。これらのバージョンでは、EMR Notebooks はアタッチされた Amazon EMR クラスターでカーネルを実行します。カーネルとライブ

ラリは、クラスターのプライマリノードに直接インストールすることができます。EMR Notebooks をこれらのクラスターバージョンで使用すると、以下の利点があります。

- パフォーマンスの向上 - ノートブックのカーネルは、選択した EC2 インスタンスタイプのクラスターで実行されます。以前のバージョンでは、サイズ変更、アクセス、カスタマイズできない特殊なインスタンスでカーネルが実行されていました。
- カーネルを追加およびカスタマイズする機能 - クラスターに接続して、conda および pip を使用してカーネルのパッケージをインストールすることができます。また、ノートブックのセルでのターミナルコマンドを使用した pip インストールもサポートされています。以前のバージョンでは、プリインストールされたカーネル (Python、、Spark PySpark、SparkR) のみが使用できました。詳細については、「クラスターのプライマリノードへのカーネルと Python ライブラリのインストール」を参照してください。
- Python ライブラリをインストールする機能 - conda および pip を使用して、クラスターのプライマリノードに Python ライブラリをインストールすることができます。conda を使用することをお勧めします。以前のバージョンでは、 のノートブックスコープのライブラリのみがサポート PySpark されています。

クラスターのリリースによってサポートされている EMR Notebooks の機能

クラスターリリースバージョン	のノートブックスコープのライブラリ PySpark	クラスターへのカーネルのインストール	プライマリノードへの Python ライブラリのインストール
5.18.0 より前	EMR Notebooks はサポートされていません		
5.18.0 ~ 5.25.0	いいえ	いいえ	いいえ
5.26.0 ~ 5.29.0	はい	いいえ	いいえ
5.30.0	はい	はい	はい
6.0.0	いいえ	いいえ	いいえ
5.32.0 以降、および 6.2.0 以降	はい	はい	はい

同時にアタッチする EMR Notebooks の制限

ノートブックをサポートしているクラスターを作成するときは、クラスタープライマリノードの EC2 インスタンスタイプを考慮してください。この EC2 インスタンスのメモリの制約によって、クラスターでコードとクエリを実行するために同時に準備できるノートブックの数が決まります。

プライマリノード EC2 インスタンスタイプ	EMR Notebooks の数
*.medium	2
*.large	4
*.xlarge	8
*.2xlarge	16
*.4xlarge	24
*.8xlarge	24
*.16xlarge	24

Jupyter Notebook と Python のバージョン

EMR Notebooks では、アタッチされたクラスターの Amazon EMR リリースバージョンに関係なく、Jupyter Notebook バージョン 6.0.2 および Python 3.6.5 が実行されます。

セキュリティに関する考慮事項

暗号化された S3 ロケーションを使用する

ノートブックファイルを保存するために Amazon S3 で暗号化された場所を指定する場合は、EMR Notebooks のサービスロール をキーユーザーとして設定する必要があります。デフォルトのサービスロールは `EMR_Notebooks_DefaultRole` です。暗号化に AWS KMS キーを使用している場合は、「 AWS Key Management Service デベロッパーガイド」のAWS 「KMS でのキーポリシーの使用」および「キーユーザーを追加するためのサポート記事」を参照してください。

ホスティングドメインでの Cookie の使用

Amazon EMR で使用するオフコンソールアプリケーションのセキュリティを強化するために、アプリケーションホスティングドメインはパブリックサフィックスリスト (PSL) に登録されます。これらのホスティングドメインの例には以下が含まれます: `emrstudio-prod.us-east-1.amazonaws.com`、`emrnotebooks-prod.us-east-1.amazonaws.com`、`emrappui-prod.us-east-1.amazonaws.com`セキュリティ強化のため、デフォルトのドメイン名に機密性の高い Cookie を設定する必要がある場合は、`__Host-` プレフィックスの付いた Cookie を使用することをお勧めします。これは、クロスサイトリクエストフォージェリ (CSRF) 攻撃からドメインを防ぐ際に役立ちます。詳細については、「Mozilla 開発者ネットワーク」の「Set-Cookie」ページを参照してください。

ノートブックの作成

> **Note**
>
> EMR Notebooks は、コンソールで EMR Studio Workspace として使用できます。コンソールの「ワークスペースの作成」ボタンを使用すると、新しいノートブックを作成できます。EMR Notebooks ユーザーが Workspace にアクセスしたり作成したりするには、追加の IAM ロール権限が必要です。詳細については、「Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console」および「Amazon EMR console」を参照してください。

古い Amazon EMR コンソールを使用して EMR Notebooks を作成します。 AWS CLI または Amazon EMR API を使用したノートブックの作成はサポートされていません。

EMR ノートブックを作成するには

1. Amazon EMR コンソール (https://console.aws.amazon.com/elasticmapreduce/) を開きます。
2. [Notebooks (ノートブック)]、[Create notebook (ノートブックの作成)] を選択します。
3. [Notebook name (ノートブック名)] とオプションで [Notebook description (ノートブックの説明)] を入力します。
4. ノートブックをアタッチするアクティブなクラスターがある場合は、デフォルトの [既存のクラスターを選択] を選択したままにし、[選択] をクリックしてリストからクラスターを選択してから、[クラスターを選択] をクリックします。EMR Notebooks のクラスター要件の詳細については、「EMR Notebooks を使用するときの考慮事項」を参照してください。

- または -

[クラスターを作成] を選択し、[クラスター名] にクラスター名を入力し、以下のガイドラインに従ってオプションを選択します。クラスターは、オンデマンドインスタンスを使用して、アカウントのデフォルトの VPC に作成されます。

設定	説明
クラスター名	クラスターを識別するために使用するわかりやすい名前。
[Release] (リリース)	変更できません。デフォルトは最新の Amazon EMR リリースバージョン (5.36.2) です。
アプリケーション	変更できません。クラスターにインストールされているアプリケーションの一覧が表示されます。
インスタンス	インスタンスの数を入力し、EC2 インスタンスタイプを選択します。1 つのインスタンスがプライマリノードに使用されます。残りはコアノードに使用されます。インスタンスタイプによって、クラスターに同時にアタッチできるノートブックの数が決まります。詳細については、「同時にアタッチする EMR Notebooks の制限」を参照してください。
EMR ロール	デフォルトのままにするか、Amazon EMR のカスタムサービスロールを指定するリンクを選択します。詳細については、「Amazon EMR のサービスロール (EMR ロール)」を参照してください。

設定	説明
EC2 インスタンスプロファイル	デフォルトのままにするか、EC2 インスタンスのカスタムサービスロールを指定するリンクを選択します。詳細については、「クラスター EC2 インスタンスのサービスロール (EC2 インスタンスプロファイル)」を参照してください。
EC2 キーペア	クラスターインスタンスに接続できるようにする EC2 キーペアを選択します。詳細については、「SSH を使用してプライマリノードに接続する」を参照してください。
自動終了	Amazon EMR バージョン 5.30.0 および 6.1.0 以降では、自動終了がサポートされます。 チェックボックスを選択して自動終了を有効にし、クラスターが自動的にシャットダウンするまでのアイドル時間を指定します。詳細については、「自動終了ポリシーを使用する」を参照してください。

5. [Security groups (セキュリティグループ)] で、[Use default security groups (デフォルトのセキュリティグループの使用)] を選択します。または、[セキュリティグループの選択] を選択して、クラスターの VPC で使用できるカスタムセキュリティグループを選択します。1 つはプライマリインスタンス用、もう 1 つはノートブックのクライアントインスタンス用に選択します。詳細については、「the section called “EMR Notebooks のセキュリティグループ”」を参照してください。
6. [AWS サービスロール] で、デフォルトのままにするか、リストからカスタムロールを選択します。ノートブックのクライアントインスタンスは、このロールを使用します。詳細については、「EMR Notebooks のサービスロール」を参照してください。
7. [ノートブックの場所] で、Amazon S3 内のノートブックファイルが保存される場所を選択するか、独自の場所を指定します。バケットとフォルダが存在しない場合は、Amazon EMR によって作成されます。

Amazon EMR によってフォルダ名ノートブック ID のフォルダが作成され、*NotebookName*.ipynb という名前のファイルにノートブックが保存されます。たとえば、MyFirstEMRManagedNotebook という名前のノートブックに対して Amazon S3 の場所 s3://MyBucket/MyNotebooks を指定した場合、ノートブックファイルは s3://MyBucket/MyNotebooks/*NotebookID*/MyFirstEMRManagedNotebook.ipynb に保存されます。

Amazon S3 内に暗号化された場所を指定する場合は、EMR Notebooks のサービスロール をキーユーザーとして設定する必要があります。デフォルトのサービスロールは EMR_Notebooks_DefaultRole です。暗号化に AWS KMS キーを使用している場合は、「 AWS Key Management Service デベロッパーガイド」のAWS 「KMS でのキーポリシーの使用」およびキーユーザーを追加するためのサポート記事を参照してください。

8. このノートブックに関連付ける Git ベースのリポジトリを Amazon EMR に追加している場合は、[Git リポジトリ] を選択し、[リポジトリを選択] を選択して、リストからリポジトリを選択します。詳細については、「Git ベースのリポジトリと EMR Notebooks の関連付け」を参照してください。
9. 必要に応じて、[Tags (タグ)] を選択して、ノートブックのキーと値のタグを追加します。

> **Important**
>
> Key 文字列が creatorUserID に設定されて値が IAM ユーザー ID に設定されたデフォルトタグが、アクセスの目的で適用されます。アクセスを制御するために使用できるため、このタグを変更または削除しないことをお勧めします。詳細については、「IAM ポリシーを持つクラスターとノートブックのタグをアクセスコントロールに使用する」を参照してください。

10. [ノートブックの作成] を選択します。

EMR Notebooks の使用

> **Note**
>
> EMR Notebooks は、コンソールで EMR Studio Workspace として使用できます。コンソールの「ワークスペースの作成」ボタンを使用すると、新しいノートブックを作成できます。EMR Notebooks ユーザーが Workspace にアクセスしたり作成したりするには、追加の

IAM ロール権限が必要です。詳細については、「Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console」および「Amazon EMR console」を参照してください。

EMR notebooks を作成すると、ノートブックはすぐに開始されます。[ノートブック] リストの [ステータス] に、[開始中] と表示されます。ステータスが [Ready (準備完了)] のときにノートブックを開くことができます。クラスターも一緒に作成した場合は、ノートブックが [Ready (準備完了)] になるまでの時間が少し長くなることがあります。

> **Tip**
>
> ノートブックのステータスを更新するには、ブラウザを更新するか、ノートブックリストの上にある更新アイコンを選択します。

ノートブックのステータスについて

EMR notebooks では、[ノートブック] リストの [ステータス] に以下が表示されます。

ステータス	意味
準備完了	ノートブックエディタを使用してノートブックを開くことができます。ノートブックが [Ready (準備完了)] ステータスの間に、停止または削除できます。クラスターを変更するには、最初にノートブックを停止する必要があります。[Ready (準備完了)] ステータスのノートブックが長時間アイドル状態の場合、自動的に停止されます。
スタート	ノートブックが作成されていて、クラスターにアタッチされます。ノートブックを開始している間に、ノートブックエディタを開いたり、停止、削除したり、クラスターを変更することはできません。
保留中	ノートブックが作成され、クラスターとの統合が完了するのを待機しています。クラスターで

ステータス	意味
	は、リソースのプロビジョニングや他のリクエストへの応答がまだ実行中である可能性があります。ローカルモードのノートブックを使用してノートブックエディタを開くことができます。クラスタープロセスに依存するすべてのコードは実行されず、失敗します。
停止中	ノートブックがシャットダウンしているか、ノートブックがアタッチされているクラスターが終了しています。ノートブックが停止している間に、ノートブックエディタを開いたり、停止、削除したり、クラスターを変更することはできません。
停止	ノートブックはシャットダウンしました。クラスターが実行されている限り、同じクラスターでノートブックを起動できません。クラスターを変更して、削除することができます。
[削除中]	使用可能なクラスターのリストからクラスターが削除されています。ノートブックファイル *NotebookName* .ipynb は Amazon S3 に残り、該当するストレージ料金が発生し続けます。

ノートブックエディタの使用

EMR ノートブックを使用する利点は、Jupyter でノートブックを起動することも、コンソール JupyterLab から直接起動することもできます。

EMR Notebooks では、Amazon EMR コンソールからアクセスするノートブックエディタが、使い慣れたオープンソースの Jupyter Notebook エディタまたは です JupyterLab。ノートブックエディタは Amazon EMR コンソール内で起動されるため、Amazon EMR クラスターでホストされているノートブックで起動するよりもアクセスの設定がより効率的です。SSH、セキュリティグループルール、およびプロキシ設定を通じたウェブアクセスのために、ユーザーのクライアントを設定する必要は

ありません。十分なアクセス許可のあるユーザーは、Amazon EMR コンソール内でノートブックエディタを簡単に開くことができます。

ただし、Amazon EMR 内から EMR notebooks を同時に開くことができるユーザーは 1 人のみです。すでに開いている EMR notebooks を別のユーザーが開こうとすると、エラーが発生します。

> **Important**
>
> Amazon EMR では、ノートブックエディタセッションごとに一意の署名付き URL が作成されます。この URL は、短期間だけ有効です。ノートブックエディタの URL を共有しないことをお勧めします。共有すると、URL の受取人がその有効期間中にノートブックの編集とノートブックコードの実行のためのアクセス許可を持つため、セキュリティ上のリスクが生じます。他のユーザーがノートブックにアクセスする必要がある場合は、アクセス許可ポリシーを使用してユーザーにアクセス許可を付与し、EMR Notebooks のサービスロールが Amazon S3 の場所にアクセスできるようにしてください。詳細については、「the section called “セキュリティ”」および「EMR Notebooks のサービスロール」を参照してください。

EMR notebooks のノートブックエディタを開くには

1. [ノートブック] リストから、[ステータス] が [使用可能] または [保留中] のノートブックを選択します。
2. Jupyter で Open in JupyterLab または Open を選択します。

   新しいブラウザタブが JupyterLab または Jupyter Notebook エディタに表示されます。
3. [Kernel (カーネル)] メニューで [Change kernel (カーネルの変更)] を選択し、プログラミング言語のカーネルを選択します。

   ノートブック内からコードを記述して実行する準備ができました。

ノートブックの内容の保存

ノートブックエディタで操作する際には、ノートブックセルと出力の内容は Amazon S3 内で定期的にノートブックファイルに自動で保存されます。前回セルが編集されたため変更がないノートブックには、エディタ内のノートブック名の横に [(autosaved) ((自動保存))] が表示されます。変更内容がまだ保存されていない場合、[unsaved changes (保存されていない変更)] が表示されます。

ノートブックを手動で保存できます。[ファイル] メニューで [Save and Checkpoint] (保存とチェックポイント) を選択するか、Ctrl+S キーを押します。これにより、Amazon S3 のノートブック

フォルダ内の [checkpoints] (チェックポイント) フォルダに *NotebookName*.ipynb という名前のファイルが作成されます。例えば s3://*MyBucket*/*MyNotebookFolder*/*NotebookID*/checkpoints/*NotebookName*.ipynb です。この場所には最新のチェックポイントファイルのみが保存されます。

クラスターの変更

ノートブック自体の内容を変更せずに、EMR notebooks がアタッチされたクラスターを変更できます。クラスターを変更できるのは、[Stopped (停止)] ステータスのノートブックのみです。

EMR notebooks のクラスターを変更するには

1. 変更するノートブックが実行されている場合は、[Notebooks (ノートブック)] リストから選択して [Stop (停止)] を選択します。
2. ノートブックのステータスが [Stopped (停止)] の場合、[Notebooks (ノートブック)] リストからノートブックを選択して、[View details (詳細を表示)] を選択します。
3. [Change cluster (クラスターの変更)] を選択します。
4. ノートブックをアタッチする Hadoop、Spark、Livy を実行しているアクティブなクラスターがある場合、デフォルトのままにして、リストからクラスターを選択します。要件を満たすクラスターのみが表示されます。

   - または -

   [Create a cluster (クラスターの作成)] を選択してクラスターのオプションを選択します。詳細については、「クラスターの要件」を参照してください。
5. [Security groups (セキュリティグループ)] のオプションを選択して、[Change cluster and start notebook (クラスターを変更してノートブックを開始)] を選択します。

ノートブックとノートブックファイルの削除

Amazon EMR コンソールを使用して EMR notebooks を削除すると、使用可能なノートブックのリストからノートブックが削除されます。ただし、ノートブックファイルは Amazon S3 に残り、ストレージ料金が発生し続けます。

ノートブックを削除して関連ファイルを削除するには

1. Amazon EMR コンソール (https://console.aws.amazon.com/elasticmapreduce/) を開きます。

2. [Notebooks (ノートブック)] を選択し、リストからノートブックを選択して、[View details (詳細を表示)] を選択します。
3. [Notebook location (ノートブックの場所)] の横にあるフォルダアイコンを選択して [URL] をコピーします。そのパターンは `s3://MyNotebookLocationPath/NotebookID/` です。
4. [削除] を選択します。

   ノートブックはリストから削除され、ノートブックの詳細は表示されなくなります。
5. 「Amazon Simple Storage Service ユーザーガイド」の「S3 バケットからフォルダを削除する方法」の手順に従います。ステップ 3 からバケットとフォルダに移動します。

   -または-

   AWS CLI がインストールされている場合は、コマンドプロンプトを開き、この段落の最後に コマンドを入力します。Amazon S3 の場所を上記でコピーした場所に置き換えます。 AWS CLI が Amazon S3 の場所を削除する権限を持つユーザーのアクセスキーで設定されていることを確認します。詳細については、「AWS CLIユーザーガイド」の「AWS Command Line Interface の設定。」を参照してください。

   ```
   aws s3 rm s3://MyNotebookLocationPath/NotebookID
   ```

ノートブックファイルの共有

各 EMR notebooks は、*NotebookName*.ipynb という名前のファイルとして Amazon S3 に保存されます。ノートブックファイルが、EMR Notebooks がベースにしている Jupyter Notebook の同じバージョンと互換性がある限り、ノートブックを EMR notebooks として開くことができます。

別のユーザーからノートブックファイルを開く最も簡単な方法は、別のユーザーからローカルファイルシステムに *.ipynb ファイルを保存し、Jupyter と JupyterLab エディタでアップロード機能を使用することです。

このプロセスを使用して、他のユーザーによって共有された EMR ノートブック、Jupyter コミュニティで共有されたノートブックを使用できます。また、まだノートブックファイルがあるときにコンソールから削除されたノートブックを復元することもできます。

EMR notebooks のベースとして別のノートブックファイルを使用するには

1. 先に進む前に、使用するすべてのノートブックのノートブックエディタを閉じて、EMR notebooks の場合はノートブックを停止します。

2. EMR notebooks を作成して名前を入力します。ノートブックに入力した名前は、置き換える必要があるファイルの名前になります。新しいファイル名は、このファイル名と正確に一致する必要があります。
3. ノートブック用に選択した Amazon S3 内の場所をメモしておきます。置き換えたファイルは、次のパターンのようなパスとファイル名を持つフォルダにあります。`s3://MyNotebookLocation/NotebookID/MyNotebookName.ipynb`
4. ノートブックを停止します。
5. まったく同じ名前を使用して、Amazon S3 の場所にある古いノートブックファイルを新しいものと置き換えます。

   Amazon S3 の次の AWS CLI コマンドは、EMR Notebook `SharedNotebook.ipynb` の というローカルマシンに保存されたファイルを、名前 MyNotebook、ID が でe-12A3BCDEFJHIJKLMNO45PQRST、Amazon S3 で`MyBucket/MyNotebooksFolder`指定された で作成されたファイルに置き換えます。Amazon S3 コンソールを使用してファイルをコピーおよび置き換える方法については、「Amazon Simple Storage Service ユーザーガイド」の「オブジェクトのアップロード、ダウンロード、管理」を参照してください。

   ```
   aws s3 cp SharedNotebook.ipynb s3://MyBucket/
   MyNotebooksFolder/-12A3BCDEFJHIJKLMNO45PQRST/MyNotebook.ipynb
   ```

EMR Notebooks をプログラムで実行するサンプルコマンド

> **Note**
>
> EMR Notebooks は、コンソールで EMR Studio Workspace として使用できます。コンソールの「ワークスペースの作成」ボタンを使用すると、新しいノートブックを作成できます。EMR Notebooks ユーザーが Workspace にアクセスしたり作成したりするには、追加の IAM ロール権限が必要です。詳細については、「Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console」および「Amazon EMR console」を参照してください。

概要

EMR Notebooks は、スクリプトまたはコマンドラインから実行 API で実行できます。 AWS コンソールの外部で EMR Notebooks の実行を開始、停止、一覧表示、および記述すると、EMR Notebooks をプログラムで制御できます。パラメータ化されたノートブックセルがあるノートブッ

クには、さまざまなパラメータ値を渡すことができます。これにより、パラメータ値の新しいセットごとにノートブックのコピーを作成する必要がなくなります。詳細については、「Amazon EMR API actions」を参照してください。

Amazon CloudWatch イベント および を使用して、EMR Notebooks の実行をスケジュールまたはバッチ処理できます AWS Lambda。詳細については、「Amazon Events AWS Lambda での の使用 CloudWatch 」を参照してください。

プログラムによる実行のロール権限

EMR Notebooks でプログラムによる実行を使用するには、以下のポリシーを使用してユーザー権限を設定する必要があります。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowExecutionActions",
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:StartNotebookExecution",
                "elasticmapreduce:DescribeNotebookExecution",
                "elasticmapreduce:ListNotebookExecutions"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowPassingServiceRole",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::account-id:role/EMR_Notebooks_DefaultRole"
        }
    ]
}
```

EMR Notebooks クラスターで EMR Notebooks をプログラムで実行する場合、以下の権限を追加する必要があります。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowRetrievingManagedEndpointCredentials",
            "Effect": "Allow",
            "Action": [
                "emr-containers:GetManagedEndpointSessionCredentials"
            ],
            "Resource": [
                "arn:aws:emr-containers:region:account-id:/virtualclusters/virtual-
cluster-id/endpoints/managed-endpoint-id"
            ],
            "Condition": {
                "StringEquals": {
                    "emr-containers:ExecutionRoleArn": [
                        "arn:aws:iam::account-id:role/emr-on-eks-execution-role"
                    ]
                }
            }
        },
        {
            "Sid": "AllowDescribingManagedEndpoint",
            "Effect": "Allow",
            "Action": [
                "emr-containers:DescribeManagedEndpoint"
            ],
            "Resource": [
                "arn:aws:emr-containers:region:account-id:/virtualclusters/virtual-
cluster-id/endpoints/managed-endpoint-id"
            ]
        }
    ]
}
```

プログラムによる実行の制限事項

- アカウント AWS リージョン ごとに最大 100 の同時実行がサポートされています。
- 30 日以上実行された場合、実行は終了します。
- Notebooks のプログラムによる実行は、Amazon EMR Serverless インタラクティブアプリケーションではサポートされていません。

プログラムによる EMR Notebooks の実行例

以下のセクションでは、、Boto3 SDK (Python) AWS CLI、および Ruby を使用したプログラムによる EMR Notebook 実行の例をいくつか紹介します。

- ノートブック実行の CLI コマンドの例
- ノートブック実行の Python サンプル
- ノートブック実行の Ruby サンプル

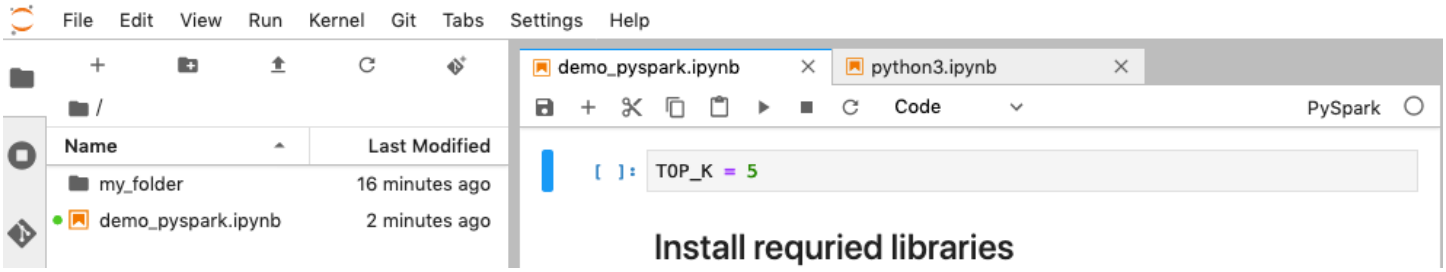
Apache Airflow や Amazon Managed Workflows for Apache Airflow (MWAA) などのオーケストレーションツールを使用して、スケジュールされたワークフローの一部としてパラメータ化されたノートブックを実行することもできます。詳細については、「AWS Big Data Blog」の「Orchestrating analytics jobs on EMR Notebooks using MWAA」を参照してください。

ノートブック実行の CLI コマンドの例

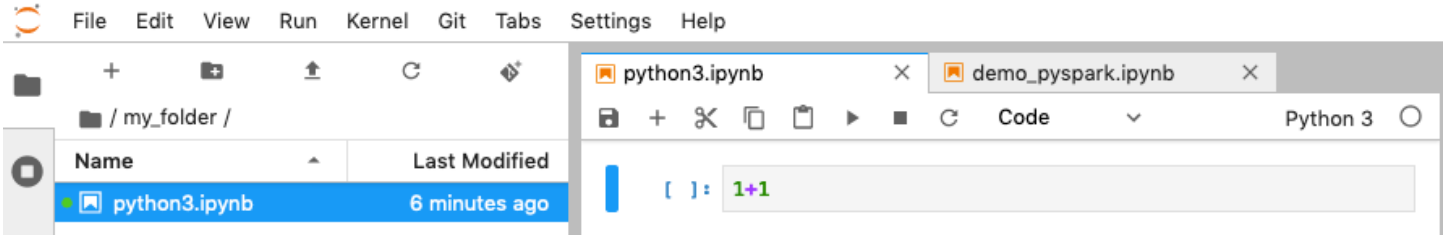
> **Note**
>
> EMR Notebooks は、コンソールで EMR Studio Workspace として使用できます。コンソールの「ワークスペースの作成」ボタンを使用すると、新しいノートブックを作成できます。EMR Notebooks ユーザーが Workspace にアクセスしたり作成したりするには、追加の IAM ロール権限が必要です。詳細については、「Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console」および「Amazon EMR console」を参照してください。

以下の例では、EMR Notebooks コンソールのデモノートブックを使用します。ノートブックを見つけるには、ホームディレクトリへの相対ファイルパスを使用します。この例では、`demo_pyspark.ipynb` と `my_folder/python3.ipynb` の 2 つのノートブックファイルを実行できます。

ファイル `demo_pyspark.ipynb` の相対パスは、以下に示す `demo_pyspark.ipynb` です。



python3.ipynb の相対パスは、以下に示す my_folder/python3.ipynb です。



Amazon EMR API NotebookExecution アクションの詳細については、「Amazon EMR API actions」を参照してください。

ノートブックを実行する

次の例で示すように、 を使用して start-notebook-executionアクションでノートブック AWS CLI を実行できます。

Example — Amazon EMR (Amazon EC2 上で実行されている) クラスターを使用して EMR Studio Workspace で EMR ノートブックを実行する

```
aws emr --region us-east-1 \
start-notebook-execution \
--editor-id e-ABCDEFG123456 \
--notebook-params '{"input_param":"my-value", "good_superhero":["superman", "batman"]}'
 \
--relative-path test.ipynb \
--notebook-execution-name my-execution \
--execution-engine '{"Id" : "j-1234ABCD123"}' \
--service-role EMR_Notebooks_DefaultRole

{
    "NotebookExecutionId": "ex-ABCDEFGHIJ1234ABCD"
}
```

Example — EMR Notebooks クラスターを使用する EMR Studio Workspace で EMR Notebooks を実行する

```
aws emr start-notebook-execution \
    --region us-east-1 \
    --service-role EMR_Notebooks_DefaultRole \
    --environment-variables '{"KERNEL_EXTRA_SPARK_OPTS": "--conf
 spark.executor.instances=1", "KERNEL_LAUNCH_TIMEOUT": "350"}' \
```

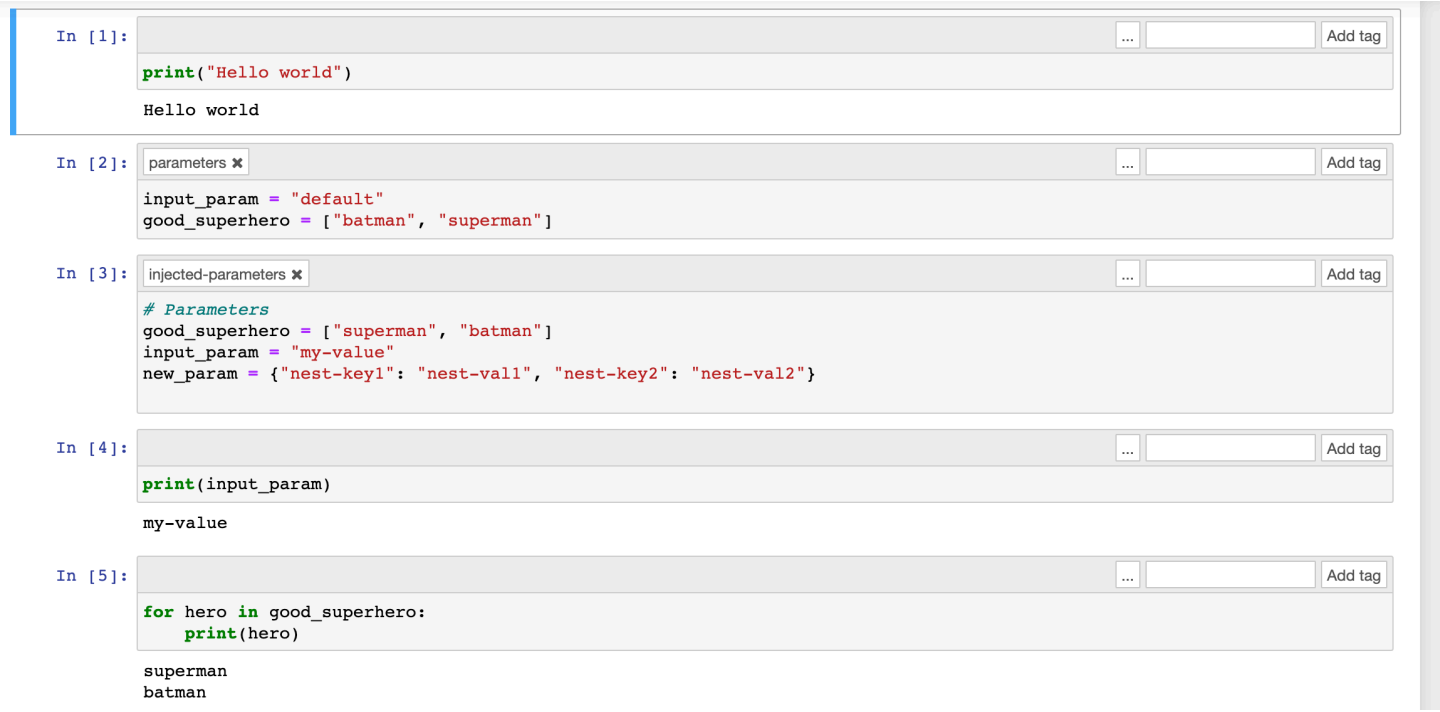
```
    --output-notebook-format HTML \
    --execution-engine Id=arn:aws:emr-containers:us-west-2:account-id:/
virtualclusters/ABCDEFG/
endpoints/ABCDEF,Type=EMR_ON_EKS,ExecutionRoleArn=arn:aws:iam::account-
id:role/execution-role \
    --editor-id e-ABCDEFG \
    --relative-path EMRonEKS-spark_python.ipynb
```

Example — Amazon S3 ロケーションを指定して EMR Notebooks を実行する

```
aws emr start-notebook-execution \
    --region us-east-1 \
    --notebook-execution-name my-execution-on-emr-on-eks-cluster \
    --service-role EMR_Notebooks_DefaultRole \
    --environment-variables '{"KERNEL_EXTRA_SPARK_OPTS": "--conf
 spark.executor.instances=1", "KERNEL_LAUNCH_TIMEOUT": "350"}' \
    --output-notebook-format HTML \
    --execution-engine Id=arn:aws:emr-containers:us-west-2:account-id:/
virtualclusters/ABCDEF/
endpoints/ABCDEF,Type=EMR_ON_EKS,ExecutionRoleArn=arn:aws:iam::account-
id:role/execution-role \
    --notebook-s3-location '{"Bucket": "your-s3-bucket","Key": "s3-prefix-to-notebook-
location/EMRonEKS-spark_python.ipynb"}' \
    --output-notebook-s3-location '{"Bucket": "your-s3-bucket","Key": "s3-prefix-for-
storing-output-notebook"}'
```

ノートブック出力

サンプルノートブックからの出力を以下に示します。セル 3 は、新しく挿入されたパラメータ値を示します。



ノートブックを記述する

describe-notebook-execution アクションを使用して、特定のノートブック実行に関する情報にアクセスできます。

```
aws emr --region us-east-1 \
describe-notebook-execution --notebook-execution-id ex-IZWZZVR9DKQ9WQ7VZWXJZR29UGHTE

{
    "NotebookExecution": {
        "NotebookExecutionId": "ex-IZWZZVR9DKQ9WQ7VZWXJZR29UGHTE",
        "EditorId": "e-BKTM2DIHXBEDRU44ANWRKIU8N",
        "ExecutionEngine": {
            "Id": "j-2QMOV6JAX1TS2",
            "Type": "EMR",
            "MasterInstanceSecurityGroupId": "sg-05ce12e58cd4f715e"
        },
        "NotebookExecutionName": "my-execution",
        "NotebookParams": "{\"input_param\":\"my-value\", \"good_superhero\":
[\"superman\", \"batman\"]}",
        "Status": "FINISHED",
        "StartTime": 1593490857.009,
        "Arn": "arn:aws:elasticmapreduce:us-east-1:123456789012:notebook-execution/ex-
IZWZZVR9DKQ9WQ7VZWXJZR29UGHTE",
```

```
        "LastStateChangeReason": "Execution is finished for cluster j-2QMOV6JAX1TS2.",
        "NotebookInstanceSecurityGroupId": "sg-0683b0a39966d4a6a",
        "Tags": []
    }
}
```

ノートブックを停止する

実行中のノートブックを停止するには、stop-notebook-execution コマンドを使用して停止できます。

```
# stop a running execution
aws emr --region us-east-1 \
stop-notebook-execution --notebook-execution-id ex-IZWZX78UVPAATC8LHJR129B1RBN4T


# describe it
aws emr --region us-east-1 \
describe-notebook-execution --notebook-execution-id ex-IZWZX78UVPAATC8LHJR129B1RBN4T

{
    "NotebookExecution": {
        "NotebookExecutionId": "ex-IZWZX78UVPAATC8LHJR129B1RBN4T",
        "EditorId": "e-BKTM2DIHXBEDRU44ANWRKIU8N",
        "ExecutionEngine": {
            "Id": "j-2QMOV6JAX1TS2",
            "Type": "EMR"
        },
        "NotebookExecutionName": "my-execution",
        "NotebookParams": "{\"input_param\":\"my-value\", \"good_superhero\":
[\"superman\", \"batman\"]}",
        "Status": "STOPPED",
        "StartTime": 1593490876.241,
        "Arn": "arn:aws:elasticmapreduce:us-east-1:123456789012:editor-execution/ex-
IZWZX78UVPAATC8LHJR129B1RBN4T",
        "LastStateChangeReason": "Execution is stopped for cluster j-2QMOV6JAX1TS2.
 Internal error",
        "Tags": []
    }
}
```

ノートブックの実行を開始時間別に一覧表示する

ノートブックの実行を開始時間別に一覧表示するには、list-notebook-executions に --from パラメータを渡します。

```
# filter by start time
aws emr --region us-east-1 \
list-notebook-executions --from 1593400000.000

{
    "NotebookExecutions": [
        {
            "NotebookExecutionId": "ex-IZWZX78UVPAATC8LHJR129B1RBN4T",
            "EditorId": "e-BKTM2DIHXBEDRU44ANWRKIU8N",
            "NotebookExecutionName": "my-execution",
            "Status": "STOPPED",
            "StartTime": 1593490876.241
        },
        {
            "NotebookExecutionId": "ex-IZWZZVR9DKQ9WQ7VZWXJZR29UGHTE",
            "EditorId": "e-BKTM2DIHXBEDRU44ANWRKIU8N",
            "NotebookExecutionName": "my-execution",
            "Status": "RUNNING",
            "StartTime": 1593490857.009
        },
        {
            "NotebookExecutionId": "ex-IZWZYRS0M14L5V95WZ9OQ399SKMNW",
            "EditorId": "e-BKTM2DIHXBEDRU44ANWRKIU8N",
            "NotebookExecutionName": "my-execution",
            "Status": "STOPPED",
            "StartTime": 1593490292.995
        },
        {
            "NotebookExecutionId": "ex-IZX009ZK83IVY5E33VH8MDMELVK8K",
            "EditorId": "e-BKTM2DIHXBEDRU44ANWRKIU8N",
            "NotebookExecutionName": "my-execution",
            "Status": "FINISHED",
            "StartTime": 1593489834.765
        },
        {
            "NotebookExecutionId": "ex-IZWZXOZF88JWDF9J09GJ91R57VI0N",
            "EditorId": "e-BKTM2DIHXBEDRU44ANWRKIU8N",
            "NotebookExecutionName": "my-execution",
```

```
            "Status": "FAILED",
            "StartTime": 1593488934.688
        }
    ]
}
```

ノートブックの実行を開始時間とステータス別に一覧表示する

list-notebook-executions コマンドでは、--status パラメータを使用して結果をフィルタリングすることもできます。

```
# filter by start time and status
aws emr --region us-east-1 \
list-notebook-executions --from 1593400000.000 --status FINISHED
{
    "NotebookExecutions": [
        {
            "NotebookExecutionId": "ex-IZWZZVR9DKQ9WQ7VZWXJZR29UGHTE",
            "EditorId": "e-BKTM2DIHXBEDRU44ANWRKIU8N",
            "NotebookExecutionName": "my-execution",
            "Status": "FINISHED",
            "StartTime": 1593490857.009
        },
        {
            "NotebookExecutionId": "ex-IZX009ZK83IVY5E33VH8MDMELVK8K",
            "EditorId": "e-BKTM2DIHXBEDRU44ANWRKIU8N",
            "NotebookExecutionName": "my-execution",
            "Status": "FINISHED",
            "StartTime": 1593489834.765
        }
    ]
}
```

ノートブック実行の Python サンプル

> **Note**
>
> EMR Notebooks は、コンソールで EMR Studio Workspace として使用できます。コンソールの「ワークスペースの作成」ボタンを使用すると、新しいノートブックを作成できます。EMR Notebooks ユーザーが Workspace にアクセスしたり作成したりするには、追加の

IAM ロール権限が必要です。詳細については、「Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console」および「Amazon EMR console」を参照してください。

次のコードサンプルは、ノートブック実行 API を示す demo.py という名前の SDK for Python (Boto3) ファイルです。

Amazon EMR API NotebookExecution アクションの詳細については、「Amazon EMR API actions」を参照してください。

```
import boto3,time

emr = boto3.client(
    'emr',
    region_name='us-west-1'
)

start_resp = emr.start_notebook_execution(
    EditorId='e-40AC8ZO6EGGCPJ4DLO48KGGGI',
    RelativePath='boto3_demo.ipynb',
    ExecutionEngine={'Id':'j-1HYZS6JQKV11Q'},
    ServiceRole='EMR_Notebooks_DefaultRole'
)

execution_id = start_resp["NotebookExecutionId"]
print(execution_id)
print("\n")

describe_response = emr.describe_notebook_execution(NotebookExecutionId=execution_id)

print(describe_response)
print("\n")

list_response = emr.list_notebook_executions()
print("Existing notebook executions:\n")
for execution in list_response['NotebookExecutions']:
    print(execution)
    print("\n")

print("Sleeping for 5 sec...")
time.sleep(5)

print("Stop execution " + execution_id)
```

```
emr.stop_notebook_execution(NotebookExecutionId=execution_id)
describe_response = emr.describe_notebook_execution(NotebookExecutionId=execution_id)
print(describe_response)
print("\n")
```

demo.py を実行したときの出力を以下に示します。

```
ex-IZX56YJDW1D29Q1PHR32WABU2SAPK

{'NotebookExecution': {'NotebookExecutionId': 'ex-IZX56YJDW1D29Q1PHR32WABU2SAPK',
 'EditorId': 'e-40AC8ZO6EGGCPJ4DLO48KGGGI', 'ExecutionEngine': {'Id':
 'j-1HYZS6JQKV11Q', 'Type': 'EMR'}, 'NotebookExecutionName': '', 'Status': 'STARTING',
 'StartTime': datetime.datetime(2020, 8, 19, 0, 49, 19, 418000, tzinfo=tzlocal()),
 'Arn': 'arn:aws:elasticmapreduce:us-west-1:123456789012:notebook-execution/ex-
IZX56YJDW1D29Q1PHR32WABU2SAPK', 'LastStateChangeReason': 'Execution is starting
 for cluster j-1HYZS6JQKV11Q.', 'Tags': []}, 'ResponseMetadata': {'RequestId':
 '70f12c5f-1dda-45b7-adf6-964987d373b7', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-
amzn-requestid': '70f12c5f-1dda-45b7-adf6-964987d373b7', 'content-type': 'application/
x-amz-json-1.1', 'content-length': '448', 'date': 'Wed, 19 Aug 2020 00:49:22 GMT'},
 'RetryAttempts': 0}}

Existing notebook executions:

{'NotebookExecutionId': 'ex-IZX56YJDW1D29Q1PHR32WABU2SAPK', 'EditorId':
 'e-40AC8ZO6EGGCPJ4DLO48KGGGI', 'NotebookExecutionName': '', 'Status': 'STARTING',
 'StartTime': datetime.datetime(2020, 8, 19, 0, 49, 19, 418000, tzinfo=tzlocal())}


{'NotebookExecutionId': 'ex-IZX5ABS5PR1E5AHMFYEMX3JJIORRB', 'EditorId':
 'e-40AC8ZO6EGGCPJ4DLO48KGGGI', 'NotebookExecutionName': '', 'Status': 'RUNNING',
 'StartTime': datetime.datetime(2020, 8, 19, 0, 48, 36, 373000, tzinfo=tzlocal())}


{'NotebookExecutionId': 'ex-IZX5GLVXIU1HNI8BWVW057F6MF4VE', 'EditorId':
 'e-40AC8ZO6EGGCPJ4DLO48KGGGI', 'NotebookExecutionName': '', 'Status': 'FINISHED',
 'StartTime': datetime.datetime(2020, 8, 19, 0, 45, 14, 646000, tzinfo=tzlocal()),
 'EndTime': datetime.datetime(2020, 8, 19, 0, 46, 26, 543000, tzinfo=tzlocal())}


{'NotebookExecutionId': 'ex-IZX5CV8YDUO8JAIWMXN2VH32RUIT1', 'EditorId':
 'e-40AC8ZO6EGGCPJ4DLO48KGGGI', 'NotebookExecutionName': '', 'Status': 'FINISHED',
```

```
 'StartTime': datetime.datetime(2020, 8, 19, 0, 43, 5, 807000, tzinfo=tzlocal()),
 'EndTime': datetime.datetime(2020, 8, 19, 0, 44, 31, 632000, tzinfo=tzlocal())}


{'NotebookExecutionId': 'ex-IZX5AS0PPW55CEDEURZ9NSOWSUJZ6', 'EditorId':
 'e-40AC8ZO6EGGCPJ4DLO48KGGGI', 'NotebookExecutionName': '', 'Status': 'FINISHED',
 'StartTime': datetime.datetime(2020, 8, 19, 0, 42, 29, 265000, tzinfo=tzlocal()),
 'EndTime': datetime.datetime(2020, 8, 19, 0, 43, 48, 320000, tzinfo=tzlocal())}


{'NotebookExecutionId': 'ex-IZX57YF5Q53BKWLR4I5QZ14HJ7DRS', 'EditorId':
 'e-40AC8ZO6EGGCPJ4DLO48KGGGI', 'NotebookExecutionName': '', 'Status': 'FINISHED',
 'StartTime': datetime.datetime(2020, 8, 19, 0, 38, 37, 81000, tzinfo=tzlocal()),
 'EndTime': datetime.datetime(2020, 8, 19, 0, 40, 39, 646000, tzinfo=tzlocal())}

Sleeping for 5 sec...
Stop execution ex-IZX56YJDW1D29Q1PHR32WABU2SAPK
{'NotebookExecution': {'NotebookExecutionId': 'ex-IZX56YJDW1D29Q1PHR32WABU2SAPK',
 'EditorId': 'e-40AC8ZO6EGGCPJ4DLO48KGGGI', 'ExecutionEngine': {'Id':
 'j-1HYZS6JQKV11Q', 'Type': 'EMR'}, 'NotebookExecutionName': '', 'Status': 'STOPPING',
 'StartTime': datetime.datetime(2020, 8, 19, 0, 49, 19, 418000, tzinfo=tzlocal()),
 'Arn': 'arn:aws:elasticmapreduce:us-west-1:123456789012:notebook-execution/ex-
IZX56YJDW1D29Q1PHR32WABU2SAPK', 'LastStateChangeReason': 'Execution is being stopped
 for cluster j-1HYZS6JQKV11Q.', 'Tags': []}, 'ResponseMetadata': {'RequestId':
 '2a77ef73-c1c6-467c-a1d1-7204ab2f6a53', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-
amzn-requestid': '2a77ef73-c1c6-467c-a1d1-7204ab2f6a53', 'content-type': 'application/
x-amz-json-1.1', 'content-length': '453', 'date': 'Wed, 19 Aug 2020 00:49:30 GMT'},
 'RetryAttempts': 0}}
```

ノートブック実行の Ruby サンプル

> **Note**
>
> EMR Notebooks は、コンソールで EMR Studio Workspace として使用できます。コンソールの「ワークスペースの作成」ボタンを使用すると、新しいノートブックを作成できます。EMR Notebooks ユーザーが Workspace にアクセスしたり作成したりするには、追加の IAM ロール権限が必要です。詳細については、「Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console」および「Amazon EMR console」を参照してください。

次に、ノートブック実行 API の使用方法を示す Ruby コードサンプルを示します。

```
# prepare an Amazon EMR client

emr = Aws::EMR::Client.new(
  region: 'us-east-1',
  access_key_id: 'AKIA...JKPKA',
  secret_access_key: 'rLMeu...vU0OLrAC1',
)
```

ノートブックの実行を開始し、実行 ID を取得する

この例では、Amazon S3 エディタと EMR Notebooks は `s3://mybucket/notebooks/e-EA8VGAA429FEQTC8HC9ZHWISK/test.ipynb` です。

Amazon EMR API `NotebookExecution` アクションの詳細については、「Amazon EMR API actions」を参照してください。

```
start_response = emr.start_notebook_execution({
    editor_id: "e-EA8VGAA429FEQTC8HC9ZHWISK",
    relative_path: "test.ipynb",

    execution_engine: {id: "j-3U82I95AMALGE"},

    service_role: "EMR_Notebooks_DefaultRole",
})


notebook_execution_id = start_resp.notebook_execution_id
```

ノートブックの実行を記述し、詳細を出力する

```
describe_resp = emr.describe_notebook_execution({
    notebook_execution_id: notebook_execution_id
})
puts describe_resp.notebook_execution
```

上記のコマンドの出力は以下のとおりです。

```
{
:notebook_execution_id=>"ex-IZX3VTVZWVWPP27KUB90BZ7V9IEDG",
:editor_id=>"e-EA8VGAA429FEQTC8HC9ZHWISK",
```

```
:execution_engine=>{:id=>"j-3U82I95AMALGE", :type=>"EMR", :master_instance_security_group_id=>n
:notebook_execution_name=>"",
:notebook_params=>nil,
:status=>"STARTING",
:start_time=>2020-07-23 15:07:07 -0700,
:end_time=>nil,
:arn=>"arn:aws:elasticmapreduce:us-east-1:123456789012:notebook-execution/ex-
IZX3VTVZWVWPP27KUB90BZ7V9IEDG",
:output_notebook_uri=>nil,
:last_state_change_reason=>"Execution is starting for cluster
 j-3U82I95AMALGE.", :notebook_instance_security_group_id=>nil,
:tags=>[]
}
```

ノートブックフィルター

```
"EditorId": "e-XXXX",           [Optional]
"From" : "1593400000.000",    [Optional]
"To" :
```

ノートブックの実行を停止する

```
stop_resp = emr.stop_notebook_execution({
    notebook_execution_id: notebook_execution_id
})
```

Spark ユーザーおよびジョブのアクティビティをモニタリングするためのユーザー偽装の有効化

> **Note**
>
> EMR Notebooks は、コンソールで EMR Studio Workspace として使用できます。コンソールの「ワークスペースの作成」ボタンを使用すると、新しいノートブックを作成できます。EMR Notebooks ユーザーが Workspace にアクセスしたり作成したりするには、追加の IAM ロール権限が必要です。詳細については、「Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console」および「Amazon EMR console」を参照してください。

EMR Notebooks では、Spark クラスターでユーザー偽装を設定することができます。この機能を使用すると、ノートブックエディタ内から開始されたジョブのアクティビティを追跡できます。また、EMR Notebooks には組み込みの Jupyter Notebook ウィジェットがあり、Spark のジョブの詳細とクエリの出力をノートブックエディタで確認できます。ウィジェットはデフォルトで使用可能で、特別な設定は必要ありません。ただし、履歴サーバーを表示するには、プライマリノードにホストされている Amazon EMR ウェブインターフェイスを表示するようにクライアントを設定する必要があります。

Spark ユーザー偽装の設定

デフォルトでは、ノートブックエディタを使用してユーザーが送信する Spark ジョブは、不明瞭な `livy` ユーザー ID から発生しているように見えます。クラスターにユーザー偽装を設定して、代わりに、それらのジョブをコードを実行したユーザー ID に関連付けることができます。プライマリノードの HDFS ユーザーディレクトリは、ノートブックでコードを実行する各ユーザー ID に対して作成されます。例えば、ユーザー `NbUser1` がノートブックエディタからコードを実行する場合、プライマリノードに接続して、`hadoop fs -ls /user` がディレクトリ `/user/user_NbUser1` を示していることを確認できます。

`core-site` および `livy-conf` の設定分類でプロパティを設定して、この機能を有効にします。この機能は、Amazon EMR でクラスターと共にノートブックを作成する場合、デフォルトでは使用できません。アプリケーションをカスタマイズするための設定分類の使用の詳細については、「Amazon EMR リリース ガイド」の「アプリケーションの設定」を参照してください。

EMR Notebooks のユーザー偽装を有効にするには、次の設定分類と値を使用します。

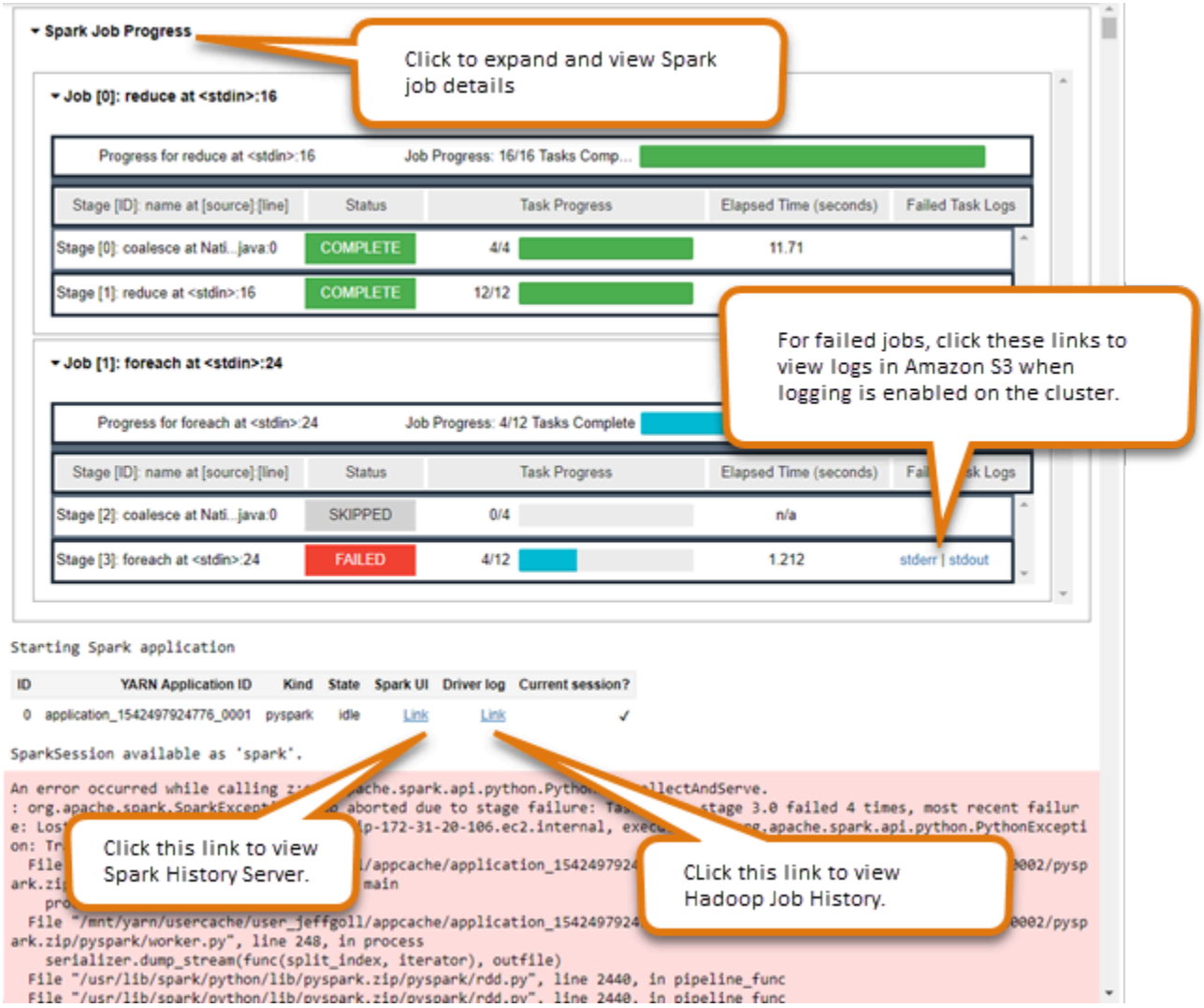
```
[
    {
        "Classification": "core-site",
        "Properties": {
          "hadoop.proxyuser.livy.groups": "*",
          "hadoop.proxyuser.livy.hosts": "*"
        }
    },
    {
        "Classification": "livy-conf",
        "Properties": {
          "livy.impersonation.enabled": "true"
        }
    }
]
```

Spark ジョブモニタリングウィジェットの使用

EMR クラスター上で Spark ジョブを実行するノートブックエディタでコードを実行する場合、出力には Spark ジョブモニタリングの Jupyter Notebook が含まれます。ウィジェットには、ジョブの詳細と、Spark の履歴サーバーページや Hadoop のジョブ履歴ページへの便利なリンクの他に、失敗したジョブに関する Amazon S3 内のジョブログへの便利なリンクがあります。

クラスタープライマリノードで履歴サーバーのページを表示するには、必要に応じて SSH クライアントとプロキシをセットアップする必要があります。詳細については、「Amazon EMR クラスターでホストされているウェブインターフェイスを表示する」を参照してください。Amazon S3 内のログを表示するには、クラスターのログ記録を有効にする必要があります。この設定は新しいクラスターではデフォルトです。詳細については、「Amazon S3 にアーカイブされたログファイルを表示する」を参照してください。

以下は Spark ジョブモニタリングの例です。



EMR Notebooks のセキュリティとアクセスコントロール

> **Note**
>
> EMR Notebooks は、コンソールで EMR Studio Workspace として使用できます。コンソールの「ワークスペースの作成」ボタンを使用すると、新しいノートブックを作成できます。EMR Notebooks ユーザーが Workspace にアクセスしたり作成したりするには、追加の IAM ロール権限が必要です。詳細については、「Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console」および「Amazon EMR console」を参照してください。

EMR Notebooks のセキュリティ体制を調整するのに役立つ機能がいくつかあります。これにより、承認されたユーザーのみが EMR notebooks へのアクセス権限を持ち、ノートブックを使用でき、ノートブックエディタを使用してクラスターでコードを実行できるようにします。これらの機能は、Amazon EMR および Amazon EMR クラスターで使用可能なセキュリティ機能と共に動作します。詳細については、「Amazon EMR でのセキュリティ」を参照してください。

- AWS Identity and Access Management ポリシーステートメントをノートブックタグと一緒に使用して、アクセスを制限できます。詳細については、「Amazon EMR で IAM が機能する仕組み」および「EMR Notebooks のアイデンティティベースのポリシーステートメントの例」を参照してください。
- Amazon EC2 セキュリティグループは、クラスターのプライマリインスタンスとノートブックエディタ間のネットワークトラフィックを制御する仮想ファイアウォールとして機能します。デフォルトまたはカスタムのセキュリティグループを使用できます。詳細については、「EMR Notebooks の EC2 セキュリティグループの指定」を参照してください。
- 他の AWS サービスとやり取りするときに EMR Notebooks が持つアクセス許可を決定する AWS サービスロールを指定します。詳細については、「EMR Notebooks のサービスロール」を参照してください。

カーネルとライブラリのインストールと使用

> **Note**
>
> EMR Notebooks は、コンソールで EMR Studio Workspace として使用できます。コンソールの「ワークスペースの作成」ボタンを使用すると、新しいノートブックを作成できます。EMR Notebooks ユーザーが Workspace にアクセスしたり作成したりするには、追加の

IAM ロール権限が必要です。詳細については、「Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console」および「Amazon EMR console」を参照してください。

各 EMR notebooks には、プリインストールされたライブラリとカーネルのセットが付属しています。EMR クラスターがカーネルとライブラリのあるリポジトリにアクセスできる場合は、クラスターに追加のライブラリとカーネルをインストールすることができます。たとえば、クラスターがプライベートサブネットにある場合は、ネットワークアドレス変換 (NAT) を設定して、クラスターがパブリックの PyPI リポジトリにアクセスしてライブラリをインストールするためのパスを指定する必要があります。さまざまなネットワーク設定の外部アクセス設定の詳細については、「Amazon VPC ユーザーガイド」の「シナリオと例」を参照してください。

EMR Serverless アプリケーションには、Python および 用の以下のライブラリがプリインストールされています PySpark。

- Python ライブラリ - ggplot、matplotlib、numpy、pandas、plotly、bokeh、scikit-learn、scipy、scipy
- PySpark ライブラリ – ggplot、matplotlib、numpy、pandas、plotly、bokeh、scikit-learn、scipy、scipy

クラスターのプライマリノードへのカーネルと Python ライブラリのインストール

Amazon EMR リリースバージョン 5.30.0 以降 (6.0.0 を除く) では、クラスターのプライマリノードに追加の Python ライブラリとカーネルをインストールすることができます。インストールしたカーネルとライブラリは、クラスターにアタッチされている EMR notebooks を実行するすべてのユーザーが使用できます。この方法でインストールした Python ライブラリは、プライマリノードで実行されるプロセスでのみ使用できます。ライブラリはコアノードまたはタスクノードにはインストールされず、それらのノードで実行されているエグゼキューターでは使用できません。

> **Note**
>
> Amazon EMR バージョン 5.30.1、5.31.0、および 6.1.0 では、クラスターのプライマリノードにカーネルとライブラリをインストールするためには、追加のステップを実行する必要があります。
> この機能を有効にするには、以下の操作を行います。

1. EMR Notebooks のサービスロールにアタッチされているアクセス許可ポリシーで、次のアクションが許可されていることを確認します。

   ```
   elasticmapreduce:ListSteps
   ```

   詳細については、「EMR Notebooks のサービスロール」を参照してください。

2. 次の例に示すように、 を使用して EMR Notebooks をセットアップするステップをクラスターで AWS CLI 実行します。ステップ名 EMRNotebooksSetup を使用する必要があります。*us-east-1* を、クラスターが存在するリージョンに置き換えます。詳細については、「AWS CLIを使用したクラスターへのステップの追加」を参照してください。

   ```
   aws emr add-steps --cluster-id MyClusterID --steps
    Type=CUSTOM_JAR,Name=EMRNotebooksSetup,ActionOnFailure=CONTINUE,Jar=s3://us-
   east-1.elasticmapreduce/libs/script-runner/script-runner.jar,Args=["s3://
   awssupportdatasvcs.com/bootstrap-actions/EMRNotebooksSetup/emr-notebooks-
   setup.sh"]
   ```

カーネルとライブラリをインストールするには、プライマリノードの /emr/notebook-env/bin ディレクトリで pip または conda を使用します。

Example — Python ライブラリのインストール

Python3 カーネルから、ノートブックセル内から %pip マジックをコマンドとして実行し、Python ライブラリをインストールします。

```
%pip install pmdarima
```

更新されたパッケージを使用するには、カーネルの再起動が必要になる場合があります。%%sh Spark マジックを使って pip を呼び出すこともできます。

```
%%sh
/emr/notebook-env/bin/pip install -U matplotlib
/emr/notebook-env/bin/pip install -U pmdarima
```

PySpark カーネルを使用する場合は、 pip コマンドを使用してクラスターにライブラリをインストールするか、ノートブック内から PySpark ノートブックスコープのライブラリを使用できます。

ターミナルからクラスターで pip コマンドを実行するには、以下のコマンドが示すように、まず SSH を使用してプライマリノードに接続します。

```
sudo pip3 install -U matplotlib
sudo pip3 install -U pmdarima
```

ノートブックスコープのライブラリを使用することもできます。ノートブックスコープのライブラリでは、ライブラリのインストールはセッションの範囲に限定され、すべての Spark エグゼキューターで行われます。詳細については、「ノートブックスコープのライブラリの使用」を参照してください。

PySpark カーネル内に複数の Python ライブラリをパッケージ化する場合は、分離された Python 仮想環境を作成することもできます。例については、「Using Virtualenv」を参照してください。

セッションに Python 仮想環境を作成するには、以下の例に示すように、ノートブックの最初のセルにある %%configure マジックコマンドの Spark spark.yarn.dist.archives プロパティを使用します。

```
%%configure -f
{
   "conf": {
   "spark.yarn.appMasterEnv.PYSPARK_PYTHON":"./environment/bin/python",
   "spark.yarn.appMasterEnv.PYSPARK_DRIVER_PYTHON":"./environment/bin/python",
   "spark.yarn.dist.archives":"s3://DOC-EXAMPLE-BUCKET/prefix/
my_pyspark_venv.tar.gz#environment",
   "spark.submit.deployMode":"cluster"
   }
}
```

Spark エグゼキューター環境も同様に作成できます。

```
%%configure -f
{
   "conf": {
   "spark.yarn.appMasterEnv.PYSPARK_PYTHON":"./environment/bin/python",
   "spark.yarn.appMasterEnv.PYSPARK_DRIVER_PYTHON":"./environment/bin/python",
   "spark.executorEnv.PYSPARK_PYTHON":"./environment/bin/python",
   "spark.yarn.dist.archives":"s3://DOC-EXAMPLE-BUCKET/prefix/
my_pyspark_venv.tar.gz#environment",
   "spark.submit.deployMode":"cluster"
   }
```

```
}
```

conda を使用して Python ライブラリをインストールすることもできます。conda を使用するのに sudo アクセスは必要ありません。プライマリノードに SSH を使用して接続して、ターミナルから conda を実行する必要があります。詳細については、「SSH を使用してプライマリノードに接続する」を参照してください。

Example - カーネルをインストールする

以下の例は、クラスターのプライマリノードに接続されている間にターミナルコマンドを使用して Kotlin カーネルをインストールする方法を示しています。

```
sudo /emr/notebook-env/bin/conda install kotlin-jupyter-kernel -c jetbrains
```

> **Note**
>
> これらの手順では、カーネルの依存関係はインストールされません。カーネルにサードパーティーの依存関係がある場合、ノートブックでカーネルを使用するためには、その前に、追加のセットアップ手順を実行する必要があります。

ノートブックスコープのライブラリの考慮事項および制限

ノートブックスコープのライブラリを使用する場合は、以下の点を考慮してください。

- ノートブックスコープのライブラリは、Amazon EMR リリース 5.26.0 以降で作成したクラスターで使用できます。
- ノートブックスコープのライブラリは、カーネルでのみ PySpark使用することを目的としています。
- すべてのユーザーがノートブックのセルから追加のノートブックスコープのライブラリをインストールできます。ノートブックのユーザーは、それらのライブラリを 1 回のノートブックセッションでのみ使用できます。他のユーザーが同じライブラリを必要とする場合や同じユーザーが別のセッションで同じライブラリを必要とする場合は、ライブラリを再インストールする必要があります。
- アンインストールできるのは、`install_pypi_package` API を使用してインストールされたライブラリのみです。クラスターにプリインストールされたライブラリをアンインストールすることはできません。

- 異なるバージョンの同じライブラリがノートブックスコープのライブラリとしてクラスターにインストールされている場合、ノートブックスコープのライブラリのバージョンはクラスター全体のライブラリのバージョンを上書きします。

ノートブックスコープのライブラリの操作

ライブラリをインストールするには、そのライブラリが存在する PyPI リポジトリに Amazon EMR クラスターがアクセスできる必要があります。

次の例は、 PySpark カーネル と APIs を使用してノートブックセル内からライブラリを一覧表示、インストール、アンインストールするための簡単なコマンドを示しています。その他の例については、 AWS Big Data Blog の「EMR Notebooks を使用して実行中のクラスターに Python ライブラリをインストールする」を参照してください。

Example - 現在のライブラリの一覧表示

以下のコマンドは、現在の Spark ノートブックセッションで使用できる Python パッケージを一覧表示します。これにより、クラスターにインストールされたライブラリと、ノートブックスコープのライブラリが一覧表示されます。

```
sc.list_packages()
```

Example - Celery ライブラリのインストール

以下のコマンドは、Celery ライブラリをノートブックスコープのライブラリとしてインストールします。

```
sc.install_pypi_package("celery")
```

このライブラリのインストール後、以下のコマンドは、そのライブラリが Spark のドライバーと実行プログラムで使用できることを確認します。

```
import celery
sc.range(1,10000,1,100).map(lambda x: celery.__version__).collect()
```

Example - Arrow ライブラリのインストール、バージョンとリポジトリの指定

以下のコマンドは、Arrow ライブラリのバージョンとリポジトリ URL を指定して、そのライブラリをノートブックスコープのライブラリとしてインストールします。

```
sc.install_pypi_package("arrow==0.14.0", "https://pypi.org/simple")
```

Example - ライブラリのアンインストール

以下のコマンドは、Arrow ライブラリをアンインストールし、現在のセッションからノートブックスコープのライブラリとして削除します。

```
sc.uninstall_package("arrow")
```

Git ベースのリポジトリと EMR Notebooks の関連付け

> **Note**
>
> EMR Notebooks は、コンソールで EMR Studio Workspace として使用できます。コンソールの「ワークスペースの作成」ボタンを使用すると、新しいノートブックを作成できます。EMR Notebooks ユーザーが Workspace にアクセスしたり作成したりするには、追加の IAM ロール権限が必要です。詳細については、「Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console」および「Amazon EMR console」を参照してください。

Git ベースのリポジトリを Amazon EMR Notebooks に関連付けて、バージョン管理された環境でノートブックを保存できます。ノートブックには最大 3 つのリポジトリを関連付けることができます。以下の Git ベースのサービスがサポートされています。

- AWS CodeCommit
- GitHub
- Bitbucket
- GitLab

Git ベースのリポジトリをノートブックに関連付けることには、以下のような利点があります。

- バージョン管理 - コードの変更をバージョン管理システムに記録できるため、変更の履歴を確認し、変更を選択して元に戻すことができます。
- コラボレーション - 異なるノートブックで作業を行う同僚がリモートの Git ベースのリポジトリを使用してコードを共有できます。ノートブックは、リモートリポジトリからコードを複製またはマージし、それらのリモートリポジトリに変更を返すことができます。

- コードの再利用 — データ分析や機械学習の手法を示す多くの Jupyter Notebook は、 などのパブリックにホストされているリポジトリで利用できます GitHub。ノートブックをリポジトリに関連付けて、リポジトリに含まれる Jupyter Notebook を再利用できます。

EMR Notebooks で Git ベースのリポジトリを使用するには、Amazon EMR コンソールでリポジトリをリソースとして追加し、認証が必要なリポジトリに認証情報を関連付けて、リポジトリをノートブックにリンクします。アカウントに保存されているリポジトリのリストと各リポジトリの詳細は、Amazon EMR コンソールで表示できます。ノートブックを作成するとき、既存の Git ベースのリポジトリを関連付けることができます。

トピック

前提条件と考慮事項

> **Note**
>
> EMR Notebooks は、コンソールで EMR Studio Workspace として使用できます。コンソールの「ワークスペースの作成」ボタンを使用すると、新しいノートブックを作成できます。EMR Notebooks ユーザーが Workspace にアクセスしたり作成したりするには、追加の IAM ロール権限が必要です。詳細については、「Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console」および「Amazon EMR console」を参照してください。

Git ベースのリポジトリを EMR Notebooks と統合することを計画している場合は、以下の点を考慮してください。

AWS CodeCommit

CodeCommit リポジトリを使用する場合は、 で Git 認証情報と HTTPS を使用する必要があります CodeCommit。SSH キー、および AWS CLI 認証情報ヘルパーを使用した HTTPS はサポートされて

いません。 CodeCommit は、個人用アクセストークン (PATsをサポートしていません。詳細については、「IAM ユーザーガイド」の「 での IAM の使用 CodeCommit: Git 認証情報、SSH キー、 AWS アクセスキー」および「 ユーザーガイド」の「Git 認証情報を使用した HTTPS ユーザーのセットアップ」を参照してください。 AWS CodeCommit

アクセスとアクセス許可に関する考慮事項

リポジトリをノートブックに関連付ける前に、クラスター、EMR Notebooks の IAM ロール、およびセキュリティグループの設定とアクセス許可が正しいことを確認してください。「EMR Notebooks 用にプライベートにホストされた Git リポジトリを設定する」の手順に従って、プライベートネットワークでホストしている Git ベースのリポジトリを設定することもできます。

- クラスターのインターネットアクセス - 起動されるネットワークインターフェイスにはプライベート IP アドレスしかありません。つまり、ノートブックが接続するクラスターは、ネットワークアドレス変換 (NAT) ゲートウェイに接続されたプライベートサブネット内にあるか、仮想プライベートゲートウェイを介してインターネットにアクセスできる必要があります。詳細については、「Amazon VPC のオプション」を参照してください。

  ノートブックのセキュリティグループには、クラスターからインターネットにトラフィックをルーティングすることをノートブックに許可するアウトバウンドルールが含まれている必要があります。独自のセキュリティグループを作成することをお勧めします。詳細については、「EMR Notebooks の EC2 セキュリティグループの指定」を参照してください。

  > **Important**
  >
  > ネットワークインターフェイスがパブリックサブネットに起動された場合、インターネットゲートウェイ (IGW) を介してインターネットと通信できなくなります。

- のアクセス許可 AWS Secrets Manager – Secrets Manager を使用してリポジトリへのアクセスに使用するシークレットを保存する場合、 には `secretsmanager:GetSecretValue`アクションを許可するアクセス許可ポリシーがアタッチされているthe section called “EMR Notebooks ロール”必要があります。

EMR Notebooks 用にプライベートにホストされた Git リポジトリを設定する

次の手順を使用して、EMR Notebooks 用にプライベートにホストされたリポジトリを設定します。DNS サーバーおよび Git サーバーに関する情報が含まれた設定ファイルを用意する必要があ

ります。Amazon EMR は、この情報を使用して、プライベートにホストされたリポジトリにトラフィックをルーティングできる EMR notebooks を設定します。

前提条件

EMR Notebooks 用にプライベートにホストされた Git リポジトリを設定する前に、次のものが必要です。

- EMR Notebooks のファイルが保存される Amazon S3 Control 場所。

EMR Notebooks 用にプライベートにホストされた 1 つ以上の Git リポジトリを設定するには

1. 提供されたテンプレートを使用して、設定ファイルを作成します。設定で指定する Git サーバーごとに次の値を含めます。

   - **`DnsServerIpV4`** - DNS サーバーの IPv4 アドレス。`DnsServerIpV4` と `GitServerIpV4List` の両方に値を指定した場合、`DnsServerIpV4` の値が優先され、`GitServerDnsName` を解決するために使用されます。

     > **Note**
     >
     > プライベートにホストされた Git リポジトリを使用するには、DNS サーバーで EMR Notebooks からのインバウンドアクセスを許可する必要があります。DNS サーバーを他の不正アクセスから保護することを強くお勧めします。

   - **`GitServerDnsName`** - Git サーバーの DNS 名。例えば、`"git.example.com"` です。
   - **`GitServerIpV4List`** - Git サーバーに属する IPv4 アドレスのリスト。

```
[
    {
        "Type": "PrivatelyHostedGitConfig",
        "Value": [
            {
                "DnsServerIpV4": "<10.24.34.xxx>",
                "GitServerDnsName": "<enterprise.git.com>",
                "GitServerIpV4List": [
                    "<xxx.xxx.xxx.xxx>",
                    "<xxx.xxx.xxx.xxx>"
                ]
            },
```

```
            {
                "DnsServerIpV4": "<10.24.34.xxx>",
                "GitServerDnsName": "<git.example.com>",
                "GitServerIpV4List": [
                    "<xxx.xxx.xxx.xxx>",
                    "<xxx.xxx.xxx.xxx>"
                ]
            }
        ]
    }
]
```

2. configuration.json という名前で設定ファイルを保存します。
3. 設定ファイルを指定された Amazon S3 ストレージの場所にある life-cycle-configuration というフォルダーにアップロードします。例えば、デフォルトの S3 の場所が s3://DOC-EXAMPLE-BUCKET/notebooks の場合、設定ファイルは s3://DOC-EXAMPLE-BUCKET/notebooks/**life-cycle-configuration/configuration.json** に配置する必要があります。

> **Important**
>
> life-cycle-configuration フォルダへのアクセスを EMR Notebooks 管理者および EMR Notebooks のサービスロールのみに制限することを強くお勧めします。また、configuration.json を不正アクセスから保護する必要があります。手順については、「ユーザーポリシーを使用したバケットへのアクセスの制御」または「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。

アップロードの手順については、「Amazon Simple Storage Service ユーザーガイド」の「フォルダの作成」と「オブジェクトのアップロード」を参照してください。

Git ベースのリポジトリを Amazon EMR に追加する

> **Note**
>
> EMR Notebooks は、コンソールで EMR Studio Workspace として使用できます。コンソールの「ワークスペースの作成」ボタンを使用すると、新しいノートブックを作成できます。EMR Notebooks ユーザーが Workspace にアクセスしたり作成したりするには、追加の

IAM ロール権限が必要です。詳細については、「Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console」および「Amazon EMR console」を参照してください。

Git ベースのリポジトリを古いコンソールの EMR Notebooks、または新しいコンソールの EMR Studio Workspace に追加する方法については、以下のセクションを参照してください。

New console

EMR Notebooks は新しいコンソールの EMR Studio Workspace なので、「GIT ベースのリポジトリを EMR Studio Workspace にリンクする」の手順に従って最大 3 つの Git リポジトリを Workspace に関連付けることができます。

JupyterLab Git 拡張機能を使用することもできます。Jupyterlab ノートブックの左側のサイドバーから [Git] アイコンを選択し、拡張機能にアクセスします。拡張機能の詳細については、jupyterlab-git GitHub リポジトリを参照してください。

Git リポジトリを Workspace に関連付けるには、Studio 管理者が Git リポジトリのリンクを許可するように Studio を設定する手順を実行する必要があります。詳細については、「Git ベースのリポジトリのアクセス権とアクセス許可を設定する」を参照してください。

Old console

古いコンソールを使用して Amazon EMR アカウントのリソースとして Git ベースのリポジトリを追加するには

1. 古い Amazon EMR コンソール (https://console.aws.amazon.com/elasticmapreduce) を開きます。
2. [Git リポジトリ] を選択し、[Add repository (リポジトリの追加)] を選択します。
3. [リポジトリ名] に、Amazon EMR でリポジトリに使用する名前を入力します。

   名前には、英数字、ハイフン (-)、またはアンダースコア (_) のみを含めることができます。
4. [Git repository URL (Git リポジトリ URL)] に、リポジトリの URL を入力します。CodeCommit リポジトリを使用する場合、これは URL のクローンを作成し、HTTPS のクローンを作成するときにコピーされる URL です。例えば、 ですhttps://git-codecommit.*us-west-2*.amazonaws.com/v1/repos/*MyCodeCommitRepoName*。
5. [ブランチ] に、ブランチ名を入力します。
6. [Git credentials (Git 認証情報)] で、以下のガイドラインに従ってオプションを選択します。Git ユーザー名とパスワード、または個人用アクセストークン (PAT) を使用して、リポ

ジトリに対する認証を行うことができます。EMR Notebooks は、Secrets Manager に保存されたシークレットを使用して Git 認証情報にアクセスします。

> **Note**
>
> GitHub リポジトリを使用する場合は、個人アクセストークン (PAT) を使用して認証することをお勧めします。2021 年 8 月 13 日以降、 GitHub は Git オペレーションの認証時にパスワードを受け入れなくなります。詳細については、 ブログの「Git オペレーションのトークン認証要件」の投稿を参照してください。 GitHub

オプション	説明
既存の AWS シークレットを使用する	認証情報を Secrets Manager でシークレットとしてすでに保存している場合は、このオプションを選択し、リストからシークレット名を選択します。 Git ユーザー名とパスワードに関連付けられたシークレットを選択する場合、シークレットは `{"gitUsername": "MyUserName ", "gitPassword": "MyPassword "}` の形式にする必要があります。

オプション	説明
新しいシークレットを作成する	既存の Git 認証情報を、Secrets Manager で作成した新しいシークレットに関連付けるには、このオプションを選択します。リポジトリに使用する Git 認証情報に基づいて、以下のいずれかの操作を行います。 Git のユーザー名とパスワードを使用してリポジトリにアクセスする場合は、[Username and password] (ユーザー名とパスワード) を選択し、Secrets Manager で使用する [Secret name] (シークレット名) を入力してから、シークレットに関連付ける [Username] (ユーザー名) および [Password] (パスワード) を入力します。 または 個人用アクセストークンを使用してリポジトリにアクセスする場合は、[Personal access token (PAT)] (Personal access token (PAT)) を選択し、Secrets Manager で使用する [Secret name] (シークレット名) を入力してから、[personal access token] (個人用アクセストークン) を入力します。 詳細については、「 のコマンドライン用の個人用アクセストークンの作成 GitHub」および「Bitbucket 用の個人用アクセストークン」を参照してください。 CodeCommit リポジトリはこのオプションをサポートしていません。
認証情報なしでパブリックリポジトリを使用する	パブリックリポジトリにアクセスするには、このオプションを選択します。

7. [リポジトリの追加] を選択します。

Git ベースのリポジトリを更新または削除する

> **Note**
>
> EMR Notebooks は、コンソールで EMR Studio Workspace として使用できます。コンソールの「ワークスペースの作成」ボタンを使用すると、新しいノートブックを作成できます。EMR Notebooks ユーザーが Workspace にアクセスしたり作成したりするには、追加の IAM ロール権限が必要です。詳細については、「Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console」および「Amazon EMR console」を参照してください。

Git ベースのリポジトリを古いコンソールの EMR Notebooks、または新しいコンソールの EMR Studio Workspace から削除する方法については、以下のセクションを参照してください。

New console

EMR Notebooks は新しいコンソールの EMR Studio Workspace なので、Workspace 内の Git リポジトリの操作について詳しくは、「GIT ベースのリポジトリを EMR Studio Workspace にリンクする」を参照してください。ただし、現時点では、Git リポジトリを Workspace から削除することはできません。

Old console

古いコンソールの Git ベースのリポジトリを更新するには

1. [Git リポジトリ] ページで、更新するリポジトリを選択します。
2. リポジトリページで、[Edit repository (リポジトリの編集)] を選択します。
3. リポジトリページで、[Git credentials (Git 認証情報)] を更新します。

古いコンソールで Git リポジトリを削除するには

1. [Git リポジトリ] ページで、削除するリポジトリを選択します。
2. リポジトリページで、リポジトリに現在リンクされているすべてのノートブックを選択します。[Unlink notebook (ノートブックをリンク解除)] を選択します。
3. リポジトリページで、[Delete (削除)] を選択します。

> **Note**
>
> Amazon EMR からローカル Git リポジトリを削除するには、まずこのリポジトリからノートブックをリンク解除する必要があります。詳細については、「Git ベースのリポジトリをリンクまたはリンク解除する」を参照してください。Git リポジトリを削除しても、リポジトリ用に作成されたシークレットは削除されません。シークレットは AWS Secrets Managerで削除できます。

Git ベースのリポジトリをリンクまたはリンク解除する

> **Note**
>
> EMR Notebooks は、コンソールで EMR Studio Workspace として使用できます。コンソールの「ワークスペースの作成」ボタンを使用すると、新しいノートブックを作成できます。EMR Notebooks ユーザーが Workspace にアクセスしたり作成したりするには、追加の IAM ロール権限が必要です。詳細については、「Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console」および「Amazon EMR console」を参照してください。

以下の手順を使用して、Git ベースのリポジトリを古いコンソールの EMR Notebooks、または新しいコンソールの EMR Studio Workspace にリンクまたはリンク解除します。

New console

EMR Notebooks は新しいコンソールの EMR Studio Workspace なので、Workspace 内の Git リポジトリの操作について詳しくは、「GIT ベースのリポジトリを EMR Studio Workspace にリンクする」を参照してください。ただし、現時点では、Git リポジトリを Workspace から削除することはできません。

Old console

Git ベースのリポジトリを EMR ノートブックにリンクするには

ノートブックが [準備完了] になると、リポジトリをノートブックにリンクできます。

1. [ノートブック] リストから、更新するノートブックを選択します。
2. [ノートブック] ページの [Git リポジトリ] セクションで、[新しいリポジトリをリンク] を選択します。

3. [Link Git repository to notebook (Git リポジトリをノートブックにリンク)] ウィンドウのリポジトリリストで、ノートブックにリンクする 1 つ以上のリポジトリを選択し、[Link repository (リポジトリをリンク)] を選択します。

または

1. [Git リポジトリ] ページで、ノートブックにリンクするリポジトリを選択します。
2. [EMR notebooks (EMR ノートブック)] のリストで、[Link new notebook (新しいノートブックをリンク)] を選択して、このリポジトリを既存のノートブックにリンクします。

Git リポジトリを EMR ノートブックからリンク解除するには

1. [ノートブック] リストから、更新するノートブックを選択します。
2. [Git リポジトリ] のリストで、ノートブックからリンク解除するリポジトリを選択してから、[リポジトリをリンク解除] を選択します。

または

1. [Git リポジトリ] ページで、更新するリポジトリを選択します。
2. [EMR notebooks (EMR ノートブック)] のリストで、リポジトリからリンク解除するノートブックを選択した後、[Unlink notebook (ノートブックをリンク解除)] を選択します。

> **Note**
>
> Git リポジトリをノートブックにリンクすると、リモートリポジトリがローカルの Jupyter Notebook に複製されます。Git リポジトリをノートブックからリンク解除すると、リモートリポジトリからノートブックが切断されるだけで、ローカル Git リポジトリは削除されません。

リポジトリのステータスについて

Git リポジトリは、リポジトリリスト内で以下のいずれかのステータスを持つことができます。EMR Notebooks と Git リポジトリのリンクの詳細については、「Git ベースのリポジトリをリンクまたはリンク解除する」を参照してください。

ステータス	意味
Linking (リンク中)	Git リポジトリがノートブックにリンク中です。リポジトリのステータスが [Linking (リンク中)] である間は、ノートブックを停止できません。
リンク済み	Git リポジトリがノートブックにリンク済みです。リポジトリのステータスが [リンク済み] である間は、リモートリポジトリにノートブックが接続されています。
Link Failed (リンク失敗)	Git リポジトリがノートブックへのリンクに失敗しました。リンクを再試行できます。
Unlinking (リンク解除中)	Git リポジトリがノートブックからリンク解除中です。リポジトリのリンクが [Unlinking (リンク解除中)] である間は、ノートブックを停止できません。Git リポジトリをノートブックからリンク解除すると、リモートリポジトリからノートブックが切断されるだけで、ノートブックからコードは削除されません。
Unlink Failed (リンク解除に失敗)	Git リポジトリがノートブックからのリンク解除に失敗しました。リンク解除を再試行できます。

Git リポジトリが関連付けられた新しいノートブックを作成する

> **Note**
>
> EMR Notebooks は、コンソールで EMR Studio Workspace として使用できます。コンソールの「ワークスペースの作成」ボタンを使用すると、新しいノートブックを作成できます。EMR Notebooks ユーザーが Workspace にアクセスしたり作成したりするには、追加の IAM ロール権限が必要です。詳細については、「Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console」および「Amazon EMR console」を参照してください。

古い Amazon EMR コンソールでノートブックを作成して Git リポジトリに関連付けるには

1. ノートブックの作成 の指示に従います。
2. [セキュリティグループ] で、[Use your own security group (独自のセキュリティグループを使用)] を選択します。

   > **Note**
   >
   > ノートブックのセキュリティグループには、クラスター経由でインターネットにトラフィックをルーティングすることをノートブックに許可するアウトバウンドルールが含まれている必要があります。独自のセキュリティグループを作成することをお勧めします。詳細については、「EMR Notebooks の EC2 セキュリティグループの指定」を参照してください。

3. [Git リポジトリ] の [リポジトリの選択] を選択して、ノートブックに関連付けるリポジトリを選択します。

   1. アカウントにリソースとして保存されているリポジトリを選択し、[Save (保存)] を選択します。
   2. アカウントに新しいリポジトリをリソースとして追加するには、[新しいリポジトリの追加] を選択します。新しいウィンドウで [Add repository (リポジトリの追加)] ワークフローを完了します。

ノートブックで Git リポジトリを使用する

> **Note**
>
> EMR Notebooks は、コンソールで EMR Studio Workspace として使用できます。コンソールの「ワークスペースの作成」ボタンを使用すると、新しいノートブックを作成できます。EMR Notebooks ユーザーが Workspace にアクセスしたり作成したりするには、追加の IAM ロール権限が必要です。詳細については、「Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console」および「Amazon EMR console」を参照してください。

ノートブックを開く JupyterLabときに、Jupyter で開くか開くかを選択できます。

Jupyter でノートブックを開くことを選択すると、ノートブック内の展開可能なファイルとフォルダのリストが表示されます。ノートブックセルで以下のような Git コマンドを手動で実行できます。

```
!git pull origin primary
```

追加のリポジトリを開くには、他のフォルダに移動します。

JupyterLab インターフェイスでノートブックを開く場合は、プリインストールされた JupyterLab Git 拡張機能を使用できます。拡張機能の詳細については、「jupyterlab-git」を参照してください。

クラスターの計画と設定

このセクションでは、Amazon EMR を使用したクラスターの計画、設定、および起動に関する設定オプションと手順を説明します。クラスタを起動する前に、処理しているデータと、コスト、速度、容量、可用性、セキュリティ、および管理の要件に基づくシステムに関する選択をします。選択肢には次が含まれます。

- クラスタを実行するリージョン、データを保存する場所と方法、結果を出力する方法。クラスターの場所とデータストレージの設定 を参照してください。
- Outposts または Local Zones で Amazon EMR クラスターを実行しているかどうか。「の EMR クラスター AWS Outposts」または「AWS ローカルゾーンの EMR クラスター」を参照してください。
- クラスタを長期にわたり実行するか、または一時的に実行するか、どのソフトウェアが実行されるか。「ステップ実行後に継続または終了するようにクラスターを設定する」および「クラスターソフトウェアを設定する」を参照してください。
- クラスターが 1 つのプライマリノードを持つか、3 つのプライマリノードを持つか。プライマリノードの計画と設定 を参照してください。
- アプリケーションのコスト、パフォーマンス、および可用性を最適化するハードウェアおよびネットワークオプション。クラスターハードウェアとネットワークを設定する を参照してください。
- より容易に管理でき、アクティビティ、パフォーマンス、状態を監視できるようにクラスタを設定する方法。「クラスターのログ記録とデバッグを設定する」および「クラスターのタグ付け」を参照してください。
- クラスターのリソースへのアクセスを認証し許可する方法、およびデータを暗号化する方法。Amazon EMR でのセキュリティ を参照してください。
- 他のソフトウェアとサービスと統合する方法。ドライバーとサードパーティーアプリケーション統合 を参照してください。

クラスターをすばやく起動

コンソールを使用してクラスターをすばやく起動するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr/clusters で Amazon EMR コンソールを開きます。

2. 左側のナビゲーションペインの [EMR on EC2] で、[クラスター] を選択し、[クラスターの作成] を選択します
3. [クラスターの作成] ページで、提供されたフィールドの値を入力または選択します。永続的な概要パネルには、現在選択されているクラスターオプションのリアルタイムビューが表示されます。サマリーパネルの見出しを選択し、対応するセクションに移動して、調整できます。クラスター名に <、>、$、|、` (バックティック) の文字を含めることはできません。[クラスターの作成] を選択する前に、必要な設定をすべて完了する必要があります。
4. [クラスターの作成] を選択して、表示されている設定を確定します。
5. クラスターの詳細ページが開きます。クラスター名の横のクラスターの [ステータス] を見つけます。ステータスは、クラスター作成プロセスに応じて [開始中]、[実行中]、[待機中] へと変化します。アップデートを受けるには、右上にある更新アイコンを選択するか、ブラウザを更新する必要があります。

   ステータスが [待機中] に変わると、クラスターが起動して実行中になり、ステップと SSH 接続を受け入れる準備が整います。

クラスターの場所とデータストレージの設定

このセクションでは、クラスターのリージョンを設定する方法、Amazon EMR を使用するときに利用できるさまざまなファイルシステム、およびそれらを使用する方法について説明します。さらに、必要に応じてデータを Amazon EMR に準備またはアップロードする方法のほか、ログファイルの出力場所と設定する任意の出力ファイルの出力場所を準備する方法についても説明します。

トピック

- AWS リージョンを選択する
- ストレージシステムとファイルシステムで作業する
- 入力データを準備する
- 出力場所を設定する

AWS リージョンを選択する

Amazon Web Services は、世界中のデータセンターにあるサーバーで実行されています。データセンターは、地理的リージョン別に整理されています。Amazon EMR クラスターを起動するときは、リージョンを指定する必要があります。レイテンシーを削減し、コストを最小限に抑えて規制用件に対応できるリージョンを選ぶとよいでしょう。Amazon EMR でサポートされているリージョンと工

ンドポイントの一覧については、「Amazon Web Services 全般のリファレンス」の「リージョンとエンドポイント」を参照してください。

最高のパフォーマンスのため、クラスターはデータと同じリージョンで起動する必要があります。例えば、入力データを格納している Simple Storage Service (Amazon S3) バケットが米国西部 (オレゴン) リージョンにある場合、リージョン内データ転送料を避けるため、米国西部 (オレゴン) リージョンでクラスターを起動する必要があります。Simple Storage Service (Amazon S3) バケットを使用して、クラスターの出力を受け取る場合、それも米国西部 (オレゴン) リージョンで作成する必要があります。

Amazon EC2 キーペアをクラスターと関連付ける予定がある場合 (SSH を使用して、マスターノードにログオンする場合に必要)、キーペアをクラスターと同じリージョンで作成する必要があります。同様に、Amazon EMR がクラスターを管理するために作成するセキュリティグループは、クラスターと同じリージョンに作成します。

2017 年 5 月 17 日以降 AWS アカウント に にサインアップした場合、 からリソースにアクセスするときの AWS Management Console デフォルトリージョンは米国東部 (オハイオ) (us-east-2) です。古いアカウントの場合、デフォルトリージョンは米国西部 (オレゴン) (us-west-2) または米国東部 (バージニア北部) (us-east-1) です。詳細については、「リージョンとエンドポイント」を参照してください。

一部の AWS 機能は、限定されたリージョンでのみ使用できます。例えば、クラスターコンピューティングインスタンスは米国東部 (バージニア北部) リージョンでのみ使用でき、アジアパシフィック (シドニー) リージョンでは Hadoop 1.0.3 以降のみサポートされます。リージョンを選択する場合は、リージョンで使用したい機能がサポートされていることを確認してください。

最高のパフォーマンスを得るには、クラスターで使用するすべての AWS リソースに同じリージョンを使用します。次の表に、サービス間でリージョン名をマッピングします。Amazon EMR リージョンのリストについては、「Amazon Web Services 全般のリファレンス」の「AWS リージョン とエンドポイント」を参照してください。

コンソールでリージョンを選択する

デフォルトのリージョンは、ナビゲーションバーのアカウント情報の左側に表示されます。新しいコンソールと古いコンソールの両方でリージョンを切り替えるには、[リージョン] ドロップダウンメニューを選択し、新しいオプションを選択します。

を使用してリージョンを指定する AWS CLI

aws configure コマンドまたはAWS_DEFAULT_REGION環境変数 AWS CLI を使用して、 でデフォルトのリージョンを指定します。詳細については、「 ユーザーガイド」の AWS 「リージョンの設定AWS Command Line Interface 」を参照してください。

SDK または API でリージョンを選択する

SDK を使用してリージョンを選択するには、リージョンのエンドポイントを使用するようにアプリケーションを設定します。 AWS SDK を使用してクライアントアプリケーションを作成している場合、以下の例のように setEndpoint を呼び出すことによってクライアントのエンドポイントを変更できます。

```
client.setEndpoint("elasticmapreduce.us-west-2.amazonaws.com");
```

アプリケーションでエンドポイントを設定して、リージョンを指定したら、クラスターの EC2 インスタンスのアベイラビリティーゾーンを設定できます。アベイラビリティーゾーンはそれぞれが地理的に独立しており、他のゾーンの影響は受けません。また、同じリージョン内の他のアベイラビリティーゾーンへのネットワーク接続が低コストで、レイテンシーも短くなります。リージョンには 1 つまたは複数のアベイラビリティーゾーンが含まれます。パフォーマンスを最適化し、レイテンシーを低減するには、すべてのリソースを、それらを使用するクラスターと同じアベイラビリティーゾーンに置く必要があります。

ストレージシステムとファイルシステムで作業する

Amazon EMR および Hadoop には、クラスターステップの処理に使用できるさまざまなファイルシステムが用意されています。どのファイルシステムを使用するかは、データへのアクセスに使用する URI のプレフィックスで指定します。例えば、s3://DOC-EXAMPLE-BUCKET1/path は、EMRFS を使用して Simple Storage Service (Amazon S3) バケットを参照します。次の表に、使用可能なファイルシステムと、それぞれの使用が推奨される条件を示します。

Amazon EMR および Hadoop は通常、クラスターを処理するときに以下のうち少なくとも 2 つのファイルシステムを使用します。HDFS と EMRFS は、Amazon EMR で使用される 2 つの主なファイルシステムです。

> **Important**
>
> Amazon EMR リリース 5.22.0 以降、Amazon EMR は AWS 署名バージョン 4 のみを使用して Amazon S3 へのリクエストを認証します。以前の Amazon EMR リリースでは、リリース

ノートで AWS Signature Version 4 のみが使用されていることが示されていない限り、場合によっては Signature Version 2 が使用されます。詳細については、Amazon Simple Storage Service デベロッパーガイドの「リクエストの認証 (AWS 署名バージョン 4)」および「リクエストの認証 (AWS 署名バージョン 2)」を参照してください。

ファイルシステム	プレフィックス	説明
HDFS	`hdfs://` (またはプレフィックスなし)	HDFS は分散型のスケーラブルかつポータブルな Hadoop 用ファイルシステムです。HDFS の利点は、クラスターを管理する Hadoop クラスターノードと個別のステップを管理する Hadoop クラスターノードの間でのデータ認識です。詳細については、「Hadoop のドキュメント」を参照してください。 HDFS は、マスターノードおよびコアノードによって使用されます。1 つの利点は高速であることです。欠点は、エフェメラルなストレージであり、クラスターが終了すると回収されてしまうことです。最適な用途は、ジョブフローの中間ステップで得られた結果のキャッシュ場所です。
EMRFS	`s3://`	EMRFS は、通常のファイルを Amazon EMR から Simple Storage Service (Amazon S3) に直接読み書きするために使用される Hadoop ファイルシステムの実装です。EMRFS は、Hadoop で使用できるように永続的データを Amazon S3 に保存できるだけでなく、Amazon S3 のサーバー側の暗号化、read-after-write 整合性、リスト整合性などの機能も提供します。 **Note** 以前は、Amazon EMR は、`s3n` および `s3a` ファイルシステムを使用していました。どちらも現在でも動作しますが、最適なパフォーマンス、セキュリティ、および信頼性のため

ファイルシステム	プレフィックス	説明
		には、s3 URI スキームを使用することをお勧めします。
ローカルファイルシステム		ローカルファイルシステムとは、ローカルに接続されているディスクを指します。Hadoop クラスターを作成すると、インスタンスストアと呼ばれる、あらかじめアタッチされたディスクストレージのブロックが事前設定されている EC2 インスタンスから、各ノードが作成されます。インスタンスストアボリューム上のデータは、EC2 インスタンスの存続中のみ使用できます。インスタンスストアボリュームは、バッファやキャッシュ、作業データのように絶えず変化する一時的データを保存するのに最適です。詳細については、「Amazon EC2 インスタンスストレージ」を参照してください。 ローカルファイルシステムは HDFS によって使用されますが、Python もローカルファイルシステムから実行されるため、追加のアプリケーションファイルをインスタンスストアボリュームに保存することを選択できます。

ファイルシステム	プレフィックス	説明
(レガシーの) Simple Storage Service (Amazon S3) ブロックファイルシステム	`s3bfs://`	Simple Storage Service (Amazon S3) ブロックファイルシステムは、レガシーのファイルストレージシステムです。このシステムは、使用しないことを強くお勧めします。 **Important** このファイルシステムは、クラスター障害の原因となる競合状態を引き起こすことがあるため、利用はお勧めしません。ただし、レガシーアプリケーションでは必要になることがあります。

ファイルシステムへのアクセス

どのファイルシステムを使用するかは、データへのアクセスに使用するユニフォームリソースアイデンティファイア (URI) のプレフィックスで指定します。次の手順は、数種類のファイルシステムを参照する方法を示しています。

ローカルの HDFS にアクセスするには

- URI に `hdfs:///` プレフィックスを指定します。Amazon EMR は、URI にプレフィックスを指定しないパスをローカル HDFS に解決します。たとえば、次の URI はどちらも HDFS 内の同じ場所に解決されます。

```
hdfs:///path-to-data

/path-to-data
```

リモートの HDFS にアクセスするには

- 次の例に示すように、URI にマスターノードの IP アドレスを含めます。

```
hdfs://master-ip-address/path-to-data

master-ip-address/path-to-data
```

Simple Storage Service (Amazon S3) にアクセスするには

- s3:// プレフィックスを使用します。

```
s3://bucket-name/path-to-file-in-bucket
```

Simple Storage Service (Amazon S3) ブロックファイルシステムにアクセスするには

- Simple Storage Service (Amazon S3) ブロックファイルシステムを必要とするレガシーアプリケーションでのみ使用してください。このファイルシステムでデータにアクセス、またはデータを格納するには、URI で s3bfs:// プレフィックスを使用します。

  Simple Storage Service (Amazon S3) ブロックファイルシステムは、Simple Storage Service (Amazon S3) への 5 GB を超えるサイズのアップロードをサポートするために使われていたレガシーファイルシステムです。Amazon EMR が AWS Java SDK を通じて提供するマルチパートアップロード機能を使用すると、最大 5 TB のサイズのファイルを Amazon S3 ネイティブファイルシステムにアップロードでき、Amazon S3 ブロックファイルシステムは非推奨になります。

  > **Warning**
  >
  > このレガシーファイルシステムは競合状態を引き起こす可能性があり、それによってファイルシステムが破壊されるおそれがあるため、この形式は避け、代わりに EMRFS を使用してください。

```
s3bfs://bucket-name/path-to-file-in-bucket
```

入力データを準備する

大部分のクラスターは入力データを読み込んで、そのデータを処理します。データを読み込むには、該当するデータが、クラスターでアクセスできる場所に位置し、かつクラスターで処理できる形式になっている必要があります。最も一般的なシナリオは、入力データを Simple Storage Service (Amazon S3) にアップロードすることです。Amazon EMR には、クラスターが Simple Storage Service (Amazon S3) からデータをインポートまたは読み取るためのツールが用意されています。

Hadoop におけるデフォルトの入力形式はテキストファイルです。ただし、Hadoop をカスタマイズすれば、他の形式で格納されているデータをインポートできます。

トピック

Amazon EMR が受け入れることができる入力のタイプ

クラスターのデフォルトの入力形式は、各行が改行 (\n) 文字で区切られているテキストファイルです。これは、最もよく使用される入力形式です。

入力データがデフォルトのテキストファイル以外の形式である場合は、Hadoop インターフェイス `InputFormat` を使用して他の入力の種類を指定できます。カスタムデータの種類を処理するために、`FileInputFormat` クラスのサブクラスを作成することもできます。詳細については、http://hadoop.apache.org/docs/current/api/org/apache/hadoop/mapred/InputFormat.html を参照してください。

Hive を使用している場合は、シリアライザ/デシリアライザ (SerDe) を使用して、 内のデータを特定の形式から HDFS に読み取ることができます。詳細については、https://cwiki.apache.org/confluence/display/Hive/SerDe を参照してください。

Amazon EMR にデータを入力する方法

Amazon EMR では、複数の方法でデータをクラスターに配置することができます。最も一般的な方法は、Simple Storage Service (Amazon S3) にデータをアップロードし、Amazon EMR の組み込み機能を使用してクラスターにデータをロードするというものです。また、Hadoop の

DistributedCache 機能を使用して、分散ファイルシステムからローカルファイルシステムにファイルを転送することもできます。Amazon EMR によって提供される Hive (Hive バージョン 0.7.1.1 以降) の実装には、DynamoDB と Amazon EMR クラスターの間でデータのインポートおよびエクスポートを行う場合に使用できる機能が含まれています。処理する社内データが大量にある場合、 AWS Direct Connect サービスが役に立つ場合があります。

トピック

- データを Simple Storage Service (Amazon S3) にアップロードする
- AWS DataSyncでのデータのアップロード
- 分散キャッシュによるファイルのインポート
- 圧縮ファイルの処理方法
- DynamoDB データを Hive にインポートする
- AWS Direct Connectでデータに接続する
- AWS Snowballで大量のデータをアップロードする

データを Simple Storage Service (Amazon S3) にアップロードする

Simple Storage Service (Amazon S3) バケットにファイルをアップロードする方法については、「Amazon Simple Storage Service ユーザーガイド」の「バケットにオブジェクトを追加する」を参照してください。Hadoop での Simple Storage Service (Amazon S3) の使用の詳細については、http://wiki.apache.org/hadoop/AmazonS3 を参照してください。

トピック

- Amazon S3 バケットの作成と設定
- Simple Storage Service (Amazon S3) 用のマルチパートアップロードを設定する
- ベストプラクティス
- Amazon S3 Express One Zone にデータをアップロードする

Amazon S3 バケットの作成と設定

Amazon EMR は、Amazon S3 AWS SDK for Java で を使用して、入力データ、ログファイル、出力データを保存します。Simple Storage Service (Amazon S3) は、これらのストレージロケーションをバケットとして参照します。バケットには、Simple Storage Service (Amazon S3) と DNS の要件に従って一定の制約と制限があります。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「バケットの制約と制限」を参照してください。

このセクションでは、Amazon S3 を使用して Amazon AWS Management Console Amazon S3バケットのアクセス許可を作成して設定する方法について説明します。また、Simple Storage Service (Amazon S3) API または AWS CLIを使用して、Simple Storage Service (Amazon S3) バケットのアクセス許可を作成および設定できます。変更と共に curl を使用して、Simple Storage Service (Amazon S3) の適切な認証パラメータを渡すこともできます。

以下のリソースを参照してください。

- コンソールを使用してバケットを作成する方法については、「Simple Storage Service (Amazon S3) ユーザーガイド」の「バケットの作成」を参照してください。
- を使用してバケットを作成および操作するには AWS CLI、「Amazon S3 ユーザーガイド」の「 で高レベルの S3 コマンド AWS Command Line Interfaceを使用する」を参照してください。Amazon S3
- SDK を使用してバケットを作成する方法については、「Amazon Simple Storage Service ユーザーガイド」の「バケットを作成する例」を参照してください。
- curl を使用してバケットを操作する方法については、「curl 用の Simple Storage Service (Amazon S3) 認証ツール」を参照してください。
- リージョンの詳細については、「Amazon Simple Storage Service ユーザーガイド」の「バケットへのアクセス」を参照してください。
- Amazon S3 Access Points を使用してバケットで作業する方法については、「Simple Storage Service (Amazon S3) ユーザーガイドの「アクセスポイントでのバケット形式のエイリアスの使用」を参照してください。Simple Storage Service (Amazon S3) バケット名の代わりに、Amazon S3 Access Points と Amazon S3 Access Points のエイリアスを合わせて簡単に使用できます。Simple Storage Service (Amazon S3) アクセスポイントのエイリアスは、既存アプリケーションと新しいアプリケーションの両方 (Spark、Hive、Presto など) に使用できます。

> **Note**
>
> バケットのロギングを有効にした場合、有効になるのはバケットアクセスログのみです。Amazon EMR クラスターログは有効にはなりません。

バケットの作成中またはそれ以降に、アプリケーションに応じてバケットにアクセスするための適切なアクセス許可を設定できます。一般的に、お客様自身（オーナー）に読み書きのアクセス、認証されたユーザーに対しては読み込みアクセスを付与します。

クラスターを作成するには、必要な Simple Storage Service (Amazon S3) バケットが存在していなければなりません。クラスターで参照される必要なスクリプトまたはデータはすべて、Simple Storage Service (Amazon S3) にアップロードする必要があります。次の表では、サンプルデータ、スクリプト、およびログファイルの場所について説明しています。

Simple Storage Service (Amazon S3) 用のマルチパートアップロードを設定する

Amazon EMR は AWS SDK for Java による Amazon S3 マルチパートアップロードをサポートしています。マルチパートアップロードを使用すると、単一のオブジェクトをパートのセットとしてアップロードすることができます。これらのオブジェクトパートは、任意の順序で個別にアップロードできます。いずれかのパートの送信が失敗すると、他のパートに影響を与えることなくそのパートを再送することができます。オブジェクトのすべてのパートがアップロードされたら、Simple Storage Service (Amazon S3) はこれらのパートを組み立ててオブジェクトを作成します。

詳細については、Amazon Simple Storage Service ユーザーガイドのマルチパートアップロードの概要を参照してください。

さらに、Amazon EMR には、失敗したマルチパートアップロードのパートのクリーンアップをより正確に制御できるプロパティも用意されています。

マルチパートアップロードのための Amazon EMR の設定プロパティを次の表で説明します。これらは、`core-site` 設定分類を使用して設定することができます。詳細については、「Amazon EMR リリースガイド」の「アプリケーションの設定」を参照してください。

設定パラメータ名	デフォルト値	説明
`fs.s3n.multipart.uploads.enabled`	`true`	マルチパートアップロードを有効にするかどうかを示すブールタイプ。EMRFS の整合性のあるビューが有効になっていると、マルチパートアップロードがデフォルトで有効になっており、この値を `false` に設定しても無視されます。
`fs.s3n.multipart.uploads.split.size`	`134217728`	マルチパートアップロードが有効になっている場合に、EMRFS で新しいパートのアップロードを開始する前に、パートの最大サイズ (バイト単位) を指定します。最小値は `5242880` (5 MB) です。指定した値が小さい場合は、`5242880` が使用されます。最

設定パラメータ名	デフォルト値	説明
		大数は 5368709120 (5 GB) です。指定した値が大きい場合は、5368709120 が使用されます。 EMRFS のクライアント側で暗号化が無効になっていて、Simple Storage Service (Amazon S3) 向けに最適化されたコミッターも無効になっている場合は、この値によって、EMRFS でファイルのアップロードに PutObject リクエストではなくマルチパートアップロードを使用するようになるまでの、許容されるデータファイルの最大サイズも制御されます。詳細については、「」を参照してください。
fs.s3n.ssl.enabled	true	http と https のどちらを使用するかを示すブールタイプです。
fs.s3.buckets.create.enabled	false	バケットが存在しない場合に作成する必要があるかどうかを示すブールタイプです。false に設定すると、CreateBucket 操作で例外が発生します。
fs.s3.multipart.clean.enabled	false	未完了のマルチパートアップロードの定期的なバックグラウンドクリーンアップを有効にするかどうかを示すブールタイプです。
fs.s3.multipart.clean.age.threshold	604800	マルチパートアップロードがクリーンアップの対象となるまでの最小期間 (秒単位) を指定する long 型です。デフォルトは 1 週間です。

設定パラメータ名	デフォルト値	説明
fs.s3.multipart.clean.jitter.max	10000	次回クリーンアップをスケジュールする前の 15 分間の固定遅延に追加するランダムのジッター遅延時間 (秒単位) の最大数を指定する整数型です。

マルチパートアップロードの無効化

Console

コンソールでマルチパートアップロードを無効にするには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で、[クラスター] を選択し、[クラスターの作成] を選択します
3. [ソフトウェアの設定] の下で、次の設定を入力します: `classification=core-site,properties=[fs.s3n.multipart.uploads.enabled=false]`。
4. クラスターに適用するその他のオプションを選択します。
5. クラスターを起動するには、[クラスターの作成] を選択します。

CLI

を使用してマルチパートアップロードを無効にするには AWS CLI

この手順では、 AWS CLIを使用してマルチパートアップロードを無効にする方法を説明します。マルチパートアップロードを無効にするには、`create-cluster` コマンドを入力し、`--bootstrap-actions` パラメータを指定します。

1. 次の内容で myConfig.json ファイルを作成してコマンドを実行するのと同じディレクトリに保存します。

```
[
  {
    "Classification": "core-site",
    "Properties": {
      "fs.s3n.multipart.uploads.enabled": "false"
```

```
    }
  }
]
```

2. 次のコマンドを入力し、*myKey* を EC2 キーペアの名前に置き換えます。

> **Note**
>
> 読みやすくするために、Linux 行連続文字 (\) が含まれています。Linux コマンドでは、これらは削除することも、使用することもできます。Windows の場合、削除するか、キャレット (^) に置き換えてください。

```
aws emr create-cluster --name "Test cluster" \
--release-label emr-7.1.0 --applications Name=Hive Name=Pig \
--use-default-roles --ec2-attributes KeyName=myKey --instance-type m5.xlarge \
--instance-count 3 --configurations file://myConfig.json
```

API

API を使用してマルチパートアップロードを無効にするには

- プログラムによる Simple Storage Service (Amazon S3) マルチパートアップロードについては、「Amazon Simple Storage Service ユーザーガイド」の「マルチパートアップロードに AWS SDK for Java を使用する」を参照してください。

  AWS SDK for Java の詳細については、AWS 「 SDK for Java」を参照してください。

ベストプラクティス

EMR クラスターで Simple Storage Service (Amazon S3) バケットを使用するための推奨事項を次に示します。

バージョニングの有効化

バージョニングは、Amazon S3 バケット用の推奨設定です。バージョニングを有効にすると、データが誤って削除または上書きされても復元できます。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「バージョニングの使用」を参照してください。

失敗したマルチパートアップロードをクリーンアップする

EMR クラスターコンポーネントは、Amazon S3 APIs で AWS SDK for Java を介したマルチパートアップロードを使用して、デフォルトでログファイルと出力データを Amazon S3 に書き込みます。Amazon EMR を使用した、この設定に関連するプロパティの変更の詳細については、「Simple Storage Service (Amazon S3) 用のマルチパートアップロードを設定する」を参照してください。大きなファイルのアップロードでは、Simple Storage Service (Amazon S3) マルチパートアップロードが未完了になることがあります。マルチパートアップロードを正常に完了できないと、進行中のマルチパートアップロードによって継続的にバケットが使用され、ストレージ料金が発生します。過剰なファイルストレージを避けるために以下のオプションをお勧めします。

- Amazon EMR で使用するバケットについては、Simple Storage Service (Amazon S3) でライフサイクル設定ルールを使用して、アップロードの開始日から 3 日後に未完了のマルチパートアップロードを削除します。ライフサイクル設定ルールを使用すると、オブジェクトのストレージクラスと有効期限を制御できます。詳細については、「オブジェクトのライフサイクル管理」および「バケットライフサイクルポリシーを使用した不完全なマルチパートアップロードの中止」を参照してください。
- `fs.s3.multipart.clean.enabled` を `true` に設定し、他のクリーンアップパラメータをチューニングすることで、Amazon EMR のマルチパートクリーンアップ機能を有効にします。この機能は、大容量、大規模、および稼働時間が限られているクラスターで便利です。この場合は、ライフサイクル設定ルールの `DaysAfterIntitiation` パラメータが、最小に設定しても長すぎるため、Simple Storage Service (Amazon S3) ストレージでスパイクを発生させています。Amazon EMR のマルチパートクリーンアップでは、より正確に制御できます。詳細については、「Simple Storage Service (Amazon S3) 用のマルチパートアップロードを設定する」を参照してください。

バージョンマーカーを管理する

Simple Storage Service (Amazon S3) でライフサイクル設定ルールを有効にして、Amazon EMR で使用するバージョニング対応バケットについて期限切れのオブジェクト削除マーカーを削除することをお勧めします。バージョニング対応のバケットからオブジェクトを削除すると、削除マーカーが作成されます。オブジェクトの以前のバージョンすべてがその後有効期限切れになると、有効期限が切れたオブジェクトの削除マーカー 1 つがバケット内に残ります。これらの削除マーカーに対する料金はかかりませんが、期限切れのマーカーを削除すると LIST リクエストのパフォーマンスが向上する可能性があります。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「バージョニングが有効なバケットのライフサイクル設定」を参照してください。

パフォーマンスに関するベストプラクティス

EMR クラスターやこれらのクラスターのアプリケーションでは、使い方やワークロードに応じて、バケットに対する多数のリクエストが発生することがあります。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「リクエスト率とパフォーマンスに関する考慮事項」を参照してください。

Amazon S3 Express One Zone にデータをアップロードする

概要

Amazon EMR 6.15.0 以降では、Amazon EMR と Apache Spark を Amazon S3 Express One Zone ストレージクラスと組み合わせて使用することで、Spark ジョブのパフォーマンスを向上させることができます。S3 Express One Zone は、1 秒間に数十万件単位のリクエストによりデータに頻繁にアクセスするようなアプリケーション用の S3 ストレージクラスです。リリース時点で、S3 Express One Zone は、Amazon S3 の中でレイテンシーが最も低く、パフォーマンスの最も高いクラウドオブジェクトストレージを提供しています。

前提条件

- S3 Express One Zone のアクセス許可 - S3 Express One Zone が S3 オブジェクトに対して GET、LIST、PUT などのアクションを最初に実行すると、ストレージクラスがユーザーに代わって CreateSession を呼び出します。S3A コネクタが CreateSession API を呼び出せるように、お使いの IAM ポリシーで s3express:CreateSession アクセス許可を付与する必要があります。このアクセス許可ポリシーの例については、「Amazon S3 Express One Zone の使用を開始する」を参照してください。
- S3A コネクタ - S3 Express One Zone ストレージクラスを使用する Amazon S3 バケットのデータにアクセスするように Spark クラスターを設定するには、Apache Hadoop コネクタ S3A を使用する必要があります。コネクタを使用するには、すべての S3 URI が s3a スキームを使用していることを確認してください。使用していない場合は、s3 スキームと s3n スキーム用にファイルシステム実装を変更してください。

s3 スキームを変更するには、以下のクラスター設定を指定します。

```
[
  {
    "Classification": "core-site",
    "Properties": {
      "fs.s3.impl": "org.apache.hadoop.fs.s3a.S3AFileSystem",
      "fs.AbstractFileSystem.s3.impl": "org.apache.hadoop.fs.s3a.S3A"
```

```
    }
  }
]
```

s3n スキームを変更するには、以下のクラスター設定を指定します。

```
[
  {
    "Classification": "core-site",
    "Properties": {
      "fs.s3n.impl": "org.apache.hadoop.fs.s3a.S3AFileSystem",
      "fs.AbstractFileSystem.s3n.impl": "org.apache.hadoop.fs.s3a.S3A"
    }
  }
]
```

Amazon S3 Express One Zone の使用を開始する

トピック

- アクセス権ポリシーを作成する
- クラスターを作成および設定する
- 設定の概要

アクセス権ポリシーを作成する

Amazon S3 Express One Zone を使用するクラスターを作成する前に、クラスターの Amazon EC2 インスタンスプロファイルにアタッチする IAM ポリシーを作成する必要があります。ポリシーには S3 Express One Zone ストレージクラスにアクセスするためのアクセス許可が必要です。次のポリシー例は、必要なアクセス許可を付与する方法を示します。ポリシーを作成したら、クラスターを作成および設定する セクションで説明されているように、お使いの EMR クラスターの作成に使用するインスタンスプロファイルロールにポリシーをアタッチします。

```
{
  "Version":"2012-10-17",
  "Statement": [
        {
            "Effect": "Allow",
            "Resource": "arn:aws:s3express:region-code:account-id:bucket/DOC-EXAMPLE-
BUCKET",
            "Action": [
```

```
                "s3express:CreateSession"
            ]
        }
    ]

}
```

クラスターを作成および設定する

次に、S3 Express One Zone で Spark を実行するクラスターを作成します。以下の手順では、 AWS Management Consoleにクラスターを作成するための大まかな概要を説明します。

1. Amazon EMR コンソールに移動し、サイドバーから [クラスター] を選択します。次に、[クラスターを作成] を選択します。
2. Amazon EMR リリース `emr-6.15.0` 以降を選択します。
3. [Spark インタラクティブ] アプリケーションバンドルを選択し、クラスターに含める他のアプリケーションを選択します。少なくとも Spark および Hadoop をクラスターに組み込む必要があります。
4. Amazon S3 Express One Zone を有効にするには、[ソフトウェア設定] セクションで次の例のような設定を入力します。設定と推奨値については、この手順の後の 設定の概要 セクションで説明しています。

```
[
  {
    "Classification": "core-site",
    "Properties": {
      "fs.s3a.aws.credentials.provider":
 "software.amazon.awssdk.auth.credentials.InstanceProfileCredentialsProvider",
      "fs.s3a.change.detection.mode": "none",
      "fs.s3a.endpoint.region": "aa-example-1",
      "fs.s3a.select.enabled": "false"
    }
  },
  {
    "Classification": "spark-defaults",
    "Properties": {
      "spark.sql.sources.fastS3PartitionDiscovery.enabled": "false"
    }
  }
]
```

5. [Amazon EMR 用 EC2 インスタンスプロファイル] セクションで、既存のロールを使用するように選択し、上記の アクセス権ポリシーを作成する セクションで作成したポリシーがアタッチされたロールを使用します。
6. 残りのクラスター設定をアプリケーションに合わせて設定し、[クラスターを作成] を選択します。

設定の概要

次の表では、クラスターを作成および設定する セクションで説明されているように、Amazon EMR で S3 Express One Zone を使用するクラスターをセットアップするときに指定する必要がある設定と推奨値について説明しています。

S3A 設定

パラメータ	デフォルト値	推奨値	説明
`fs.s3a.aws.credentials.provider`	指定しない場合は、`TemporaryAWSCredentialsProvider` 、`SimpleAWSCredentialsProvider` 、`EnvironmentVariableCredentialsProvider` 、`IAMInstanceCredentialsProvider` の順序で `AWSCredentialProviderList` が使用されます。	`software.amazon.awssdk.auth.credentials.InstanceProfileCredentialsProvider`	Amazon EMR インスタンスプロファイルロールには、S3A ファイルシステムが `s3express:CreateSession` を呼び出すことを許可するポリシーが必要です。S3 Express One Zone のアクセス許可があれば、他の認証情報プロバイダーも機能します。
	null		

パラメータ	デフォルト値	推奨値	説明
fs.s3a.endpoint.region		バケットを AWS リージョン 作成した 。	リージョン解決ロジックは S3 Express One Zone ストレージクラスでは機能しません。
fs.s3a.select.enabled	true	false	Amazon S3 select は S3 Express One Zone ストレージクラスではサポートされていません。
fs.s3a.change.detection.mode	server	なし	S3A による変更検出は、MD5 ベースの etags をチェックすることで動作します。S3 Express One Zone ストレージクラスは MD5 checksums をサポートしていません。

Spark 設定

パラメータ	デフォルト値	推奨値	説明
spark.sql.sources.fastS3PartitionDiscovery.enabled	true	false	内部最適化は、S3 Express One Zone ストレージクラスではサポートされていない S3 API パラメータを使用します。

パラメータ	デフォルト値	推奨値	説明

考慮事項

Amazon EMR 上の Apache Spark を S3 Express One Zone ストレージクラスと統合する場合は、次の点を考慮してください。

- Amazon S3 Express One Zone は、Amazon EMR リリース 6.15.0 以降でサポートされています。
- Amazon EMR で S3 Express One Zone を使用するには、S3A コネクタが必要です。S3 Express One Zone との通信に必要な機能とストレージクラスを備えているのは S3A だけです。コネクタを設定する手順については、「the section called “前提条件”」を参照してください。
- Amazon S3 Express One Zone ストレージクラスは、Amazon EC2 上で実行される Amazon EMR クラスター上の Spark でのみサポートされます。
- Amazon S3 Express One Zone ストレージクラスは SSE-S3 の暗号化のみをサポートします。詳細については、「Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3)」を参照してください。
- Amazon S3 Express One Zone ストレージクラスは S3A `FileOutputCommitter` での書き込みをサポートしていません。S3 Express One Zone のバケットに S3A `FileOutputCommitter` を使用して書き込みを行うと、エラー InvalidStorageClass: The storage class you specified is not valid が発生します。
- Amazon S3 Express One Zone ストレージクラスは、Amazon EMR Serverless または EKS 上の Amazon EMR ではサポートされていません。

AWS DataSyncでのデータのアップロード

AWS DataSync は、オンプレミスストレージとストレージサービス間、または AWS ストレージサービス間でデータを移動するプロセスを簡素化、自動化、および高速化するオンラインデータ転送 AWS サービスです。 は、Hadoop Distributed File System (HDFS)、NAS ファイルサーバー、セルフマネージドオブジェクトストレージなどのさまざまなオンプレミスストレージシステム DataSync をサポートします。

クラスターにデータを入力する最も一般的な方法は、Simple Storage Service (Amazon S3) にデータをアップロードし、Amazon EMR の組み込み機能を使用してクラスターにデータをロードするというものです。

DataSync は、次のタスクを実行するのに役立ちます。

- ビジネス継続性を実現するために、Hadoop クラスター上の HDFS を Simple Storage Service (Amazon S3) にレプリケートする
- HDFS を Simple Storage Service (Amazon S3) にコピーして、データレイクに入力する
- 分析と処理のために Hadoop クラスターの HDFS と Simple Storage Service (Amazon S3) の間でデータを転送する

S3 バケットにデータをアップロードするには、まずオンプレミスストレージと同じネットワークに 1 つ以上の DataSync エージェントをデプロイします。エージェントは、セルフマネージドの場所からデータを読み取ったり、そこにデータを書き込むために使用される仮想マシン (VM) です。次に、S3 バケット AWS リージョン がある AWS アカウント と でエージェントをアクティブ化します。

エージェントがアクティブ化されたら、オンプレミスストレージの送信元の場所、S3 バケットの送信先の場所、およびタスクを作成します。タスクは、2 つの場所 (送信元と送信先) からなる一式とタスクの動作を制御するために使用する一連のデフォルトオプションです。

最後に、 DataSync タスクを実行して、送信元から送信先にデータを転送します。

詳細については、「AWS DataSyncの開始方法」を参照してください。

分散キャッシュによるファイルのインポート

トピック
- [サポートされているファイルの種類]
- キャッシュしたファイルの場所
- ストリーミングアプリケーションからのキャッシュしたファイルへのアクセス
- ストリーミングアプリケーションからのキャッシュしたファイルへのアクセス

DistributedCache は、マップまたはリデュースタスクで共通のデータにアクセスする必要があるときに効率を高められる Hadoop の機能です。クラスターが依存する既存のアプリケーションまたはバイナリがクラスターの作成時にインストールされていない場合は、DistributedCache を使用するとそれらのファイルをインポートできます。この機能により、クラスターノードは、ファイルを他のクラスターノードから取得する代わりに、ローカルファイルシステムからインポートされたファイルを読み取ることができます。

詳細については、http://hadoop.apache.org/docs/stable/api/org/apache/hadoop/filecache/DistributedCache.html を参照してください。

クラスターの作成時に DistributedCache を呼び出します。ファイルは、Hadoop ジョブが起動される直前にキャッシュされ、ジョブが終わるまでキャッシュ内に置かれます。キャッシュしたファイルは、HDFS や Simple Storage Service (Amazon S3) など、Hadoop 対応のファイルシステムに置くことができます。ファイルキャッシュのデフォルトサイズは 10 GB です。キャッシュのサイズを変更するには、`local.cache.size` ブートストラップアクションを使用して Hadoop のパラメータの設定を変更します。詳細については、「追加のソフトウェアをインストールするためのブートストラップアクションの作成」を参照してください。

[サポートされているファイルの種類]

DistributedCache では、単一のファイルとアーカイブの両方が使用できます。個別のファイルは、読み取り専用としてキャッシュされます。実行可能ファイルおよびバイナリファイルには、実行許可が設定されます。

アーカイブは、`gzip` などのユーティリティを使用してパッケージ化された 1 つ以上のファイルの集まりです。DistributedCache が圧縮されたファイルを各コアノードに渡し、キャッシュの一部としてアーカイブを解凍します。DistributedCacheは、以下の圧縮形式をサポートしています。

- zip
- tgz
- tar.gz
- tar
- jar

キャッシュしたファイルの場所

DistributedCache はファイルをコアノードのみにコピーします。クラスター内にコアノードが存在しない場合、DistributedCache はファイルをプライマリノードにコピーします。

DistributedCache は、symlink を使用することにより、キャッシュされたファイルをマッパーおよびリデューサーの現在の作業ディレクトリに関連付けます。symlink はファイルの場所のエイリアスであり、実際のファイルの場所ではありません。パラメータの値 (`yarn-site.xml` の `yarn.nodemanager.local-dirs`) は、一時ファイルの場所を示します。Amazon EMR はこのパラメータを `/mnt/mapred` か、またはインスタンスタイプと EMR バージョンに基づいていくつかのバリエーションに設定します。たとえば、インスタンスタイプに 2 つのエフェメラルボリュームが含まれているため、設定に `/mnt/mapred` と `/mnt1/mapred` が含まれていることがあります。キャッシュファイルは、一時ファイルが置かれる場所のサブディレクトリ `/mnt/mapred/taskTracker/archive` に置かれます。

1 つのファイルをキャッシュすると、ファイルは DistributedCache によって `archive` ディレクトリに置かれます。アーカイブをキャッシュすると、DistributedCache はファイルを解凍し、`/archive` 内にアーカイブファイルと同じ名前でサブディレクトリを作成します。各ファイルは、新しいサブディレクトリ内に置かれます。

DistributedCache は、ストリーミングを使用するときにのみ使用できます。

ストリーミングアプリケーションからのキャッシュしたファイルへのアクセス

キャッシュしたファイルにマッパーまたはリデューサーアプリケーションからアクセスするには、現在の作業ディレクトリ (./) をアプリケーションのパスに追加し、キャッシュしたファイルが現在の作業ディレクトリにあるかのように参照していることを確認してください。

ストリーミングアプリケーションからのキャッシュしたファイルへのアクセス

AWS Management Console および を使用して AWS CLI 、分散キャッシュを使用するクラスターを作成できます。

> **Note**
>
> Amazon EMR コンソールは、再設計され、使いやすくなりました。古いコンソールと新しいコンソールエクスペリエンスの違いについては、「Amazon EMR コンソール」を参照してください。

New console

新しいコンソールで分散キャッシュファイルを指定するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で、[クラスター] を選択し、[クラスターの作成] を選択します
3. [ステップ] で [ステップの追加] を選択します。[ステップの追加]] ダイアログが開きます。[引数] フィールドに、キャッシュに保存するファイルおよびアーカイブを指定します。ファイルのサイズ (またはアーカイブに含まれるファイルのサイズの合計) は、割り当てられているキャッシュサイズ未満でなければなりません。

個々のファイルを分散キャッシュに追加する場合は、-cacheFile、ファイルの名前と場所、ポンド記号 (#)、およびファイルをローカルキャッシュに置いたときに付ける名前を順番に指定します。次の例は、個々のファイルを分散キャッシュに追加する方法を示します。

```
-cacheFile \
s3://DOC-EXAMPLE-BUCKET/file-name#cache-file-name
```

アーカイブファイルを分散キャッシュに追加する場合は、-cacheArchive、Amazon S3 でのファイルの場所、ポンド記号 (#)、およびファイルの集合をローカルキャッシュに置いたときに付ける名前を順番に入力します。次の例は、アーカイブファイルを分散キャッシュに追加する方法を示します。

```
-cacheArchive \
s3://DOC-EXAMPLE-BUCKET/archive-name#cache-archive-name
```

その他のダイアログフィールドに適切な値を入力します。オプションは、ステップタイプによって異なります。ステップを追加してダイアログを終了するには、[ステップの追加] を選択します。

4. クラスターに適用するその他のオプションを選択します。
5. クラスターを起動するには、[クラスターの作成] を選択します。

Old console

古いコンソールで分散キャッシュファイルを指定するには

1. 新しい Amazon EMR コンソールに移動し、サイドナビゲーションから [古いコンソールに切り替え] を選択します。古いコンソールに切り替えたときの動作の詳細については、「Using the old console」を参照してください。
2. [クラスターを作成] を選択します。
3. 起動モードとして [Step execution (ステップ実行)] を選択します。
4. [ステップ] セクションの [Add step (ステップの追加)] フィールドで、リストから [Streaming program (ストリーミングプログラム)] を選択し、[Configure and add (設定と追加)] をクリックします。

5. [引数] フィールドに、キャッシュに保存するファイルおよびアーカイブを指定し、[追加] を選択します。ファイルのサイズ (またはアーカイブに含まれるファイルのサイズの合計) は、割り当てられているキャッシュサイズ未満でなければなりません。

   個々のファイルを分散キャッシュに追加する場合は、-cacheFile、ファイルの名前と場所、ポンド記号 (#)、およびファイルをローカルキャッシュに置いたときに付ける名前を順番に指定します。次の例は、個々のファイルを分散キャッシュに追加する方法を示します。

   ```
   -cacheFile \
   s3://DOC-EXAMPLE-BUCKET/file_name#cache_file_name
   ```

   アーカイブファイルを分散キャッシュに追加する場合は、-cacheArchive、Amazon S3 でのファイルの場所、ポンド記号 (#)、およびファイルの集合をローカルキャッシュに置いたときに付ける名前を順番に入力します。次の例は、アーカイブファイルを分散キャッシュに追加する方法を示します。

   ```
   -cacheArchive \
   s3://DOC-EXAMPLE-BUCKET/archive_name#cache_archive_name
   ```

6. クラスターの設定と起動に進みます。クラスターでは、キャッシュの場所にファイルをコピーしてから、クラスターステップを処理します。

CLI

を使用して分散キャッシュファイルを指定するには AWS CLI

- クラスターの作成時にストリーミングステップを送信するには、create-cluster コマンドを入力し、--steps パラメータを指定します。を使用して分散キャッシュファイルを指定するには AWS CLI、ストリーミングステップを送信するときに適切な引数を指定します。

  個々のファイルを分散キャッシュに追加する場合は、-cacheFile、ファイルの名前と場所、ポンド記号 (#)、およびファイルをローカルキャッシュに置いたときに付ける名前を順番に指定します。

  アーカイブファイルを分散キャッシュに追加する場合は、-cacheArchive、Amazon S3 でのファイルの場所、ポンド記号 (#)、およびファイルの集合をローカルキャッシュに置いたときに付ける名前を順番に入力します。次の例は、アーカイブファイルを分散キャッシュに追加する方法を示します。

での Amazon EMR コマンドの使用の詳細については AWS CLI、「」を参照してくださいhttps://docs.aws.amazon.com/cli/latest/reference/emr。

Example 1

次のコマンドを入力して、クラスターを起動し、ストリーミングステップを送信します。このコマンドでは -cacheFile を使用して、sample_dataset_cached.dat というファイルをキャッシュに追加します。

```
aws emr create-cluster --name "Test cluster" --release-label emr-4.0.0 --
applications Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey
 --instance-type m5.xlarge --instance-count 3 --steps Type=STREAMING,Name="Streaming
 program",ActionOnFailure=CONTINUE,Args=["--files","s3://my_bucket/my_mapper.py
 s3://my_bucket/my_reducer.py","-mapper","my_mapper.py","-reducer","my_reducer.py,"-
input","s3://my_bucket/my_input","-output","s3://my_bucket/my_output", "-
cacheFile","s3://my_bucket/sample_dataset.dat#sample_dataset_cached.dat"]
```

--instance-groups パラメータを使用せずにインスタンス数を指定すると、1 つのプライマリノードが起動され、残りのインスタンスはコアノードとして起動されます。すべてのノードで、コマンドで指定したインスタンスタイプが使用されます。

以前にデフォルトの EMR サービスロールと EC2 インスタンスプロファイルを作成していない場合は、「aws emr create-default-roles」と入力してそれらを作成してから、create-cluster サブコマンドを入力します。

Example 2

次のコマンドでは、ストリーミングクラスターを作成し、-cacheArchive を使用してファイルのアーカイブを 1 つキャッシュに追加しています。

```
aws emr create-cluster --name "Test cluster" --release-label emr-4.0.0 --
applications Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey
 --instance-type m5.xlarge --instance-count 3 --steps Type=STREAMING,Name="Streaming
 program",ActionOnFailure=CONTINUE,Args=["--files","s3://my_bucket/my_mapper.py
 s3://my_bucket/my_reducer.py","-mapper","my_mapper.py","-reducer","my_reducer.py,"-
input","s3://my_bucket/my_input","-output","s3://my_bucket/my_output", "-
cacheArchive","s3://my_bucket/sample_dataset.tgz#sample_dataset_cached"]
```

`--instance-groups` パラメータを使用せずにインスタンス数を指定すると、1 つのプライマリノードが起動され、残りのインスタンスはコアノードとして起動されます。すべてのノードで、コマンドで指定したインスタンスタイプが使用されます。

以前にデフォルトの EMR サービスロールと EC2 インスタンスプロファイルを作成していない場合は、「`aws emr create-default-roles`」と入力してそれらを作成してから、`create-cluster` サブコマンドを入力します。

圧縮ファイルの処理方法

Hadoop は、ファイル拡張子をチェックして圧縮ファイルを検出します。Hadoop でサポートされている圧縮タイプは、gzip、bzip2、LZO です。これらのタイプの圧縮を使用してファイルを抽出する際は、追加のアクションは不要です。Hadoop が処理します。

LZO ファイルのインデックスを作成するには、https://github.com/kevinweil/hadoop-lzo からダウンロード可能な hadoop-lzo ライブラリを使用できます。これはサードパーティーライブラリであるため、Amazon EMR はこのツールを使用する方法について開発者サポートを提供しないことに注意してください。使用法については、hadoop-lzo の readme ファイルを参照してください。

DynamoDB データを Hive にインポートする

Amazon EMR によって提供される Hive の実装には、DynamoDB と Amazon EMR クラスターの間でデータのインポートおよびエクスポートを行う場合に使用できる機能が含まれています。これは、入力データが DynamoDB に保存されている場合に役に立ちます。詳細については、「Amazon EMR による DynamoDB 内テーブルのエクスポート、インポート、クエリ、結合」を参照してください。

AWS Direct Connectでデータに接続する

AWS Direct Connect は、データセンター、オフィス、またはコロケーション環境から Amazon Web Services へのプライベート専用ネットワーク接続を確立するために使用できるサービスです。大量の入力データがある場合、 AWS Direct Connect を使用すると、ネットワークコストを削減し、帯域幅スループットを向上させ、インターネットベースの接続よりも一貫したネットワークエクスペリエンスを提供できます。詳細については、「AWS Direct Connect ユーザーガイド」を参照してください。

AWS Snowballで大量のデータをアップロードする

AWS Snowball は、Amazon Simple Storage Service (Amazon S3) とオンサイトのデータストレージロケーション間で大量のデータを高速 faster-than-internetに転送するために使用できるサービスで

す。Snowball では、インポートジョブとエクスポートジョブの 2 つのジョブタイプをサポートしています。インポートジョブには、オンプレミスのソースから Simple Storage Service (Amazon S3) バケットへのデータ転送が含まれます。エクスポートジョブには、Simple Storage Service (Amazon S3) バケットからオンプレミスのソースへのデータ転送が含まれます。どちらのジョブタイプでも、地域の配送業者が Simple Storage Service (Amazon S3) とお客様のオンサイトデータストレージロケーション間を輸送する間、Snowball デバイスは安全であり、データが保護されます。Snowball デバイスは物理的に堅牢で、 AWS Key Management Service () によって保護されていますAWS KMS。詳細については、「AWS Snowball Edge デベロッパーガイド」を参照してください。

出力場所を設定する

Amazon EMR クラスターの最も一般的な出力形式は、圧縮または非圧縮形式のテキストファイルです。一般に、これらは Simple Storage Service (Amazon S3) バケットに書き込まれます。このバケットは、クラスターを起動する前に作成しておく必要があります。クラスターの起動時に、出力場所として S3 バケットを指定します。

詳細については、次のトピックを参照してください。

トピック

- Amazon S3 バケットの作成と設定
- Amazon EMR が返すことができる形式
- 所有していない Amazon S3 バケットにデータを書き込む方法
- クラスターの出力を圧縮する

Amazon S3 バケットの作成と設定

Amazon EMR (Amazon EMR) は、Simple Storage Service (Amazon S3) を使用して入力データ、ログファイル、および出力データを保存します。Simple Storage Service (Amazon S3) は、これらのストレージロケーションをバケットとして参照します。バケットには、Simple Storage Service (Amazon S3) と DNS の要件に従って一定の制約と制限があります。詳細については、「Amazon Simple Storage Service デベロッパーガイド」の「バケットの制約と制限」を参照してください。

Simple Storage Service (Amazon S3) バケットを作成するには、「Amazon Simple Storage Service デベロッパーガイド」の「バケットの作成」の手順に従います。

> **Note**
>
> [Create a Bucket] (バケットの作成) ウィザードでロギングを有効にした場合、有効になるのはバケットアクセスログのみで、クラスターログは有効にはなりません。

> **Note**
>
> リージョン固有のバケットの指定の詳細については、「Amazon Simple Storage Service デベロッパーガイド」の「バケットとリージョン」およびAWS SDKs」を参照してください。

バケットを作成したら、そこに適切なアクセス許可を設定できます。通常、お客様自身 (所有者) に読み取りと書き込みのアクセス権限を付与します。バケットを設定するには、「Amazon S3 のセキュリティベストプラクティス」に従うことを強くお勧めします。

クラスターを作成するには、必要な Simple Storage Service (Amazon S3) バケットが存在していなければなりません。クラスターで参照される必要なスクリプトまたはデータはすべて、Simple Storage Service (Amazon S3) にアップロードする必要があります。次の表では、サンプルデータ、スクリプト、およびログファイルの場所について説明しています。

情報	Simple Storage Service (Amazon S3) の場所の例
スクリプトまたはプログラム	`s3://DOC-EXAMPLE-BUCKET1/script/MapperScript.py`
ログファイル	`s3://DOC-EXAMPLE-BUCKET1/logs`
入力データ	`s3://DOC-EXAMPLE-BUCKET1/input`
出力データ	`s3://DOC-EXAMPLE-BUCKET1/output`

Amazon EMR が返すことができる形式

クラスターのデフォルトの出力形式は、テキストファイルの個々の行にキー、値のペアが書き込まれたテキストです。これは、最も良く使われている出力形式です。

出力データをデフォルトのテキストファイル以外の形式で書き込む必要がある場合は、Hadoop インターフェイス `OutputFormat` を使用して、他の出力タイプを指定できます。カスタムデータの種類を処理するために、`FileOutputFormat` クラスのサブクラスを作成することもできます。詳細については、http://hadoop.apache.org/docs/current/api/org/apache/hadoop/mapred/OutputFormat.html を参照してください。

Hive クラスターを起動する場合は、シリアライザ/デシリアライザ (SerDe) を使用して HDFS から特定の形式にデータを出力できます。詳細については、https://cwiki.apache.org/confluence/display/Hive/SerDe を参照してください。

所有していない Amazon S3 バケットにデータを書き込む方法

Amazon Simple Storage Service (Amazon S3) バケットにファイルを書き込むと、デフォルトではお客様以外はそのファイルを読むことができません。これは、自分専用のバケットにファイルを書き込むことが前提になっており、このデフォルト設定によってファイルのプライバシーを保護しています。

ただし、クラスターを実行し、出力を別の AWS ユーザーの Amazon S3 バケットに書き込む場合、その AWS 他のユーザーがその出力を読み取れるようにするには、次の 2 つの操作を行う必要があります。

- AWS 他のユーザーに Amazon S3 バケットの書き込みアクセス許可を付与してもらいます。起動するクラスターは AWS 認証情報で実行されるため、起動するクラスターは他のユーザーの AWS バケットに書き込むこともできます。
- ユーザーまたはクラスターが Amazon S3 バケットに書き込むファイルに、 AWS 他のユーザーの読み取りアクセス許可を設定します。この読み取りアクセス許可を設定する最も簡単な方法は、Simple Storage Service (Amazon S3) で事前定義されているアクセスポリシーのセットである既定アクセスコントロールリスト (ACL) を使用することです。

他のユーザーが AWS 他のユーザーの Amazon S3 バケットにファイルを書き込むアクセス許可を付与する方法については、Amazon Simple Storage Service ユーザーガイドの「バケットのアクセス許可の編集」を参照してください。

お客様のクラスターが Simple Storage Service (Amazon S3) へのファイル書き込み時に既定の ACL を使用するようにするには、`fs.s3.canned.acl` クラスター設定オプションに、使用する既定の ACL を設定します。次の表に、現在定義されている既定 ACL のリストを示します。

既定 ACL	説明
AuthenticatedRead	所有者には Permission.FullControl が、GroupGrantee.AuthenticatedUsers グループには Permission.Read アクセスが与えられるように指定します。
BucketOwnerFullControl	バケットの所有者に Permission.FullControl が与えられるように指定します。バケットの所有者は、オブジェクトの所有者と同じであるとは限りません。
BucketOwnerRead	バケットの所有者に Permission.Read が与えられるように指定します。バケットの所有者は、オブジェクトの所有者と同じであるとは限りません。
LogDeliveryWrite	アクセスログを作成できるように、所有者には Permission.FullControl が、GroupGrantee.LogDelivery グループには Permission.Write アクセスが与えられるように指定します。
Private	所有者に Permission.FullControl が与えられるように指定します。
PublicRead	所有者には Permission.FullControl が、GroupGrantee.AllUsers グループには Permission.Read アクセスが与えられるように指定します。
PublicReadWrite	所有者には Permission.FullControl が、GroupGrantee.AllUsers グループには Permission.Read および Permission.Write アクセスが与えられるように指定します。

クラスターの設定オプションを設定する方法は、実行するクラスターのタイプに応じてさまざまです。以下の手順では、一般的なケースでオプションを設定する方法を示します。

Hive で既定 ACL を使用してファイルを書き込むには

- Hive のコマンドプロンプトで、クラスターが Simple Storage Service (Amazon S3) に書き込むファイルに対して設定する既定 ACL を `fs.s3.canned.acl` 設定オプションに設定します。Hive のコマンドプロンプトにアクセスするには、SSH を使用してマスターノード接続し、Hadoop のコマンドプロンプトに "Hive" と入力します。詳細については、「SSH を使用してプライマリノードに接続する」を参照してください。

  次の例では、`fs.s3.canned.acl` 設定オプションを `BucketOwnerFullControl` に設定することにより、Simple Storage Service (Amazon S3) バケットの所有者に対してファイルの完全なコントロールを許可しています。set コマンドでは大文字と小文字が区別され、引用符やスペースは使用できません。

```
hive> set fs.s3.canned.acl=BucketOwnerFullControl;
create table acl (n int) location 's3://acltestbucket/acl/';
insert overwrite table acl select count(*) from acl;
```

  例の最後の 2 行では、Simple Storage Service (Amazon S3) に保存されるテーブルを作成し、そのテーブルにデータを書き込みます。

Pig で既定 ACL を使用してファイルを書き込むには

- Pig のコマンドプロンプトで、クラスターが Simple Storage Service (Amazon S3) に書き込むファイルに対して設定する既定 ACL を `fs.s3.canned.acl` 設定オプションに設定します。Pig のコマンドプロンプトにアクセスするには、SSH を使用してマスターノード接続し、Hadoop のコマンドプロンプトに "Pig" と入力します。詳細については、「SSH を使用してプライマリノードに接続する」を参照してください。

  次の例では、設定`fs.s3.canned.acl`オプションを に設定し BucketOwnerFullControl、Amazon S3 バケットの所有者に ファイルに対する完全な制御を許可します。set コマンドでは、既定 ACL 名の前にスペースを 1 つ挿入し、引用符は使用しません。

```
pig> set fs.s3.canned.acl BucketOwnerFullControl;
store some data into 's3://acltestbucket/pig/acl';
```

カスタム JAR で既定 ACL を使用してファイルを書き込むには

- -D フラグを指定して Hadoop を使用して、`fs.s3.canned.acl` 設定オプションを設定します。これを次の例に示します。

```
hadoop jar hadoop-examples.jar wordcount
-Dfs.s3.canned.acl=BucketOwnerFullControl s3://mybucket/input s3://mybucket/output
```

クラスターの出力を圧縮する

トピック

- 出力データ圧縮
- 中間データ圧縮
- Amazon EMR での Snappy ライブラリの使用

出力データ圧縮

Hadoop ジョブの出力を圧縮します。 TextOutputFormat 結果を使用している場合、結果は gzip で囲まれたテキストファイルです。に書き込む SequenceFiles 場合、結果は内部的に圧縮 SequenceFile された になります。これは、mapred.output.compress を true に設定することによって有効にできます。

ストリーミングジョブを実行している場合は、そのストリーミングジョブにこれらの引数を渡すことで有効にできます。

```
-jobconf mapred.output.compress=true
```

また、ブートストラップアクションを使用することによって自動的にすべてのジョブ出力を圧縮することもできます。ここでは、Ruby クライアントでこれを行う方法を示します。

```
--bootstrap-actions s3://elasticmapreduce/bootstrap-actions/configure-hadoop \
--args "-s,mapred.output.compress=true"
```

最後に、Custom Jar を作成している場合は、ジョブの作成時に次の行を使用して出力圧縮を有効にすることができます。

```
FileOutputFormat.setCompressOutput(conf, true);
```

中間データ圧縮

マッパーからリデューサーに大量のデータを移動するジョブでは、中間圧縮を行うことでパフォーマンスが大幅に向上することがあります。マップ出力を圧縮し、コアノードに到達したらそれを解凍します。設定は、mapred.compress.map.output で行います。これは、出力圧縮についても同様に有効にすることができます。

Custom Jar を作成している場合は、次のコマンドを使用します。

```
conf.setCompressMapOutput(true);
```

Amazon EMR での Snappy ライブラリの使用

Snappy は高速圧縮/解凍ライブラリです。Amazon EMR AMI バージョン 2.0 以降で使用でき、中間圧縮用のデフォルトとして使用されています。Snappy の詳細については、http://code.google.com/p/snappy/ を参照してください。

プライマリノードの計画と設定

Amazon EMR クラスターを起動するときに、クラスター内に 1 つのプライマリノードを持つか、3 つのプライマリノードを持つか選択できます。インスタンスフリートの高可用性は、Amazon EMR リリース 5.36.1、5.36.2、6.8.1、6.9.1、6.10.1、6.11.1、6.12.0 以降でサポートされています。インスタンスグループの場合、Amazon EMR リリース 5.23.0 以降で高可用性がサポートされています。Amazon EMR は、クラスターの可用性をさらに向上させるために、Amazon EC2 プレイスメントグループを利用して、プライマリノードを個別の基盤ハードウェア上に配置させることができま

す。詳細については、「Amazon EMR と EC2 プレイスメントグループの統合」を参照してください。

複数のプライマリノードを持つ Amazon EMR クラスターには、次のような利点があります。

- プライマリノードは単一障害点ではなくなる。いずれかのプライマリノードに障害が発生した場合、クラスターは他の 2 つのプライマリノードを使用して、中断なしに実行される。その間に、Amazon EMR は障害が発生したプライマリノードを、同じ設定とブートストラップアクションでプロビジョニングされた新しいマスターノードに自動的に置き換えます。
- Amazon EMR は、HDFS NameNode と YARN の Hadoop 高可用性機能を有効に ResourceManager し、他のいくつかのオープンソースアプリケーションでの高可用性をサポートします。

  複数のプライマリノードを持つ Amazon EMR クラスターがオープンソースアプリケーションおよび他の Amazon EMR 機能をサポートする仕組みについては、「サポートされるアプリケーションと機能」を参照してください。

> **Note**
>
> クラスターは、1 つのアベイラビリティーゾーンまたはサブネットにのみ存在できます。

このセクションでは、サポートされるアプリケーションと複数のマスターノードを持つ Amazon EMR クラスターの機能に関する情報に加えて、設定の詳細、ベストプラクティス、クラスター起動の考慮事項について説明します。

トピック

サポートされるアプリケーションと機能

このトピックでは、Amazon EMR クラスター ResourceManager の HDFS NameNode と YARN の Hadoop 高可用性機能、および高可用性機能がオープンソースアプリケーションやその他の Amazon EMR 機能でどのように機能するかについて説明します。

高可用性 HDFS

複数のプライマリノードを持つ Amazon EMR クラスターにより、Hadoop 内の HDFS NameNode 高可用性機能が有効になります。詳細については、「HDFS High Availability」を参照してください。

Amazon EMR クラスターでは、2 つ以上の個別のノードが として設定されます NameNodes。1 つは NameNode `active`状態、もう 1 つは `standby`状態です。を持つノードが`active` NameNode 失敗すると、Amazon EMR は自動 HDFS フェイルオーバープロセスを開始します。を持つノードは `standby`NameNode になり`active`、クラスター内のすべてのクライアントオペレーションを引き継ぎます。Amazon EMR は障害が発生したノードを新しいノードで置き換え、`standby` として再結合します。

> **Note**
>
> 5.30.1 までの Amazon EMR バージョン 5.23.0 では、3 つのプライマリノードのうち HDFS を実行するのは 2 つだけです NameNode。

が であるかどうかを確認する必要がある場合 NameNode は`active`、SSH を使用してクラスター内の任意のプライマリノードに接続し、次のコマンドを実行できます。

```
hdfs haadmin -getAllServiceState
```

出力 NameNode には、 がインストールされているノードとそのステータスが一覧表示されます。例えば、 などです

```
ip-##-#-#-##1.ec2.internal:8020 active
ip-##-#-#-##2.ec2.internal:8020 standby
ip-##-#-#-##3.ec2.internal:8020 standby
```

高可用性 YARN ResourceManager

複数のプライマリノードを持つ Amazon EMR クラスターは、Hadoop で YARN ResourceManager 高可用性機能を有効にします。詳細については、ResourceManager「高可用性」を参照してください。

複数のプライマリノードを持つ Amazon EMR クラスターでは、YARN は 3 つのプライマリノードすべてで ResourceManager 実行されます。1 ResourceManager つは `active` 状態、

もう 2 つは standby状態です。のプライマリノードがactive ResourceManager 失敗すると、Amazon EMR は自動フェイルオーバープロセスを開始します。を持つプライマリノードstandby ResourceManager は、すべてのオペレーションを引き継ぎます。Amazon EMR は、障害が発生したプライマリノードを新しいプライマリノードに置き換え、ク ResourceManager ォーラムを として再結合しますstandby。

任意のプライマリノードの「http://*master-public-dns-name*:8088/cluster」に接続できます。これにより、自動的にactiveリソースマネージャーに誘導されます。どのリソースマネージャーが active であるかを確認するには、SSH を使用して、クラスターのいずれかのプライマリノードに接続します。続いて、次のコマンドを実行して 3 つのプライマリノードとそのステータスのリストを取得します。

```
yarn rmadmin -getAllServiceState
```

複数のプライマリノードを持つ Amazon EMR クラスターでサポートされているアプリケーション

複数のプライマリノードを持つ Amazon EMR クラスターには、次のアプリケーションをインストールして実行できます。アプリケーションごとに、プライマリノードのフェイルオーバープロセスは異なります。

アプリケーション	プライマリノードのフェイルオーバー中の可用性	メモ
Flink	プライマリノードのフェイルオーバーによって影響を受けない可用性	Amazon EMR での Flink ジョブは YARN アプリケーションとして実行されます。Flink のは、コアノード ApplicationMasters で YARN の としてJobManagers 実行されます。JobManager は、プライマリノードのフェイルオーバープロセスの影響を受けません。 Amazon EMR バージョン 5.27.0 以前を使用している場合、 JobManager は単一障害点です。が JobManager 失敗すると、すべてのジョブ状態が失われ、実行中のジョブは再開されません。アプリケーションの試行回数、チェックポイント、Flink の状態ストレージ ZooKeeper として を有効にすることで、 JobManager 高可

アプリケーション	プライマリノードのフェイルオーバー中の可用性	メモ
		用性を有効にできます。詳細については、「複数のプライマリノードがある Amazon EMR クラスターでの Flink の設定」を参照してください。 Amazon EMR バージョン 5.28.0 以降では、JobManager 高可用性を有効にするために手動設定は必要ありません。
Ganglia	プライマリノードのフェイルオーバーによって影響を受けない可用性	Ganglia はすべてのプライマリノードで利用できるため、Ganglia はプライマリノードのフェイルオーバープロセス中に継続して実行できます。
Hadoop	高可用性	アクティブなプライマリノードに障害が発生すると、HDFS NameNode と YARN ResourceManager は自動的にスタンバイノードにフェイルオーバーします。
HBase	高可用性	アクティブなプライマリノードで障害が発生した場合、HBase は自動的にスタンバイノードにフェイルオーバーします。 REST または Thrift サーバーを通じて HBase に接続している場合、アクティブなプライマリノードに障害が発生したときは、別のプライマリノードに切り替える必要があります。
HCatalog	プライマリノードのフェイルオーバーによって影響を受けない可用性	HCatalog は、クラスター外部に存在する Hive メタストア上に構築されます。HCatalog は、プライマリノードのフェイルオーバープロセス中も引き続き利用可能です。

アプリケーション	プライマリノードのフェイルオーバー中の可用性	メモ
JupyterHub	高可用性	JupyterHub は、3 つのプライマリインスタンスすべてにインストールされます。プライマリノードの障害時にノートブックが失われないように、ノートブックの永続性を設定することを強くお勧めします。詳細については、「Simple Storage Service (Amazon S3) でノートブックの永続性を設定する」を参照してください。
Livy	高可用性	Livy は 3 つすべてのプライマリノードにインストールされます。アクティブなプライマリノードで障害が発生した場合、現在の Livy セッションへのアクセスは失われ、別のプライマリノードまたは新しい代替ノードで新しい Livy セッションを作成する必要があります。
Mahout	プライマリノードのフェイルオーバーによって影響を受けない可用性	Mahout にはデーモンがないため、プライマリノードのフェイルオーバープロセスによる影響を受けません。
MXNet	プライマリノードのフェイルオーバーによって影響を受けない可用性	MXNet にはデーモンがないため、プライマリノードのフェイルオーバープロセスによる影響を受けません。
フェニックス	高可用性	Phoenix は、3 つのプライマリノードのいずれかでのみ QueryServer 実行されます。3 つのマスターすべての Phoenix は、Phoenix を接続するように設定されています QueryServer。`/etc/phoenix/conf/phoenix-env.sh` ファイルを使用して、Phoenix の QueryServer のプライベート IP を見つけることができます。

アプリケーション	プライマリノードのフェイルオーバー中の可用性	メモ
Pig	プライマリノードのフェイルオーバーによって影響を受けない可用性	Pig にはデーモンがないため、プライマリノードのフェイルオーバープロセスによる影響を受けません。
Spark	高可用性	すべての Spark アプリケーションは YARN コンテナで実行され、高可用性 YARN 機能と同じ方法で、プライマリノードのフェイルオーバーに対応できます。
Sqoop	高可用性	デフォルトでは、sqoop-job および sqoop-metastore は、コマンドを実行するマスターのローカルディスクにデータ (ジョブの説明) を保存します。外部データベースにメタストアデータを保存する場合は、Apache Sqoop ドキュメントを参照してください
Tez	高可用性	Tez コンテナは YARN で実行されるため、Tez はプライマリノードのフェイルオーバープロセス中は YARN と同じ方法で動作します。
TensorFlow	プライマリノードのフェイルオーバーによって影響を受けない可用性	TensorFlow にはデーモンがないため、プライマリノードのフェイルオーバープロセスの影響を受けません。

アプリケーション	プライマリノードのフェイルオーバー中の可用性	メモ
Zeppelin	高可用性	Zeppelin は 3 つすべてのプライマリノードにインストールされます。Zeppelin は、データの損失を防ぐために、デフォルトでノートとインタープリタの設定を HDFS に保存します。インタープリタセッションは、3 つすべてのプライマリインスタンス間で完全に分離されます。セッションデータは、マスターで障害が発生すると失われます。異なるプライマリインスタンスで同じノートを同時に変更しないことをお勧めします。
ZooKeeper	高可用性	ZooKeeper は、HDFS 自動フェイルオーバー機能の基盤です。 は、調整データの維持、そのデータの変更に関するクライアントへの通知、およびクライアントに障害のモニタリングを行うための高可用性サービス ZooKeeper を提供します。詳細については、「HDFS Automatic Failover」を参照してください。

複数のプライマリノードを含む Amazon EMR クラスターで以下のアプリケーションを実行するには、外部データベースを設定する必要があります。外部データベースはクラスター外部に存在し、プライマリノードのフェイルオーバープロセス中にデータを永続的に保ちます。以下のアプリケーションでは、サービスコンポーネントはプライマリノードのフェイルオーバープロセス中に自動的に復旧されますが、アクティブなジョブは失敗し、再試行が必要になる場合があります。

アプリケーション	プライマリノードのフェイルオーバー中の可用性	メモ
[Hive]	サービスコンポーネントのみに対する高可用性	Hive の外部メタストアが必要です。PostgreSQL はマルチマスタークラスターではサポートされていないため、これは MySQL の外部メタストアでなければなりません。詳細につ

アプリケーション	プライマリノードのフェイルオーバー中の可用性	メモ
		いては、「Hive の外部メタストアの設定」を参照してください。
Hue	サービスコンポーネントのみに対する高可用性	Hue の外部データベースが必要です。詳細については、「Amazon RDS でのリモートデータベースと Hue の使用」を参照してください。
Oozie	サービスコンポーネントのみに対する高可用性	Oozie の外部データベースが必要です。詳細については、「Amazon RDS でのリモートデータベースと Oozie の使用」を参照してください。 oozie-server と oozie-client は 3 つのプライマリノードすべてにインストールされます。oozie-clients は、デフォルトで正しい oozie-server に接続するように設定されています。

アプリケーション	プライマリノードのフェイルオーバー中の可用性	メモ
PrestoDB または PrestoSQL/Trino	サービスコンポーネントのみに対する高可用性	PrestoDB (Amazon EMR 6.1.0-6.3.0 では PrestoSQL、Amazon EMR 6.4.0 以降では Trino) 用の外部 Hive メタストアが必要です。Presto を AWS Glue データカタログで使用するか、Hive の外部 MySQL データベースを使用できます。 Presto CLI は 3 つのプライマリノードすべてにインストールされるため、これを使用して、任意のプライマリノードから Presto コーディネーターにアクセスできます。Presto コーディネーターは 1 つのプライマリノードにのみインストールされます。Presto コーディネーターがインストールされているプライマリノードの DNS 名は、Amazon EMR `describe-cluster` API を呼び出し、レスポンス内の `MasterPublicDnsName` フィールドの戻り値を読み取ることで見つけることができます。

> **Note**
>
> プライマリノードで障害が発生した場合、Java Database Connectivity (JDBC) または Open Database Connectivity (ODBC) はプライマリノードへの接続を終了します。Hive メタストアデーモンはすべてのプライマリノードで実行されるため、残りのいずれかのプライマリノードに接続して作業を続行できます。または、障害が発生したプライマリノードが置き換えられるのを待つことができます。

複数プライマリノードを持つクラスターでの Amazon EMR 機能の動作の仕組み

SSH を使用したプライマリノードへの接続

1 つのプライマリノードに接続するのと同じ方法で、SSH を使用して Amazon EMR クラスターで 3 つのプライマリノードのいずれかに接続できます。詳細については、「SSH を使用してプライマリノードに接続する」を参照してください。

プライマリノードで障害が発生した場合、そのプライマリノードへの SSH 接続は終了します。作業を続行するには、2 つのプライマリノードのうち、もう 1 つのノードに接続できます。あるいは、障害が発生したプライマリノードを Amazon EMR が置き換えた後で、新しいプライマリノードにアクセスできます。

> **Note**
>
> 代替プライマリノードのプライベート IP アドレスは、前のノードと同じままになります。代替プライマリノードのパブリック IP アドレスは変更される場合があります。新しい IP アドレスはコンソールで取得するか、 AWS CLI の `describe-cluster` コマンドを使用して取得できます。
>
> NameNode は、2 つのプライマリノードでのみ実行されます。ただし、`hdfs` CLI コマンドを実行し、ジョブを運用して 3 つすべてのプライマリノードで HDFS にアクセスできます。

複数のプライマリノードを持つ Amazon EMR クラスターでのステップの操作

1 つのプライマリノードを持つクラスターでステップを操作するのと同じ方法で、複数のプライマリノードを持つ Amazon EMR クラスターにステップを送信できます。詳細については、「クラスターへの作業の送信」を参照してください。

以下に、複数のプライマリノードを持つ Amazon EMR クラスターでステップを操作する際の考慮事項を示します。

- プライマリノードで障害が発生した場合、プライマリノードで実行中のステップは FAILED とマークされます。ローカルに書き込まれたデータはすべて失われます。ただし、FAILED ステータスがステップの実際の状態を反映しているとは限りません。
- プライマリノードで障害が発生したときに実行中のステップが YARN アプリケーションを開始した場合、プライマリノードの自動フェイルオーバーにより、ステップは続行して成功できます。

- ジョブの出力を参照して、ステップのステータスを確認することをお勧めします。例えば、MapReduce ジョブは _SUCCESS ファイルを使用して、ジョブが正常に完了したかどうかを判断します。
- TERMINATE_JOB_FLOW または TERMINATE_CLUSTER の代わりに、 ActionOnFailure パラメータを CONTINUE または CANCEL_AND_WAIT に設定することをお勧めします。

自動終了保護

Amazon EMR は、複数のプライマリノードを持つすべてのクラスターに対して終了保護を自動的に有効にし、クラスターの作成時に指定したステップ実行設定をオーバーライドします。クラスターの起動後に、終了保護を無効にできます。実行中のクラスターに対する終了保護の設定 を参照してください。複数のプライマリノードを持つクラスターをシャットダウンするには、まずクラスター属性を変更して、終了保護を無効にする必要があります。手順については、「複数のプライマリノードを持つ Amazon EMR クラスターの終了」を参照してください。

終了保護の詳細については、「終了保護の使用」を参照してください。

複数のプライマリノードを持つ Amazon EMR クラスターでサポートされていない機能

現在、次の Amazon EMR の機能は、複数のプライマリノードを持つ Amazon EMR クラスターで使用できません。

- EMR Notebooks
- 永続 Spark 履歴サーバーへのワンクリックアクセス
- 永続アプリケーションユーザーインターフェイス
- 永続アプリケーションユーザーインターフェイスへのワンクリックアクセスは、現在、複数のプライマリノードを持つ Amazon EMR クラスターや AWS Lake Formation と統合された Amazon EMR クラスターでは使用できません。

> **Note**
>
> クラスターで Kerberos 認証を使用するには、外部 KDC を設定する必要があります。Amazon EMR バージョン 5.27.0 以降では、複数のプライマリノードを持つ Amazon EMR クラスターに HDFS 透過的暗号化を設定することができます。詳細については、「Amazon EMR における HDFS での透過的暗号化」を参照してください。

複数のプライマリノードを持つ Amazon EMR クラスターの起動

このトピックでは、複数のプライマリノードを持つ Amazon EMR クラスターを起動するための設定の詳細と例を示します。

> **Note**
>
> Amazon EMR は、複数のプライマリノードを持つすべてのクラスターに対して自動終了保護を有効にし、クラスターの作成時に提供した自動終了設定をオーバーライドします。複数のプライマリノードを持つクラスターをシャットダウンするには、まずクラスター属性を変更して、終了保護を無効にする必要があります。手順については、「複数のプライマリノードを持つ Amazon EMR クラスターの終了」を参照してください。

前提条件

- パブリックおよびプライベート VPC サブネットの両方で、複数のプライマリノードを持つ Amazon EMR クラスターを起動できます。EC2-Classic はサポートされません。パブリックサブネットで複数のプライマリノードを持つ Amazon EMR クラスターを起動するには、コンソールで [IPv4 の自動割り当て] を選択するか、次のコマンドを実行して、このサブネットのインスタンスがパブリック IP アドレスを受け取ることができるようにする必要があります。*22XXXX01* をサブネット ID に置き換えます。

```
aws ec2 modify-subnet-attribute --subnet-id subnet-22XXXX01 --map-public-ip-on-launch
```

- 複数のプライマリノードを持つ Amazon EMR クラスターで Hive、Hue、または Oozie を実行するには、外部メタストアを作成する必要があります。詳細については、「Hive の外部メタストアの設定」、「Amazon RDS でのリモートデータベースと Hue の使用」、または「Apache Oozie」を参照してください。
- クラスターで Kerberos 認証を使用するには、外部 KDC を設定する必要があります。詳細については、「Amazon EMR 上の Kerberos の設定」を参照してください。

複数のプライマリノードを持つ Amazon EMR クラスターの起動

インスタンスグループまたはインスタンスフリートを使用する場合、複数のプライマリノードを持つクラスターを起動できます。複数のプライマリノードを持つインスタンスグループを使用するときは、プライマリノードインスタンスグループにインスタンスカウント値 3 を指定する必要があり

ます。複数のプライマリノードでインスタンスフリートを使用する場合、プライマリインスタンスフリートには 3 の TargetOnDemandCapacity、0 の TargetSpotCapacity を、そしてプライマリフリート用に設定する各インスタンスタイプには 1 の WeightedCapacity を指定する必要があります。

次の例は、インスタンスグループとインスタンスフリートの両方にデフォルト AMI またはカスタム AMI を使用してクラスターを起動する方法を示します。

> **Note**
>
> AWS CLIを使用して複数のプライマリノードを持つ Amazon EMR クラスターを起動するときに、サブネット ID を指定する必要があります。次の例の *22XXXX01* および *22XXXX02* は、お使いのサブネット ID と置き換えてください。

Default AMI, instance groups

Example 例 – デフォルト AMI を使用した、複数のプライマリノードを持つ Amazon EMR インスタンスグループクラスターの起動

```
aws emr create-cluster \
--name "ha-cluster" \
--release-label emr-6.15.0 \
--instance-groups InstanceGroupType=MASTER,InstanceCount=3,InstanceType=m5.xlarge
 InstanceGroupType=CORE,InstanceCount=4,InstanceType=m5.xlarge \
--ec2-attributes
 KeyName=ec2_key_pair_name,InstanceProfile=EMR_EC2_DefaultRole,SubnetId=subnet-22XXXX01
 \
--service-role EMR_DefaultRole \
--applications Name=Hadoop Name=Spark
```

Default AMI, instance fleets

Example 例 – デフォルト AMI を使用した、複数のプライマリノードを持つ Amazon EMR インスタンスフリートクラスターの起動

```
aws emr create-cluster \
--name "ha-cluster" \
--release-label emr-6.15.0 \
--instance-fleets '[
    {
        "InstanceFleetType": "MASTER",
```

```
        "TargetOnDemandCapacity": 3,
        "TargetSpotCapacity": 0,
        "LaunchSpecifications": {
            "OnDemandSpecification": {
                "AllocationStrategy": "lowest-price"
            }
        },
        "InstanceTypeConfigs": [
            {
                "WeightedCapacity": 1,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.xlarge"
            },
            {
                "WeightedCapacity": 1,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.2xlarge"
            },
            {
                "WeightedCapacity": 1,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.4xlarge"
            }
        ],
        "Name": "Master - 1"
    },
    {
        "InstanceFleetType": "CORE",
        "TargetOnDemandCapacity": 5,
        "TargetSpotCapacity": 0,
        "LaunchSpecifications": {
            "OnDemandSpecification": {
                "AllocationStrategy": "lowest-price"
            }
        },
        "InstanceTypeConfigs": [
            {
                "WeightedCapacity": 1,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.xlarge"
            },
            {
                "WeightedCapacity": 2,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
```

```
                "InstanceType": "m5.2xlarge"
            },
            {
                "WeightedCapacity": 4,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.4xlarge"
            }
        ],
        "Name": "Core - 2"
    }
]' \
 --ec2-attributes '{"InstanceProfile":"EMR_EC2_DefaultRole","SubnetIds":
["subnet-22XXXX01", "subnet-22XXXX02"]}' \
--service-role EMR_DefaultRole \
--applications Name=Hadoop Name=Spark
```

Custom AMI, instance groups

Example 例 – カスタム AMI を使用した、複数のプライマリノードを持つ Amazon EMR インスタンスグループクラスターの起動

```
aws emr create-cluster \
--name "custom-ami-ha-cluster" \
--release-label emr-6.15.0 \
--instance-groups InstanceGroupType=MASTER,InstanceCount=3,InstanceType=m5.xlarge
 InstanceGroupType=CORE,InstanceCount=4,InstanceType=m5.xlarge \
--ec2-attributes
 KeyName=ec2_key_pair_name,InstanceProfile=EMR_EC2_DefaultRole,SubnetId=subnet-22XXXX01
 \
--service-role EMR_DefaultRole \
--applications Name=Hadoop Name=Spark \
--custom-ami-id ami-MyAmiID
```

Custom AMI, instance fleets

Example 例 – カスタム AMI を使用した、複数のプライマリノードを持つ Amazon EMR インスタンスフリートクラスターの起動

```
aws emr create-cluster \
--name "ha-cluster" \
--release-label emr-6.15.0 \
--instance-fleets '[
    {
```

```
        "InstanceFleetType": "MASTER",
        "TargetOnDemandCapacity": 3,
        "TargetSpotCapacity": 0,
        "LaunchSpecifications": {
            "OnDemandSpecification": {
                "AllocationStrategy": "lowest-price"
            }
        },
        "InstanceTypeConfigs": [
            {
                "WeightedCapacity": 1,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.xlarge"
            },
            {
                "WeightedCapacity": 1,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.2xlarge"
            },
            {
                "WeightedCapacity": 1,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.4xlarge"
            }
        ],
        "Name": "Master - 1"
    },
    {
        "InstanceFleetType": "CORE",
        "TargetOnDemandCapacity": 5,
        "TargetSpotCapacity": 0,
        "LaunchSpecifications": {
            "OnDemandSpecification": {
                "AllocationStrategy": "lowest-price"
            }
        },
        "InstanceTypeConfigs": [
            {
                "WeightedCapacity": 1,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.xlarge"
            },
            {
                "WeightedCapacity": 2,
```

```
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.2xlarge"
            },
            {
                "WeightedCapacity": 4,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.4xlarge"
            }
        ],
        "Name": "Core - 2"
    }
]' \
--ec2-attributes '{"InstanceProfile":"EMR_EC2_DefaultRole","SubnetIds":
["subnet-22XXXX01", "subnet-22XXXX02"]}' \
--service-role EMR_DefaultRole \
--applications Name=Hadoop Name=Spark \
--custom-ami-id ami-MyAmiID
```

複数のプライマリノードを持つ Amazon EMR クラスターの終了

複数のプライマリノードを持つ Amazon EMR クラスターを終了するには、次の例に示すように、クラスターの終了前に終了保護を無効にする必要があります。*j-3KVTXXXXXX7UG* をクラスター ID に置き換えます。

```
aws emr modify-cluster-attributes --cluster-id j-3KVTXXXXXX7UG --no-termination-
protected
aws emr terminate-clusters --cluster-id j-3KVTXXXXXX7UG
```

Amazon EMR と EC2 プレイスメントグループの統合

Amazon EC2 で Amazon EMR の複数のプライマリノードクラスターを起動する場合、プレイスメントグループ戦略を使用して、ハードウェア障害から保護するためにプライマリノードインスタンスをデプロイする方法を指定することができます。

プレイスメントグループ機能は、複数のプライマリノードクラスターのオプションとして Amazon EMR バージョン 5.23.0 以降でサポートされています。現在、プレイスメントグループ機能ではプライマリノードタイプのみがサポートされており、SPREAD 戦略は、これらのプライマリノードに適用されます。SPREAD 戦略では、ハードウェア障害の発生時に複数のプライマリノードが失われるのを防ぐため、少数のインスタンスを別個の基盤となるハードウェア全体に配置します。リクエストを実行するための固有のハードウェアが不足している場合、インスタンスの起動リクエストが失敗する可

能性があることに注意してください。EC2 プレイスメント戦略および制限については、「Linux インスタンス用 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。

Amazon EC2 には、 AWS リージョンごとに起動できるプレイスメントグループ戦略対応クラスターが 500 個という初期制限があります。 AWS サポートに連絡して、許可されたプレイスメントグループの数の増加をリクエストしてください。Amazon EMR が Amazon EMR プレイスメントグループ戦略に関連付けているキーと値のペアを追跡することで、Amazon EMR が作成する EC2 プレイスメントグループを識別できます。EC2 クラスターのインスタンスタグの詳細については、「Amazon EC2 でクラスターインスタンスを表示する」を参照してください。

Amazon EMRrole へのプレイスメントグループの管理ポリシーのアタッチ

Amazon EMR が Amazon EC2 でプレイスメントグループを作成、削除、および記述できるようにするために、プレイスメントグループ戦略には `AmazonElasticMapReducePlacementGroupPolicy` という管理ポリシーが必要です。複数のプライマリノードを持つ Amazon EMR クラスターを起動する前に、`AmazonElasticMapReducePlacementGroupPolicy` を Amazon EMR のサービスロールにアタッチする必要があります。

プレイスメントグループ管理ポリシーの代わりに `AmazonEMRServicePolicy_v2` 管理ポリシーを Amazon EMR ロールにアタッチできます。`AmazonEMRServicePolicy_v2` は Amazon EC2 のプレイスメントグループに対して、`AmazonElasticMapReducePlacementGroupPolicy` と同じアクセスを許可します。詳細については、「Amazon EMR のサービスロール (EMR ロール)」を参照してください。

`AmazonElasticMapReducePlacementGroupPolicy` 管理ポリシーは、Amazon EMR が作成および管理する次の JSON テキストです。

> **Note**
>
> `AmazonElasticMapReducePlacementGroupPolicy` 管理ポリシーは自動的に更新されるため、ここに示すポリシーは である可能性があります out-of-date。 AWS マネジメントコンソールを使用して、現在のポリシーを表示します。

```
{
   "Version":"2012-10-17",
   "Statement":[
```

```
      {
         "Resource":"*",
         "Effect":"Allow",
         "Action":[
            "ec2:DeletePlacementGroup",
            "ec2:DescribePlacementGroups"
         ]
      },
      {
         "Resource":"arn:aws:ec2:*:*:placement-group/pg-*",
         "Effect":"Allow",
         "Action":[
            "ec2:CreatePlacementGroup"
         ]
      }
   ]
}
```

プレイスメントグループ戦略を使用して、複数のプライマリノードを持つ Amazon EMR クラスターを起動する

プレイスメントグループ戦略を使用して複数のプライマリノードを持つ Amazon EMR クラスターを起動するには、プレイスメントグループマネージドポリシー `AmazonElasticMapReducePlacementGroupPolicy` を Amazon EMR ロールにアタッチします。詳細については、「Amazon EMRrole へのプレイスメントグループの管理ポリシーのアタッチ」を参照してください。

このロールを使用して複数のプライマリノードを持つ Amazon EMR クラスターを起動するたびに、Amazon EMR は、それらのプライマリノードに適用される `SPREAD` 戦略を使用してクラスターを起動しようとします。プレイスメントグループマネージドポリシー `AmazonElasticMapReducePlacementGroupPolicy` がアタッチされていないロールを使用する場合、Amazon EMR は、プレイスメントグループ戦略なしで複数のプライマリノードを持つ Amazon EMR クラスターを起動しようとします。

Amazon EMR EMRAPI または CLI を使用して `placement-group-configs` パラメータを指定して複数のプライマリノードを持つ Amazon EMR クラスターを起動する場合、Amazon EMR は Amazon EMR ロールにプレイスメントグループマネージドポリシー `AmazonElasticMapReducePlacementGroupPolicy` がアタッチされている場合にのみクラスターを起動します。Amazon EMR ロールにポリシーがアタッチされていない場合、複数のプライマリノードを持つ Amazon EMR クラスターの起動は失敗します。

Amazon EMR API

Example 例 - プレイスメントグループ戦略を使用して、Amazon EMR API から複数のプライマリノードを持つインスタンスグループクラスターを起動する

RunJobFlow アクションを使用して複数のプライマリノードを持つ Amazon EMR クラスターを作成する場合は、 PlacementGroupConfigsプロパティを次のように設定します。現在、MASTER インスタンスロールは、自動的に SPREAD をプレイスメントグループ戦略として使用します。

```
{
   "Name":"ha-cluster",
   "PlacementGroupConfigs":[
      {
         "InstanceRole":"MASTER"
      }
   ],
   "ReleaseLabel": emr-6.15.0,
   "Instances":{
      "ec2SubnetId":"subnet-22XXXX01",
      "ec2KeyName":"ec2_key_pair_name",
      "InstanceGroups":[
         {
            "InstanceCount":3,
            "InstanceRole":"MASTER",
            "InstanceType":"m5.xlarge"
         },
         {
            "InstanceCount":4,
            "InstanceRole":"CORE",
            "InstanceType":"m5.xlarge"
         }
      ]
   },
   "JobFlowRole":"EMR_EC2_DefaultRole",
   "ServiceRole":"EMR_DefaultRole"
}
```

- *ha-cluster* を自分の高可用性クラスター名に置き換えます。
- *subnet-22XXXX01* をサブネット ID に置き換えます。

- *ec2_key_pair_name* をこのクラスターの EC2 キーペアの名前に置き換えます。EC2 キーペアはオプションであり、SSH を使用してクラスターにアクセスする場合にのみ必須です。

AWS CLI

Example 例 - プレイスメントグループ戦略を使用して、 AWS Command Line Interfaceから複数のプライマリノードを持つインスタンスフリートクラスターを起動する

RunJobFlow アクションを使用して複数のプライマリノードを持つ Amazon EMR クラスターを作成する場合は、 PlacementGroupConfigsプロパティを次のように設定します。現在、MASTER インスタンスロールは、自動的に SPREAD をプレイスメントグループ戦略として使用します。

```
aws emr create-cluster \
--name "ha-cluster" \
--placement-group-configs InstanceRole=MASTER \
--release-label emr-6.15.0 \
--instance-fleets '[
    {
        "InstanceFleetType": "MASTER",
        "TargetOnDemandCapacity": 3,
        "TargetSpotCapacity": 0,
        "LaunchSpecifications": {
            "OnDemandSpecification": {
                "AllocationStrategy": "lowest-price"
            }
        },
        "InstanceTypeConfigs": [
            {
                "WeightedCapacity": 1,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.xlarge"
            },
            {
                "WeightedCapacity": 1,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.2xlarge"
            },
            {
                "WeightedCapacity": 1,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.4xlarge"
```

```
            }
        ],
        "Name": "Master - 1"
    },
    {
        "InstanceFleetType": "CORE",
        "TargetOnDemandCapacity": 5,
        "TargetSpotCapacity": 0,
        "LaunchSpecifications": {
            "OnDemandSpecification": {
                "AllocationStrategy": "lowest-price"
            }
        },
        "InstanceTypeConfigs": [
            {
                "WeightedCapacity": 1,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.xlarge"
            },
            {
                "WeightedCapacity": 2,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.2xlarge"
            },
            {
                "WeightedCapacity": 4,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.4xlarge"
            }
        ],
        "Name": "Core - 2"
    }
]' \
--ec2-attributes '{
    "KeyName": "ec2_key_pair_name",
    "InstanceProfile": "EMR_EC2_DefaultRole",
    "SubnetIds": [
        "subnet-22XXXX01",
        "subnet-22XXXX02"
    ]
}' \
--service-role EMR_DefaultRole \
--applications Name=Hadoop Name=Spark
```

- *ha-cluster* を自分の高可用性クラスター名に置き換えます。
- *ec2_key_pair_name* をこのクラスターの EC2 キーペアの名前に置き換えます。EC2 キーペアはオプションであり、SSH を使用してクラスターにアクセスする場合にのみ必須です。
- *subnet-22XXXX01* および *subnet-22XXXX02* は、お使いのサブネット ID に置き換えてください。

プレイスメントグループ戦略を使用せずに複数プライマリノードを持つクラスターを起動する

プレイスメントグループ戦略を使用しないで複数プライマリノードを持つクラスターでプライマリノードを起動する場合、次のいずれかを実行する必要があります。

- プレイスメントグループ管理ポリシー AmazonElasticMapReducePlacementGroupPolicy を Amazon EMRrole から削除する、または
- Amazon EMRAPI または CLI を使用して placement-group-configs パラメータを指定し、NONE をプレイスメントグループ戦略として選択して、複数のプライマリノードを持つクラスターを起動します。

Amazon EMR API

Example – Amazon EMRAPI を使用するプレイスメントグループ戦略を使用しないで、複数のプライマリノードを持つクラスターを起動します。

RunJobFlow アクションを使用して複数のプライマリノードを持つクラスターを作成する場合は、 PlacementGroupConfigsプロパティを次のように設定します。

```
{
   "Name":"ha-cluster",
   "PlacementGroupConfigs":[
      {
         "InstanceRole":"MASTER",
         "PlacementStrategy":"NONE"
      }
   ],
   "ReleaseLabel":"emr-5.30.1",
   "Instances":{
      "ec2SubnetId":"subnet-22XXXX01",
      "ec2KeyName":"ec2_key_pair_name",
```

```
      "InstanceGroups":[
         {
            "InstanceCount":3,
            "InstanceRole":"MASTER",
            "InstanceType":"m5.xlarge"
         },
         {
            "InstanceCount":4,
            "InstanceRole":"CORE",
            "InstanceType":"m5.xlarge"
         }
      ]
   },
   "JobFlowRole":"EMR_EC2_DefaultRole",
   "ServiceRole":"EMR_DefaultRole"
}
```

- *ha-cluster* を自分の高可用性クラスター名に置き換えます。
- *subnet-22XXXX01* をサブネット ID に置き換えます。
- *ec2_key_pair_name* をこのクラスターの EC2 キーペアの名前に置き換えます。EC2 キーペアはオプションであり、SSH を使用してクラスターにアクセスする場合にのみ必須です。

Amazon EMR CLI

Example — Amazon EMRCLI を使用するプレイスメントグループ戦略を使用せずに、複数のプライマリノードを持つクラスターを起動します。

RunJobFlow アクションを使用して複数のプライマリノードを持つクラスターを作成する場合は、 `PlacementGroupConfigs`プロパティを次のように設定します。

```
aws emr create-cluster \
--name "ha-cluster" \
--placement-group-configs InstanceRole=MASTER,PlacementStrategy=NONE \
--release-label emr-5.30.1 \
--instance-groups InstanceGroupType=MASTER,InstanceCount=3,InstanceType=m5.xlarge
 InstanceGroupType=CORE,InstanceCount=4,InstanceType=m5.xlarge \
--ec2-attributes
 KeyName=ec2_key_pair_name,InstanceProfile=EMR_EC2_DefaultRole,SubnetId=subnet-22XXXX01
 \
--service-role EMR_DefaultRole \
--applications Name=Hadoop Name=Spark
```

- *ha-cluster* を自分の高可用性クラスター名に置き換えます。
- *subnet-22XXXX01* をサブネット ID に置き換えます。
- *ec2_key_pair_name* をこのクラスターの EC2 キーペアの名前に置き換えます。EC2 キーペアはオプションであり、SSH を使用してクラスターにアクセスする場合にのみ必須です。

複数のプライマリノードを持つクラスターにアタッチされたプレイスメントグループ戦略設定の確認

Amazon EMR クラスター記述 API を使用して、複数のプライマリノードを持つクラスターにアタッチされたプレイスメントグループ戦略の設定を確認できます。

Example

```
aws emr describe-cluster --cluster-id "j-xxxxx"
{
   "Cluster":{
      "Id":"j-xxxxx",
      ...
      ...
      "PlacementGroups":[
         {
            "InstanceRole":"MASTER",
            "PlacementStrategy":"SPREAD"
         }
      ]
   }
}
```

考慮事項とベストプラクティス

複数のプライマリノードを持つ Amazon EMR クラスターを作成する場合は、以下の事項を考慮します。

> **Important**
>
> 複数のプライマリノードを持つ高可用性 EMR クラスターを起動するには、最新の Amazon EMR リリースを使用することを強くお勧めします。これにより、高可用性クラスターで最高レベルの耐障害性と安定性を得ることができます。

- インスタンスフリートの高可用性は、Amazon EMR リリース 5.36.1、5.36.2、6.8.1、6.9.1、6.10.1、6.11.1、6.12.0 以降でサポートされています。インスタンスグループの場合、Amazon EMR リリース 5.23.0 以降で高可用性がサポートされています。詳細については、「Amazon EMR リリースについて」を参照してください。
- 高可用性クラスターでは、Amazon EMR はオンデマンドインスタンスを備えたプライマリノードの起動のみをサポートします。これにより、クラスターの可用性を最大限に高めることができます。
- プライマリフリートには引き続き複数のインスタンスタイプを指定できますが、高可用性クラスターのすべてのプライマリノードは、異常のあるプライマリノードの代替を含め、同じインスタンスタイプで起動されます。
- 運用を継続するには、複数のプライマリノードがある高可用性クラスターの 3 つのプライマリノードのうちの 2 つが正常である必要があります。そのため、2 つのプライマリノードで同時に障害が発生した場合、EMR クラスターは機能しません。
- 高可用性クラスターを含むすべての EMR クラスターは、単一の可用性ゾーンで起動します。そのため、可用性ゾーンの障害に対する耐性がありません。可用性ゾーンが停止した場合、クラスターへのアクセスは失われます。
- Amazon EMR では、複数のプライマリノードを持つ Amazon EMR クラスターでサポートされているアプリケーション で指定されているものを除き、オープンソースアプリケーションの高可用性は保証していません。
- Amazon EMR リリース 5.23.0 から 5.30.1 では、1 つのインスタンスグループクラスターの 3 つのプライマリノードのうち 2 つだけが HDFS NameNode を実行します。

サブネット設定の考慮事項:

- 複数のプライマリノードを持つ Amazon EMR クラスターは、1 つのアベイラビリティーゾーンまたはサブネットにのみ存在できます。フェイルオーバーの際にサブネットが完全に利用されている、またはオーバーサブスクライブされている場合、Amazon EMR は障害が発生したプライマリノードを置き換えることができません。このシナリオを回避するには、サブネット全体を Amazon EMR クラスター専用にすることをお勧めします。さらに、サブネットで利用できる十分なプライベート IP アドレスがあることを確認します。

コアノード設定の考慮事項:

- コアノードの高可用性も維持するには、少なくとも 4 つのコアノードを起動することをお勧めします。3 つ以下のコアノードを持つ、より小さいクラスターの起動を決定した場合は、HDFS の

dfs.replication parameter を少なくとも 2 に設定して、十分な DFS レプリケーションになるようにします。詳細については、「HDFS 構成」を参照してください。

> **Warning**
>
> 1. ノードが 4 つ未満のクラスターで dfs.replication を 1 に設定すると、単一ノードがダウンした場合に HDFS データが失われる可能性があります。本番環境のワークロードには、少なくとも 4 つのコアノードを持つクラスターを使用することをお勧めします。
> 2. Amazon EMR では、クラスターはコアノードを dfs.replication 未満にスケールすることはできません。例えば、dfs.replication = 2 の場合、コアノードの最小数は 2 です。
> 3. マネージドスケーリングや自動スケーリングを使用する場合や、クラスターのサイズを手動で変更する場合は、dfs.replication を 2 以上に設定することをお勧めします。

メトリクスでのアラーム設定の考慮事項:

- Amazon EMR は HDFS または YARN に関するアプリケーション固有のメトリクスを提供していません。アラームを設定して、プライマリノードのインスタンス数をモニタリングすることをお勧めします。、MultiMasterInstanceGroupNodesRunning、MultiMasterInstanceGroupNodesRunningたは の Amazon CloudWatch メトリクスを使用してアラームを設定しますMultiMasterInstanceGroupNodesRequested。プライマリノードに障害が発生し、置き換えられた場合は、 から通知 CloudWatch されます。
  - MultiMasterInstanceGroupNodesRunningPercentage が 0.5 より大きく 1.0 より小さい場合、クラスターでプライマリノードが失われた可能性があります。このような状況では、Amazon EMR はプライマリノードの置換を試みます。
  - MultiMasterInstanceGroupNodesRunningPercentage が 0.5 を下回った場合、2 つのプライマリノードで障害が発生した可能性があります。このような状況では、クォーラムは失われており、クラスターを回復することはできません。このクラスターからデータを手動で移行する必要があります。

  詳細については、「メトリクスでアラームを設定する」を参照してください。

の EMR クラスター AWS Outposts

Amazon EMR 5.28.0 以降では、. AWS Outposts AWS Outposts enables ネイティブ AWS サービス、インフラストラクチャ、オンプレミス施設での運用モデルで EMR クラスターを作成して実行できます。 AWS Outposts 環境では、 AWS クラウドで使用するのと同じ AWS APIsツール、インフラストラクチャを使用できます。Amazon EMR on AWS Outposts は、オンプレミスのデータやアプリケーションの近くで実行する必要がある低レイテンシーのワークロードに最適です。の詳細については AWS Outposts、「 AWS Outposts ユーザーガイド」を参照してください。

前提条件

Amazon EMR を AWS Outpostsで使用するための前提条件を次に示します。

- オンプレミスデータセンター AWS Outposts に をインストールして設定しておく必要があります。
- Outpost 環境と AWS リージョンの間に信頼性の高いネットワーク接続が必要です。
- Amazon EMR がサポートするインスタンスタイプが Outpost で使用できる十分な容量が必要です。

制限事項

AWS Outpostsで Amazon EMR を使用する際の制限事項を次に示します。

- オンデマンドインスタンスは、Amazon EC2 インスタンスでサポートされる唯一のオプションです。スポットインスタンスは、 AWS Outpostsの Amazon EMR では使用できません。
- 追加の Amazon EBS ストレージボリュームが必要な場合は、汎用 SSD (GP2) のみがサポートされます。
- Amazon EMR リリース 5.28 から 6.x AWS Outposts で を使用する場合、 AWS リージョン 指定した にオブジェクトを保存する S3 バケットのみを使用できます。Amazon EMR 7.0.0 以降では、の Amazon EMR AWS Outposts はS3Aファイルシステムクライアントのプレフィックス でもサポートされていますs3a://。
- AWS Outpostsの Amazon EMR では、次のインスタンスタイプのみがサポートされています。

インスタンスクラス	インスタンスのタイプ
汎用	m5.xlarge |m5.2xlarge | m5.4xlarge |m5.12xlarge | m5.24xlarge |m5d.xlarge | m5d.2xlarge | m5d.4xlarge | m5d.12xlarge | m5d.24xlarge
コンピューティング最適化	c5.xlarge |c5.2xlarge | c5.4xlarge |c5.18xlarge | c5d.xlarge |c5d.2xlarge | c5d.4xlarge | c5d.18xlarge
メモリ最適化	r5.xlarge |r5.2xlarge | r5.4xlarge |r5.12xlarge | r5d.xlarge |r5d.2xlarge | r5d.4xlarge | r5d.12xlarge | r5d.24xlarge
ストレージの最適化	i3en.xlarge |i3en.2xlarge | i3en.3xlarge | i3en.6xlarge | i3en.12xlarge |i3en.24xlarge

ネットワーク接続に関する考慮事項

- Outpost とその AWS リージョン間のネットワーク接続が失われた場合、クラスターは引き続き実行されます。ただし、接続が復元されるまで、新しいクラスターを作成したり、既存のクラスターで新しいアクションを実行したりすることはできません。インスタンスに障害が発生した場合、インスタンスは自動的に置き換えられません。さらに、実行中のクラスターへのステップの追加、ステップの実行ステータスの確認、 CloudWatch メトリクスとイベントの送信などのアクションは遅延します。
- Outpost と AWS リージョン間で、信頼性が高く可用性の高いネットワーク接続を提供することをお勧めします。Outpost とその AWS リージョン間のネットワーク接続が数時間以上失われた場合、終了保護を有効にしたクラスターは引き続き実行され、終了保護を無効にしたクラスターは終了する可能性があります。
- 定期的なメンテナンスによってネットワーク接続に影響が生じる場合は、事前に終了保護を有効にすることをお勧めします。より一般的には、接続の中断とは、Outpost またはカスタマーネットワークに対してローカルでない外部依存関係が利用できなくなることを意味します。これには、Simple Storage Service (Amazon S3)、EMRFS 整合性ビューで使用される DynamoDB、およ

び複数のプライマリノードを持つ Amazon EMR クラスターでリージョン内インスタンスが使用される場合の Amazon RDS が含まれます。

での Amazon EMR クラスターの作成 AWS Outposts

での Amazon EMR クラスターの作成 AWS Outposts は、 AWS クラウドでの Amazon EMR クラスターの作成と似ています。で Amazon EMR クラスターを作成するときは AWS Outposts、Outpost に関連付けられた Amazon EC2 サブネットを指定する必要があります。

Amazon VPC は、 AWS リージョン内のすべてのアベイラビリティーゾーンにまたがることができます。 AWS Outposts はアベイラビリティーゾーンの拡張であり、アカウント内の Amazon VPC を拡張して、複数のアベイラビリティーゾーンおよび関連する Outpost ロケーションにまたがることができます。Outpost を設定するとき、サブネットをそれに関連付けて、リージョン VPC 環境をオンプレミス施設に拡張します。Outpost インスタンスおよび関連サービスは、サブネットが関連付けられたアベイラビリティーゾーンと同様に、リージョンの VPC の一部として表示されます。詳細については、AWS Outposts ユーザーガイドを参照してください。

コンソール

AWS Outposts を使用して で新しい Amazon EMR クラスターを作成するには AWS Management Console、Outpost に関連付けられている Amazon EC2 サブネットを指定します。

> **Note**
>
> Amazon EMR コンソールは、再設計され、使いやすくなりました。古いコンソールと新しいコンソールエクスペリエンスの違いについては、「Amazon EMR コンソール」を参照してください。

New console

新しいコンソール AWS Outposts を使用して でクラスターを作成するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で、[クラスター] を選択し、[クラスターの作成] を選択します
3. [クラスターの設定] で、[インスタンスグループ] または [インスタンスフリート] を選択します。次に、[EC2 インスタンスタイプの選択] ドロップダウンメニューからインスタンス

タイプを選択するか、[アクション] を選択して [EBS ボリュームの追加] を選択します。の Amazon EMR AWS Outposts では、Amazon EBS ボリュームとインスタンスタイプが制限されています。

4. [ネットワーク] で、op-123456789 という形式の Outpost ID を持つ EC2 サブネットを選択します。
5. クラスターに適用するその他のオプションを選択します。
6. クラスターを起動するには、[クラスターの作成] を選択します。

Old console

古いコンソール AWS Outposts を使用して でクラスターを作成するには

1. 新しい Amazon EMR コンソールに移動し、サイドナビゲーションから [古いコンソールに切り替え] を選択します。古いコンソールに切り替えたときの動作の詳細については、「Using the old console」を参照してください。
2. [クラスターを作成] を選択します。
3. [Go to advanced options] を選択します。
4. [ソフトウェア設定] の [リリース] で、5.28.0 以降を選択します。
5. ハードウェア設定 の EC2 サブネット で、Outpost ID が op-123456789 の Amazon EC2 サブネットを選択します。
6. ユニフォームインスタンスグループまたはインスタンスフリート用のインスタンスタイプを選択するか、Amazon EBS ストレージボリュームを追加します。 AWS Outpostsの Amazon EMR では、限られた Amazon EBS ボリュームタイプとインスタンスタイプがサポートされています。

CLI

AWS Outposts を使用して でクラスターを作成するには AWS CLI

- AWS Outposts を使用して で新しい Amazon EMR クラスターを作成するには AWS CLI、次の例のように、Outpost に関連付けられている EC2 サブネットを指定します。*subnet-22XXXX01* を独自の Amazon EC2 サブネット ID に置き換えます。

```
aws emr create-cluster \
--name "Outpost cluster" \
--release-label emr-7.1.0 \
```

```
--applications Name=Spark \
--ec2-attributes KeyName=myKey SubnetId=subnet-22XXXX01 \
--instance-type m5.xlarge --instance-count 3 --use-default-roles
```

AWS ローカルゾーンの EMR クラスター

Amazon EMR バージョン 5.28.0 以降では、 AWS Local Zones をサポートする AWS リージョンの論理拡張として、Local Zones サブネットで Amazon EMR クラスターを作成して実行できます。Local Zone を使用すると、Amazon EMR の機能や、コンピューティングやストレージ AWS サービスなどのサービスのサブセットをユーザーの近くに配置して、ローカルで実行されているアプリケーションに非常に低レイテンシーでアクセスできるようになります。使用可能な Local Zones のリストについては、「AWS Local Zones」を参照してください。利用可能な AWS ローカルゾーンへのアクセスについては、「 リージョン、アベイラビリティーゾーン、およびローカルゾーン」を参照してください。

サポートされるインスタンスタイプ

Local Zones の Amazon EMR クラスターでは、次のインスタンスタイプを使用できます。使用可能なインスタンスタイプは、リージョンごとに異なる場合があります。

インスタンスクラス	インスタンスのタイプ
汎用	m5.xlarge | m5.2xlarge | m5.4xlarge | m5.12xlarge | m5.24xlarge | m5d.xlarge | m5d.2xlarge | m5d.4xlarge | m5d.12xlarge | m5d.24xlarge
コンピューティング最適化	c5.xlarge | c5.2xlarge | c5.4xlarge | c5.9xlarge | c5.18xlarge | c5d.xlarge | c5d.2xlarge | c5d.4xlarge| c5d.9xlarge | c5d.18xlarge
メモリ最適化	r5.xlarge | r5.2xlarge | r5.4xlarge | r5.12xlarge | r5d.xlarge | r5d.2xlarge | r5d.4xlarge | r5d.12xlarge | r5d.24xlarge
ストレージの最適化	i3en.xlarge | i3en.2xlarge | i3en.3xlarge | i3en.6xlarge | i3en.12xlarge | i3en.24xlarge

Local Zones での Amazon EMR クラスターの作成

AWS ローカルゾーンに関連付けられた Amazon VPC サブネットで Amazon EMR クラスターを起動して、ローカルゾーンに Amazon EMR クラスターを作成します。クラスターにアクセスするには、米国西部 (オレゴン) コンソールの us-west-2-lax-1a など、Local Zone 名を使用します。

Local Zones は現在、Amazon EMR Notebooks またはインターフェイス VPC エンドポイント () を使用した Amazon EMR への直接接続をサポートしていませんAWS PrivateLink。

> **Note**
>
> Amazon EMR コンソールは、再設計され、使いやすくなりました。古いコンソールと新しいコンソールエクスペリエンスの違いについては、「Amazon EMR コンソール」を参照してください。

New console

新しいコンソールを使用してローカルゾーンでクラスターを作成するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で、[クラスター] を選択し、[クラスターの作成] を選択します
3. [ネットワーク] で、Local Zone ID の形式を「subnet 123abc | us-west-2-lax-1a」とする EC2 サブネットを選択します。
4. ユニフォームインスタンスグループまたはインスタンスフリート用のインスタンスタイプを選択するか、Amazon EBS ストレージボリュームを追加します。
5. クラスターに適用するその他のオプションを選択します。
6. クラスターを起動するには、[クラスターの作成] を選択します。

Old console

古いコンソールを使用して Local Zone でクラスターを作成するには

1. 新しい Amazon EMR コンソールに移動し、サイドナビゲーションから [古いコンソールに切り替え] を選択します。古いコンソールに切り替えたときの動作の詳細については、「Using the old console」を参照してください。

2. [クラスターを作成] を選択します。
3. [Go to advanced options] を選択します。
4. [ソフトウェア設定] の [リリース] で、5.28.0 以降を選択します。
5. [ハードウェア構成] の [EC2 サブネット] で、Local Zone ID の形式を「subnet 123abc | us-west-2-lax-1a」とする EC2 サブネットを選択します。
6. ユニフォームインスタンスグループまたはインスタンスフリート用の Amazon EBS ストレージボリュームを追加し、インスタンスタイプを選択します。

CLI

を使用してローカルゾーンにクラスターを作成するには AWS CLI

- 次の例に示すように、create-cluster コマンドとローカルゾーン SubnetId の を使用します。subnet-22XXXX1234567 を Local Zone に置き換え SubnetId 、必要に応じて他のオプションを置き換えます。詳細については、「https://docs.aws.amazon.com/cli/latest/reference/emr/create-cluster.html」を参照してください。

```
aws emr create-cluster \
--name "Local Zones cluster" \
--release-label emr-5.29.0 \
--applications Name=Spark \
--ec2-attributes KeyName=myKey,SubnetId=subnet-22XXXX1234567 \
--instance-type m5.xlarge --instance-count 3 --use-default-roles
```

Docker の設定

Amazon EMR 6.x は Hadoop 3 をサポートしています。これにより、YARN は Amazon EMR クラスターまたは Docker コンテナ内でコンテナを直接起動 NodeManager できます。Docker コンテナは、アプリケーションコードを実行するカスタム実行環境を提供します。カスタム実行環境は、YARN NodeManager やその他のアプリケーションの実行環境から分離されます。

Docker コンテナには、アプリケーションで使用する特別なライブラリを含めることができます。また、Docker コンテナは、R や Python など、ネイティブツールやライブラリの異なるバージョンを提供できます。使い慣れた Docker ツールを使用して、アプリケーションのライブラリとランタイム依存関係を定義できます。

Amazon EMR 6.x クラスターは、デフォルトの設定では、Docker コンテナを使用して YARN アプリケーション (Spark など) を実行できます。コンテナ設定をカスタマイズするには、`/etc/hadoop/conf` ディレクトリの `yarn-site.xml` ファイルと `container-executor.cfg` ファイルに定義されている Docker サポートオプションを編集します。各設定オプションとその使用方法の詳細については、「Launching Applications Using Docker Containers」を参照してください。

ジョブを送信するときに Docker を使用することを選択できます。Docker ランタイムと Docker イメージを指定するには、次の変数を使用します。

- `YARN_CONTAINER_RUNTIME_TYPE=docker`
- `YARN_CONTAINER_RUNTIME_DOCKER_IMAGE={DOCKER_IMAGE_NAME}`

Docker コンテナを使用して YARN アプリケーションを実行すると、YARN はジョブの送信時に指定した Docker イメージをダウンロードします。この Docker イメージを YARN で解決するには、Dockerレジストリを設定する必要があります。Docker レジストリの設定オプションは、クラスターのデプロイにパブリックサブネットとプライベートサブネットのどちらを使用するかによって異なります。

Docker レジストリ

Docker レジストリは、Docker イメージのストレージおよびディストリビューションシステムです。Amazon EMR の場合、フルマネージド型の Docker コンテナレジストリである Amazon ECR を使用することをお勧めします。これにより、独自のカスタムイメージを作成し、これらのイメージを可用性とスケーラビリティに優れたアーキテクチャでホストできます。

デプロイに関する考慮事項

Docker レジストリは、クラスター内の各ホストからのネットワークアクセスを必要とします。これは、YARN アプリケーションがクラスターで実行されているときに、各ホストが Docker レジストリからイメージをダウンロードするためです。これらのネットワーク接続要件により、Amazon EMR クラスターのデプロイ先をパブリックサブネットとプライベートサブネットのどちらにするかで、Docker レジストリの選択が制限される場合があります。

パブリックサブネット

EMR クラスターがパブリックサブネットにデプロイされると、YARN を実行しているノード NodeManager はインターネット経由で利用可能な任意のレジストリに直接アクセスできます。

プライベートサブネット

EMR クラスターがプライベートサブネットにデプロイされている場合、YARN を実行しているノードはインターネットに直接アクセス NodeManager できません。Docker イメージは Amazon ECR でホストし、 経由でアクセスできます AWS PrivateLink。

AWS PrivateLink を使用してプライベートサブネットシナリオで Amazon ECR へのアクセスを許可する方法の詳細については、「Amazon ECS のセットアップ」および「Amazon ECR AWS PrivateLink」を参照してください。

Docker レジストリの設定

Amazon EMR で Docker レジストリを使用するには、Docker イメージの解決に使用する特定のレジストリを信頼するように Docker を設定する必要があります。デフォルトの信頼レジストリは、ローカル (プライベート) と centos です。他のパブリックリポジトリや Amazon ECR を使用する場合は、EMR 分類 API を `container-executor` 分類キーで使用して、`/etc/hadoop/conf/container-executor.cfg` の `docker.trusted.registries` 設定をオーバーライドできます。

次の例は、`your-public-repo` という名前のパブリックリポジトリと、ECR レジストリエンドポイント `123456789123.dkr.ecr.us-east-1.amazonaws.com` の両方を信頼するようにクラスターを設定する方法を示しています。ECR を使用する場合は、このエンドポイントを特定の ECR エンドポイントに置き換えます。

```
[
  {
    "Classification": "container-executor",
    "Configurations": [
        {
            "Classification": "docker",
            "Properties": {
                "docker.trusted.registries": "local,centos,your-public-
repo,123456789123.dkr.ecr.us-east-1.amazonaws.com",
                "docker.privileged-containers.registries": "local,centos,your-public-
repo,123456789123.dkr.ecr.us-east-1.amazonaws.com"
            }
        }
    ]
  }
]
```

AWS Command Line Interface（ AWS CLI) を使用してこの設定で Amazon EMR 6.0.0 クラスターを起動するには、前述の ontainer-executor JSON 設定の内容container-executor.jsonを含む という名前のファイルを作成します。次に、次のコマンドを使用してクラスターを起動します。

```
export KEYPAIR=<Name of your Amazon EC2 key-pair>
export SUBNET_ID=<ID of the subnet to which to deploy the cluster>
export INSTANCE_TYPE=<Name of the instance type to use>
export REGION=<Region to which to deploy the cluster>

aws emr create-cluster \
    --name "EMR-6.0.0" \
    --region $REGION \
    --release-label emr-6.0.0 \
    --applications Name=Hadoop Name=Spark \
    --service-role EMR_DefaultRole \
    --ec2-attributes KeyName=$KEYPAIR,InstanceProfile=EMR_EC2_DefaultRole,SubnetId=
$SUBNET_ID \
    --instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=
$INSTANCE_TYPE InstanceGroupType=CORE,InstanceCount=2,InstanceType=$INSTANCE_TYPE \
    --configuration file://container-executor.json
```

EMR 6.0.0 以前で Amazon ECR にアクセスするための YARN の設定

Amazon ECR を初めて使用する場合は、「Amazon ECR の開始方法」の手順に従い、Amazon EMR クラスター内の各インスタンスから Amazon ECR にアクセスできることを確認します。

EMR 6.0.0 以前では、Docker コマンドを使用して Amazon ECR にアクセスするには、まず認証情報を生成する必要があります。YARN が Amazon ECR からイメージにアクセスできることを確認するには、コンテナ環境変数 YARN_CONTAINER_RUNTIME_DOCKER_CLIENT_CONFIG を使用して、生成した認証情報への参照を渡します。

コアノードの 1 つで次のコマンドを実行して、ECR アカウントのログイン行を取得します。

```
aws ecr get-login --region us-east-1 --no-include-email
```

get-login コマンドは、認証情報を作成するために実行する必要がある正しい Docker CLI コマンドを生成します。get-login から出力をコピーして実行します。

```
sudo docker login -u AWS -p <password> https://<account-id>.dkr.ecr.us-
east-1.amazonaws.com
```

このコマンドは、/root/.docker フォルダ内に config.json ファイルを生成します。このファイルを HDFS にコピーします。クラスターに送信されたジョブは、このファイルを使用して Amazon ECR に対して認証することができます。

次のコマンドを実行して、config.json ファイルをホームディレクトリにコピーします。

```
mkdir -p ~/.docker
sudo cp /root/.docker/config.json ~/.docker/config.json
sudo chmod 644 ~/.docker/config.json
```

次のコマンドを実行して、config.json を HDFS に配置し、クラスターで実行されるジョブで使用できるようにします。

```
hadoop fs -put ~/.docker/config.json /user/hadoop/
```

YARNは、Docker イメージレジストリとしての ECR にアクセスし、ジョブ実行中にコンテナをプルできます。

Docker レジストリと YARN を設定したら、Docker コンテナを使用して YARN アプリケーションを実行できます。詳細については、「Amazon EMR 6.0.0 を使用して Docker で Spark アプリケーションを実行する」を参照してください。

EMR 6.1.0 以降では、Amazon ECR への認証を手動でセットアップする必要はありません。container-executor 分類キーで Amazon ECR レジストリが検出された場合、Amazon ECR 自動認証機能がアクティブになり、ECR イメージで Spark ジョブを送信する際に YARN が認証プロセスを処理します。yarn-site で yarn.nodemanager.runtime.linux.docker.ecr-auto-authentication.enabled を調べると、自動認証が有効になっているかどうかを確認することができます。docker.trusted.registries に ECR レジストリ URL が含まれている場合、自動認証が有効であり、YARN 認証設定が true に設定されています。

Amazon ECR への自動認証を使用するための前提条件

- EMR バージョン 6.1.0 以降
- 設定に含まれる ECR レジストリは、クラスターと同じリージョンにあります
- 認証トークンを取得し、任意のイメージをプルする許可を持つ IAM ロール

詳細については、「Amazon ECR を使用してセットアップする」を参照してください。

自動認証を有効にする方法

「Docker レジストリの設定」に従って Amazon ECR レジストリを信頼できるレジストリとして設定し、Amazon ECR リポジトリとクラスターが同じリージョンにあることを確認します。

信頼できるレジストリに ECR レジストリが設定されていない場合でもこの機能を有効にするには、設定分類を使用して `yarn.nodemanager.runtime.linux.docker.ecr-auto-authentication.enabled` を `true` に設定します。

自動認証を無効にする方法

デフォルトでは、信頼できるレジストリで Amazon ECR レジストリが検出されない場合、自動認証は無効になります。

信頼できるレジストリに Amazon ECR レジストリが設定されている場合でも自動認証を無効にするには、設定分類を使用して `yarn.nodemanager.runtime.linux.docker.ecr-auto-authentication.enabled` を `false` に設定します。

クラスターで自動認証が有効になっているかどうかを確認する方法

マスターノードで、vi などのテキストエディタを使用して、ファイル `vi /etc/hadoop/conf.empty/yarn-site.xml` のコンテンツを表示します。`yarn.nodemanager.runtime.linux.docker.ecr-auto-authentication.enabled` の値を確認します。

クラスターの終了を制御する

このセクションでは、Amazon EMR クラスターをシャットダウンするためのオプションについて説明します。自動終了と終了保護、およびそれらが他の Amazon EMR 機能とやり取りする方法の詳細を取り上げます。

Amazon EMR クラスターは、次の方法でシャットダウンできます。

- 最後のステップ実行後の終了 - すべてのステップが完了した後にシャットダウンする一時的なクラスターを作成します。
- 自動終了 (アイドル後) - 指定したアイドル時間後にシャットダウンする自動終了ポリシーを使用してクラスターを作成します。詳細については、「自動終了ポリシーを使用する」を参照してください。
- 手動終了 - クラスターを意図的に終了するまで実行を継続する長時間稼働クラスターを作成します。手動でクラスターを終了する方法については、「クラスターを終了する」を参照してください。

また、クラスターで終了保護を設定して、事故やエラーによる EC2 インスタンスのシャットダウンを回避することもできます。

Amazon EMR がクラスターをシャットダウンすると、クラスター内のすべての Amazon EC2 インスタンスがシャットダウンします。インスタンスストアおよび EBS ボリューム内のデータを使用したり、復元したりすることはできなくなります。Simple Storage Service (Amazon S3) への書き込みとコストのバランス調整によってデータを管理および保持する戦略を策定するにあたっては、クラスターの終了について理解し、それを管理することが重要です。

トピック

ステップ実行後に継続または終了するようにクラスターを設定する

このトピックでは、長時間稼働クラスターの使用と、最後のステップの実行後にシャットダウンする一時的なクラスターの作成の違いについて説明します。また、クラスターのステップ実行を設定する方法についても説明します。

長時間稼働クラスターを作成する

デフォルトでは、コンソールまたは で作成したクラスター AWS CLI は長時間実行されます。長時間稼働クラスターは、シャットダウンするアクションを実行するまで、実行を続け、作業を受け入れ、料金が発生します。

長時間稼働クラスターは、次のような状況で効果的です。

- 対話式または自動的にデータをクエリする必要がある場合。
- クラスターでホストされるビッグデータアプリケーションと継続的にやり取りする必要がある場合。
- 非常に大きいデータセットを定期的に処理したり、頻繁に処理したりするため、毎回新しいクラスターを起動してデータをロードするのが非効率である場合。

また、長時間稼働クラスターで終了保護を設定して、事故やエラーによる EC2 インスタンスのシャットダウンを回避することもできます。詳細については、「終了保護の使用」を参照してください。

> **Note**
>
> Amazon EMR は、複数のプライマリノードを持つすべてのクラスターに対して終了保護を自動的に有効にし、クラスターの作成時に指定したステップ実行設定をオーバーライドします。クラスターの起動後に、終了保護を無効にできます。実行中のクラスターに対する終了保護の設定 を参照してください。複数のプライマリノードを持つクラスターをシャットダウンするには、まずクラスター属性を変更して、終了保護を無効にする必要があります。手順については、「複数のプライマリノードを持つ Amazon EMR クラスターの終了」を参照してください。

ステップ実行後に終了するようにクラスターを設定する

ステップ実行後に終了するように設定する場合、クラスターが起動し、ブートストラップアクションが実行され、指定したステップが実行されます。最後のステップが完了するとすぐに、Amazon EMR はクラスターの Amazon EC2 インスタンスを終了します。Amazon EMR API を使用して起動するクラスターでは、ステップ実行がデフォルトで有効になっています。

ステップ実行後の終了は、毎日のデータ処理など、定期的な処理タスクを実行するクラスターに効果的です。また、ステップ実行により、データの処理に必要な時間分の料金のみが課金されるようになります。ステップの詳細については、「クラスターへの作業の送信」を参照してください。

> **Note**
>
> Amazon EMR コンソールは、再設計され、使いやすくなりました。古いコンソールと新しいコンソールエクスペリエンスの違いについては、「Amazon EMR コンソール」を参照してください。

Console

コンソールでステップ実行後に終了を有効にするには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で、[クラスター] を選択し、[クラスターの作成] を選択します

3. [ステップ] で [ステップの追加] を選択します。[ステップを追加] ダイアログボックスで、適切なフィールド値を入力します。オプションは、ステップタイプによって異なります。ステップを追加してダイアログを終了するには、[ステップの追加] を選択します。
4. [クラスターの終了] の [最後のステップの完了後にクラスターを終了] チェックボックスを選択します。
5. クラスターに適用するその他のオプションを選択します。
6. クラスターを起動するには、[クラスターの作成] を選択します。

AWS CLI

を使用してステップ実行後に終了を有効にするには AWS CLI

- `--auto-terminate` コマンドを使用して一時的なクラスターを作成するときに、`create-cluster` パラメータを指定します。

  次の例は `--auto-terminate` パラメータを使用する方法を示しています。次のコマンドを入力し、*myKey* を EC2 キーペアの名前に置き換えます。

  > **Note**
  >
  > 読みやすくするために、Linux 行連続文字 (\) が含まれています。Linux コマンドでは、これらは削除することも、使用することもできます。Windows の場合、削除するか、キャレット (^) に置き換えてください。

```
aws emr create-cluster --name "Test cluster" --release-label emr-7.1.0 \
--applications Name=Hive Name=Pig --use-default-roles --ec2-attributes
 KeyName=myKey \
--steps Type=PIG,Name="Pig Program",ActionOnFailure=CONTINUE,\
Args=[-f,s3://mybucket/scripts/pigscript.pig,-p,\
INPUT=s3://mybucket/inputdata/,-p,OUTPUT=s3://mybucket/outputdata/,\
$INPUT=s3://mybucket/inputdata/,$OUTPUT=s3://mybucket/outputdata/]
--instance-type m5.xlarge --instance-count 3 --auto-terminate
```

API

クラスター起動時に Amazon EMR API を使用してステップ実行後に終了を無効にするには

1. RunJobFlow アクションを使用してクラスターを作成する場合は、KeepJobFlowAliveWhenNoStepsプロパティを に設定しますfalse。
2. クラスター起動後の Amazon EMR API を使用してステップ実行後に終了の設定を変更するには：

   SetKeepJobFlowAliveWhenNoSteps アクションを使用します。

自動終了ポリシーを使用する

自動終了ポリシーを使用すると、未使用のクラスターをモニタリングして手動で終了することなく、クラスターのクリーンアップをオーケストレーションできます。クラスターに自動終了ポリシーを追加する場合、クラスターが自動的にシャットダウンするまでのアイドル時間を指定します。

リリースバージョンに応じて、Amazon EMR は異なる基準を使用してクラスターをアイドルとしてマークします。次の表は、Amazon EMR がクラスターのアイドル状態を決定する方法の概要を示しています。

使用するバージョン	クラスターがアイドル状態と見なされる基準
Amazon EMR バージョン 5.34.0 以降、および 6.4.0 以降	• アクティブな YARN アプリケーションがない • HDFSの使用率が 10% を下回っている • アクティブな EMR ノートブックまたは EMR Studio 接続がない • クラスター上のアプリケーションユーザインターフェイスが使用されていない • 保留中のステップはありません
Amazon EMR バージョン 5.30.0 ~ 5.33.0 および 6.1.0 ~ 6.3.0	•

使用するバージョン	クラスターがアイドル状態と見なされる基準
	アクティブな YARN アプリケーションがない • クラスターにアクティブな Spark ジョブがない **Note** Amazon EMR はクラスターをアイドルとしてマークし、アクティブな Python3 カーネルがある場合でも、クラスターを自動的に終了することがあります。これは、Python3 カーネルを実行しても Spark ジョブがクラスターで送信されないためです。Python3 カーネルで自動終了を使用するには、Amazon EMR バージョン 6.4.0 以降を使用することをお勧めします。

Note

Amazon EMR バージョン 6.4.0 以降では、プライマリノードでのアクティビティを検出するためのクラスター上のファイル `/emr/metricscollector/isbusy` がサポートされています。クラスターを使用してシェルスクリプトまたは非 Yarn アプリケーションを実行する場合、`isbusy` を定期的に変更または更新して、クラスターがアイドル状態ではないことを Amazon EMR に伝えることができます。

クラスターの作成時に自動終了ポリシーをアタッチしたり、既存のクラスターにポリシーを追加したりできます。自動終了を変更または無効にするには、ポリシーを更新または削除することができます。

考慮事項

自動終了ポリシーを使用する前に、次の機能と制限を考慮してください。

- 次の では AWS リージョン、Amazon EMR 6.14.0 以降で Amazon EMR 自動終了を使用できます。
  - アジアパシフィック (ハイデラバード) (ap-south-2)
  - アジアパシフィック (ジャカルタ) (ap-southeast-3)
  - 欧州 (スペイン) (eu-south-2)
- 次の では AWS リージョン、Amazon EMR 5.30.0 および 6.1.0 以降で Amazon EMR 自動終了を使用できます。
  - 米国東部 (バージニア北部) (us-east-1)
  - 米国東部 (オハイオ) (us-east-2)
  - 米国西部 (オレゴン) (us-west-2)
  - 米国西部 (北カリフォルニア) (us-west-1)
  - アフリカ (ケープタウン) (af-south-1)
  - アジアパシフィック (香港) (ap-east-1)
  - アジアパシフィック (ムンバイ) (ap-south-1)
  - アジアパシフィック (ソウル) (ap-northeast-2)
  - アジアパシフィック (シンガポール) (ap-southeast-1)
  - アジアパシフィック (シドニー) (ap-southeast-2)
  - アジアパシフィック (東京) (ap-northeast-1)
  - カナダ (中部) (ca-central-1)
  - 南米 (サンパウロ) (sa-east-1)
  - ヨーロッパ (フランクフルト) (eu-central-1)
  - 欧州 (アイルランド) (eu-west-1)
  - ヨーロッパ (ロンドン) (eu-west-2)
  - 欧州 (ミラノ) (eu-south-1)
  - 欧州 (パリ) (eu-west-3)
  - 欧州 (ストックホルム) (eu-north-1)
  - 中国 (北京) (cn-north-1)
  - 中国 (寧夏) (cn-northwest-1)

  - AWS GovCloud（米国東部) (us-gov-east-1)
  - AWS GovCloud（米国西部) (us-gov-west-1)
- アイドルタイムアウトは、時間を指定しない場合、デフォルトで 60 分 (1 時間) です。最小アイドルタイムアウトには 1 分、最大アイドルタイムアウトには 7 日間を指定できます。
- Amazon EMR バージョン 6.4.0 以降では、Amazon EMR コンソールで新しいクラスターを作成すると、自動終了がデフォルトで有効になります。
- Amazon EMR は、クラスターの自動終了を有効にすると、高解像度 Amazon CloudWatch メトリクスを発行します。これらのメトリクスを使用して、クラスターのアクティビティとアイドル状態を追跡できます。詳細については、「クラスター容量メトリクス」を参照してください。
- Presto、Trino、HBase などの非 Yarn アプリケーションを使用する場合、自動終了はサポートされません。
- 自動終了を使用するには、メトリクスコレクタープロセスが API Gateway の自動終了用のパブリック API エンドポイントに接続できる必要があります。でプライベート DNS 名を使用する場合 Amazon Virtual Private Cloud、自動終了は正しく機能しません。自動終了が動作するようにするには、次のアクションの 1 つを実行することをお勧めします。
  - Amazon VPC から API Gateway のインターフェイス VPC エンドポイントを削除します。
  - 「VPC から API Gateway API に接続するときに「HTTP 403 Forbidden」エラーが発生するのはなぜですか?」の手順に従い、プライベート DNS 名の設定を無効にします。
  - 代わりに、クラスターをプライベートサブネットで起動します。詳細については、「プライベートサブネット」のトピックを参照してください。
- (EMR 5.30.0 以降) プライマリセキュリティグループのデフォルトの [すべて許可] アウトバウンドルールを削除して 0.0.0.0/ にした場合、サービスアクセス用のセキュリティグループへのアウトバウンド TCP 接続をポート 9443 で許可するルールを追加する必要があります。サービスアクセス用のセキュリティグループで、プライマリセキュリティグループからのインバウンド TCP トラフィックをポート 9443 で許可する必要もあります。セキュリティグループの設定の詳細については、「Amazon EMR-managed security group for the primary instance (private subnets))」を参照してください。

自動終了を使用するためのアクセス許可

Amazon EMR の自動終了ポリシーを適用および管理するには、次の EMR クラスターを管理する IAM リソースへの IAM アクセス許可ポリシーの例に記載されている、アクセス許可にアタッチする必要があります。

```
{
  "Version": "2012-10-17",
  "Statement": {
      "Sid": "AllowAutoTerminationPolicyActions",
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:PutAutoTerminationPolicy",
        "elasticmapreduce:GetAutoTerminationPolicy",
        "elasticmapreduce:RemoveAutoTerminationPolicy"
      ],
      "Resource": "<your-resources>"
    }
}
```

自動終了ポリシーをアタッチ、更新、または削除する

このセクションでは、Amazon EMR クラスターに自動終了ポリシーをアタッチ、更新、または削除する手順について説明します。自動終了ポリシーを使用する前に、必要な IAM アクセス許可があることを確認してください。自動終了を使用するためのアクセス許可 を参照してください。

> **Note**
>
> Amazon EMR コンソールは、再設計され、使いやすくなりました。古いコンソールと新しいコンソールエクスペリエンスの違いについては、「Amazon EMR コンソール」を参照してください。

New console

新しいコンソールでクラスターの作成時に自動終了ポリシーをアタッチするには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で、[クラスター] を選択し、[クラスターの作成] を選択します
3. [クラスターの終了] で、[アイドル時間後にクラスターを終了] を選択します。
4. クラスターが自動終了するまでのアイドル時間および分数を指定します。デフォルトのアイドル時間は 1 時間です。
5. クラスターに適用するその他のオプションを選択します。

6. クラスターを起動するには、[クラスターの作成] を選択します。

新しいコンソールを使用して、実行中のクラスターで自動終了ポリシーをアタッチ、更新、または削除するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で [クラスター] を選択し、更新するクラスターを選択します。
3. クラスターの詳細ページの [プロパティ] タブで、[クラスターの終了] を見つけて [編集] を選択します。
4. [自動終了を有効にする] を選択または選択解除して、機能を有効または無効にします。自動終了を有効にする場合は、クラスターが自動終了するまでのアイドル時間および分数を指定します。次に [変更の保存] を選択して確定します。

Old console

古いコンソールでクラスターの作成時に自動終了ポリシーをアタッチするには

1. 新しい Amazon EMR コンソールに移動し、サイドナビゲーションから [古いコンソールに切り替え] を選択します。古いコンソールに切り替えたときの動作の詳細については、「Using the old console」を参照してください。
2. [クラスターを作成] を選択します。
3. [ハードウェア構成] を選択して、[Auto-termination] (自動終了) を選択します。
4. クラスターが自動終了するまでのアイドル時間および分数を指定します。デフォルトのアイドル時間は 1 時間です。
5. 必要に応じてアプリケーションの他の設定を選択し、[Create cluster (クラスターの作成)] を選択します。

古いコンソールを使用して、実行中のクラスターで自動終了ポリシーをアタッチ、更新、または削除するには

1. 新しい Amazon EMR コンソールに移動し、サイドナビゲーションから [古いコンソールに切り替え] を選択します。古いコンソールに切り替えたときの動作の詳細については、「Using the old console」を参照してください。

2. [クラスター] を選択し、更新するクラスターを選択します。
3. クラスター詳細ページで、[ハードウェア] を選択します。
4. [自動終了を有効にする] を選択または選択解除して、機能を有効または無効にします。自動終了を有効にする場合は、クラスターが自動終了するまでのアイドル時間および分数を指定します。

AWS CLI

開始する前に

自動終了ポリシーを使用する前に、 AWS CLIを最新バージョンに更新することをお勧めします。手順については、「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を使用して自動終了ポリシーをアタッチまたは更新するには

- `aws emr put-auto-termination-policy` コマンドを使用して、クラスターに自動終了ポリシーをアタッチまたは更新します。

  次の例では、 に 3600 秒を指定します*IdleTimeout*。を指定しない場合*IdleTimeout*、値はデフォルトで 1 時間になります。

```
aws emr put-auto-termination-policy \
--cluster-id <your-cluster-id> \
--auto-termination-policy IdleTimeout=3600
```

> **Note**
>
> 読みやすくするために、Linux 行連続文字 (\) が含まれています。Linux コマンドでは、これらは削除することも、使用することもできます。Windows の場合、削除するか、キャレット (^) に置き換えてください。

`aws emr create-cluster` コマンドを使用する場合は、`--auto-termination-policy` の値を指定することもできます。での Amazon EMR コマンドの使用の詳細については AWS CLI、AWS CLI 「 コマンドリファレンス」を参照してください。

を使用して自動終了ポリシーを削除するには AWS CLI

- `aws emr remove-auto-termination-policy` コマンドを使用して、クラスターから自動終了ポリシーを削除します。での Amazon EMR コマンドの使用の詳細については AWS CLI、AWS CLI 「コマンドリファレンス」を参照してください。

```
aws emr remove-auto-termination-policy --cluster-id <your-cluster-id>
```

終了保護の使用

終了保護は、クラスターを偶発的な終了から保護します。これは、重要なワークロードを処理する長時間稼働クラスターに特に役立ちます。長時間稼働クラスターで削除保護が有効になっていてもクラスターを終了することはできますが、最初にクラスターから明示的に削除保護を削除する必要があります。これにより、事故やエラーで EC2 インスタンスがシャットダウンされることがなくなります。削除保護はクラスターの作成時に有効にできます。また、実行中のクラスターで設定を変更することもできます。

終了保護が有効になっていると、Amazon EMR API の `TerminateJobFlows` アクションは機能しません。ユーザーは、この API や AWS CLIの `terminate-clusters` コマンドでクラスターを終了することはできません。この API はエラーを返し、CLI はゼロ以外のリターンコードで終了します。Amazon EMR コンソールを使用してクラスターを終了する場合、終了保護を無効にする追加ステップが示されます。

> **Warning**
>
> 終了保護は、ヒューマンエラーや回避策が発生した場合にデータが保持されることを保証するものではありません。例えば、SSH を使用してインスタンスに接続しているときにコマンドラインから再起動コマンドが発行された場合、インスタンスで実行されているアプリケーションまたはスクリプトによって再起動コマンドが発行された場合、または Amazon EC2 や Amazon EMR API が使用されて終了保護が無効にされた場合などです。これは、Amazon EMR リリース 7.1 以降を実行していて、インスタンスが異常で回復不能になった場合にも当てはまります。終了保護が有効になっている場合でも、HDFS データを含むインスタンスストレージに保存されたデータは失われる可能性があります。Simple Storage Service (Amazon S3) のロケーションにデータ出力を書き込み、ビジネス継続性の要件に応じてバックアップ戦略を作成します。

削除保護は、以下のアクションによるクラスターリソースのスケールに影響することはありません。

- AWS Management Console または を使用してクラスターを手動でサイズ変更します AWS CLI。詳細については、「実行中のクラスターのサイズを手動で変更する」を参照してください。
- スケールインポリシーと自動スケーリングを使用して、コアまたはタスクインスタンスグループからインスタンスを削除する。詳細については、「カスタムポリシーによる自動スケーリングをインスタンスグループに使用する」を参照してください。
- ターゲット容量を減らしてインスタンスフリートからインスタンスを削除する。詳細については、「インスタンスフリートオプション」を参照してください。

終了保護と Amazon EC2

Amazon EMR クラスターの終了保護設定は、クラスター内のすべての Amazon EC2 インスタンスの `DisableApiTermination` 属性に対応します。例えば、EMR クラスターで終了保護を有効にすると、Amazon EMR は EMR クラスター内のすべての EC2 インスタンスに対して を自動的に `DisableApiTermination` true に設定します。終了保護を無効にする場合も同様です。Amazon EMR は`DisableApiTermination`、EMR クラスター内のすべての EC2 インスタンスに対して を自動的に false に設定します。Amazon EMR からクラスターを終了またはスケールダウンし、EC2 インスタンスの Amazon EC2 設定が競合する場合、Amazon EMR は Amazon EC2 の `DisableApiStop`および 設定よりも Amazon EMR `DisableApiTermination`設定を優先し、EC2 インスタンスを終了し続けます。 Amazon EC2

例えば、Amazon EC2 コンソールを使用して、終了保護が無効になっている EMR クラスター内の Amazon EC2 インスタンスで終了保護を有効にできます。Amazon EMR コンソール、 AWS CLI、または Amazon EMR API を使用してクラスターを終了またはスケールダウンすると、Amazon EMR は `DisableApiTermination`設定を上書きし、false に設定して、他のインスタンスとともにインスタンスを終了します。

Amazon EC2 コンソールを使用して、終了保護が無効になっている EMR クラスター内の Amazon EC2 インスタンスで停止保護を有効にすることもできます。クラスターを終了またはスケールダウンすると、Amazon EMR は Amazon EC2 で `DisableApiStop`を false に設定し、他のインスタンスとともにインスタンスを終了します。

Amazon EMR は、クラスターを終了またはスケールダウンする場合にのみ、 `DisableApiStop` 設定を上書きします。EMR クラスターで終了保護を有効または無効にしても、Amazon EMR はそれぞれの EMR クラスター内の EC2 インスタンスの`disableApiStop`設定を変更しません。

> **Important**
>
> 終了保護付きの Amazon EMR クラスターの一部としてインスタンスを作成し、Amazon EC2 API または AWS CLI コマンドを使用して `DisableApiTermination`が になるようにインスタンスを変更し`false`、Amazon EC2 API または AWS CLI コマンドで `TerminateInstances`オペレーションを実行すると、Amazon EC2 インスタンスは終了します。

終了保護と異常な状態の YARN ノード

Amazon EMR は、クラスターのコアおよびタスクの Amazon EC2 インスタンスで実行されている Apache Hadoop YARN のノードのステータスを定期的にチェックします。ヘルスステータスは、NodeManager ヘルスチェッカーサービス によって報告されます。ノードが を報告するとUNHEALTHY、Amazon EMR インスタンスコントローラーはノードを拒否リストに追加し、再び正常になるまで YARN コンテナを割り当てません。終了保護、異常なノード交換、Amazon EMR リリースバージョンのステータスに応じて、Amazon EMR は異常なインスタンスを置き換えるか、インスタンス へのコントローラーの割り当てを停止します。

ステップ実行後の終了保護と終了

ステップ実行後に終了を有効にし、終了保護を有効にすると、Amazon EMR は終了保護を無視します。

クラスターにステップを送信するときに `ActionOnFailure` プロパティを設定することにより、エラーが原因でステップの実行を完了できなかった場合に何が起きるのかを確認できます。この設定に使用できる値は、`TERMINATE_CLUSTER` (以前のバージョンでは `TERMINATE_JOB_FLOW`)、`CANCEL_AND_WAIT`、および `CONTINUE` です。詳細については、「クラスターへの作業の送信」を参照してください。

を `ActionOnFailure`に設定して設定されたステップが失敗した場合`CANCEL_AND_WAIT`、ステップ実行が有効になった後に終了すると、クラスターは後続のステップを実行せずに終了します。

`ActionOnFailure` が `TERMINATE_CLUSTER` に設定されているステップが失敗した場合は、以下の設定の表で結果を確認してください。

ActionOnFailure	ステップ実行後の終了	終了保護	結果

ActionOnFailure	ステップ実行後の終了	終了保護	結果
TERMINATE_CLUSTER	有効	無効	クラスターが終了
	有効	有効	クラスターが終了
	無効	有効	クラスターの稼働が継続
	無効	無効	クラスターが終了

終了保護とスポットインスタンス

Amazon EMR の終了保護を有効にしても、スポット料金が最大スポット料金を超えた場合の Amazon EC2 スポットインスタンスの終了を防ぐことはできません。

クラスターを起動するときに終了保護を設定する

コンソール、、または API を使用してクラスターを起動するときに、終了保護を有効 AWS CLIまたは無効にできます。

単一ノードクラスターの場合、デフォルトの終了保護設定は次のとおりです。

- Amazon EMR コンソールによるクラスターの起動 — 終了保護はデフォルトで無効になっています。
- によるクラスターの起動 AWS CLI `aws emr create-cluster`— が指定されていない限り、終了保護は無効`--termination-protected`になります。
- Amazon EMR API RunJobFlow コマンドによるクラスターの起動 — ブール値が `TerminationProtected` に設定されていない限り、終了保護は無効になります`true`。

高可用性クラスターの場合、デフォルトの終了保護設定は次のとおりです。

- Amazon EMR コンソールによるクラスターの起動 — 終了保護はデフォルトで有効になっています。

- によるクラスターの起動 AWS CLI `aws emr create-cluster`— `--termination-protected`が指定されていない限り、Termination Protection は無効です。
- Amazon EMR API RunJobFlow コマンドによるクラスターの起動 — ブール値が `TerminationProtected` に設定されていない限り、終了保護は無効になりますtrue。

Console

コンソールでクラスターを作成するときに終了保護をオンまたはオフにするには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で、[クラスター] を選択し、[クラスターの作成] を選択します
3. [EMR リリースバージョン] で、[emr-6.6.0] 以降を選択します。
4. クラスターの終了とノードの交換 で、終了保護の使用が事前に選択されていることを確認するか、選択をクリアしてオフにします。
5. クラスターに適用するその他のオプションを選択します。
6. クラスターを起動するには、[クラスターの作成] を選択します。

AWS CLI

を使用してクラスターを作成するときに終了保護をオンまたはオフにするには AWS CLI

- では AWS CLI、 `--termination-protected`パラメータを指定して `create-cluster` コマンドを使用して、終了保護を有効にしたクラスターを起動できます。削除保護はデフォルトで無効になっています。

  次の例では、削除保護を有効にしたクラスターを作成します。

  > **Note**
  >
  > 読みやすくするために、Linux 行連続文字 (\) が含まれています。Linux コマンドでは、これらは削除することも、使用することもできます。Windows の場合、削除するか、キャレット (^) に置き換えてください。

```
aws emr create-cluster --name "TerminationProtectedCluster" --release-
label emr-7.1.0 \
--applications Name=Hadoop Name=Hive Name=Pig \
--use-default-roles --ec2-attributes KeyName=myKey --instance-type m5.xlarge \
--instance-count 3 --termination-protected
```

での Amazon EMR コマンドの使用の詳細については AWS CLI、「」を参照してくださいhttps://docs.aws.amazon.com/cli/latest/reference/emr。

実行中のクラスターに対する終了保護の設定

コンソールまたは AWS CLIを使用して、実行中のクラスターに対して終了保護を設定することができます。

Console

コンソールを使用して実行中のクラスターの終了保護をオンまたはオフにするには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で [クラスター] を選択し、更新するクラスターを選択します。
3. クラスターの詳細ページの [プロパティ] タブで、[クラスターの終了] を見つけて [編集] を選択します。
4. [終了保護を使用] チェックボックスを選択または選択解除」して、機能をオンまたはオフにします。次に [変更の保存] を選択して確定します。

AWS CLI

を使用して実行中のクラスターの終了保護をオンまたはオフにするには AWS CLI

- AWS CLIを使用して実行中のクラスターで終了保護を有効にするには、`modify-cluster-attributes` コマンドと `--termination-protected` パラメータを使用します。削除保護を無効にするには、`--no-termination-protected` パラメータを使用します。

  次の例では、ID *j-3KVTXXXXXX7UG* のクラスターで削除保護を有効にします。

```
aws emr modify-cluster-attributes --cluster-id j-3KVTXXXXXX7UG --termination-
protected
```

次の例では、同じクラスターで削除保護を無効にします。

```
aws emr modify-cluster-attributes --cluster-id j-3KVTXXXXXX7UG --no-termination-
protected
```

異常なノードの交換

Amazon EMR は、Apache Hadoop のNodeManager ヘルスチェッカーサービスを定期的に使用して、Amazon EC2 クラスター上の Amazon EMR のコアノードのステータスをモニタリングします。ノードが機能的に最適でない場合、ヘルスチェッカーはそのノードを Amazon EMR コントローラーに報告します。Amazon EMR コントローラーはノードを拒否リストに追加し、ノードのステータスが改善するまでノードが新しい YARN アプリケーションを受信できないようにします。ノードが異常になる一般的な理由の 1 つは、ディスクを過剰に使用していることです。異常なノードの特定と復旧の詳細については、「リソースエラー」を参照してください。

Amazon EMR が異常なノードを終了するか、クラスターに保持するかを選択できます。異常なノード交換をオフにすると、異常なノードは拒否リストに残り、クラスター容量にカウントされ続けます。引き続き Amazon EC2 コアインスタンスに接続して設定と復旧を行うことができます。そのため、クラスターのサイズを変更して容量を追加できます。終了保護がオンになっていても、Amazon EMR は異常なノードを置き換えることに注意してください。

異常なノード置換がオンの場合、Amazon EMR は異常なコアノードを終了し、インスタンスグループ内のインスタンス数またはインスタンスフリートのターゲット容量に基づいて新しいインスタンスをプロビジョニングします。複数またはすべてのコアノードが 45 分以上異常である場合、Amazon EMR はノード を正常に置き換えます。

> **Important**
>
> Amazon EMR が異常のあるコアインスタンスを適切に置き換えるため、HDFS データを完全に失う可能性を回避するには、常にデータをバックアップすることをお勧めします。

Amazon EMR は、異常なノード置換のために Amazon CloudWatch Events を発行するため、異常なコアインスタンスで何が起こっているかを追跡できます。詳細については、「異常なノード交換イベント」を参照してください。

デフォルトのノード交換および終了保護設定

異常なノード交換はすべての Amazon EMR リリースで使用できますが、デフォルト設定は選択したリリースラベルによって異なります。これらの設定は、新しいクラスターを作成するときに異常なノード置換を設定するか、いつでもクラスター設定に移動することで変更できます。

Amazon EMR リリース 7.0 以下を実行している単一ノードクラスターまたは高可用性クラスターを作成する場合、異常のあるノード置換のデフォルト設定は終了保護によって異なります。

- 終了保護を有効にすると、異常なノード交換が無効になります。
- 終了保護を無効にすると、異常なノード交換が可能になります。

クラスターの起動時に異常なノード交換を設定する

コンソール、、または API を使用してクラスターを起動するときに、異常なノード交換を有効 AWS CLIまたは無効にできます。

デフォルトの異常なノード交換設定は、クラスターの起動方法によって異なります。

- Amazon EMR コンソール — 異常のあるノード交換はデフォルトで有効になっています。
- AWS CLI `aws emr create-cluster` — を指定しない限り、異常なノード置換はデフォルトで有効になっています`--no-unhealthy-node-replacement`。
- Amazon EMR RunJobFlow API コマンド — ブ`UnhealthyNodeReplacement`ール値を `True`または に設定しない限り、異常なノード置換はデフォルトで有効になっています`False`。

Console

コンソールでクラスターを作成するときに、異常なノード交換をオンまたはオフにするには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で、[クラスター] を選択し、[クラスターの作成] を選択します

3. EMR リリースバージョン では、必要な Amazon EMR リリースラベルを選択します。
4. クラスターの終了とノードの置換 で、異常のあるノードの置換 (推奨) が事前に選択されていることを確認するか、選択を解除してオフにします。
5. クラスターに適用するその他のオプションを選択します。
6. クラスターを起動するには、[クラスターの作成] を選択します。

AWS CLI

を使用してクラスターを作成するときに、異常なノード交換をオンまたはオフにするには AWS CLI

- では AWS CLI、 `--unhealthy-node-replacement` パラメータを指定して `create-cluster` コマンドを使用して、異常なノード置換を有効にしたクラスターを起動できます。異常のあるノード交換は、デフォルトでオンになっています。

  次の例では、異常なノード置換を有効にしてクラスターを作成します。

  > **Note**
  >
  > 読みやすくするために、Linux 行連続文字 (\) が含まれています。Linux コマンドでは、これらは削除することも、使用することもできます。Windows の場合、削除するか、キャレット (^) に置き換えてください。

```
aws emr create-cluster --name "SampleCluster" --release-label emr-7.1.0 \
--applications Name=Hadoop Name=Hive Name=Pig \
--use-default-roles --ec2-attributes KeyName=myKey --instance-type m5.xlarge \
--instance-count 3 --unhealthy-node-replacement
```

  での Amazon EMR コマンドの使用の詳細については AWS CLI、「Amazon EMR AWS CLI コマンド」を参照してください。

実行中のクラスターでの異常なノード交換の設定

コンソール、、 AWS CLIまたは API を使用して、実行中のクラスターで異常なノード交換をオンまたはオフにできます。

Console

コンソールを使用して実行中のクラスターで異常なノード交換をオンまたはオフにするには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で [クラスター] を選択し、更新するクラスターを選択します。
3. クラスターの詳細ページの「プロパティ」タブで、クラスターの終了とノードの交換を見つけ、「編集」を選択します。
4. 異常のあるノード交換チェックボックスをオンまたはオフにして、機能をオンまたはオフにします。次に [変更の保存] を選択して確定します。

AWS CLI

を使用して実行中のクラスターで異常なノード交換をオンまたはオフにするには AWS CLI

- で実行中のクラスターで異常のあるノード置換を有効にするには AWS CLI、 `--unhealthy-node-replacement`パラメータを指定して `modify-cluster-attributes` コマンドを使用します。削除保護を無効にするには、`--no-unhealthy-node-replacement` パラメータを使用します。

  次の例では、ID *j-3KVTXXXXXX7UG* のクラスターで異常のあるノード置換を有効にします。

  ```
  aws emr modify-cluster-attributes --cluster-id j-3KVTXXXXXX7UG --unhealthy-node-replacement
  ```

  次の例では、同じクラスターで異常なノード交換をオフにします。

  ```
  aws emr modify-cluster-attributes --cluster-id j-3KVTXXXXXX7UG --no-unhealthy-node-replacement
  ```

Amazon EMR でAmazon Linux AMI を使用する

Amazon Linux Amazon マシンイメージ (AMI)

クラスターを作成して起動するときに、Amazon EMR では Amazon Linux Amazon マシンイメージ (AMI) を使用して Amazon EC2 インスタンスを初期化します。AMI には、各インスタンスでクラスターアプリケーションをホストするために必要な Amazon Linux オペレーティングシステム、その他のソフトウェア、および設定が含まれています。

デフォルトでは、クラスターを作成するときに、Amazon EMR は使用する Amazon EMR リリースバージョン用に作成されたデフォルトの Amazon Linux AMI を使用します。デフォルトの Amazon Linux AMI の詳細については、「Amazon EMR にデフォルトの Amazon Linux AMI を使用する」を参照してください。Amazon EMR 5.7.0 以降を使用する場合は、Amazon EMR 用のデフォルトの Amazon Linux AMI ではなく、カスタム Amazon Linux AMI を指定できます。カスタム AMI では、ルートデバイスボリュームを暗号化し、ブートストラップアクションを使用する代わりにアプリケーションと設定をカスタマイズできます。Amazon EMR クラスターのインスタンスグループまたはインスタンスフリートの設定で、インスタンスタイプごとにカスタム AMI を指定できます。複数のカスタム AMI サポートにより、クラスターで複数のアーキテクチャタイプを柔軟に使用することができます。カスタム AMI の使用 を参照してください。

Amazon EMR は、すべての AMI 用のルートデバイスとして Amazon EBS の汎用 SSD ボリュームを自動的にアタッチします。EBS に支えられた AMI はパフォーマンスを強化します。Amazon Linux AMI の詳細については、「Amazon マシンイメージ (AMI)」を参照してください。Amazon EMR インスタンスのインスタンスストレージの詳細については、「インスタンスストレージ」を参照してください。

Amazon EMR にデフォルトの Amazon Linux AMI を使用する

カスタム AMI を指定しない限り、各 Amazon EMR リリースバージョンでは Amazon EMR 用のデフォルトの Amazon Linux AMI が使用されます。Amazon EMR 5.36 以降、Amazon EMR 6.6 および Amazon EMR 7.0 は、Amazon EMR デフォルト AMI で Amazon Linux 2 (EMR 5.x の場合は AL2、EMR 7.x の場合は AL2023) を更新するためのデフォルトの動作をリリースします。これは、デフォルトの Amazon EMR AMI に最新の Amazon Linux リリースを自動的に適用することです。

Amazon EMR リリース用 Amazon Linux の自動更新

Amazon EMR 7.0 以降、6.6 以降、5.36 以降の最新のパッチリリースを使用してクラスターを起動すると、Amazon EMR はデフォルトの Amazon EMR AMI に最新の Amazon Linux リリースを使用します。例:

- `x.x.0` と `x.x.1` がリリースされている場合、`x.x.1` リリースの起動時に `x.x.0` リリースは AMI 更新を取得しなくなります。
- 同様に、`x.x.2` の起動時に `x.x.1` は AMI 更新の取得を停止します。
- 後で `x.y.0` のリリース時に、`x.x.[latest]` は `x.y.[latest]` と共に AMI 更新を引き続き受信します。

Amazon EMR リリースの第 2 小数点の後の数字 (`6.8.1`) で示される最新のパッチリリースを使用しているかどうかを確認するには、Amazon EMR リリースガイドで利用可能なリリースを参照するか、コンソールでクラスターを作成するときに Amazon EMR リリースドロップダウンを確認するか、`ListReleaseLabels` API または `list-release-labels` CLI アクションを使用してください。Amazon EMR の新リリースのリリース時に更新を取得するには、リリースガイドの「新着情報」ページで RSS フィードを購読してください。

必要に応じて、Amazon EMR リリースに最初に付属していた Amazon Linux バージョンでクラスターを起動することもできます。クラスターの Amazon Linux リリースを指定する方法については、「EMR クラスター作成時の Amazon Linux リリースの変更」を参照してください。

デフォルトの Amazon Linux バージョン

トピック

- Amazon EMR 7.0 以降のデフォルト AMIs
- Amazon EMR 6.6 以降でデフォルトの AMI
- Amazon EMR 5.x のデフォルト AMI

Amazon EMR 7.0 以降のデフォルト AMIs

次の表に、Amazon EMR リリース 7.0 以降の最新のパッチバージョンの Amazon Linux 情報を示します。

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2023 年 3 月 2024 日 0304.0	6.1.79-99.164.amzn2023	2024 年 3 月 12 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-central-2 • eu-south-1 • eu-south-2 • ap-east-1 • ap-south-1 • ap-south-2 • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
			• me-central-1 • me-south-1 • ca-central-1 • il-central-1 • ca-west-1 • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2023 年 3 月 2024 日 219.0	6.1.77-99.164.amzn2023	2024 年 3 月 1 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-central-2 • eu-south-1 • eu-south-2 • ap-east-1 • ap-south-1 • ap-south-2 • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1

OsRelea Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
			• me-central-1 • me-south-1 • ca-central-1 • il-central-1 • ca-west-1 • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2023 年 3 月 2024 日 205.0	6.1.75-99.163.amzn2023	2024 年 2 月 19 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-central-2 • eu-south-1 • eu-south-2 • ap-east-1 • ap-south-1 • ap-south-2 • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
			• me-central-1 • me-south-1 • ca-central-1 • il-central-1 • ca-west-1 • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2023 年 3 月 2024 日 122.0	6.1.72-96.166.amzn2023	2024 年 2 月 5 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-central-2 • eu-south-1 • eu-south-2 • ap-east-1 • ap-south-1 • ap-south-2 • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1

OsRelea Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
			• me-central-1 • me-south-1 • ca-central-1 • il-central-1 • ca-west-1 • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2023 年 3 月 2024 日 108.0	6.1.72-96.166.amzn2023	2024 年 1 月 24 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-central-2 • eu-south-1 • eu-south-2 • ap-east-1 • ap-south-1 • ap-south-2 • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
			• me-central-1 • me-south-1 • ca-central-1 • il-central-1 • ca-west-1 • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2023 年 3 月 2023 日 1211.4	6.1.66-91.160.amzn2023	2023 年 12 月 19 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-central-2 • eu-south-1 • eu-south-2 • ap-east-1 • ap-south-1 • ap-south-2 • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
			• me-central-1 • me-south-1 • ca-central-1 • il-central-1 • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

Amazon EMR 6.6 以降でデフォルトの AMI

次の表は、Amazon EMR 6.6.x リリース 以降の最新のパッチバージョンに関する Amazon Linux 情報を示しています。

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20240223.0	4.14.336	2024 年 3 月 8 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
			• eu-west-2 • eu-west-3 • eu-central-1 • eu-central-2 (6.10.1 以降) • eu-south-1 • eu-south-2 (6.10.1 以降) • ap-east-1 • ap-south-1 • ap-south-2 (6.10.1 以降) • ap-southeast-3 • ap-southeast-4 (6.8.1+ および 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 (6.10.1 以降) • me-south-1 • ca-central-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
			• il-central-1 (6.8.1+ および 5.36.1) • ca-west-1 (6.9.1+ および 5.36.1) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20240131.0	4.14.336	2024 年 2 月 14 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-central-2 (6.10.1 以降) • eu-south-1 • eu-south-2 (6.10.1 以降) • ap-east-1 • ap-south-1 • ap-south-2 (6.10.1 以降) • ap-southeast-3 • ap-southeast-4 (6.8.1+ および 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
			• ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 (6.10.1 以降) • me-south-1 • ca-central-1 • il-central-1 (6.8.1+ および 5.36.1) • ca-west-1 (6.9.1+ および 5.36.1) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20240124.0	4.14.336	2024 年 2 月 7 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-central-2 (6.10.1 以降) • eu-south-1 • eu-south-2 (6.10.1 以降) • ap-east-1 • ap-south-1 • ap-south-2 (6.10.1 以降) • ap-southeast-3 • ap-southeast-4 (6.8.1+ および 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
			• ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 (6.10.1 以降) • me-south-1 • ca-central-1 • il-central-1 (6.8.1+ および 5.36.1) • ca-west-1 (6.9.1+ および 5.36.1) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20240109.0	4.14.334	2024 年 1 月 24 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-central-2 (6.10.1 以降) • eu-south-1 • eu-south-2 (6.10.1 以降) • ap-east-1 • ap-south-1 • ap-south-2 (6.10.1 以降) • ap-southeast-3 • ap-southeast-4 (6.8.1+ および 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
			• ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 (6.10.1 以降) • me-south-1 • ca-central-1 • il-central-1 (6.8.1+ および 5.36.1) • ca-west-1 (6.9.1+ および 5.36.1) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20231218.0	4.14.330	2024 年 1 月 2 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-central-2 (6.10+) • eu-south-1 • eu-south-2 (6.10+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10+) • ap-southeast-3 • ap-southeast-4 (6.8+ と 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
			• sa-east-1 • me-central-1 (6.10+) • me-south-1 • ca-central-1 • il-central-1 (6.8+ と 5.36.1) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20231206.0	4.14.330	2023 年 12 月 22 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-central-2 (6.10+) • eu-south-1 • eu-south-2 (6.10+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10+) • ap-southeast-3 • ap-southeast-4 (6.8+ と 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1

OsRelea Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
			• sa-east-1 • me-central-1 (6.10+) • me-south-1 • ca-central-1 • il-central-1 (6.8+ と 5.36.1) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20231116.0	4.14.328	2023 年 12 月 11 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-central-2 (6.10+) • eu-south-1 • eu-south-2 (6.10+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10+) • ap-southeast-3 • ap-southeast-4 (6.8+ と 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
			• sa-east-1 • me-central-1 (6.10+) • me-south-1 • ca-central-1 • il-central-1 (6.8+ と 5.36.1) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20231101.0	4.14.327	2023 年 11 月 17 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-central-2 (6.10+) • eu-south-1 • eu-south-2 (6.10+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10+) • ap-southeast-3 • ap-southeast-4 (6.8+ と 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1

OsRelea Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
			• sa-east-1 • me-central-1 (6.10+) • me-south-1 • ca-central-1 • il-central-1 (6.8+ と 5.36.1) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20230920.1	4.14.326	2023 年 11 月 7 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-central-2 (6.10+) • eu-south-1 • eu-south-2 (6.10+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10+) • ap-southeast-3 • ap-southeast-4 (6.8+ と 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
			• sa-east-1 • me-central-1 (6.10+) • me-south-1 • ca-central-1 • il-central-1 (6.8+ と 5.36.1) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20231012.1	4.14.326	2023 年 10 月 26 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-central-2 (6.10+) • eu-south-1 • eu-south-2 (6.10+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10+) • ap-southeast-3 • ap-southeast-4 (6.8+ と 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1

OsRelea Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
			• sa-east-1 • me-central-1 (6.10+) • me-south-1 • ca-central-1 • il-central-1 (6.8+ と 5.36.1) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20230926.0	4.14.322	2023 年 10 月 19 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-central-2 (6.10+) • eu-south-1 • eu-south-2 (6.10+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10+) • ap-southeast-3 • ap-southeast-4 (6.8+ と 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
			• sa-east-1 • me-central-1 (6.10+) • me-south-1 • ca-central-1 • il-central-1 (6.8+ と 5.36.1) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20238906.0	4.14.322	2023 年 10 月 4 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-central-2 (6.10+) • eu-south-1 • eu-south-2 (6.10+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10+) • ap-southeast-3 • ap-southeast-4 (6.8+ と 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1

OsRelea Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
			• sa-east-1 • me-central-1 (6.10+) • me-south-1 • ca-central-1 • il-central-1 (6.9+ と 5.36.1)

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20230822.0	4.14.322	2023 年 8 月 30 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-central-2 (6.10+) • eu-south-1 • eu-south-2 (6.10+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10+) • ap-southeast-3 • ap-southeast-4 (6.8+ と 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1

OsRelea Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
			• sa-east-1 • me-central-1 (6.10+) • me-south-1 • ca-central-1 • il-central-1 (6.9+ と 5.36.1)

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20230808.0	4.14.320	2023 年 8 月 24 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-central-2 (6.10+) • eu-south-1 • eu-south-2 (6.10+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10+) • ap-southeast-3 • ap-southeast-4 (6.8+ と 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1

OsRelea Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
			• sa-east-1 • me-central-1 (6.10+) • me-south-1 • ca-central-1 • il-central-1 (6.9+ と 5.36.1) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20230727.0	4.14.320	2023 年 8 月 14 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-central-2 (6.10+) • eu-south-1 • eu-south-2 (6.10+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10+) • ap-southeast-3 • ap-southeast-4 (6.8+ と 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1

OsRelea Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
			• sa-east-1 • me-central-1 (6.10+) • me-south-1 • ca-central-1 • il-central-1 (6.9+ と 5.36.1)

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20230719.0	4.14.320	2023 年 8 月 2 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-central-2 (6.10+) • eu-south-1 • eu-south-2 (6.10+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10+) • ap-southeast-3 • ap-southeast-4 (6.8+ と 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1

OsRelea Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
			• sa-east-1 • me-central-1 (6.10+) • me-south-1 • ca-central-1 • il-central-1 (6.9+ と 5.36.1)

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20230628.0	4.14.318	2023 年 7 月 12 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-central-2 (6.10+) • eu-south-1 • eu-south-2 (6.10+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10+) • ap-southeast-3 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1

OsRelea Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
			• me-central-1 (6.10+) • me-south-1 • ca-central-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20230612.0	4.14.314	2023 年 6 月 23 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-central-2 (6.10+) • eu-south-1 • eu-south-2 (6.10+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10+) • ap-southeast-3 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1

OsRelea Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
			• me-central-1 (6.10+) • me-south-1 • ca-central-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20230504.1	4.14.313	2023 年 5 月 16 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-central-2 (6.10+) • eu-south-1 • eu-south-2 (6.10+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10+) • ap-southeast-3 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1

OsRelea Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
			• me-south-1 • ca-central-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20230418.0	4.14.311	2023 年 5 月 3 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-central-2 (6.10 のみ) • eu-south-1 • eu-south-2 (6.10 のみ) • ap-east-1 • ap-south-1 • ap-south-2 (6.10 のみ) • ap-southeast-3 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1

OsRelea Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
			• sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20230404.1	4.14.311	2023 年 4 月 18 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-central-2 • eu-south-1 • eu-south-2 • ap-east-1 • ap-south-1 • ap-south-2 • ap-southeast-3 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1

OsRelea Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
			• me-south-1 • ca-central-1
2.0.2023 404.0	4.14.311	2023 年 4 月 10 日	• us-east-1 • eu-west-3

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20230320.0	4.14.309	2023 年 3 月 30 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-central-2 • eu-south-1 • eu-south-2 • ap-east-1 • ap-south-1 • ap-south-2 • ap-southeast-3 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1

OsRelea Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
			• me-south-1 • ca-central-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20230307.0	4.14.305	2023 年 3 月 15 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-central-2 • eu-south-1 • eu-south-2 • ap-east-1 • ap-south-1 • ap-south-2 • ap-southeast-3 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1

OsRelea Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
			• me-south-1 • ca-central-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20230207.0	4.14.304	2023 年 3 月 3 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-central-2 • eu-south-1 • eu-south-2 • ap-east-1 • ap-south-1 • ap-south-2 • ap-southeast-3 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
			• me-south-1 • ca-central-1
2.0.20230119.1	4.14.301	2023 年 2 月 9 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-south-1 • ap-east-1 • ap-south-1 • ap-southeast-3 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-south-1 • ca-central-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20221210.1	4.14.301	2023 年 1 月 12 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-south-1 • ap-east-1 • ap-south-1 • ap-southeast-3 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-south-1 • ca-central-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20221103.3	4.14.296	2022 年 12 月 5 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-south-1 • ap-east-1 • ap-south-1 • ap-southeast-3 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-south-1 • ca-central-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20220004.0	4.14.294	2022 年 11 月 2 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-south-1 • ap-east-1 • ap-south-1 • ap-southeast-3 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-south-1 • ca-central-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20220912.1	4.14.291	2022 年 10 月 7 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-south-1 • ap-east-1 • ap-south-1 • ap-southeast-3 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-south-1 • ca-central-1
2.0.20220805.0	4.14.287	2022 年 8 月 30 日	• us-west-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20220719.0	4.14.287	2022 年 8 月 10 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-south-1 • ap-east-1 • ap-south-1 • ap-southeast-3 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-south-1 • ca-central-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20220426.0	4.14.281	2022 年 6 月 10 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-south-1 • ap-east-1 • ap-south-1 • ap-southeast-3 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-south-1 • ca-central-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20220406.1	4.14.275	2022 年 5 月 2 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-south-1 • ap-east-1 • ap-south-1 • ap-southeast-3 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-south-1 • ca-central-1

Amazon EMR 5.x のデフォルト AMI

次の表は、Amazon EMR 5.x リリース 5.36 以降の最新のパッチバージョンに関する Amazon Linux 情報を示しています。

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20230504.1	4.14.313	2023 年 5 月 16 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • ca-central-1 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-south-1 • ap-east-1 • ap-south-1 • ap-southeast-3 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1

OsRelea Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
			• me-south-1 • me-central-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20230418.0	4.14.311	2023 年 5 月 3 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • ca-central-1 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-south-1 • ap-east-1 • ap-south-1 • ap-southeast-3 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-south-1 • me-central-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20230404.1	4.14.311	2023 年 4 月 18 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • ca-central-1 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-south-1 • ap-east-1 • ap-south-1 • ap-southeast-3 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-south-1
2.0.20230404.0	4.14.311	2023 年 4 月 10 日	• us-east-1 • eu-west-3

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20230320.0	4.14.309	2023 年 3 月 30 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • ca-central-1 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-south-1 • ap-east-1 • ap-south-1 • ap-southeast-3 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-south-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20230307.0	4.14.305	2023 年 3 月 15 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • ca-central-1 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-south-1 • ap-east-1 • ap-south-1 • ap-southeast-3 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-south-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20230207.0	4.14.304	2023 年 3 月 3 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-south-1 • ap-east-1 • ap-south-1 • ap-southeast-3 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-south-1 • ca-central-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20221210.1	4.14.301	2023 年 1 月 12 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-south-1 • ap-east-1 • ap-south-1 • ap-southeast-3 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-south-1 • ca-central-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20221103.3	4.14.296	2022 年 12 月 5 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-south-1 • ap-east-1 • ap-south-1 • ap-southeast-3 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-south-1 • ca-central-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20220004.0	4.14.294	2022 年 11 月 2 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-south-1 • ap-east-1 • ap-south-1 • ap-southeast-3 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-south-1 • ca-central-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20220912.1	4.14.291	2022 年 10 月 7 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-south-1 • ap-east-1 • ap-south-1 • ap-southeast-3 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-south-1 • ca-central-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20220719.0	4.14.287	2022 年 8 月 10 日	• us-west-1 • eu-west-3 • eu-north-1 • eu-central-1 • ap-south-1 • me-south-1

OsRelease Label (AL バージョン)	AL カーネルバージョン	利用可能日	AWS リージョン
2.0.20220426.0	4.14.281	2022 年 6 月 14 日	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-north-1 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-south-1 • ap-east-1 • ap-south-1 • ap-southeast-3 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-south-1 • ca-central-1

ソフトウェア更新に関する考慮事項

以下のデフォルトのソフトウェア更新動作に注意してください。

Amazon EMR 7.x - Amazon Linux 2023

Amazon EMR リリース 7.0 以降は、Amazon Linux 2023 (AL2023) で実行されます。AL2023 のデフォルト動作では、AMI は Amazon Linux ソフトウェアリポジトリの特定のバージョンにロックされます。そのため、クラスターを起動するたびにセキュリティ更新が適用されるわけではありません。Amazon EMR 7.x リリースのデフォルトの動作では、クラスターを作成した時にのみデフォルトの Amazon EMR AMI の最新 AL2023 リリースを自動的に適用します。最新のセキュリティ更新を受け取るには、定期的にクラスターを再作成することをお勧めします。

Amazon EMR 5.x および 6.x - Amazon Linux および Amazon Linux 2

7.0 以前の Amazon EMR リリースでは、デフォルトの Amazon Linux (AL) または Amazon Linux 2 (AL2) をベースとするクラスターで Amazon EC2 インスタンスを初めて起動すると、AL と Amazon EMR 用に有効化されているパッケージリポジトリ内で、そのリリースバージョンに適用されるソフトウェア更新がないかチェックします。他の AL および AL2 インスタンスと同じように、これらのリポジトリから緊急および重要なセキュリティ更新プログラムが自動的にインストールされます。

また、ネットワーク設定では、Amazon S3 内の Amazon Linux リポジトリへの HTTP および HTTPS の出力が許可されている必要があります。そうでない場合、セキュリティ更新は失敗します。詳細については、「Amazon EC2 ユーザーガイド」の「Amazon Linux - パッケージリポジトリAmazon EC2」を参照してください。デフォルトでは、NVIDIA や CUDA など、再起動が必要な他のソフトウェアパッケージおよびカーネルの更新は、初回起動時に自動ダウンロードから除外されます。

Amazon EMR 5.35.0 以前、および 6.5.0 以前 - Amazon Linux AMI は Amazon EMR リリースバージョンにロックされています

Amazon EMR 5.35.0 以前、および 6.5.0 以前では、デフォルトの AMI は up-to-date Amazon EMR リリース時に利用可能な最も多くの Amazon Linux AMI に基づいています。AMI とビッグデータアプリケーション、およびリリースバージョンに含まれている Amazon EMR の機能の互換性はテストされています。

Amazon EMR 5.35.0 以前および 6.5.0 以前の Amazon EMR リリースバージョンは、互換性を保つためにそれぞれ割り当てられた Amazon Linux AMI バージョンに「ロック」されています。そのため、互換性を確保するためにより低いバージョンが必要で移行できない場合を除いて、最新の

Amazon EMR リリースバージョンを使用することをお勧めします。互換性を確保するためにより低い Amazon EMR リリースバージョンを使用する必要がある場合は、そのシリーズの最新のリリースを使用することをお勧めします。たとえば、5.12 シリーズを使用する必要がある場合は、5.12.0 や 5.12.1 ではなく 5.12.2 を使用します。シリーズで新しいリリースが使用可能になった場合は、新しいリリースへのアプリケーションの移行を検討してください。

Amazon EMR 5.36.0 以降と 6.6.0 以降で導入された自動更新動作の詳細については、「Amazon EMR リリース用 Amazon Linux の自動更新」を参照してください。

デフォルトのブート動作はカーネルの更新を除外します

Amazon EMR 用のデフォルトの Amazon Linux AMI に基づくクラスターの Amazon EC2 インスタンスを初めて起動すると、Amazon Linux と Amazon EMR で使用できる、その AMI バージョンに適用されるソフトウェア更新用のパッケージリポジトリがチェックされます。他の Amazon EC2 インスタンスと同じように、これらのリポジトリから緊急および重要なセキュリティ更新が自動的にインストールされます。

ただし、Amazon Linux AMI の古いバージョンを使用している場合は、最新のセキュリティ更新が自動的にインストールされない場合があります。これは、お使いの EMR クラスターが参照するリポジトリが Amazon Linux AMI の各バージョンに固定されているためです。

また、ネットワーク設定では、Amazon S3 内の Amazon Linux リポジトリへの HTTP および HTTPS の出力が許可されている必要があります。そうでない場合、セキュリティ更新は失敗します。詳細については、「Amazon EC2 ユーザーガイド」の「Amazon Linux - パッケージリポジトリAmazon EC2」を参照してください。デフォルトでは、NVIDIA や CUDA など、再起動が必要な他のソフトウェアパッケージおよびカーネルの更新は、初回起動時に自動ダウンロードから除外されます。

> **Important**
>
> AL2023 を実行する EMR クラスターはデフォルトの Amazon Linux 動作を使用しており、お使いの Amazon マシンイメージ (AMI) は特定のバージョンの Amazon Linux リポジトリにロックされます。デフォルトでは、クラスターは起動時に自動的にソフトウェアセキュリティ更新を受信しません。クラスターには、クラスターの作成時に選択した AL2023 AMI のバージョンで利用可能なアップデートのみが含まれます。詳細については、Amazon Linux 2023 ユーザーガイドの「Amazon Linux 2023 の更新」を参照してください。

> **Important**
>
> Amazon Linux または Amazon Linux 2 Amazon マシンイメージ (AMI) を実行する EMR クラスターは、デフォルトの Amazon Linux 動作を使用します。再起動が必要な重要かつクリティカルなカーネル更新が自動的にダウンロードされてインストールされることはありません。これは、デフォルトの Amazon Linux AMI を実行している他の Amazon EC2 インスタンスと同じ動作です。Amazon EMR リリースが利用可能になった後に、再起動が必要な新しい Amazon Linux ソフトウェアアップデート (カーネル、NVIDIA、CUDA のアップデートなど) が使用可能になった場合、デフォルトの AMI を実行する EMR クラスターインスタンスで、それらの更新が自動的にダウンロードされてインストールされることはありません。カーネルの更新を取得するには、Amazon EMR AMI をカスタマイズして、最新の Amazon Linux AMI を使用できるようにします。

更新の有無にかかわらず、クラスターは起動します

最初のクラスター起動時にパッケージリポジトリに到達できないためにソフトウェアの更新をインストールできない場合でも、クラスターインスタンスの起動は完了することに注意してください。例えば、S3 が一時的に利用できないためにリポジトリに到達できない場合や、アクセスをブロックするように VPC またはファイアウォールルールが設定されている可能性があります。

`sudo yum update` を実行しないでください

SSH を使用してクラスターインスタンスに接続すると、画面出力の最初の数行にインスタンスが使用する Amazon Linux AMI のリリースノートのリンク、最新の Amazon Linux AMI バージョンに関する通知、更新に使用可能なリポジトリのパッケージ数に関する通知、および `sudo yum update` の実行指示が示されます。

> **Important**
>
> SSH またはブートストラップアクションを使用して接続しているときは、クラスターインスタンスで `sudo yum update` を実行しないことを強くお勧めします。実行するとすべてのパッケージが無差別にインストールされ、互換性がなくなる可能性があります。

ソフトウェア更新に関するベストプラクティス

ソフトウェア更新の管理に関するベストプラクティス

- より低い Amazon EMR リリースバージョンを使用している場合は、ソフトウェアパッケージを更新する前に最新のリリースへの移行を検討してテストします。
- より高いリリースバージョンに移行するかソフトウェアパッケージをアップグレードする場合は、最初に本番稼働環境以外で実装をテストします。そのときには、Amazon EMR コンソールを使用してクラスターのクローンを作成するオプションが役に立ちます。
- アプリケーションと使用する Amazon Linux AMI バージョンのソフトウェア更新を個別に評価します。本番稼働環境では、セキュリティ体制、アプリケーション機能、またはパフォーマンスの向上のために絶対に必要であると判断したパッケージだけをテストしてインストールします。
- Amazon Linux Security Center で更新を確認します。
- SSH を使用して各クラスターインスタンスに接続することでパッケージのインストールを回避します。その代わりにブートストラップアクションを使用し、必要に応じてすべてのクラスターインスタンスでパッケージのインストールと更新を行います。そのためには、クラスターを終了して再起動する必要があります。詳細については、「追加のソフトウェアをインストールするためのブートストラップアクションの作成」を参照してください。

カスタム AMI の使用

Amazon EMR 5.7.0 以降を使用する場合は、Amazon EMR 用のデフォルトの Amazon Linux AMI ではなく、カスタム Amazon Linux AMI を指定できます。カスタム AMI は、以下を実行する場合に役立ちます。

- ブートストラップアクションを使用する代わりにアプリケーションを事前インストールして他のカスタマイズを行う。これにより、クラスターの起動時間を短縮し、スタートアップのワークフローを合理化できます。詳細と例については、「事前設定したインスタンスからのカスタム Amazon Linux AMI の作成」を参照してください。
- ブートストラップアクションが許可する以上の高度なクラスターおよびノード設定を実装する。
- 5.24.0 より低い Amazon EMR バージョンを使用している場合、クラスターの EC2 インスタンスの EBS ルートデバイスボリューム (ブートボリューム) を暗号化します。デフォルトの AMI と同様に、カスタム AMI の最小ルートボリュームサイズは、Amazon EMR リリース 6.9 以前で 10 GiB、Amazon EMR リリース 6.10 以降で 15 GiB です。詳細については、「暗号化された Amazon EBS ルートデバイスボリュームを使用したカスタム AMI の作成」を参照してください。

> **Note**
>
> Amazon EMR バージョン 5.24.0 以降では、 をキープロバイダー AWS KMS として指定すると、セキュリティ設定オプションを使用して EBS ルートデバイスとストレージボリュームを暗号化できます。詳細については、「ローカルディスク暗号化」を参照してください。

カスタム AMI は、クラスターを作成するのと同じ AWS リージョンに存在する必要があります。EC2 インスタンスアーキテクチャにも一致する必要があります。例えば、m5.xlarge インスタンスには x86_64 アーキテクチャがあります。したがって、カスタム AMI を使用して m5.xlarge をプロビジョニングするには、カスタム AMI にも x86_64 アーキテクチャが必要です。同様に、arm64 アーキテクチャを持つ m6g.xlarge インスタンスをプロビジョニングするには、カスタム AMI に arm64 アーキテクチャが必要です。インスタンスタイプの Linux AMI を識別する方法の詳細については、「Amazon EC2 ユーザーガイド」の「Linux AMI の検索」を参照してください。 Amazon EC2

> **Important**
>
> Amazon Linux または Amazon Linux 2 Amazon マシンイメージ (AMI) を実行する EMR クラスターは、デフォルトの Amazon Linux 動作を使用します。再起動が必要な重要かつクリティカルなカーネル更新が自動的にダウンロードされてインストールされることはありません。これは、デフォルトの Amazon Linux AMI を実行している他の Amazon EC2 インスタンスと同じ動作です。Amazon EMR リリースが利用可能になった後に、再起動が必要な新しい Amazon Linux ソフトウェアアップデート (カーネル、NVIDIA、CUDA のアップデートなど) が使用可能になった場合、デフォルトの AMI を実行する EMR クラスターインスタンスで、それらの更新が自動的にダウンロードされてインストールされることはありません。カーネルの更新を取得するには、Amazon EMR AMI をカスタマイズして、最新の Amazon Linux AMI を使用できるようにします。

事前設定したインスタンスからのカスタム Amazon Linux AMI の作成

Amazon EMR のカスタム Amazon Linux AMI を作成するためにソフトウェアを事前インストールして他の設定を行う基本的な手順は、次のとおりです。

- ベース Amazon Linux AMI からインスタンスを起動します。

- インスタンスに接続してソフトウェアをインストールし、他のカスタマイズを行います。
- 設定したインスタンスの新しいイメージ (AMI スナップショット) を作成します。

カスタマイズ済みのインスタンスに基づいてイメージを作成したら、そのイメージを暗号化されたターゲットにコピーできます (「暗号化された Amazon EBS ルートデバイスボリュームを使用したカスタム AMI の作成」を参照)。

チュートリアル: カスタムソフトウェアがインストールされたインスタンスから AMI を作成する

最新の Amazon Linux AMI に基づいて EC2 インスタンスを起動するには

1. AWS CLI を使用して、既存の AMI からインスタンスを作成する次のコマンドを実行します。*MyKeyName* をインスタンスへの接続に使用するキーペアに置き換え、 を適切な Amazon Linux AMI の ID *MyAmiId*に置き換えます。最新の AMI ID に関しては、Amazon Linux AMI を参照してください。

   > **Note**
   >
   > 読みやすくするために、Linux 行連続文字 (\) が含まれています。Linux コマンドでは、これらは削除することも、使用することもできます。Windows の場合、削除するか、キャレット (^) に置き換えてください。

   ```
   aws ec2 run-instances --image-id MyAmiID \
   --count 1 --instance-type m5.xlarge \
   --key-name MyKeyName --region us-west-2
   ```

   InstanceId出力値は、次のステップで *MyInstanceId* として使用します。
2. 次のコマンドを実行します。

   ```
   aws ec2 describe-instances --instance-ids MyInstanceId
   ```

   PublicDnsName 出力値は、次のステップでインスタンスに接続するために使用します。

インスタンスに接続してソフトウェアをインストールするには

1. SSH 接続を使用して Linux インスタンスでシェルコマンドを実行します。詳細については、Amazon EC2 ユーザーガイド」の「SSH を使用した Linux インスタンスへの接続」を参照してください。
2. 必要なカスタマイズを行います。例:

```
sudo yum install MySoftwarePackage
sudo pip install MySoftwarePackage
```

カスタマイズしたイメージからスナップショットを作成するには

- インスタンスをカスタマイズしたら、create-image コマンドを使用してインスタンスから AMI を作成します。

```
aws ec2 create-image --no-dry-run --instance-id MyInstanceId --name MyEmrCustomAmi
```

imageID 出力値は、クラスターの起動時または暗号化されたスナップショットの作成時に使用します。詳細については、「EMR クラスターで単一のカスタム AMI を使用する」および「暗号化された Amazon EBS ルートデバイスボリュームを使用したカスタム AMI の作成」を参照してください。

Amazon EMR クラスターでカスタム AMI を使用する方法

カスタム AMI を使用して Amazon EMR クラスターをプロビジョニングするには、次の 2 つの方法があります。

- クラスターのすべての EC2 インスタンスに対して 1 つのカスタム AMI を使用する。
- クラスターで使用されるさまざまな EC2 インスタンスタイプに異なるカスタム AMI を使用する。

EMR クラスターのプロビジョニングには、2 つのオプションのうち 1 つしか使用できず、クラスターの起動後に変更することはできません。

Amazon EMR クラスターで単一のカスタム AMI と複数のカスタム AMI を使用する場合の考慮事項

考慮事項	単一のカスタム AMI	複数のカスタム AMI
同じクラスターの複数のカスタム AMI で x86 プロセッサと Graviton2 プロセッサの両方を使用する	✗ サポート対象外	✓ サポート対象
AMI カスタマイズがインスタンスタイプによって異なる	✗ サポート対象外	✓ サポート対象
実行中のクラスターに新しいタスクインスタンスグループ/フリートを追加するときに、カスタム AMI を変更する 注意: 既存のインスタンスグループ/フリートのカスタム AMI を変更することはできません。	✗ サポート対象外	✓ サポート対象
AWS コンソールを使用してクラスターを起動する	✓ サポート対象	✗ サポート対象外
AWS CloudFormation を使用してクラスターを起動する	✓ サポート対象	✓ サポート対象

EMR クラスターで単一のカスタム AMI を使用する

クラスターの作成時にカスタム AMI ID を指定するには、次のいずれかを使用します。

- AWS Management Console
- AWS CLI
- Amazon EMR SDK
- Amazon EMR API RunJobFlow
- AWS CloudFormation（ クラスター 、クラスター InstanceGroupConfig、リソース 、または リソース InstanceTypeConfig- の `CustomAmiID`プロパティを参照) InstanceGroupConfig InstanceFleetConfigInstanceTypeConfig

Amazon EMR console

コンソールを使用して単一のカスタム AMI を指定するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で、[クラスター] を選択し、[クラスターの作成] を選択します
3. [名前とアプリケーション] で [オペレーティングシステムオプション] を探します。[カスタム AMI] を選択し、[カスタム AMI] フィールドに AMI ID を入力します。
4. クラスターに適用するその他のオプションを選択します。
5. クラスターを起動するには、[クラスターの作成] を選択します。

AWS CLI

を使用して単一のカスタム AMI を指定するには AWS CLI

- `--custom-ami-id` コマンドの実行時に `aws emr create-cluster` パラメータを使用して AMI ID を指定します。

  次の例では、ブートボリュームが 20 GiB の単一のカスタム AMI を使用するクラスターを指定しています。詳細については、「Amazon EBS ルートデバイスボリュームのカスタマイズ」を参照してください。

  > **Note**
  >
  > 読みやすくするために、Linux 行連続文字 (\) が含まれています。Linux コマンドでは、これらは削除することも、使用することもできます。Windows の場合、削除するか、キャレット (^) に置き換えてください。

```
aws emr create-cluster --name "Cluster with My Custom AMI" \
--custom-ami-id MyAmiID --ebs-root-volume-size 20 \
--release-label emr-5.7.0 --use-default-roles \
--instance-count 2 --instance-type m5.xlarge
```

Amazon EMR クラスターで複数のカスタム AMI を使用する

複数のカスタム AMI を使用してクラスターを作成するには、次のいずれかを使用します。

- AWS CLI バージョン 1.20.21 以降
- AWS SDK
- 「Amazon EMR API リファレンスRunJobFlow」の「Amazon EMR」
- AWS CloudFormation（ クラスター 、クラスター InstanceGroupConfig、リソース 、または リソース InstanceTypeConfig- の CustomAmiIDプロパティを参照) InstanceGroupConfig InstanceFleetConfigInstanceTypeConfig

AWS マネジメントコンソールは現在、複数のカスタム AMIs を使用したクラスターの作成をサポートしていません。

Example - AWS CLI を使用して、複数のカスタム AMIs を使用してインスタンスグループクラスターを作成する

AWS CLI バージョン 1.20.21 以降を使用すると、クラスター全体に 1 つのカスタム AMI を割り当てることも、クラスター内のすべてのインスタンスノードに複数のカスタム AMIs を割り当てることもできます。

次の例は、各ノードタイプ (プライマリ、コア、タスク) にわたって 2 つのインスタンスタイプ (m5.xlarge) を使用して作成された均一インスタンスグループクラスターを示します。各ノードには複数のカスタム AMI があります。この例は、複数のカスタム AMI 設定のいくつかの機能を示しています。

- クラスターレベルで割り当てられているカスタム AMI はありません。これは、クラスターの起動が失敗する原因となる、複数のカスタム AMI と単一のカスタム AMI の間の競合を回避するためです。
- クラスターは、プライマリ、コア、および個々のタスクノードにまたがる複数のカスタム AMI を持つことができます。これにより、プリインストールされたアプリケーション、高度なクラスター設定、暗号化された Amazon EBS ルートデバイスボリュームなど、個々の AMI のカスタマイズが可能になります。
- インスタンスグループコアノードは、1 つのインスタンスタイプと、対応するカスタム AMI のみを持つことができます。同様に、プライマリノードは、1 つのインスタンスタイプと、対応するカスタム AMI のみを持つことができます。
- クラスターは、複数のタスクノードを持つことができます。

```
aws emr create-cluster --instance-groups
InstanceGroupType=PRIMARY,InstanceType=m5.xlarge,InstanceCount=1,CustomAmiId=ami-123456
InstanceGroupType=CORE,InstanceType=m5.xlarge,InstanceCount=1,CustomAmiId=ami-234567
InstanceGroupType=TASK,InstanceType=m6g.xlarge,InstanceCount=1,CustomAmiId=ami-345678
InstanceGroupType=TASK,InstanceType=m5.xlarge,InstanceCount=1,CustomAmiId=ami-456789
```

Example - AWS CLI バージョン 1.20.21 以降を使用して、複数のインスタンスタイプと複数のカスタム AMIs

AWS CLI バージョン 1.20.21 以降を使用すると、実行中のクラスターに追加するインスタンスグループに複数のカスタム AMIs を追加できます。次の例に示すように、CustomAmiId 引数を add-instance-groups コマンドとともに使用できます。複数のノードで同じ複数のカスタム AMI ID (ami-123456) が使用されていることに注意してください。

```
aws emr create-cluster --instance-groups
InstanceGroupType=PRIMARY,InstanceType=m5.xlarge,InstanceCount=1,CustomAmiId=ami-123456
InstanceGroupType=CORE,InstanceType=m5.xlarge,InstanceCount=1,CustomAmiId=ami-123456
InstanceGroupType=TASK,InstanceType=m5.xlarge,InstanceCount=1,CustomAmiId=ami-234567

{
    "ClusterId": "j-123456",
    ...
}

aws emr add-instance-groups --cluster-id j-123456 --instance-groups
 InstanceGroupType=Task,InstanceType=m6g.xlarge,InstanceCount=1,CustomAmiId=ami-345678
```

Example - AWS CLI バージョン 1.20.21 以降を使用して、インスタンスフリートクラスター、複数のカスタム AMIs、複数のインスタンスタイプ、オンデマンドプライマリ、オンデマンドコア、複数のコアノードとタスクノードを作成します。

```
aws emr create-cluster --instance-fleets
InstanceFleetType=PRIMARY,TargetOnDemandCapacity=1,InstanceTypeConfigs=['{InstanceType=m5.xlarg
 CustomAmiId=ami-123456}']
InstanceFleetType=CORE,TargetOnDemandCapacity=1,InstanceTypeConfigs=['{InstanceType=m5.xlarge,C
{InstanceType=m6g.xlarge, CustomAmiId=ami-345678}']
InstanceFleetType=TASK,TargetSpotCapacity=1,InstanceTypeConfigs=['{InstanceType=m5.xlarge,Custo
{InstanceType=m6g.xlarge, CustomAmiId=ami-567890}']
```

Example - AWS CLI バージョン 1.20.21 以降を使用して、複数のインスタンスタイプと複数のカスタム AMIs

```
aws emr create-cluster --instance-fleets
InstanceFleetType=PRIMARY,TargetOnDemandCapacity=1,InstanceTypeConfigs=['{InstanceType=m5.xlarg
 CustomAmiId=ami-123456}']
InstanceFleetType=CORE,TargetOnDemandCapacity=1,InstanceTypeConfigs=['{InstanceType=m5.xlarge,C
{InstanceType=m6g.xlarge, CustomAmiId=ami-345678}']

{
    "ClusterId": "j-123456",
    ...
}

aws emr add-instance-fleet --cluster-id j-123456 --instance-fleet
InstanceFleetType=TASK,TargetSpotCapacity=1,InstanceTypeConfigs=['{InstanceType=m5.xlarge,Custo
{InstanceType=m6g.xlarge, CustomAmiId=ami-345678}']
```

AMI パッケージリポジトリの更新の管理

デフォルトでは、Amazon Linux AMI は最初の起動時にパッケージリポジトリに接続し、他のサービスが起動する前にセキュリティ更新をインストールします。Amazon EMR のカスタム AMI の指定時に、必要に応じて、これらの更新を無効にすることを選択できます。この機能を無効にするオプションは、カスタム AMI を使用する場合にのみ使用できます。デフォルトでは、Amazon Linux カーネル更新および再起動を必要とするその他のソフトウェアパッケージは更新されません。ネットワーク設定では、Simple Storage Service (Amazon S3) の Amazon Linux リポジトリへの HTTP および HTTPS の出力が許可されている必要があります。そうしないと、セキュリティ更新は成功しません。

> **Warning**
>
> カスタム AMI を指定するときに、すべてのインストールされているパッケージを再起動時に更新することを選択するようお勧めします。パッケージを更新しないことを選択すると、他のセキュリティリスクが生じます。

では AWS Management Console、カスタム AMI を選択すると、更新を無効にするオプションを選択できます。

では AWS CLI、 create-cluster コマンドを使用する`--custom-ami-id`ときに を `--repo-upgrade-on-boot NONE` とともに指定できます。

Amazon EMR API では、 RepoUpgradeOnBootパラメータ`NONE`に を指定できます。

暗号化された Amazon EBS ルートデバイスボリュームを使用したカスタム AMI の作成

Amazon EMR で Amazon Linux AMI の Amazon EBS ルートデバイスボリュームを暗号化するには、暗号化されていない AMI から暗号化されたターゲットにスナップショットイメージをコピーします。暗号化された EBS ボリュームの作成については、「Amazon EC2 ユーザーガイド」の「Amazon EBS 暗号化Amazon EC2」を参照してください。スナップショットのソース AMI としてベース Amazon Linux AMI を使用できます。または、カスタマイズしたベース Amazon Linux AMI から派生した AMI からスナップショットをコピーできます。

> **Note**
>
> Amazon EMR バージョン 5.24.0 以降では、 をキープロバイダー AWS KMS として指定すると、セキュリティ設定オプションを使用して EBS ルートデバイスとストレージボリュームを暗号化できます。詳細については、「ローカルディスク暗号化」を参照してください。

外部キープロバイダーまたは AWS KMS キーを使用して EBS ルートボリュームを暗号化できます。Amazon EMR で使用するサービスロール (通常はデフォルトの `EMR_DefaultRole`) に対しては、Amazon EMR で AMI を使用してクラスターを作成するために、少なくともボリュームの暗号化と復号を許可する必要があります。をキープロバイダー AWS KMS として使用する場合は、次のアクションを許可する必要があります。

- `kms:encrypt`
- `kms:decrypt`
- `kms:ReEncrypt*`
- `kms:CreateGrant`
- `kms:GenerateDataKeyWithoutPlaintext"`
- `kms:DescribeKey"`

これを行う最も簡単な方法としては、次のチュートリアルで説明するように、キーユーザーとしてロールを追加します。以下のポリシーステートメントの例は、ロールのポリシーをカスタマイズする必要がある場合に使用してください。

```
{
"Version": "2012-10-17",
"Statement": [
   {
   "Sid": "EmrDiskEncryptionPolicy",
   "Effect": "Allow",
   "Action": [
      "kms:Encrypt",
      "kms:Decrypt",
      "kms:ReEncrypt*",
      "kms:CreateGrant",
      "kms:GenerateDataKeyWithoutPlaintext",
      "kms:DescribeKey"
      ],
   "Resource": [
      "*"
      ]
   }
]
}
```

チュートリアル: KMS キーを使用した、暗号化されたルートデバイスボリュームを持つカスタム AMI の作成

この例の最初のステップでは、KMS キーの ARN を検索するか、新規作成します。キーの作成の詳細については、「AWS Key Management Service デベロッパーガイド」の「キーの作成」を参照してください。次の手順では、デフォルトのサービスロール `EMR_DefaultRole` をキーユーザーとしてキーポリシーに追加する方法を示します。作成または編集するキーの ARN 値を書き留めます。AMI を作成するときに、より高い ARN を使用します。

コンソールを使用して Amazon EC2 のサービスロールを暗号化キーユーザーのリストに追加するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/kms で AWS Key Management Service （ AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセレクターを使用します。

3. 使用する KMS キーのエイリアスを選択します。
4. [Key Users] のキーの詳細ページで、[Add] を選択します。
5. [アタッチ] ダイアログボックスで Amazon EMR サービスロールを選択します。デフォルトロールの名前は `EMR_DefaultRole` です。
6. 添付を選択します。

を使用して暗号化された AMI を作成するには AWS CLI

- から `aws ec2 copy-image` コマンドを使用して、暗号化された EBS ルートデバイスボリュームと変更したキーを持つ AMI AWS CLI を作成します。指定された `--kms-key-id` 値を、ユーザーが作成またはより低く変更したキーの完全な ARN に置き換えます。

> **Note**
>
> 読みやすくするために、Linux 行連続文字 (\) が含まれています。Linux コマンドでは、これらは削除することも、使用することもできます。Windows の場合、削除するか、キャレット (^) に置き換えてください。

```
aws ec2 copy-image --source-image-id MyAmiId \
--source-region us-west-2 --name MyEncryptedEMRAmi \
--encrypted --kms-key-id arn:aws:kms:us-west-2:12345678910:key/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

コマンドの出力として、作成した AMI の ID が提供されます。この ID をクラスターの作成時に指定できます。詳細については、「EMR クラスターで単一のカスタム AMI を使用する」を参照してください。また、ソフトウェアをインストールし、他の設定を行うことにより、この AMI をカスタマイズすることもできます。詳細については、「事前設定したインスタンスからのカスタム Amazon Linux AMI の作成」を参照してください。

ベストプラクティスと考慮事項

Amazon EMR のカスタム AMI を作成する場合は、以下の点を考慮してください。

- Amazon EMR 7.x シリーズは Amazon Linux 2023 に基づいています。これらの Amazon EMR バージョンでは、カスタム AMIs に Amazon Linux 2023 に基づくイメージを使用する必要があります。基本カスタム AMI を見つけるには、「Linux AMI の検索」を参照してください。
- 7.x より前のバージョンの Amazon EMR では、Amazon Linux 2023 AMIs はサポートされていません。
- Amazon EMR 5.30.0 以降、および Amazon EMR 6.x シリーズは、Amazon Linux 2 に基づいています。これらの Amazon EMR バージョンでカスタム AMI を作成する場合は、Amazon Linux 2 に基づくイメージを使用する必要があります。基本カスタム AMI を見つけるには、「Linux AMI の検索」を参照してください。
- 5.30.0 および 6.x より低い Amazon EMR バージョンでは、Amazon Linux 2 AMI はサポートされません。
- 64 ビットの Amazon Linux AMI を使用する必要があります。32 ビット AMI はサポートされていません。
- 複数の Amazon EBS ボリュームを持つ Amazon Linux AMI はサポートされていません。
- 最新の EBS-backed Amazon Linux AMI に基づいてカスタマイズします。Amazon Linux AMI および対応する AMI ID のリストについては、Amazon Linux AMI を参照してください。
- カスタム AMI を作成する際に、既存の Amazon EMR インスタンスのスナップショットをコピーしないでください。これを行うと、エラーになります。
- Amazon EMR と互換性がある HVM 仮想化タイプおよびインスタンスのみがサポートされます。必ず Amazon EMR と互換性がある HVM イメージおよびインスタンスタイプを選択して、AMI のカスタマイズプロセスを進めてください。互換性のあるインスタンスおよび仮想化タイプについては、「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。
- サービスロールには AMI での起動許可が必要であるため、AMI はパブリックであるか、ユーザーが AMI の所有者であるか、AMI を所有者と共有している必要があります。
- AMI で作成するユーザーをアプリケーションと同じ名前 (`hadoop`、`hdfs`、`yarn`、`spark` など) にすると、エラーが発生します。
- `/tmp`、`/var`、および `/emr` のコンテンツ (AMI に存在する場合) は起動時にそれぞれ `/mnt/tmp`、`/mnt/var`、および `/mnt/emr` に移動されます。ファイルは保持されますが、大量のデータがあると、スタートアップに予想以上の時間がかかる場合があります。
- 作成日が 2018-08-11 の Amazon Linux AMI に基づくカスタム Amazon Linux AMI を使用すると、Oozie サーバーの起動に失敗します。Oozie を使用する場合は、作成日が異なる Amazon Linux AMI ID に基づいてカスタム AMI を作成します。次の AWS CLI コマンドを使用して、2018.03 バージョンのすべての HVM Amazon Linux AMIs のイメージ IDs のリストをリリース

日とともに返すことができます。これにより、ベースとして適切な Amazon Linux AMI を選択できます。を us-west-2 などのリージョン識別子 MyRegion に置き換えます。

```
aws ec2 --region MyRegion describe-images --owner amazon --query 'Images[?
Name!=`null`]|[?starts_with(Name, `amzn-ami-hvm-2018.03`) == `true`].
[CreationDate,ImageId,Name]' --output text | sort -rk1
```

- 標準以外のドメイン名と AmazonProvidedDNS を持つ VPC を使用する場合は、オペレーティングシステムの DNS 設定で `rotate`オプションを使用しないでください。

詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EBS-backed Linux AMI の作成」を参照してください。 Amazon EC2

EMR クラスター作成時の Amazon Linux リリースの変更

Amazon EMR 6.6.0 以降を使用してクラスターを起動すると、デフォルトの Amazon EMR AMI 用に検証された最新の Amazon Linux 2 リリースが自動的に使用されます。Amazon EMR コンソールまたは AWS CLIを使用して、クラスターに別の Amazon Linux リリースを指定できます。

Amazon EMR console

コンソールでクラスターを作成するときに Amazon Linux リリースを変更するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で、[クラスター] を選択し、[クラスターの作成] を選択します
3. [EMR バージョン] には、[emr-6.6.0] 以降を選択してください。
4. [オペレーティングシステムオプション] で [Amazon Linux バージョン] を選択し、[最新の Amazon Linux 更新を自動的に適用する] チェックボックスを選択します。
5. クラスターに適用するその他のオプションを選択します。
6. クラスターを起動するには、[クラスターの作成] を選択します。

AWS CLI

AWS CLIを使用してクラスターを作成するときに Amazon Linux リリースを変更するには

- `aws emr` create-cluster コマンドを実行するときに、`--os-release-label` パラメータを使用して [Amazon Linux リリース] を指定します。

```
aws emr create-cluster --name "Cluster with Different Amazon Linux Release" \
--os-release-label 2.0.20210312.1 \
--release-label emr-6.6.0 --use-default-roles \
--instance-count 2 --instance-type m5.xlarge
```

Amazon EBS ルートデバイスボリュームのカスタマイズ

EBS ルートボリュームのデフォルト

Amazon EMR 4.x 以降では、クラスターを作成するときにルートボリュームサイズを指定できます。Amazon EMR リリース 6.15.0 以降では、ルートボリュームの IOPS とスループットも指定できます。これらの属性は Amazon EBS ルートデバイスボリュームにのみ適用され、クラスター内のすべてのインスタンスに適用されます。属性はストレージボリュームには適用されず、ストレージボリュームのサイズは、クラスター作成時にインスタンスタイプごとに指定します。

- Amazon EMR 6.10.0 以降では、デフォルトのルートボリュームサイズは 15 GiB です。以前のリリースでは、デフォルトのルートボリュームサイズは 10 GiB でした。ルートボリュームサイズは最大 100 GiB まで調整できます。
- デフォルトのルートボリューム IOPS は 3000 です。最大 16000 まで調整できます。
- デフォルトのルートボリュームスループットは 125 MiB/秒です。これは最大 1000 MiB/秒まで調整できます。

> **Note**
>
> ルートボリュームサイズと IOPS の比率は、1 ボリューム対 500 IOPS (1:500) を超えることはできません。また、ルートボリューム IOPS とスループットの比率は 1 IOPS 対 0.25 スループット (1:0.25) を超えることはできません。

Amazon EBS の詳細については、「Amazon EC2 ルートデバイスボリューム」を参照してください。

デフォルト AMI のルートデバイスボリュームタイプ

デフォルトの AMI を使用する場合、ルートデバイスボリュームタイプは使用する Amazon EMR リリースによって決まります。

- Amazon EMR リリース 6.15.0 以降では、Amazon EMR はルートデバイスボリュームタイプとして汎用 SSD (gp3) をアタッチします。
- Amazon EMR リリース 6.15.0 以前では、Amazon EMR はルートデバイスボリュームタイプとして汎用 SSD (gp2) をアタッチします。

カスタム AMI のルートデバイスボリュームタイプ

カスタム AMI には、複数の異なるルートデバイスボリュームタイプがある場合があります。Amazon EMR は常にカスタム AMI ボリュームタイプを使用します。

- Amazon EMR リリース 6.15.0 以降では、カスタム AMI のルートボリュームサイズ、IOPS、スループットを設定できます。ただし、これらの属性がそのカスタム AMI ボリュームタイプに適用できる場合に限ります。
- Amazon EMR リリース 6.15.0 以前では、カスタム AMI のルートボリュームサイズのみを設定できます。

クラスターの作成時にルートボリュームサイズ、IOPS、またはスループットを設定しない場合、Amazon EMR は該当する場合はカスタム AMI の値を使用します。クラスターの作成時にこれらの値を指定した場合、Amazon EMR は、指定された値がカスタム AMI ルートボリュームと互換性があり、サポートされている限り、指定された値を使用します。詳細については、「カスタム AMI の使用」を参照してください。

ルートデバイスボリュームのサイズの料金

EBS ルートデバイスボリュームのコストは、クラスターが実行されるリージョンでのボリュームタイプの月次 EBS 料金に基づき、時間単位で計算されます。ストレージボリュームについても同様です。料金は GB 単位ですが、ルートボリュームサイズは GiB で指定するため、これに基づいて見積もることができます (1 GB は 0.931323 GiB です)。

汎用 SSD gp2 と gp3 では請求方法が異なります。クラスターの EBS ルートデバイスボリュームに関連する料金を見積もるには、次の計算式を使用します。

汎用 SSD gp2

gp2 のコストには EBS ボリュームサイズ (GB 単位) のみが含まれます。

```
($EBS size in GB/month) * 0.931323 / 30 / 24 * EMR_EBSRootVolumesizeInGiB *
 InstanceCount
```

たとえば、プライマリノードとコアノードがあるクラスターでルートデバイスボリュームがデフォルトの 10 GiB であるベース Amazon Linux AMI を使用しているとします。リージョンの EBS コストが月額 0.10 USD/GB である場合、1 インスタンスあたりの料金は約 0.00129 USD/時、クラスター全体の料金は約 0.00258 USD/時となります (月額 0.10 USD/GB を 30 日で割り、さらに 24 時間で割って、この値に 10 GB を乗算し、さらに 2 クラスターインスタンスを乗算した結果)。

汎用 SSD gp3

gp3 のコストには、EBS ボリュームサイズ料金 (GB 単位)、3000 を超える IOPS 料金 (3000 IOPS までは無料)、125 MB/秒を超えるスループット 料金 (125 MB/秒までは無料) が含まれます。

```
($EBS size in GB/month) * 0.931323 / 30 / 24 * EMR_EBSRootVolumesizeInGiB *
 InstanceCount
+
($EBS IOPS/Month)/30/24* (EMR_EBSRootVolumeIops - 3000) * InstanceCount
+
($EBS throughput/Month)/30/24* (EMR_EBSRootVolumeThroughputInMb/s - 125) *
 InstanceCount
```

たとえば、プライマリノードとコアノードがあるクラスターで、ルートデバイスボリュームがデフォルトの 15 GiB、IOPS が 4000、スループットが 140 であるベース Amazon Linux AMI を使用するとします。リージョンの EBS コストが月額 0.10 USD/GB、3000 を超えてプロビジョニングされた IOPS の月額が 0.005 USD/IOPS、125 を超えてプロビジョニングされたスループットの月額が 0.040 USD/MB/秒とします。この場合、1 時間当り 1 インスタンス当りの費用は約 0.009293 USD、クラスター全体の費用は 1 時間当り約 0.018586 USD になります。

ルートデバイスボリュームのカスタム設定の指定

> **Note**
>
> ルートボリュームサイズと IOPS の比率は、1 ボリューム対 500 IOPS (1:500) を超えることはできません。また、ルートボリューム IOPS とスループットの比率は 1 IOPS 対 0.25 スループット (1:0.25) を超えることはできません。

Console

Amazon EMR コンソールを使用して Amazon EBS ルートデバイスボリュームの属性を指定するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で、[クラスター] を選択し、[クラスターの作成] を選択します
3. Amazon EMR リリース 6.15.0 以降を選択します。
4. [クラスター設定] で [EBS ルートボリューム] セクションに移動し、設定する属性の値を入力します。
5. クラスターに適用するその他のオプションを選択します。
6. クラスターを起動するには、[クラスターの作成] を選択します。

CLI

AWS CLIを使用して Amazon EBS ルートデバイスボリュームの属性を指定するには

- 次の例のように、create-cluster コマンドの `--ebs-root-volume-size`、`--ebs-root-volume-iops`、`--ebs-root-volume-throughput` のパラメータを使用します。

  > **Note**
  >
  > 読みやすくするために、Linux 行連続文字 (\) が含まれています。Linux コマンドでは、これらは削除することも、使用することもできます。Windows の場合、削除するか、キャレット (^) に置き換えてください。

```
aws emr create-cluster --release-label emr-6.15.0\
--ebs-root-volume-size 20 \
--ebs-root-volume-iops 3000\
--ebs-root-volume-throughput 135\
--instance-groups InstanceGroupType=MASTER,\
InstanceCount=1,InstanceType=m5.xlarge
 InstanceGroupType=CORE,InstanceCount=2,InstanceType=m5.xlarge
```

クラスターソフトウェアを設定する

ソフトウェアリリースを選択すると、Amazon EMR は Amazon マシンイメージ (AMI) と Amazon Linux を使用して、クラスターの起動時に選択したソフトウェア (Hadoop、Spark、Hive など) をインストールします。Amazon EMR は、新しいリリースを定期的に提供しており、新機能、新しいアプリケーション、全般的な更新が追加されます。可能な限り最新のリリースを使用して、クラスターを起動することをお勧めします。最新リリースは、コンソールからクラスターを起動する場合のデフォルトオプションです。

各リリースで使用できるソフトウェアの Amazon EMR リリースとバージョンの詳細については、「Amazon EMR リリースガイド」を参照してください。クラスターにインストールされているアプリケーションおよびソフトウェアのデフォルト設定を編集する方法の詳細については、「Amazon EMR リリースガイド」の「アプリケーションの設定」を参照してください。Amazon EMR リリースに含まれるオープンソース Hadoop および Spark エコシステムコンポーネントの一部のバージョンには、「Amazon EMR リリースガイド」で説明されているパッチと改良が含まれています。

クラスター上のインストールに使用できる標準ソフトウェアおよびアプリケーションに加えて、ブートストラップアクションを使用してカスタムソフトウェアをインストールできます。ブートストラップアクションは、クラスターの起動時に実行されるスクリプトと、作成時にクラスターに追加される新しいノードで実行されるスクリプトです。ブートストラップアクションは、各ノードで AWS CLI コマンドを呼び出して Amazon S3 からクラスター内の各ノードにオブジェクトをコピーするのにも役立ちます。

> **Note**
>
> ブートストラップアクションは、Amazon EMR リリース 4.x 以降では使用方法が異なります。Amazon EMR AMI バージョン 2.x および 3.x とのこれらの違いの詳細については、

「Amazon EMR リリースガイド」の「4.x リリースバージョンの違い」を参照してください。

追加のソフトウェアをインストールするためのブートストラップアクションの作成

ブートストラップアクションを使用することにより、追加のソフトウェアのインストール、またはクラスターインスタンスの設定のカスタマイズを行うことができます。ブートストラップアクションは、Amazon EMR が Amazon Linux の Amazon マシンイメージ (AMI) を使用してインスタンスを起動した後に、クラスターで実行されるスクリプトです。ブートストラップアクションは、お客様がクラスターを作成するときに指定するアプリケーションを Amazon EMR がインストールする前、およびクラスターノードがデータの処理を開始する前に実行されます。実行中のクラスターにノードを追加した場合、ブートストラップアクションはこれらのノードでも同じ方法で実行されます。カスタムブートストラップアクションを作成し、クラスターを作成するタイミングを指定できます。

Amazon EMR AMI バージョン 2.x および 3.x 用の事前定義済みのブートストラップアクションのほとんどは、Amazon EMR リリース 4.x ではサポートされません。例えば、`configure-Hadoop` と `configure-daemons` は、Amazon EMR リリース 4.x ではサポートされません。代わりに、Amazon EMR リリース 4.x ではこの機能がネイティブで提供されます。ブートストラップアクションを Amazon EMR AMI バージョン 2.x および 3.x から Amazon EMR リリース 4.x に移行する方法の詳細については、「Amazon EMR リリースガイド」の「Amazon EMR の以前のバージョンの AMI を使用したクラスターとアプリケーション設定のカスタマイズ」を参照してください。

ブートストラップアクションの基本

デフォルトでは、ブートストラップアクションは Hadoop ユーザーとして実行されます。ブートストラップアクションは、`sudo` を使用し、ルート権限で実行できます。

すべての Amazon EMR 管理インターフェイスでブートストラップアクションがサポートされています。コンソール、または API から複数の`bootstrap-actions`パラメータを指定することで AWS CLI、クラスターごとに最大 16 のブートストラップアクションを指定できます。

クラスターの作成時に、Amazon EMR コンソールからオプションでブートストラップアクションを指定できます。

CLI を使用する場合、`create-cluster` コマンドを使用してクラスターを作成するときに `--bootstrap-actions` パラメータを追加して、Amazon EMR にブートストラップアクションスクリプトへの参照を渡すことができます。

```
--bootstrap-actions Path="s3://mybucket/filename",Args=[arg1,arg2]
```

ブートストラップアクションがゼロ以外のエラーコードを返すと、Amazon EMR はそれをエラーとして処理し、インスタンスを終了します。ブートストラップアクションがエラーになるインスタンスが多すぎると、Amazon EMR はクラスターを終了します。ごくわずかのインスタンスでエラーになった場合、Amazon EMR は失敗したインスタンスの再割り当てを試み、処理を継続します。クラスターの lastStateChangeReason エラーコードを使用して、ブートストラップアクションによって引き起こされたエラーを識別します。

ブートストラップアクションを条件付きで実行する

マスターノードでのみブートストラップアクションを実行するには、そのノードがマスターかどうかを判断するロジックを含むカスタムブートストラップアクションを使用できます。

```
#!/bin/bash
if grep isMaster /mnt/var/lib/info/instance.json | grep false;
then
    echo "This is not master node, do nothing,exiting"
    exit 0
fi
echo "This is master, continuing to execute script"
# continue with code logic for master node below
```

次の出力はコアノードから出力されます。

```
This is not master node, do nothing, exiting
```

次の出力はマスターノードから出力されます。

```
This is master, continuing to execute script
```

このロジックを使用するには、上記のコードを含むブートストラップアクションを Amazon S3 バケットにアップロードします。で AWS CLI、 aws emr create-cluster API コールに --bootstrap-actionsパラメータを追加し、ブートストラップスクリプトの場所を の値として指定しますPath。

シャットダウンアクション

ブートストラップアクションスクリプトで 1 つ以上の shutdown actions を作成するには、スクリプトを /mnt/var/lib/instance-controller/public/shutdown-actions/ ディレクトリに書

き込みます。クラスターが終了すると、ディレクトリ内のすべてのスクリプトが並行して実行されます。各スクリプトが 60 秒以内に実行され完了しなければなりません。

ノードの終了時にエラーが発生した場合、シャットダウンアクションスクリプトが実行される保証はありません。

> **Note**
>
> Amazon EMR バージョン 4.0 以降を使用する場合、マスターノードに `/mnt/var/lib/instance-controller/public/shutdown-actions/` ディレクトリを手動で作成する必要があります。デフォルトでは存在しませんが、作成後は、このディレクトリのスクリプトがシャットダウンの前に実行されます。ディレクトリを作成するためのマスターノードへの接続の詳細については、「SSH を使用してプライマリノードに接続する」を参照してください。

カスタムブートストラップアクションの使用

カスタムスクリプトを作成すると、カスタマイズされたブートストラップアクションを実行します。Amazon EMR インターフェイスはすべて、カスタムブートストラップアクションを参照できます。

> **Note**
>
> 最高のパフォーマンスを得るには、Amazon EMR で使用するカスタムブートストラップアクション、スクリプト、およびその他のファイルを、クラスター AWS リージョン と同じ にある Amazon S3 バケットに保存することをお勧めします。

内容

カスタムブートストラップアクションの追加

> **Note**
>
> Amazon EMR コンソールは、再設計され、使いやすくなりました。古いコンソールと新しいコンソールエクスペリエンスの違いについては、「Amazon EMR コンソール」を参照してください。

New console

新しいコンソールを使用してブートストラップアクションでクラスターを作成するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で、[クラスター] を選択し、[クラスターの作成] を選択します
3. [ブートストラップアクション] で [追加] を選択し、アクションの名前、スクリプトの場所、オプション引数を指定します。[ブートストラップアクションを追加] を選択します。
4. オプションで、さらにブートストラップアクションを追加します。
5. クラスターに適用するその他のオプションを選択します。
6. クラスターを起動するには、[クラスターの作成] を選択します。

Old console

古いコンソールを使用してカスタムブートストラップアクションでクラスターを作成するには

1. 新しい Amazon EMR コンソールに移動し、サイドナビゲーションから [古いコンソールに切り替え] を選択します。古いコンソールに切り替えたときの動作の詳細については、「Using the old console」を参照してください。
2. [クラスターを作成] を選択します。
3. [Go to advanced options] をクリックします。
4. [Create Cluster] - [Advanced Options] で、説明に従ってステップ 1 と 2 でオプションを選択し、「Step 3: General Cluster Settings」に進みます。
5. [Bootstrap Actions] で [Configure and add] を選択して、ブートストラップアクションの [Name]、[JAR location]、および引数を指定します。追加を選択します。

6. オプションで、必要に応じてさらにブートストラップアクションを追加します。
7. クラスターの作成に進みます。ブートストラップアクションは、クラスターがプロビジョニングおよび初期化された後で実行されます。

   クラスターのプライマリノードの実行中には、プライマリノードに接続して、/mnt/var/log/bootstrap-actions/1 ディレクトリにブートストラップアクションスクリプトが生成したログファイルを確認できます。

CLI

を使用してカスタムブートストラップアクションでクラスターを作成するには AWS CLI

を使用してブートストラップアクション AWS CLI を含める場合は、カンマ区切りリストArgsとして Pathと を指定します。次の例では、引数リストが使用されていません。

- カスタムブートストラップアクションでクラスターを起動するには、次のコマンドを入力し、*myKey* を EC2 キーペアの名前に置き換えます。--bootstrap-actions をパラメータとして含め、ブートストラップスクリプトの場所を Path の値として指定します。

  - Linux、UNIX、Mac OS X ユーザー:

    ```
    aws emr create-cluster --name "Test cluster" --release-label emr-4.0.0 \
    --use-default-roles --ec2-attributes KeyName=myKey \
    --applications Name=Hive Name=Pig \
    --instance-count 3 --instance-type m5.xlarge \
    --bootstrap-actions Path="s3://elasticmapreduce/bootstrap-actions/download.sh"
    ```

  - Windows ユーザー:

    ```
    aws emr create-cluster --name "Test cluster" --release-label emr-4.2.0 --use-
    default-roles --ec2-attributes KeyName=myKey --applications Name=Hive Name=Pig
     --instance-count 3 --instance-type m5.xlarge --bootstrap-actions Path="s3://
    elasticmapreduce/bootstrap-actions/download.sh"
    ```

  --instance-groups パラメータを使用せずにインスタンス数を指定すると、1 つのプライマリノードが起動され、残りのインスタンスはコアノードとして起動されます。すべてのノードで、コマンドで指定したインスタンスタイプが使用されます。

> **Note**
>
> 以前にデフォルトの Amazon EMR サービスロールと EC2 インスタンスプロファイルを作成していない場合は、「`aws emr create-default-roles`」と入力してそれらを作成してから、`create-cluster` サブコマンドを入力します。

での Amazon EMR コマンドの使用の詳細については AWS CLI、「」を参照してくださいhttps://docs.aws.amazon.com/cli/latest/reference/emr。

カスタムブートストラップアクションを使用した Simple Storage Service (Amazon S3) から各ノードへのオブジェクトのコピー

アプリケーションをインストールする前に、Simple Storage Service (Amazon S3) からクラスター内の各ノードにオブジェクトをコピーするブートストラップアクションを使用できます。 AWS CLI はクラスターの各ノードにインストールされるため、ブートストラップアクションは AWS CLI コマンドを呼び出すことができます。

Simple Storage Service (Amazon S3) から各クラスターノードのローカルフォルダ `/mnt1/myfolder` にファイル `myfile.jar` をコピーする簡単なブートストラップアクションスクリプトの例を次に示します。スクリプトは、以下の内容を使って、`copymyfile.sh` のファイル名で Simple Storage Service (Amazon S3) に保存されます。

```
#!/bin/bash
aws s3 cp s3://mybucket/myfilefolder/myfile.jar /mnt1/myfolder
```

クラスターの起動時にスクリプトを指定します。次の AWS CLI 例は、これを示しています。

```
aws emr create-cluster --name "Test cluster" --release-label emr-7.1.0 \
--use-default-roles --ec2-attributes KeyName=myKey \
--applications Name=Hive Name=Pig \
--instance-count 3 --instance-type m5.xlarge \
--bootstrap-actions Path="s3://mybucket/myscriptfolder/copymyfile.sh"
```

クラスターハードウェアとネットワークを設定する

Amazon EMR クラスターの作成時における重要な考慮事項は、Amazon EC2 インスタンスおよびネットワークオプションを設定する方法です。この章では、これらのオプションを詳しく説明したうえで、ベストプラクティスとガイドラインによって総括します。

- ノードタイプ – EMR クラスター内の Amazon EC2 インスタンスは、ノードタイプに分類されます。プライマリノード、コアノード、およびタスクノードの 3 つがあります。各ノードタイプは、クラスター上にインストールする分散アプリケーションにより定義される一連のロールを実行します。Hadoop MapReduce または Spark ジョブ中に、コアノードとタスクノードのコンポーネントがデータを処理し、出力を Amazon S3 または HDFS に転送し、ステータスメタデータをプライマリノードに戻します。単一ノードクラスターの場合、すべてのコンポーネントはプライマリノード上で実行されます。詳細については、「ノードタイプ (プライマリノード、コアノード、タスクノード) について理解する」を参照してください。
- EC2 インスタンス — クラスターを作成するとき、各タイプのノードが実行される Amazon EC2 インスタンスについて選択します。EC2 インスタンスタイプは、ノードの処理およびストレージプロファイルを決定します。ノードの Amazon EC2 インスタンスの選択は、クラスター内の個々のノードタイプのパフォーマンスプロファイルを決定するため、重要です。詳細については、「Amazon EC2 インスタンスを設定する」を参照してください。
- ネットワーク — Amazon EMR クラスターは、パブリックサブネット、プライベートサブネット、または共有サブネットを使用して VPC で起動できます。ネットワーク設定により、お客様とサービスがクラスターに接続して作業を実行する方法、クラスターがデータストアおよび AWS リソースに接続する方法、およびそれらの接続でトラフィックを制御するためのオプションが決定されます。詳細については、「ネットワークを設定する」を参照してください。
- インスタンスグループ – 各ノードタイプをホストする EC2 インスタンスの集合は、インスタンスフリートまたはユニフォームインスタンスグループと呼ばれます。インスタンスグループの設定は、クラスターの作成時に選択します。この選択により、実行中にクラスターにノードを追加する方法が決定されます。この設定はすべてのノードタイプに適用されます。後で変更することはできません。詳細については、「インスタンスフリートまたはユニフォームインスタンスグループでクラスターを作成する」を参照してください。

> **Note**
>
> インスタンスフリート設定は、5.0.0 および 5.0.3 を除く Amazon EMR リリース 4.8.0 以降でのみ使用できます。

ノードタイプ (プライマリノード、コアノード、タスクノード) について理解する

このセクションでは、Amazon EMR によるこれらの各ノードタイプの使用方法を説明し、クラスターの容量計画の基礎を提供します。

プライマリノード

プライマリノードはクラスターを管理し、通常は分散アプリケーションのプライマリコンポーネントを実行します。例えば、プライマリノードは YARN ResourceManager サービスを実行してアプリケーションのリソースを管理します。また、HDFS NameNode サービスを実行し、クラスターに送信されたジョブのステータスを追跡し、インスタンスグループのヘルスをモニタリングします。

クラスターの進行状況をモニタリングしてアプリケーションを直接操作するには、Hadoop ユーザーとして SSH でプライマリノードに接続します。詳細については、「SSH を使用してプライマリノードに接続する」を参照してください。プライマリノードに接続すると、Hadoop ログファイルなどのディレクトリとファイルに直接アクセスすることができます。詳細については、「ログファイルを表示する」を参照してください。プライマリノードで実行されるウェブサイトとしてアプリケーションで公開しているユーザーインターフェイスを表示することもできます。詳細については、「Amazon EMR クラスターでホストされているウェブインターフェイスを表示する」を参照してください。

> **Note**
>
> Amazon EMR 5.23.0 以降では、3 つのプライマリノードを持つクラスターを起動して、YARN Resource Manager、HDFS、Spark NameNode、Hive、Ganglia などのアプリケーションの高可用性をサポートできます。プライマリノードは、現在この機能による潜在的な単一障害点ではありません。プライマリノードのいずれかに障害が発生した場合、Amazon EMR は自動的にスタンバイプライマリノードにフェイルオーバーし、障害が発生したプライマリノードを同じ設定とブートストラップアクションを持つ新しいプライマリノードに置き換えます。詳細については、「プライマリノードの計画と設定」を参照してください。

コアノード

コアノードは、プライマリノードによって管理されます。コアノードはデータノードデーモンを実行して、Hadoop Distributed File System (HDFS)の一部としてデータストレージを調整します。さら

にタスクトラッカーデーモンを実行し、インストールされているアプリケーションが要求するデータ上で、その他の並列計算タスクを実行します。例えば、コアノードは YARN NodeManager デーモン、Hadoop MapReduce タスク、Spark エグゼキューターを実行します。

コアインスタンスグループまたはインスタンスフリートはクラスターごとに 1 つだけですが、インスタンスグループまたはインスタンスフリートでは、複数の Amazon EC2 インスタンスで複数のノードを実行できます。インスタンスグループの場合、クラスターの実行中に Amazon EC2 インスタンスを追加または削除することができます。また、オートスケーリングを設定して、メトリクスの値に基づいてインスタンスを追加することもできます。インスタンスグループ設定で Amazon EC2 インスタンスを追加または削除する方法の詳細については、「クラスターのスケーリングを使用する」を参照してください。

インスタンスフリートの場合、オンデマンドとスポットで、インスタンスフリートのターゲット容量を変更することにより、事実上インスタンスを追加および変更できます。ターゲット容量の詳細については、「 インスタンスフリートオプション」を参照してください。

> **Warning**
>
> 実行中のコアノードのHDFSデーモンを停止すること、もしくは実行中のコアノードそのものを停止することは、データ損失のリスクにつながります コアノードにスポットインスタンスを使用する場合は十分に考慮するようにしてください。詳細については、「スポットインスタンスを使用すべき場合」を参照してください。

タスクノード

タスクノードを使用して、Hadoop タスクや Spark エグゼキューターなどのデータに対して並列計算 MapReduce タスクを実行するための能力を追加できます。タスクノードは、データノードデーモン上で実行されることも、HDFS でデータを保存することもありません。コアノードと同様に、既存のユニフォームインスタンスグループに Amazon EC2 インスタンスを追加するか、タスクインスタンスフリートのターゲット容量を変更することにより、クラスターにタスクノードを追加できます。

ユニフォームインスタンスグループ設定では、最大で 48 のタスクインスタンスグループを持つことができます。この方法でインスタンスグループを追加できることで、お客様は、オンデマンドインスタンスとスポットインスタンスなど、異なる Amazon EC2 インスタンスタイプおよび料金設定オプションを組み合わせることができます。これにより、費用対効果の高い方法でワークロードの要件に対応できる柔軟性が得られます。

インスタンスフリート設定では、インスタンスタイプと購入オプションを組み合わせる機能が組み込まれているため、タスクインスタンスフリートは 1 つだけになります。

スポットインスタンスはタスクノードの実行に使用されることが多いため、Amazon EMR には、タスクノードが終了しても実行中のジョブが失敗しないように YARN ジョブをスケジュールするための機能がデフォルトで備えられています。Amazon EMR は、アプリケーションマスタープロセスをコアノードでのみ実行できるようにすることで、これを実現しています。アプリケーションマスタープロセスは実行中のジョブを制御し、ジョブが有効である間は存続する必要があります。

Amazon EMR リリース 5.19.0 以降では、組み込みの YARN ノードラベル機能を使用して、これを実現しています。(以前のバージョンではコードパッチを使用していました)。`yarn-site` と `capacity-scheduler` の設定分類のプロパティは、YARN capacity-scheduler と fair-scheduler がノードラベルを利用できるように、デフォルトで設定されます。Amazon EMR は、`CORE` ラベルでコアノードに自動的にラベルを付け、アプリケーションマスターが CORE ラベルを持つノードでのみスケジュールされるようにプロパティを設定します。yarn-site および capacity-scheduler 設定分類の関連プロパティを手動で変更したり、関連する XML ファイルで直接変更したりすると、この機能が停止したり、この機能が変更されたりする可能性があります。

Amazon EMR 6.x リリースシリーズ以降では、YARN ノードラベル機能はデフォルトで無効になっています。アプリケーションプライマリプロセスは、デフォルトでコアノードとタスクノードの両方で実行できます。次のプロパティを設定することで、YARN ノードラベル機能を有効にできます。

- `yarn.node-labels.enabled: true`
- `yarn.node-labels.am.default-node-label-expression: 'CORE'`

特定のプロパティの詳細については、タスクノードのスポットインスタンスの終了によるジョブの失敗を防ぐ Amazon EMR 設定を参照してください。

Amazon EC2 インスタンスを設定する

EC2 インスタンスの設定はさまざまであり、インスタンスタイプと呼ばれています。インスタンスタイプには、さまざまな CPU、入出力、およびストレージ容量があります。インスタンスタイプに加えて、Amazon EC2 インスタンスにはさまざまな購入オプションを選択できます。ユニフォームインスタンスグループまたはインスタンスフリート内で、さまざまなインスタンスタイプと購入オプションを指定できます。詳細については、「インスタンスフリートまたはユニフォームインスタンスグループでクラスターを作成する」を参照してください。アプリケーションのインスタンスタイプと購入オプションを選択するためのガイダンスについては、「クラスター設定のベストプラクティス」を参照してください。

> **Important**
>
> を使用してインスタンスタイプを選択すると AWS Management Console、各インスタンスタイプに表示される vCPU の数は、そのインスタンスタイプの EC2 vCPU の数ではなく、そのインスタンスタイプの YARN vCPUsの数になります。 各インスタンスタイプの vCPU 数の詳細については、「Amazon EC2 インスタンスタイプ」を参照してください。

トピック

- サポートされるインスタンスタイプ
- ネットワークを設定する
- インスタンスフリートまたはユニフォームインスタンスグループでクラスターを作成する

サポートされるインスタンスタイプ

このセクションでは、Amazon EMR がサポートしているインスタンスタイプを AWS リージョン別に整理して説明しています。インスタンスタイプの詳細については、「Amazon EC2 インスタンス」および「Amazon Linux AMI インスタンスタイプマトリックス」を参照してください。

すべてのインスタンスタイプがすべてのリージョンで使用できるわけではありません。また、インスタンスの可用性は、指定されたリージョンとアベイラビリティーゾーンでの可用性と需要の影響を受けます。インスタンスのアベイラビリティーゾーンは、クラスターの起動に使用するサブネットによって決まります。

考慮事項

Amazon EMR クラスターのインスタンスタイプを選択するときは、次の点を考慮してください。

> **Important**
>
> を使用してインスタンスタイプを選択すると AWS Management Console、各インスタンスタイプに表示される vCPU の数は、そのインスタンスタイプの EC2 vCPU の数ではなく、そのインスタンスタイプの YARN vCPUsの数になります。 各インスタンスタイプの vCPU 数の詳細については、「Amazon EC2 インスタンスタイプ」を参照してください。

- 特定のリージョンおよびアベイラビリティーゾーンで利用できないインスタンスタイプを使用してクラスターを作成すると、クラスターがプロビジョニングに失敗したり、プロビジョニングが停止

する場合があります。インスタンスが使用できるかどうかについては、「Amazon EMR 料金ページ」またはこのページの「でサポートされているインスタンスタイプ AWS リージョン」表を参照してください。

- Amazon EMR リリースバージョン 5.13.0 から、すべてのインスタンスでルートボリュームに HVM 仮想化および EBS-backed ストレージが使用されます。5.13.0 以前のリリースバージョンの Amazon EMR を使用する場合、一部の旧世代インスタンスでは PVM 仮想化が使用されます。詳細については、「Linux AMI 仮想化タイプ」を参照してください。
- 一部のインスタンスタイプは拡張ネットワーキングをサポートします。詳細については、「Linux の拡張ネットワーキング」を参照してください。
- デフォルトでは、NVIDIA ドライバーおよび CUDA ドライバーは、GPU インスタンスタイプでインストールされています。

でサポートされているインスタンスタイプ AWS リージョン

次の表は、Amazon EMR がサポートする Amazon EC2 インスタンスタイプを 別にまとめたものです AWS リージョン。表には、各インスタンスタイプをサポートする 5.x、6.x、および 7.x シリーズの最も初期の Amazon EMR リリースも含まれています。

米国東部 (バージニア北部) - us-east-1

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
汎用	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5zn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.3xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7a.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.2xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.4xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.8xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
コンピューティング最適化	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5ad.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5ad.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7a.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c7a.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
高速コンピューティング	g3.4xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3s.xlarge	emr-5.19.0, emr-6.0.0, emr-7.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g6.xlarge	emr-7.2.0
	g6.2xlarge	emr-7.2.0
	g6.4xlarge	emr-7.2.0
	g6.8xlarge	emr-7.2.0
	g6.12xlarge	emr-7.2.0
	g6.16xlarge	emr-7.2.0
	g6.24xlarge	emr-7.2.0
	g6.48xlarge	emr-7.2.0
	gr6.4xlarge	emr-7.2.0
	gr6.8xlarge	emr-7.2.0
	p2.xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.2xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	p3.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p5.48xlarge	emr-6.14.0, emr-7.0.0
メモリを最適化	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5b.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5b.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7a.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7a.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1e.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x1e.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	z1d.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.3xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	z1d.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
ストレージの最適化	d3.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	h1.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	h1.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	h1.8xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	h1.16xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	im4gn.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

米国東部 (オハイオ) - us-east-2

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
汎用	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5zn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.3xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7a.2xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7a.4xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7a.8xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7a.12xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7a.16xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7a.24xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7a.32xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7a.48xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7i.2xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7i.4xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7i.8xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7i.12xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7i.16xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7i.24xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7i.48xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7i-flex.xlarge	emr-4.6.0, emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7i-flex.2xlarge	emr-4.6.0, emr-5.0.1, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7i-flex.4xlarge	emr-4.6.0, emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7i-flex.8xlarge	emr-4.6.0, emr-5.0.1, emr-6.0.0, emr-7.0.0
コンピューティング最適化	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5ad.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7a.xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7a.2xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7a.4xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7a.8xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7a.12xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7a.16xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7a.24xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7a.32xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7a.48xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c7gn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7i.xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7i.2xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7i.4xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7i.8xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7i.12xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7i.16xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7i.24xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c7i.48xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
高速コンピューティング	g3.4xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3s.xlarge	emr-5.19.0, emr-6.0.0, emr-7.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g6.xlarge	emr-7.2.0
	g6.2xlarge	emr-7.2.0
	g6.4xlarge	emr-7.2.0
	g6.8xlarge	emr-7.2.0
	g6.12xlarge	emr-7.2.0
	g6.16xlarge	emr-7.2.0
	g6.24xlarge	emr-7.2.0
	g6.48xlarge	emr-7.2.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	gr6.4xlarge	emr-7.2.0
	gr6.8xlarge	emr-7.2.0
	p2.xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.2xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p5.48xlarge	emr-6.14.0, emr-7.0.0
メモリを最適化	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5b.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7a.xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7a.2xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7a.4xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7a.8xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7a.12xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7a.16xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7a.24xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7a.32xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7a.48xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7i.2xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7i.4xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7i.8xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7i.12xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7i.16xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7i.24xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7i.48xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7iz.xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7iz.2xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7iz.4xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7iz.8xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7iz.12xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7iz.16xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7iz.32xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1e.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x2gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	z1d.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.3xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
ストレージの最適化	d3.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	h1.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	h1.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	h1.8xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	h1.16xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	im4gn.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

米国西部 (北カリフォルニア) us-west-1

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
汎用	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5zn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.3xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.2xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.4xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.8xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
コンピューティング最適化	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
高速コンピューティング	g3.4xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
メモリを最適化	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	z1d.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.3xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
ストレージの最適化	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

米国東部 (オレゴン) - us-west-2

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
汎用	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5zn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.3xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.2xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.4xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.8xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
コンピューティング最適化	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5ad.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7a.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c7a.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c7gn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
高速コンピューティング	g3.4xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	g3.16xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3s.xlarge	emr-5.19.0, emr-6.0.0, emr-7.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g6.xlarge	emr-7.2.0
	g6.2xlarge	emr-7.2.0
	g6.4xlarge	emr-7.2.0
	g6.8xlarge	emr-7.2.0
	g6.12xlarge	emr-7.2.0
	g6.16xlarge	emr-7.2.0
	g6.24xlarge	emr-7.2.0
	g6.48xlarge	emr-7.2.0
	gr6.4xlarge	emr-7.2.0
	gr6.8xlarge	emr-7.2.0
	p2.xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	p2.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.2xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p5.48xlarge	emr-6.14.0, emr-7.0.0
メモリを最適化	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5b.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7a.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7a.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1e.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x2gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	z1d.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	z1d.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.3xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
ストレージの最適化	d3.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	d3en.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	h1.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	h1.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	h1.8xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	h1.16xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	im4gn.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	is4gen.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

AWS GovCloud（米国西部) - us-gov-west-1

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
汎用	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
コンピューティング最適化	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
高速コンピューティング	g3.4xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	p2.xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.2xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
メモリを最適化	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1e.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x1e.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
ストレージの最適化	d3.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i4i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

AWS GovCloud（米国東部) - us-gov-east-1

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
汎用	m5.xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5d.16xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
コンピューティング最適化	c5.xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5d.xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
高速コンピューティング	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
メモリを最適化	r5.xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5d.xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5d.24xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	r5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r7i.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7i.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1e.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x1e.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
ストレージの最適化	i3.xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

アフリカ (ケープタウン) – af-south-1

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
汎用	m5.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5d.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
コンピューティング最適化	c5.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5.9xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5ad.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5ad.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5d.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5d.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
高速コンピューティング	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
メモリを最適化	r5.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5d.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1e.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
ストレージの最適化	i3.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i3.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

アジアパシフィック (香港) – ap–east–1

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
汎用	m5.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
コンピューティング最適化	c5.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5d.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5d.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
高速コンピューティング	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
メモリを最適化	r5.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5d.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5d.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
ストレージの最適化	i3.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

アジアパシフィック (ジャカルタ): ap-southeast-3

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
汎用	m5.xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m5.2xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m5.4xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m5.8xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m5.12xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m5.16xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m5.24xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m5d.xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m5d.2xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m5d.4xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m5d.8xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5d.12xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m5d.16xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m5d.24xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m6g.xlarge	emr-5.30.2, emr-6.1.1, emr-7.0.0
	m6g.2xlarge	emr-5.30.2, emr-6.1.1, emr-7.0.0
	m6g.4xlarge	emr-5.30.2, emr-6.1.1, emr-7.0.0
	m6g.8xlarge	emr-5.30.2, emr-6.1.1, emr-7.0.0
	m6g.12xlarge	emr-5.30.2, emr-6.1.1, emr-7.0.0
	m6g.16xlarge	emr-5.30.2, emr-6.1.1, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
コンピューティング最適化	c5.xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5.2xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	c5.4xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	c5.9xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	c5.12xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	c5.18xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	c5.24xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	c5d.xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	c5d.2xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	c5d.4xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	c5d.9xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	c5d.12xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	c5d.18xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5d.24xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	c5n.xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	c5n.2xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	c5n.4xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	c5n.9xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	c5n.18xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	c6g.xlarge	emr-5.31.1, emr-6.1.1, emr-7.0.0
	c6g.2xlarge	emr-5.31.1, emr-6.1.1, emr-7.0.0
	c6g.4xlarge	emr-5.31.1, emr-6.1.1, emr-7.0.0
	c6g.8xlarge	emr-5.31.1, emr-6.1.1, emr-7.0.0
	c6g.12xlarge	emr-5.31.1, emr-6.1.1, emr-7.0.0
	c6g.16xlarge	emr-5.31.1, emr-6.1.1, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
高速コンピューティング	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
メモリを最適化	r5.xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r5.2xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r5.4xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r5.8xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r5.12xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r5.16xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r5.24xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r5d.xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5d.2xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r5d.4xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r5d.8xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r5d.12xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r5d.16xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r5d.24xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r6g.xlarge	emr-5.31.1, emr-6.1.1, emr-7.0.0
	r6g.2xlarge	emr-5.31.1, emr-6.1.1, emr-7.0.0
	r6g.4xlarge	emr-5.31.1, emr-6.1.1, emr-7.0.0
	r6g.8xlarge	emr-5.31.1, emr-6.1.1, emr-7.0.0
	r6g.12xlarge	emr-5.31.1, emr-6.1.1, emr-7.0.0
	r6g.16xlarge	emr-5.31.1, emr-6.1.1, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r7i.xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r7i.2xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r7i.4xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r7i.8xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r7i.12xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r7i.16xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7i.24xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r7i.48xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
ストレージの最適化	i3.xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	i3.2xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	i3.4xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	i3.8xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	i3.16xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	i3en.xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	i3en.2xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	i3en.3xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	i3en.6xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	i3en.12xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	i3en.24xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

アジアパシフィック (ムンバイ) ap-south-1

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
汎用	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.xlarge	emr-4.6.0, emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7i-flex.2xlarge	emr-4.6.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.4xlarge	emr-4.6.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.8xlarge	emr-4.6.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
コンピューティング最適化	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
高速コンピューティング	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	p2.xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	p2.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
メモリを最適化	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1e.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x1e.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	z1d.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.3xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
ストレージの最適化	d3.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	is4gen.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

アジアパシフィック (ハイデラバード): ap-south-2

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
汎用	m5.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5d.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6g.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6i.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.32xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
コンピューティング最適化	c5.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.9xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.18xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5d.9xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.18xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6i.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6i.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6i.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6i.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6i.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6i.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6i.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6i.32xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
メモリを最適化	r5.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6g.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.32xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
ストレージの最適化	i3.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.3xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i3en.6xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

アジアパシフィック (大阪) – ap-northeast-3

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
汎用	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
コンピューティング最適化	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
高速コンピューティング	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
メモリを最適化	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1e.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
ストレージの最適化	i3.xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

アジアパシフィック (ソウル) ap-northeast-2

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
汎用	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5zn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.3xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5zn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.2xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.4xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.8xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
コンピューティング最適化	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
高速コンピューティング	g3.4xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	g3s.xlarge	emr-5.19.0, emr-6.0.0, emr-7.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	p2.xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.2xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
メモリを最適化	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5b.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5b.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1e.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	z1d.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	z1d.3xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
ストレージの最適化	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

アジアパシフィック (シンガポール) ap-southeast-1

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
汎用	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5zn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.3xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.2xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.4xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.8xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
コンピューティング最適化	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5ad.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
高速コンピューティング	g3.4xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	p2.xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	p2.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.2xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
メモリを最適化	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5b.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1e.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x1e.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	z1d.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.3xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
ストレージの最適化	d3.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	d3en.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	im4gn.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

アジアパシフィック (シドニー) - ap-southeast-2

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
汎用	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5zn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.3xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7i-flex.2xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.4xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.8xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
コンピューティング最適化	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5ad.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
高速コンピューティング	g3.4xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	g3.8xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3s.xlarge	emr-5.19.0, emr-6.0.0, emr-7.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	p2.xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.2xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
メモリを最適化	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5b.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5b.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.24xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	r6a.32xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	r6a.48xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1e.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x1e.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	z1d.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.3xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
ストレージの最適化	d3.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	im4gn.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	is4gen.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

アジアパシフィック (東京) - ap-northeast-1

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
汎用	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5zn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.3xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7a.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.2xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.4xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.8xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
コンピューティング最適化	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7a.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c7gn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
高速コンピューティング	g3.4xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3s.xlarge	emr-5.19.0, emr-6.0.0, emr-7.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	p2.xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.2xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
メモリを最適化	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5b.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5b.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.24xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	r6a.32xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	r6a.48xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7a.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7a.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7iz.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1e.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x1e.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	z1d.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.3xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
ストレージの最適化	d3.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	d3en.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	im4gn.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	im4gn.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

カナダ (中部) ca-central-1

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
汎用	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	m7i.2xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	m7i.4xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7i.8xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	m7i.12xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	m7i.16xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	m7i.24xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	m7i.48xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	m7i-flex.xlarge	emr-4.8.2, emr-5.0.2, emr-6.0.0, emr-7.0.0
	m7i-flex.2xlarge	emr-4.8.2, emr-5.0.2, emr-6.0.0, emr-7.0.0
	m7i-flex.4xlarge	emr-4.8.2, emr-5.0.2, emr-6.0.0, emr-7.0.0
	m7i-flex.8xlarge	emr-4.8.2, emr-5.0.2, emr-6.0.0, emr-7.0.0
コンピューティング最適化	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c7i.xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	c7i.2xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	c7i.4xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	c7i.8xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	c7i.12xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	c7i.16xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	c7i.24xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	c7i.48xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
高速コンピューティング	g3.4xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	p3.2xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
メモリを最適化	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5b.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7i.xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	r7i.2xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	r7i.4xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	r7i.8xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	r7i.12xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	r7i.16xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	r7i.24xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	r7i.48xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1e.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x1e.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
ストレージの最適化	d3.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	im4gn.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	is4gen.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

カナダ西部 (カルガリー) - ca-west-1

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
汎用	m5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m5d.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m5d.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m5d.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m5d.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m5d.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m5d.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m5d.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6g.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6g.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6g.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6g.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6g.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6g.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6gd.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6gd.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6gd.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6gd.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6gd.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6gd.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6i.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6i.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6i.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6i.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6i.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6i.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6i.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6i.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6id.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6id.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6id.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6id.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6id.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6id.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6id.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6id.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
コンピューティング最適化	c5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c5.9xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c5.18xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6g.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6g.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6g.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6g.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6g.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6g.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6gn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6gn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6gn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6gn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6gn.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6gn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6i.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6i.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6i.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6i.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6i.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6i.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6i.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6i.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6id.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6id.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6id.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6id.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6id.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6id.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6id.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6id.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
メモリを最適化	r5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6g.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6g.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6g.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6g.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6g.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6g.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6i.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6i.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6i.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6i.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6i.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6i.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6i.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6i.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6id.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6id.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6id.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6id.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6id.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6id.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6id.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6id.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
ストレージの最適化	i3en.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	i3en.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	i3en.3xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	i3en.6xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	i3en.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	i3en.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i4i.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

中国 (寧夏) – cn-northwest-1

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
汎用	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
コンピューティング最適化	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
高速コンピューティング	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	p3.2xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
メモリを最適化	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	z1d.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.3xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
ストレージの最適化	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

中国 (北京) cn-north-1

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
汎用	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
コンピューティング最適化	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
高速コンピューティング	g3.4xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	g3.16xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3s.xlarge	emr-5.19.0, emr-6.0.0, emr-7.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	p2.xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.2xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
メモリを最適化	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
ストレージの最適化	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i4i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

欧州 (フランクフルト) eu-central-1

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
汎用	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5zn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.3xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7a.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.2xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.4xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.8xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
コンピューティング最適化	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5ad.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5ad.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7a.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c7a.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7i.xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
高速コンピューティング	g3.4xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3s.xlarge	emr-5.19.0, emr-6.0.0, emr-7.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	p2.xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.2xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	p3.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
メモリを最適化	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5b.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5b.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7a.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7a.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7iz.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1e.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x1e.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	z1d.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.3xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
ストレージの最適化	d3.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	d3en.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	im4gn.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	im4gn.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

欧州 (チューリッヒ): eu-central-2

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
汎用	m5.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5d.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6gd.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.32xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
コンピューティング最適化	c5.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.9xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.18xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5d.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.9xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.18xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6g.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6gd.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6gd.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6gd.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6gd.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6gd.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6gd.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
メモリを最適化	r5.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5d.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6gd.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6gd.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6gd.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6gd.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6gd.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6gd.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6i.32xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
ストレージの最適化	d3.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	d3.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	d3.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	d3.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i3.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.3xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.6xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

ヨーロッパ (アイルランド) eu-west-1

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
汎用	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5zn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.3xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7a.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.2xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.4xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.8xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
コンピューティング最適化	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5ad.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7a.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c7a.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
高速コンピューティング	g3.4xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3s.xlarge	emr-5.19.0, emr-6.0.0, emr-7.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	p2.xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.2xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
メモリを最適化	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5b.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7a.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1e.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x2gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	z1d.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.3xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
ストレージの最適化	d3.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	d3en.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	h1.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	h1.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	h1.8xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	h1.16xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	im4gn.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	is4gen.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

欧州 (ロンドン) eu-west-2

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
汎用	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	m7i.2xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	m7i.4xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	m7i.8xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	m7i.12xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	m7i.16xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7i.24xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	m7i.48xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	m7i-flex.xlarge	emr-4.8.2, emr-5.0.3, emr-6.0.0, emr-7.0.0
	m7i-flex.2xlarge	emr-4.8.2, emr-5.0.3, emr-6.0.0, emr-7.0.0
	m7i-flex.4xlarge	emr-4.8.2, emr-5.0.3, emr-6.0.0, emr-7.0.0
	m7i-flex.8xlarge	emr-4.8.2, emr-5.0.3, emr-6.0.0, emr-7.0.0
コンピューティング最適化	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7i.xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	c7i.2xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	c7i.4xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	c7i.8xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	c7i.12xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	c7i.16xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	c7i.24xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c7i.48xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
高速コンピューティング	g3.4xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3s.xlarge	emr-5.19.0, emr-6.0.0, emr-7.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	p3.2xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
メモリを最適化	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5b.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5b.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	r7i.2xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	r7i.4xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	r7i.8xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	r7i.12xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	r7i.16xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	r7i.24xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	r7i.48xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	z1d.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	z1d.3xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
ストレージの最適化	d3.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i4i.24xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	im4gn.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

欧州 (ミラノ): eu-south-1

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
汎用	m5.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5a.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5a.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
コンピューティング最適化	c5.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5ad.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5d.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5d.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
高速コンピューティング	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
メモリを最適化	r5.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5a.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5b.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5d.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5d.16xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
ストレージの最適化	i3.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i3en.6xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

欧州 (スペイン): eu-south-2

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
汎用	m5.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5d.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6gd.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7a.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m7i.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m7i.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m7i.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m7i.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m7i.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m7i.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m7i.48xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7i-flex.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m7i-flex.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m7i-flex.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m7i-flex.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
コンピューティング最適化	c5.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.9xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.18xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5d.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.9xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.18xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7a.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c7a.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c7a.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c7a.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c7a.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c7a.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c7a.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c7a.32xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c7a.48xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7i.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c7i.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c7i.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c7i.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c7i.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c7i.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c7i.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c7i.48xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
メモリを最適化	r5.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5d.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r7a.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r7a.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r7a.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7a.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r7a.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r7a.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r7a.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r7a.32xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r7a.48xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r7i.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r7i.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r7i.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r7i.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r7i.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7i.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r7i.48xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
ストレージの最適化	i3.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.3xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.6xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

欧州 (パリ) eu-west-3

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
汎用	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	m7i.2xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	m7i.4xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	m7i.8xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	m7i.12xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	m7i.16xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	m7i.24xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	m7i.48xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	m7i-flex.xlarge	emr-4.9.2, emr-5.5.3, emr-6.0.0, emr-7.0.0
	m7i-flex.2xlarge	emr-4.9.2, emr-5.5.3, emr-6.0.0, emr-7.0.0
	m7i-flex.4xlarge	emr-4.9.2, emr-5.5.3, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7i-flex.8xlarge	emr-4.9.2, emr-5.5.3, emr-6.0.0, emr-7.0.0
コンピューティング最適化	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7i.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c7i.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7i.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7i.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7i.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7i.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7i.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7i.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
高速コンピューティング	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
メモリを最適化	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r7i.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
ストレージの最適化	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	im4gn.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	is4gen.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

欧州 (ストックホルム) eu-north-1

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
汎用	m5.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5.24xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7a.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m7i.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m7i.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m7i.8xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m7i.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m7i.16xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m7i.24xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7i.48xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m7i-flex.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m7i-flex.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m7i-flex.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m7i-flex.8xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
コンピューティング最適化	c5.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5d.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5d.18xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c5d.24xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7a.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7a.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7a.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7a.8xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7a.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7a.16xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7a.24xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c7a.32xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7a.48xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7i.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7i.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7i.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7i.8xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7i.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7i.16xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7i.24xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7i.48xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
高速コンピューティング	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	p5.48xlarge	emr-6.14.0, emr-7.0.0
メモリを最適化	r5.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r5b.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5b.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5d.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7a.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r7a.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r7a.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r7a.8xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r7a.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r7a.16xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r7a.24xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r7a.32xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r7a.48xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r7i.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r7i.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7i.8xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r7i.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r7i.16xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r7i.24xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r7i.48xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
ストレージの最適化	i3.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

中東 (バーレーン) – me-south-1

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
汎用	m5.xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5d.12xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
コンピューティング最適化	c5.xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5ad.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5d.xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
高速コンピューティング	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
メモリを最適化	r5.xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5.4xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	r5d.xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
ストレージの最適化	i3.xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

中東 (UAE): me-central-1

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
汎用	m5.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5d.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6gd.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.32xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
コンピューティング最適化	c5.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.9xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.18xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.9xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.18xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5d.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
高速コンピューティング	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
メモリを最適化	r5.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5d.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6i.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.32xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
ストレージの最適化	i3.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i3en.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.3xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.6xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

南米 (サンパウロ) sa-east-1

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
汎用	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5zn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.3xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m5zn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.2xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	m7i-flex.4xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.8xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
コンピューティング最適化	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5ad.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7i.xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	c7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
高速コンピューティング	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
メモリを最適化	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5b.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	r7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1e.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
ストレージの最適化	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

インスタンスクラス	インスタンスタイプ	サポートされている Amazon EMR の最小バージョン (5.x、6.x、7.x)
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

旧世代のインスタンス

Amazon EMR は旧世代のインスタンスをサポートし、それらのインスタンスに最適化され、まだアップグレードされていないアプリケーションをサポートします。それらのインスタンスタイプとアップグレードパスの詳細については、「旧世代のインスタンス」を参照してください。

インスタンスクラス	インスタンスのタイプ
General Purpose	m1.small[1] | m1.medium[1] | m1.large[1] | m1.xlarge[1] | m3.xlarge[1] | m3.2xlarge[1] | m4.large | m4.xlarge | m4.2xlarge | m4.4xlarge | m4.10xlarge | m4.16xlarge
Compute Optimized	c1.medium[1] [2] | c1.xlarge[1] | c3.xlarge[1] | c3.2xlarge[1] | c3.4xlarge[1] | c3.8xlarge[1] | c4.large | c4.xlarge | c4.2xlarge | c4.4xlarge | c4.8xlarge
Memory Optimized	m2.xlarge[1] | m2.2xlarge[1] | m2.4xlarge[1] | r3.xlarge | r3.2xlarge | r3.4xlarge | r3.8xlarge | r4.xlarge | r4.2xlarge | r4.4xlarge | r4.8xlarge | r4.16xlarge
Storage Optimized	d2.xlarge | d2.2xlarge | d2.4xlarge | d2.8xlarge | i2.xlarge | i2.2xlarge | i2.4xlarge | i2.8xlarge

[1] 5.13.0 よりも前の Amazon EMR リリースバージョンで PVM 仮想化 AMI を使用します。詳細については、「Linux AMI 仮想化タイプ」を参照してください。

[2]リリースバージョン 5.15.0 ではサポートされていません。

インスタンス購入オプション

クラスターをセットアップするときに、Amazon EC2 インスタンスの購入オプションを選択します。オンデマンドインスタンス、スポットインスタンス、あるいはその両方を使用することを選択できます。料金はインスタンスタイプとリージョンによって異なります。Amazon EMR の料金は、Amazon EC2 の料金 (基盤となるサーバーの料金) と Amazon EBS の料金 (Amazon EBS ボリュームをアタッチする場合) に加算されます。現在の料金については、「Amazon EMR の料金」を参照してください。

クラスターのインスタンスグループまたはインスタンスフリートを使用する選択により、クラスターの実行中のインスタンス購入オプションの変更方法が決まります。ユニフォームインスタンスグループを選択する場合、インスタンスグループの作成時にその購入オプションのみを指定できます。また、インスタンスタイプと購入オプションは各インスタンスグループのすべての Amazon EC2 インスタンスに適用されます。インスタンスフリートを選択する場合、インスタンスフリートが作成された後に購入オプションを変更でき、指定するターゲット容量を満たすように購入オプションを組み合わせることができます。これらの構成の詳細については、「インスタンスフリートまたはユニフォームインスタンスグループでクラスターを作成する」を参照してください。

オンデマンドインスタンス

オンデマンドインスタンスは、秒単位で、コンピューティング性能に対して料金をお支払いいただくものです。オプションで、これらのオンデマンドインスタンスに、リザーブドインスタンスまたはハードウェア専有インスタンス購入オプションを使用できます。リザーブドインスタンスでは、1 つのインスタンスに対して 1 回だけの支払いを行って容量を予約します。ハードウェア専有インスタンスは、他の AWS アカウントに属するインスタンスからホストハードウェアレベルで物理的に分離されます。購入オプションの詳細については、「Amazon EC2 ユーザーガイド」の「インスタンス購入オプション」を参照してください。 Amazon EC2

予約インスタンスの使用

Amazon EMR でリザーブドインスタンスを使用するには、Amazon EC2 を使ってリザーブドインスタンスを購入し、リージョンまたはアベイラビリティーゾーンのいずれかに適用される予約の範囲を含む、予約のパラメータを指定します。詳細については、Amazon EC2 ユーザーガイド」の「Amazon EC2 リザーブドインスタンス」および「リザーブドインスタンスの購入」を参照し

てください。 Amazon EC2 リザーブドインスタンスを購入した後、次の条件をすべて満たす場合、Amazon EMR はクラスターの起動時にリザーブドインスタンスを使用します。

- オンデマンドインスタンスが、リザーブドインスタンスの指定に一致するクラスター設定で指定されている
- クラスターがインスタンス予約の範囲内 (アベイラビリティーゾーンまたはリージョン) で起動されている
- 予約インスタンス容量がまだ利用できる

例えば、インスタンス予約が米国東部リージョンにスコープ設定された `m5.xlarge` リザーブドインスタンスを購入したとします。その後、2 つの `m5.xlarge` インスタンスを使用する米国東部の Amazon EMR クラスターを起動します。最初のインスタンスは、予約インスタンスのレートで請求され、もう一方はオンデマンドのレートで請求されます。予約インスタンス容量は、オンデマンドインスタンスが作成される前に使用されます。

専用インスタンスの使用

ハードウェア専有インスタンスを使用するには、Amazon EC2 を使用してハードウェア専有インスタンスを購入し、[専有] テナンシー属性で VPC を作成します。Amazon EMR 内で、クラスターがこの VPC 内で起動することを指定します。ハードウェア専有インスタンスの指定に適合するクラスター内のすべてのオンデマンドインスタンスでは、クラスターの起動時に利用可能なハードウェア専有インスタンスを使用します。

> **Note**
>
> Amazon EMR は、個々のインスタンス上で `dedicated` 属性の設定をサポートしません。

スポットインスタンス

Amazon EMR 内のスポットインスタンスは、オンデマンドの購入と比較して、低コストで Amazon EC2 インスタンス容量を購入できるオプションを提供します。スポットインスタンスを使用するデメリットは、実行中のインスタンスタイプでスポット容量が使用できなくなると、インスタンスが終了する可能性があることです。アプリケーションでスポットインスタンスを使用することが適切な場合の詳細については、「スポットインスタンスを使用すべき場合」を参照してください。

Amazon EC2 に未使用の容量がある場合、スポット料金と呼ばれる割引料金で EC2 インスタンスが提供されます。この料金は、可用性と需要に基づいて変動し、リージョンとアベイラビリティーゾーンにより設定されます。スポットインスタンスを選択するときは、各 EC2 インスタンスタイプに支

払う最大スポット料金を選択します。クラスターのアベイラビリティーゾーン内のスポット価格が、そのインスタンスタイプに指定された最大スポット料金よりも低い場合、インスタンスが起動します。インスタンスが実行されている間、現在のスポット価格 (最大スポット料金ではない) で課金されます。

> **Note**
>
> 期間が定義されたスポットインスタンス (スポットブロックとも呼ばれます) は、2021 年 7 月 1 日以降の新規のお客様は、ご利用いただけません。既に、期間が指定されたスポットインスタンスを使用した経験をお持ちのお客様については、2022 年 12 月 31 日まで、この機能を引き続きサポートいたします。

最新の料金については、「Amazon EC2 スポットインスタンスの料金」を参照してください。詳細については、「Amazon EC2 ユーザーガイド」の「スポットインスタンス」を参照してください。クラスターを作成して構成するとき、クラスターが起動するアベイラビリティーゾーンを最終的に決定するネットワークオプションを指定します。詳細については、「ネットワークを設定する」を参照してください。

> **Tip**
>
> [詳細オプション] を使用してクラスターを作成するときは、[スポット] 購入オプションの横にある情報ツールヒントにマウスカーソルを移動すると、コンソールにリアルタイムのスポット料金が表示されます。選択したリージョンの各アベイラビリティーゾーンの料金が表示されます。最低価格は緑色の行に示されます。アベイラビリティーゾーン間でスポット価格が変動するため、最初の価格が最低のアベイラビリティーゾーンを選択すると、クラスターの寿命を通じて価格が最低になる可能性があります。最適な結果を得るには、選択する前にアベイラビリティーゾーンの履歴を調べてください。詳細については、「Amazon EC2 ユーザーガイド」の「スポットインスタンスの料金履歴」を参照してください。 Amazon EC2

スポットインスタンスのオプションは、クラスター構成でユニフォームインスタンスグループを使用しているか、インスタンスフリートを使用しているかによって異なります。

ユニフォームインスタンスグループ内のスポットインスタンス

ユニフォームインスタンスグループでスポットインスタンスを使用すると、インスタンスグループ内のすべてのインスタンスはスポットインスタンスでなければなりません。クラスターに 1 つのサブ

ネットまたはアベイラビリティーゾーンを指定します。インスタンスグループごとに、1 つのスポットインスタンスと最大スポット料金を指定します。クラスターのリージョンとアベイラビリティーゾーンでのスポット料金が最大スポット料金に満たない場合は、そのタイプのスポットインスタンスが起動します。スポット価格が最大スポット料金を上回ると、インスタンスは終了します。最大スポット料金は、インスタンスグループを構成する場合にのみ設定します。後で変更することはできません。詳細については、「インスタンスフリートまたはユニフォームインスタンスグループでクラスターを作成する」を参照してください。

インスタンスフリート内のスポットインスタンス

インスタンスフリート構成を使用するとき、追加のオプションにより、スポットインスタンスの起動と終了の方法をさらにコントロールできます。基本的に、インスタンスフリートはインスタンスを起動するためにユニフォームインスタンスグループとは異なる方法を使用します。その方法は、スポットインスタンス (およびオンデマンドインスタンス) 用にターゲット容量を設定することです。インスタンスタイプごとに加重容量を指定するか、加重容量としてインスタンスタイプの vCPU (YARN vcores) を使用できます。この加重容量は、そのタイプのインスタンスがプロビジョニングされるとき、ターゲット容量に加算されます。Amazon EMR は各ターゲットのターゲット容量が満たされるまで、両方の購入オプションを用いてインスタンスをプロビジョニングします。さらに、インスタンスの起動時に Amazon EMR が選択するアベイラビリティーゾーンの範囲を定義できます。プロビジョニングのタイムアウトを含め、フリートごとに追加のスポットオプションも指定します。詳細については、「インスタンスフリートを設定する」を参照してください。

インスタンスストレージ

概要

インスタンスストアおよび Amazon EBS ボリュームストレージは HDFS データに使用されます。また、バッファ、キャッシュ、スクラッチデータ、および一部のアプリケーションがローカルファイルシステムに「流出」する可能性があるその他の一時的なコンテンツにも使用されます。

Amazon EMR 内での Amazon EBS の機能は、通常の Amazon EC2 インスタンスと異なります。Amazon EMR クラスターにアタッチされた Amazon EBS ボリュームはエフェメラルです。これらのボリュームは、クラスターとインスタンスが終了すると (たとえば、インスタンスグループを縮小する場合などに) 削除されるため、データが永続的に存在するとはみなさないでください。データの存続はエフェメラルですが、クラスター内のノードの数と仕様によっては、HDFS 内のデータがレプリケートされることもあります。Amazon EBS ストレージボリュームを追加すると、これらは追加ボリュームとしてマウントされます。これらは起動ボリュームの一部ではありません。YARN は、すべての追加ボリュームを使用するように構成されますが、ローカルストレージとしての追加ボリュームの割り当て (たとえばローカルログファイルなど) はお客様の責任にて行ってください。

考慮事項

EMR クラスターで Amazon EBS を使用するときは、以下の追加の考慮事項に留意してください。

- Amazon EBS ボリュームのスナップショットを作成し、それを Amazon EMR 内で復元することはできません。再利用可能なカスタム設定を作成するには、カスタム AMI (Amazon EMR バージョン 5.7.0 以降で入手可能) を使用します。詳細については、「カスタム AMI の使用」を参照してください。
- 暗号化された Amazon EBS ルートデバイスボリュームは、カスタム AMI を使用するときにのみサポートされます。詳細については、「暗号化された Amazon EBS ルートデバイスボリュームを使用したカスタム AMI の作成」を参照してください。
- Amazon EMR API を使用してタグを適用する場合は、それらのオペレーションが EBS ボリュームに適用されます。
- インスタンスごとに 25 ボリュームという制限があります。
- コアノードの Amazon EBS ボリュームは 5 GB 未満にすることはできません。

インスタンスのデフォルト Amazon EBS ストレージ

EBS 専用のストレージを持つ EC2 インスタンスの場合、Amazon EMR は、Amazon EBS gp2 または gp3 ストレージボリュームをそのインスタンスに割り当てます。Amazon EMR リリース 5.22.0 以降を使用してクラスターを作成する場合、デフォルトの Amazon EBS ストレージ容量はインスタンスのサイズに基づいて増加します。

増えたストレージは複数のボリュームに分割されます。これにより、IOPS のパフォーマンスだけでなく、一部の標準ワークロードのパフォーマンスも向上します。別の Amazon EBS インスタンスストレージ設定を使用する場合、EMR クラスターを作成する際、または既存のクラスターにノードを追加する際にこれを指定することができます。Amazon EBS gp2 または gp3 ボリュームはルートボリュームとして使用でき、また gp2 または gp3 ボリュームを追加のボリュームとして追加することもできます。詳細については、「追加の EBS ストレージボリュームを指定する」を参照してください。

次の表は、Amazon EBS gp2 ストレージボリュームのデフォルト数、サイズ、およびインスタンスタイプごとの合計サイズを示しています。gp2 ボリュームと gp3 ボリュームの比較については、「Amazon EBS ボリュームタイプ gp2 と gp3 の比較」を参照してください。

Amazon EMR 5.22.0 以降のインスタンスタイプ別のデフォルトの Amazon EBS gp2 ストレージボリュームとサイズについて

インスタンスサイズ	ボリューム数	ボリュームサイズ (GiB)	合計サイズ (GiB)
*.large	1	32	32
*.xlarge	2	32	64
*.2xlarge	4	32	128
*.4xlarge	4	64	256
*.8xlarge	4	128	512
*.9xlarge	4	144	576
*.10xlarge	4	160	640
*.12xlarge	4	192	768
*.16xlarge	4	256	1024
*.18xlarge	4	288	1152
*.24xlarge	4	384	1536

インスタンスのデフォルトの Amazon EBS ルートボリューム

Amazon EMR リリース 6.15 以降では、Amazon EMR が Amazon EBS 汎用 SSD (gp3) を AMI のルートデバイスとして自動的にアタッチし、パフォーマンスを強化します。それ以前のリリースでは、Amazon EMR は、ルートデバイスとして EBS 汎用 SSD (gp2) をアタッチします。

	6.15 以降	6.14 以前
デフォルトのルートボリュームタイプ		
デフォルトサイズ		

	6.15 以降	6.14 以前
デフォルト IOPS		
デフォルトのスループット		

Amazon EBS ルートデバイスボリュームをカスタマイズする方法については、「追加の EBS ストレージボリュームを指定する」を参照してください。

追加の EBS ストレージボリュームを指定する

Amazon EMR でインスタンスタイプを設定するとき、追加の EBS ボリュームを指定して、インスタンスストア (存在する場合) とデフォルト EBS ボリュームを超える容量を追加できます。Amazon EBS には、汎用 SSD、プロビジョンド IOPS (SSD)、スループット最適化 (HDD)、Cold (HDD)、磁気のボリュームタイプが用意されています。これらはパフォーマンス特性と料金が異なるため、お使いのアプリケーションの分析ニーズとビジネスニーズに応じてストレージを調整してください。たとえば、一部のアプリケーションはディスクへの書き込みが必要になる場合がありますが、メモリ内でまたは Amazon S3 を使用して安全に動作できるアプリケーションもあります。

Amazon EBS ボリュームをインスタンスにアタッチできるのは、クラスターの起動時と、別のタスクノードインスタンスグループを追加するときです。Amazon EMR クラスター内のインスタンスに障害が発生した場合は、インスタンスおよびアタッチされている Amazon EBS ボリュームの両方が、新しいボリュームに置き換えられます。結果として、手動で Amazon EBS ボリュームをデタッチする場合、Amazon EMR はそれを失敗として扱い、インスタンスストレージ (該当する場合) とボリュームストアの両方を置き換えます。

Amazon EMR では、既存の EMR クラスターのボリュームタイプを gp2 から gp3 に変更することはできません。ワークロードに gp3 を使用するには、新しい EMR クラスターを起動する必要があります。また、使用中またはプロビジョニング中のクラスターのスループットと IOPS を更新することはお勧めしません。Amazon EMR は、クラスターのスケールアップ中に追加される新しいインスタンスに対して、クラスター起動時に指定したスループットと IOPS の値を使用するためです。詳細については、「Amazon EBS ボリュームタイプ gp2 と gp3 の比較」および「gp3 への移行時の IOPS とスループットの選択」を参照してください。

> **Important**
>
> EMR クラスターで gp3 ボリュームを使用するには、新しい EMR クラスターを起動する必要があります。

Amazon EBS ボリュームタイプ gp2 と gp3 の比較

こちらは、米国東部 (バージニア北部) リージョンの gp2 ボリュームと gp3 ボリュームのコストを比較したものです。最新情報については、「Amazon EBS 汎用ボリューム」製品ページと「Amazon EBS 料金ページ」を参照してください。

ボリュームタイプ	gp3	gp2
ボリュームサイズ	1 GiB – 16 TiB	1 GiB – 16 TiB
デフォルト/ベースライン IOPS	3000	3 IOPS/GiB (最小 100 IOPS) から最大 16,000 IOPS。1 TiB 未満のボリュームでも、最大 3,000 IOPS までバーストできます。
最大 IOPS/ボリューム	16,000	16,000
デフォルト/ベースラインスループット	125 MiB/秒	スループットの制限は、ボリュームサイズに応じて 128 MiB/秒〜250 MiB/秒です。
最大スループット/ボリューム	1,000 MiB/秒	250 MiB/秒
価格	月あたり 0.08 USD/GiB、3,000 IOPS 無料および 3,000 以上月あたり 0.005 USD/プロビジョンド IOPS。125 MiB/秒無料および 125 MiB/秒以上月あたり 0.04 USD/プロビジョンド MiB/秒	月あたり 0.10 USD/GiB

gp3 への移行時の IOPS とスループットの選択

gp2 ボリュームをプロビジョニングするときは、比例する IOPS およびスループットを得るために、ボリュームのサイズを把握する必要があります。gp3 では、より高いパフォーマンスを得るために大きなボリュームをプロビジョニングする必要はありません。アプリケーションのニーズに応じて、希望のサイズとパフォーマンスを選択できます。適切なサイズと適切なパフォーマンスパラメータ (IOPS、スループット) を選択することで、パフォーマンスに影響を与えずに最大限のコスト削減を実現できます。

gp3 設定オプションの選択に役立つ表を以下に示します。

ボリュームサイズ	IOPS	スループット
1 ~ 170 GiB	3000	125 MiB/秒
170 ~ 334 GiB	3000	選択した EC2 インスタンスタイプが 125 MiB/秒以下をサポートしている場合は 125 MiB/秒、使用量に応じてより高い値を使用し、最大 250 MiB/秒*。
334 ~ 1000 GiB	3000	選択した EC2 インスタンスタイプが 125 MiB/秒以下をサポートしている場合は 125 MiB/秒、使用量に応じてより高い値を使用し、最大 250 MiB/秒*。
1000+ GiB	gp2 IOPS (GiB 単位のサイズ x 3) または現在の gp2 ボリュームによって決定される最大 IOPS に一致させる	選択した EC2 インスタンスタイプが 125 MiB/秒以下をサポートしている場合は 125 MiB/秒、使用量に応じてより高い値を使用し、最大 250 MiB/秒*。

*Gp3 には、最大 1000 MiB/秒のスループットを提供する機能があります。gp2 は最大 250 MiB/秒のスループットを提供するため、gp3 を使用する場合はこの制限を超える必要はないかもしれません。

ネットワークを設定する

ほとんどのクラスターは、Amazon Virtual Private Cloud (Amazon VPC) を使用して仮想ネットワークで起動します。VPC は 内の分離された仮想ネットワークで AWS 、アカウント内で論理的に分離されます AWS 。プライベート IP アドレス範囲、サブネット、ルーティングテーブル、ネットワークゲートウェイなどの側面を設定できます。詳細については、Amazon VPC ユーザーガイドを参照してください。

VPCは、次の機能を提供します。

- 機密データを処理する

  VPC 内でクラスターを起動することは、ルーティングテーブルや、ネットワークにアクセスできるユーザーを定義するネットワーク ACL などの追加的なツールを備えたプライベートネットワーク内でクラスターを起動することに似ています。クラスターで機密データを処理する場合は、VPC 内でクラスターを起動することで得られる追加のアクセスコントロールが必要になる可能性があります。さらに、これらのリソースのいずれにも直接的なインターネット接続を持たないプライベートサブネット内でリソースを起動することもできます。

- 内部ネットワーク上のリソースにアクセスする

  データソースがプライベートネットワークにある場合、転送するデータ量またはデータの機密性のために、Amazon EMR に AWS インポートするためにそのデータを にアップロードすることは実用的ではない、または望ましくない場合があります。代わりにクラスターを VPC 内で起動し、VPN 接続によってデータセンターと VPC を接続すると、クラスターは内部ネットワーク上のリソースにアクセスすることができます。たとえば、データセンターに Oracle データベースがある場合、クラスターを VPN でネットワークに接続した VPC 内で起動すると、クラスターはその Oracle データベースにアクセスできます。

パブリックサブネットおよびプライベートサブネット

パブリックおよびプライベート VPC サブネットの両方で Amazon EMR クラスターを起動できます。つまり、Amazon EMR クラスターを実行するためにインターネット接続は必要ありません。ただし、VPC の外部にあるサービスやリソースにアクセスするように、例えば会社のエクストラナインや などのパブリック AWS サービスエンドポイントでネットワークアドレス変換 (NAT) と VPN ゲートウェイを設定する必要がある場合があります AWS Key Management Service。

> **Important**
>
> プライベートサブネットでのクラスターの起動をサポートするのは、Amazon EMR リリースバージョン 4.2 以降のみです。

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

トピック

- Amazon VPC オプション
- VPC をホストクラスターにセットアップする
- VPC でクラスターを起動する
- プライベートサブネット用の Simple Storage Service (Amazon S3) の最小ポリシー
- VPC に関するその他のリソース

Amazon VPC オプション

VPC 内で Amazon EMR クラスターを起動するときは、パブリックサブネット、プライベートサブネット、または共有サブネット内で起動できます。クラスターに選択するサブネットタイプに応じて、構成に多少の顕著な相違があります。

パブリックサブネット

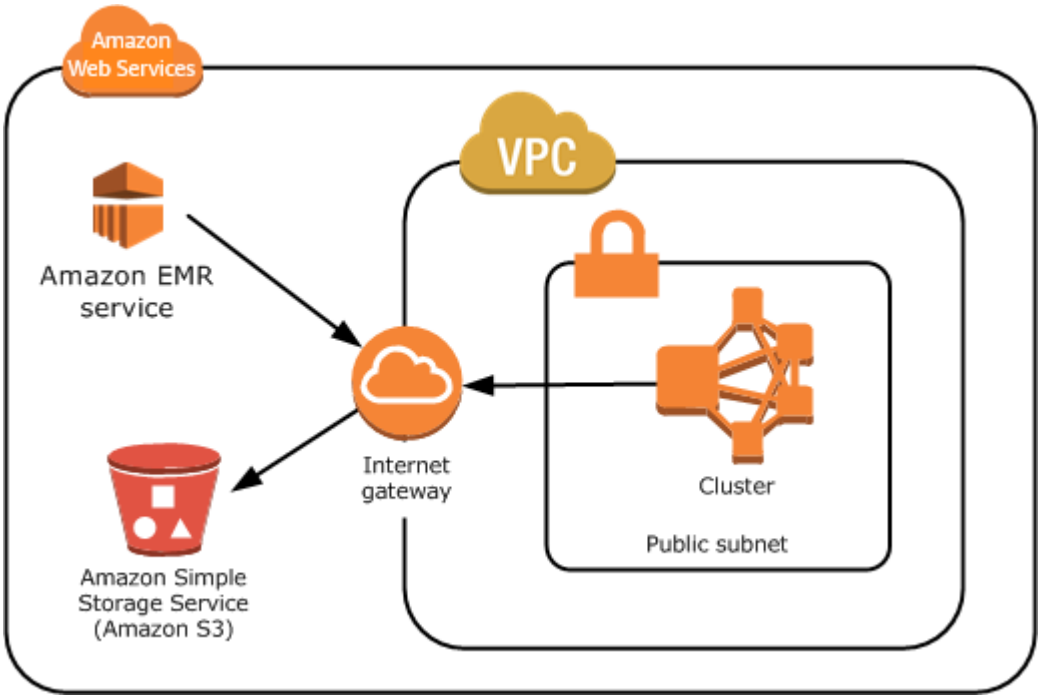
パブリックサブネットの EMR クラスターでは、接続されているインターネットゲートウェイが必要です。これは、Amazon EMR クラスターが AWS サービスと Amazon EMR にアクセスする必要があるためです。Simple Storage Service (Amazon S3) などのサービスが、VPC エンドポイントを作成する機能を提供している場合、インターネットゲートウェイを通してパブリックエンドポイントにアクセスする代わりに、エンドポイントを使用してこれらのサービスにアクセスできます。また、Amazon EMR はネットワークアドレス変換 (NAT) デバイスを通じてパブリックサブネットのクラスターと通信することはできません。この目的にはインターネットゲートウェイが必要ですが、より複雑なシナリオでは、他のトラフィックに対して NAT インスタンスまたはゲートウェイを引き続き使用できます。

クラスター内のインスタンスはいずれも、VPC エンドポイントまたはインターネットゲートウェイを通じて Simple Storage Service (Amazon S3) に接続します。現在 VPC エンドポイントをサポートしていない他の AWS サービスは、インターネットゲートウェイのみを使用します。

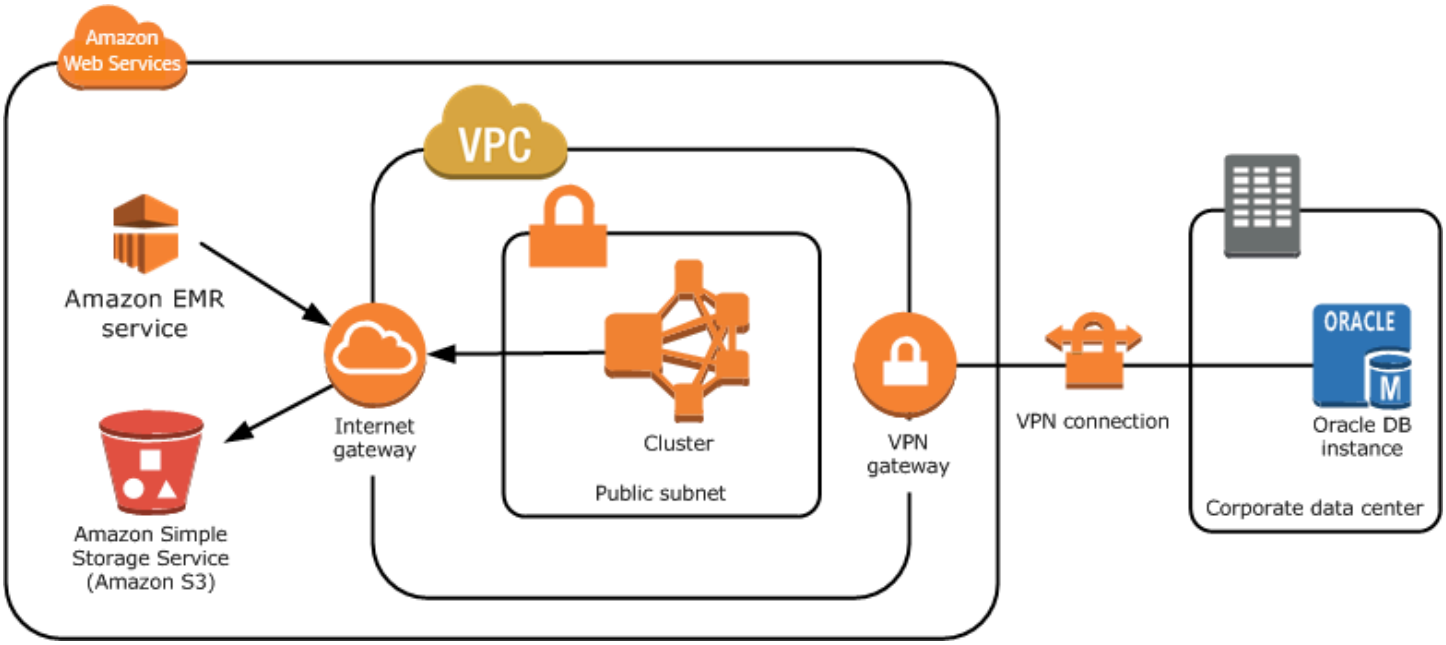
インターネットゲートウェイに接続したくない追加の AWS リソースがある場合は、VPC 内に作成したプライベートサブネットでそれらのコンポーネントを起動できます。

パブリックサブネットで実行中のクラスターは 2 つのセキュリティグループを使用します。1 つはプライマリノード用、もう 1 つはコアノードとタスクノード用です。詳細については、「セキュリティグループを使用してネットワークトラフィックを制御する」を参照してください。

次の図は、パブリックサブネットを使用して Amazon EMR クラスターが VPC でどのように実行されるかを示しています。クラスターは、インターネットゲートウェイを介して Amazon S3 バケットなどの他の AWS リソースに接続できます。



次の図は、VPC のクラスターが Oracle データベースなど、ご自身のネットワーク上のリソースにアクセスできるように VPC をセットアップする方法を示します。



プライベートサブネット

プライベートサブネットを使用すると、サブネットにインターネットゲートウェイをアタッチしなくても AWS リソースを起動できます。プライベートサブネットでのクラスターの起動をサポートするのは、Amazon EMR リリースバージョン 4.2.0 以降のみです。

> **Note**
>
> プライベートサブネットに Amazon EMR クラスターを設定するときは、Amazon S3 の VPC エンドポイントも設定することをお勧めします。EMR クラスターが Amazon S3 の VPC エンドポイントのないプライベートサブネットに存在する場合、EMR クラスターと S3 間のトラフィックは VPC 内にとどまらないため、S3 トラフィックに関連する追加の NAT ゲートウェイ料金が発生します。

プライベートサブネットとパブリックサブネットは、次の点で異なります。

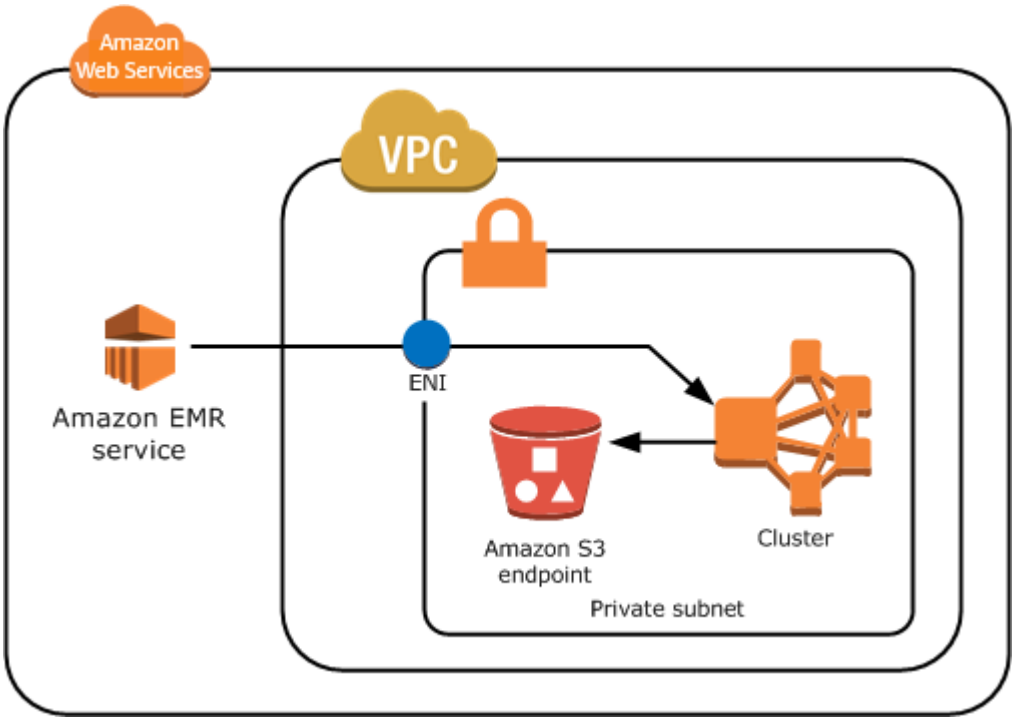
- VPC エンドポイントを提供しない AWS サービスにアクセスするには、NAT インスタンスまたはインターネットゲートウェイを使用する必要があります。
- 少なくとも、Amazon EMR サービスログバケットと Simple Storage Service (Amazon S3) の Amazon Linux リポジトリへのルートを提供する必要があります。詳細については、「プライベートサブネット用の Simple Storage Service (Amazon S3) の最小ポリシー」を参照してください。

- EMRFS 機能を使用する場合、Simple Storage Service (Amazon S3) VPC エンドポイントとプライベートサブネットから DynamoDB へのルートが必要です。
- デバッグが機能するのは、プライベートサブネットからパブリック Amazon SQS エンドポイントへのルートを提供する場合のみです。
- パブリックサブネットで NAT インスタンスまたはゲートウェイを使用したプライベートサブネット設定の作成がサポートされるのは、 AWS Management Consoleを使用する場合のみです。Amazon EMR クラスターの NAT インスタンスと Simple Storage Service (Amazon S3) VPC エンドポイントを追加および設定する最も簡単な方法は、Amazon EMR コンソールの [VPC Subnets List] (VPC サブネットリスト) ページを使用することです。NAT ゲートウェイを作成するには、「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。
- 既存の Amazon EMR クラスターがあるサブネットを、パブリックからプライベートに、またはその逆に変更することはできません。プライベートサブネット内で Amazon EMR クラスターを見つけるため、クラスターはそのプライベートサブネットで起動する必要があります。

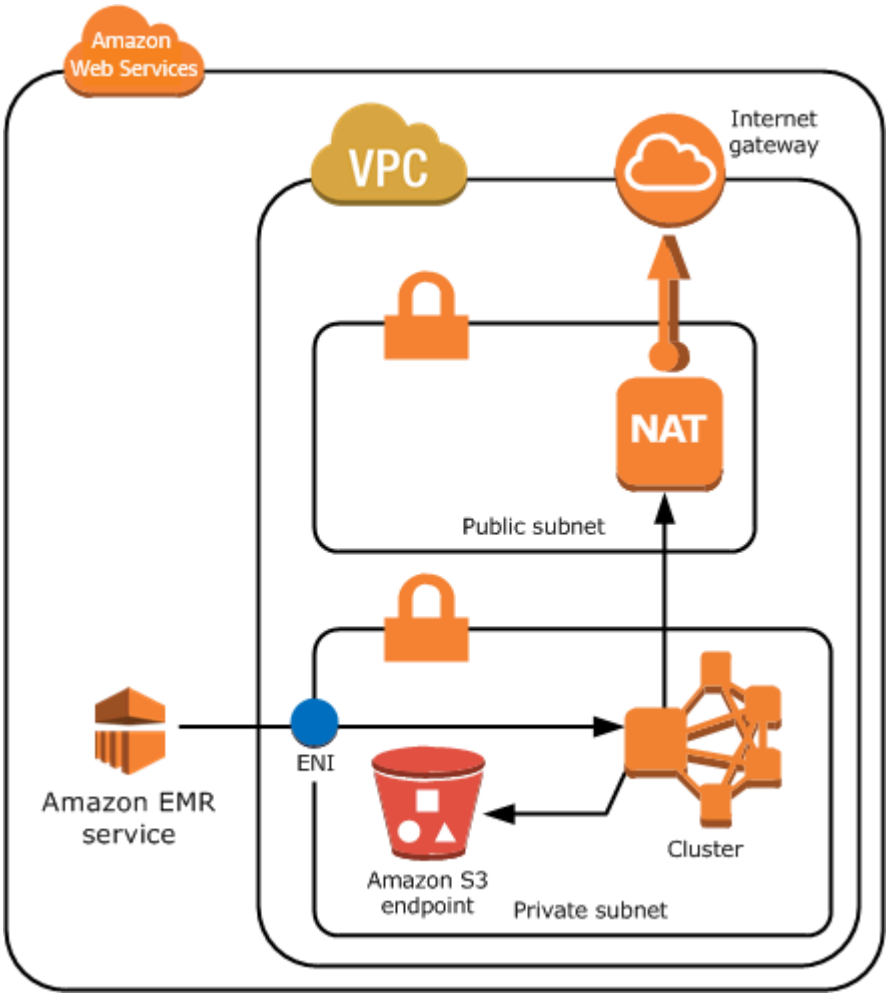
Amazon EMR は、プライベートサブネット内のクラスターに対して、-Master ElasticMapReduce-Private、-SlaveElasticMapReduce-Private、および ElasticMapReduce- の異なるデフォルトセキュリティグループを作成して使用しますServiceAccess。詳細については、「セキュリティグループを使用してネットワークトラフィックを制御する」を参照してください。

クラスターの NACL の完全なリストについては、Amazon EMR コンソールの [クラスターの詳細] ページで [プライマリのセキュリティグループ] と [コアおよびタスクのセキュリティグループ] を選択します。

次の図は、プライベートサブネット内で Amazon EMR クラスターが設定される方法を示しています。サブネット外の唯一の通信は、Amazon EMR に対するものです。



次の図は、パブリックサブネットにある NAT インスタンスに接続されているプライベートサブネット内の Amazon EMR クラスターの設定例を示しています。



共有サブネット

VPC 共有により、お客様は同じ AWS Organization 内の他の AWS アカウントとサブネットを共有できます。次の注意点に注意しながら、Amazon EMR クラスターをパブリック共有サブネットとプライベート共有サブネットの両方で起動できます。

Amazon EMR クラスターをサブネットで起動するには、サブネットの所有者とサブネットを共有している必要があります。ただし、共有サブネットは後で共有解除することができます。詳細については、「共有 VPC の使用」を参照してください。クラスターを共有サブネットで起動した後に、共有サブネットが共有解除された場合は、サブネットの共有解除時に、Amazon EMR クラスターの状態に基づいて、特定の動作が発生します。

- クラスターが正常に起動する前にサブネットが共有解除された場合 - 参加者がクラスターを起動しようとしている最中に、所有者が Amazon VPC またはサブネットの共有を停止した場合は、クラスターの起動に失敗するか、リクエストされたすべてのインスタンスがプロビジョニングされないままに、クラスターが部分的に初期化されることがあります。

- クラスターが正常に起動した後にサブネットが共有解除された場合 - 所有者が参加者とのサブネットまたは Amazon VPC の共有を停止した場合、参加者のクラスターは、新しいインスタンスの追加や、異常なインスタンスを交換するためにサイズ変更できなくなります。

Amazon EMR クラスターを起動すると、複数のセキュリティグループが作成されます。共有サブネットでは、サブネットの参加者がこれらのセキュリティグループを制御します。サブネットの所有者は、これらのセキュリティグループを表示できますが、これらのグループに対してアクションを実行することはできません。サブネットの所有者がセキュリティグループの削除または変更を希望する場合は、セキュリティグループを作成した参加者がそのアクションを実行する必要があります。

IAM で VPC のアクセス許可を制御する

デフォルトでは、すべてのユーザーはそのアカウントのすべてのサブネットを表示でき、どのサブネットでもクラスターを起動できます。

VPC でクラスターを起動する場合、Amazon EC2 Classic で起動したクラスターと同様に、 AWS Identity and Access Management (IAM) を使用してクラスターへのアクセスを制御し、 ポリシーを使用してアクションを制限できます。IAM の詳細については、「IAM ユーザーガイド」を参照してください。

また、IAM を使用すると、サブネットを作成および管理できるユーザーを制御することもできます。例えば、サブネットを管理するためのアカウントを 1 つ作成し、クラスターを起動することはできても Amazon VPC の設定を変更できないアカウントを 1 つ作成できます。Amazon EC2 および Amazon VPC でのポリシーとアクションの管理の詳細については、「Amazon EC2 ユーザーガイド Amazon EC2」の「Amazon EC2 の IAM ポリシーAmazon EC2」を参照してください。

VPC をホストクラスターにセットアップする

VPC でクラスターを起動する前に、VPC とサブネットを作成する必要があります。パブリックサブネットでは、インターネットゲートウェイを作成し、それをサブネットにアタッチする必要があります。以下の手順は、Amazon EMR クラスターをホストできる VPC を作成する方法を示しています。

Amazon EMR クラスター用のサブネットを持つ VPC 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. ページの右上で、VPC の [AWS リージョン] を選択します。
3. [Create VPC (VPC の作成)] を選択します。

4. [VPC の設定] ページで、[VPC など] を選択します。
5. [名前タグの自動生成] で [自動生成] を有効にし、VPC の名前を入力します。これは、VPC およびサブネットを作成後に Amazon VPC コンソールにおいてそれらを識別するのに役立ちます。
6. [IPv4 CIDR ブロック] フィールドに、VPC のプライベート IP アドレススペースを入力して、適切な DNS ホスト名に解決されるようにします。適切な DNS ホスト名に解決されなければ、Amazon EMR クラスターに障害が発生する場合があります。これには次の IP アドレス範囲が含まれています。

   - 10.0.0.0 – 10.255.255.255
   - 172.16.0.0 – 172.31.255.255
   - 192.168.0.0 – 192.168.255.255

7. Number of Availability Zones (AZs) (アベイラビリティーゾーンの数 (AZ)) で、サブネットを起動するアベイラビリティーゾーンの数を選択します。
8. [パブリックサブネットの数] で、VPC に追加するパブリックサブネットの数を選択します。クラスターで使用するデータがインターネットで利用可能な場合は (例えば、Amazon S3 や Amazon RDS 内)、パブリックサブネットを使用する必要があるだけで、プライベートサブネットを追加する必要はありません。
9. Number of private subnets (プライベートサブネットの数) で、VPC に追加するプライベートサブネットの数を選択します。アプリケーションのデータがご自身のネットワークに保存されている場合は (例えば、Oracle データベース内)、1 つまたは複数選択します。プライベートサブネットでの VPC の場合、すべての Amazon EC2 インスタンスは少なくとも Elastic Network Interface を介して Amazon EMR へのルートを持っている必要があります。コンソールでは、これは自動的に設定されます。
10. [NAT ゲートウェイ] で、オプションで NAT ゲートウェイを追加することを選択します。インターネットと通信する必要があるプライベートサブネットがある場合にのみ必要です。
11. オプションで、[VPC エンドポイント] で、Simple Storage Service (Amazon S3) のエンドポイントを自身のサブネットに追加することを選択します。
12. [DNS ホスト名を有効化] と [DNS 解決を有効化] がオンになっていることを確認します。詳細については、「Using DNS with Your VPC」を参照してください。
13. [Create VPC (VPC の作成)] を選択します。
14. ステータスウィンドウに、作業の進行状況が表示されます。作業が完了したら、[VPC を表示] を選択して [お使いの VPC] ページに移動します。ここにはデフォルトの VPC と作成したばかりの VPC が表示されます。デフォルト以外の VPC を作成した場合には、[Default VPC] 列に [No] と表示されます。

15. VPC をドメイン名を含まない DNS エントリと関連付ける場合は、[DHCP オプションセット] に移動し、[DHCP オプションセットの作成] を選択し、ドメイン名は省略します。オプションセットを作成したら、新しい VPC に移動し、[アクション] メニューの [DHCP オプションセットの編集] を選択し、新しいオプションセットを選択します。DNS オプションセットを作成した後にコンソールを使用してドメイン名を編集することはできません。

    Hadoop と関連アプリケーションでは、ノードの完全修飾ドメイン名 (FQDN) を解決できるようにすることを推奨します。DNS を適切に解決するには、パラメータが以下の値に設定されている DHCP オプション設定を含む VPC を設定します。

    - domain-name = **`ec2.internal`**

      リージョンが米国東部 (バージニア北部) の場合は、**`ec2.internal`** を使用します。その他のリージョンについては、*`region-name`*`.compute.internal` を使用します。`us-west-2` での例では、**`us-west-2.compute.internal`** を使用します。 AWS GovCloud （米国西部) リージョンの場合は、 を使用します**`us-gov-west-1.compute.internal`**。
    - domain-name-servers = **`AmazonProvidedDNS`**

    詳細については、「Amazon VPC ユーザーガイド」の「DHCP オプションセット」を参照してください。
16. VPC を作成した後は、[サブネット] ページに移動して、新しい VPC のサブネットの 1 つの [サブネット ID] を書き留めてください。VPC で Amazon EMR クラスターを起動するときにこの情報を使用してください。

VPC でクラスターを起動する

Amazon EMR クラスターをホストするように設定されたサブネットが作成されたら、クラスターの作成時に関連するサブネット ID を指定して、そのサブネットでクラスターを起動します。

> **Note**
>
> Amazon EMR は、リリースバージョン 4.2 以降のプライベートサブネットをサポートします。

クラスターが起動された場合、Amazon EMR はクラスターが VPC プライベートサブネットまたはパブリックサブネット内で起動しているかどうかに基づいて、セキュリティグループを追加しま

す。すべてのセキュリティグループでは、Amazon EMR サービスとの通信にポート 8443 での受信を許可しますが、IP アドレスの範囲はパブリックサブネットとプライベートサブネットで異なります。Amazon EMR はこれらのセキュリティグループをすべて管理し、時間の経過とともに AWS 範囲に IP アドレスを追加する必要がある場合があります。詳細については、「セキュリティグループを使用してネットワークトラフィックを制御する」を参照してください。

VPC 上のクラスターを管理するために、Amazon EMR はネットワークデバイスをプライマリノードにアタッチし、このデバイスを通じて管理を行います。このデバイスは、Amazon EC2 API アクション `DescribeInstances` を使用して確認できます。このデバイスをいずれかの方法で変更すると、クラスターに障害が発生する可能性があります。

> **Note**
>
> Amazon EMR コンソールは、再設計され、使いやすくなりました。古いコンソールと新しいコンソールエクスペリエンスの違いについては、「Amazon EMR コンソール」を参照してください。

New console

新しいコンソールを使用して VPC 内でクラスターを起動するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で、[クラスター] を選択し、[クラスターの作成] を選択します
3. [ネットワーク] の [仮想プライベートクラウド (VPC)] フィールドに移動します。VPC の名前を入力するか、[参照] を選択して VPC を選択します。または [VPC の作成] を選択して、クラスターに使用できる VPC を作成します。
4. クラスターに適用するその他のオプションを選択します。
5. クラスターを起動するには、[クラスターの作成] を選択します。

Old console

古いコンソールを使用して VPC 内でクラスターを起動するには

1. 新しい Amazon EMR コンソールに移動し、サイドナビゲーションから [古いコンソールに切り替え] を選択します。古いコンソールに切り替えたときの動作の詳細については、「Using the old console」を参照してください。
2. [クラスターを作成] を選択します。
3. [Go to advanced options] を選択します。
4. [Hardware Configuration (ハードウェア構成)] セクションの [Network (ネットワーク)] で、以前に作成した VPC ネットワークの ID を選択します。
5. [EC2 Subnet (EC2 サブネット)] で、以前に作成したサブネットの ID を選択します。
   a. プライベートサブネットが、NAT インスタンスと S3 エンドポイントオプションで正しく構成されている場合、サブネット名と識別子の上に (EMR 対応) と表示されます。
   b. プライベートサブネットに NAT インスタンスまたは S3 エンドポイント (またはその両方) がない場合、[Add S3 endpoint and NAT instance (S3 エンドポイントおよび NAT インスタンスの追加)]、[Add S3 endpoint (S3 エンドポイントの追加)]、または [Add NAT instance (NAT インスタンスの追加)] を選択してこれを設定できます。NAT インスタンスおよび S3 エンドポイント用に必要なオプションを選択し、[設定] を選択します。

   > **Important**
   >
   > Amazon EMR から NAT インスタンスを作成するには、ec2:CreateRoute、、`ec2:RevokeSecurityGroupEgress`、`ec2:AuthorizeSecu`よび アクセス`cloudformation:CreateStack`許可が必要です。

   > **Note**
   >
   > NAT デバイスに対して Amazon EC2 インスタンスを起動すると、追加のコストが発生します。
6. クラスターの作成に進みます。

AWS CLI

を使用して VPC でクラスターを起動するには AWS CLI

> **Note**
>
> AWS CLI では、NAT インスタンスを自動的に作成してプライベートサブネットに接続する方法はありません。ただし、サブネットで S3 エンドポイントを作成するには、Amazon VPC CLI コマンドを使用できます。コンソールを使用して NAT インスタンスを作成し、プライベートサブネットでクラスターを起動します。

VPC の設定が完了したら、`create-cluster` サブコマンドと `--ec2-attributes` パラメータを使用して、VPC 内で Amazon EMR クラスターを起動できます。クラスターに VPC サブネットを指定するには、`--ec2-attributes` パラメータを使用します。

- 特定のサブネットにクラスターを作成するには、次のコマンドを入力し、*myKey* を Amazon EC2 キーペアの名前に置き換え、*77XXXX03* をサブネット ID に置き換えます。

```
aws emr create-cluster --name "Test cluster" --release-label emr-4.2.0 --
applications Name=Hadoop Name=Hive Name=Pig --use-default-roles --ec2-attributes
 KeyName=myKey,SubnetId=subnet-77XXXX03 --instance-type m5.xlarge --instance-
count 3
```

`--instance-groups` パラメータを使用せずにインスタンス数を指定すると、1 つのプライマリノードが起動され、残りのインスタンスはコアノードとして起動されます。すべてのノードで、コマンドで指定したインスタンスタイプが使用されます。

> **Note**
>
> 以前にデフォルトの Amazon EMR サービスロールと EC2 インスタンスプロファイルを作成していない場合は、「`aws emr create-default-roles`」と入力してそれらを作成してから、`create-cluster` サブコマンドを入力します。

プライベートサブネット用の Simple Storage Service (Amazon S3) の最小ポリシー

プライベートサブネットの場合、少なくとも Amazon Linux リポジトリにアクセスするための機能を Amazon EMR に提供する必要があります。このプライベートサブネットポリシーは、Simple

Storage Service (Amazon S3) にアクセスするための VPC エンドポイントポリシーの一部です。Amazon EMR 5.25.0 以降で永続 Spark 履歴サーバーへのワンクリックアクセスを有効にするには、Spark イベントログを収集するシステムバケットへのアクセスを Amazon EMR に許可する必要があります。ログ記録を有効にする場合は、PUT アクセス許可を aws157-logs-* バケットに提供します。詳細については、「永続 Spark 履歴サーバーへのワンクリックアクセス」を参照してください。

ビジネスニーズに合ったポリシー制限は、お客様が決定します。例えば、リージョン packages.us-east-1.amazonaws.com を指定して、あいまいな Simple Storage Service (Amazon S3) バケット名を避けることができます。以下のポリシー例では、Spark イベントログを収集するための Amazon Linux リポジトリおよび Amazon EMR システムバケットにアクセスするためのアクセス許可を提供します。をログバケットが存在するリージョン*MyRegion*に置き換えますus-east-1。例えば、。

Amazon VPC エンドポイントでの IAM ポリシーの使用の詳細については、「Amazon S3 のエンドポイントポリシー」を参照してください。

```
{
   "Version": "2008-10-17",
   "Statement": [
       {
           "Sid": "AmazonLinuxAMIRepositoryAccess",
           "Effect": "Allow",
           "Principal": "*",
           "Action": "s3:GetObject",
           "Resource": [
            "arn:aws:s3:::packages.MyRegion.amazonaws.com/*",
            "arn:aws:s3:::repo.MyRegion.amazonaws.com/*",
            "arn:aws:s3:::repo.MyRegion.emr.amazonaws.com/*"
           ]
       },
       {
           "Sid": "EnableApplicationHistory",
           "Effect": "Allow",
           "Principal": "*",
           "Action": [
               "s3:Put*",
               "s3:Get*",
               "s3:Create*",
               "s3:Abort*",
               "s3:List*"
           ],
```

```
            "Resource": [
             "arn:aws:s3:::prod.MyRegion.appinfo.src/*"
            ]
        }
    ]
}
```

以下の例のポリシーは、Amazon Linux 2 リポジトリへのアクセスに必要なアクセス許可を提供します。デフォルトは Amazon Linux 2 AMI です。

```
{
    "Statement": [
        {
            "Sid": "AmazonLinux2AMIRepositoryAccess",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": [
             "arn:aws:s3:::amazonlinux.MyRegion.amazonaws.com/*",
             "arn:aws:s3:::amazonlinux-2-repos-MyRegion/*"
            ]
        }
    ]
}
```

VPC に関するその他のリソース

以下のトピックでは、VPC およびサブネットの詳細について説明します。

- VPC のプライベートサブネット
  - シナリオ 2: パブリックサブネットとプライベートサブネットを持つ VPC (NAT)
  - NAT インスタンス
  - Amazon VPC NAT インスタンスの高可用性: 例
- VPC のパブリックサブネット
  - シナリオ 1: 単一のパブリックサブネットを持つ VPC
- 一般的な VPC 情報
  - Amazon VPC User Guide
  - VPC ピアリング接続
  - VPC で Elastic Network Interface を使用する

- プライベート VPC で実行中の Linux インスタンスに安全に接続します

インスタンスフリートまたはユニフォームインスタンスグループでクラスターを作成する

クラスターを作成して、プライマリノード、コアノード、およびタスクノードの構成を指定するとき、2 つの構成オプションがあります。インスタンスフリートまたはユニフォームインスタンスグループ を使用できます。選択する構成オプションは、すべてのノードに適用されます。クラスターの使用期間にわたり適用され、インスタンスフリートとインスタンスグループはクラスター内で共存できません。インスタンスフリート設定は、5.0.xバージョンを除き、Amazon EMR のバージョン 4.8.0 以降で利用できます。

Amazon EMR コンソール、 AWS CLI、または Amazon EMR API を使用して、どちらの設定でもクラスターを作成できます。 AWS CLIから `create-cluster` コマンドを使用するとき、`--instance-fleets` パラメータでインスタンスフリートを使用してクラスターを作成するか、`--instance-groups` パラメータでユニフォームインスタンスグループを使用してクラスターを作成します。

Amazon EMR API を使用する場合も同じです。構成を使用して `InstanceGroups` オブジェクトの配列を指定するか、 `InstanceGroupConfig` 構成を使用して、 `InstanceFleets` オブジェクトの配列を指定します `InstanceFleetConfig` 。

新しい Amazon EMR コンソールでは、クラスターを作成するときにインスタンスグループまたはインスタンスフリートのいずれかを使用することを選択できます。また、それぞれにスポットインスタンスを使用するオプションもあります。古い Amazon EMR コンソールで、クラスターを作成するときにデフォルトの [クイックオプション] 設定を使用する場合、Amazon EMR は、ユニフォームインスタンスグループ設定をクラスターに適用し、オンデマンドインスタンスを使用します。ユニフォームインスタンスグループでスポットインスタンスを使用するか、インスタンスフリートと他のカスタマイゼーションを構成するには、[Advanced Options (詳細オプション)] を選択します。

インスタンスフリート

インスタンスフリート設定により、Amazon EC2 インスタンスのプロビジョニングオプションは非常に広範になります。各ノードタイプには 1 つのインスタンスフリートがあり、タスクインスタンスフリートの使用はオプションです。または Amazon EMR API とオンデマンドインスタンスとスポットインスタンスの配分戦略を使用してクラスターを作成する場合、フリートごとに最大 5 つの EC22 インスタンスタイプ、またはフリートごとに 30 の EC2 インスタンスタイプを指定できます。AWS CLI コアインスタンスフリートとタスクインスタンスフリートの場合、オンデマンドインスタ

ンスにターゲット容量を割り当て、スポットインスタンスには別の容量を割り当てます。Amazon EMR は、ターゲット容量を達成するために指定されたインスタンスタイプの組み合わせを選択し、オンデマンドインスタンスとスポットインスタンスの両方をプロビジョニングします。

プライマリノードタイプについては、Amazon EMR はインスタンスのリストから 1 つのインスタンスタイプを選択します。オンデマンドインスタンスまたはスポットインスタンスのどちらとしてプロビジョニングするかを指定してください。インスタンスフリートでも、スポットインスタンスおよびオンデマンド購入の追加のオプションが提供されます。スポットインスタンスオプションには、スポット容量をプロビジョニングできない場合のアクションを指定するタイムアウトや、スポットインスタンスフリートを起動するための優先配分戦略 (容量最適化) が含まれます。オンデマンドインスタンスフリートは、配分戦略 (最低料金) オプションを使用して起動することもできます。EMR のデフォルトのサービスロールではないサービスロールを使用する場合、またはサービスロールで EMR 管理ポリシーを使用する場合は、配分戦略オプションを有効にするために、カスタムクラスタサービスロールにアクセス許可を追加する必要があります。詳細については、「Amazon EMR のサービスロール (EMR ロール)」を参照してください。

インスタンスフリートの設定の詳細については、「インスタンスフリートを設定する」を参照してください。

ユニフォームインスタンスグループ

ユニフォームインスタンスグループは、インスタンスフリートよりも簡単なセットアップを提供します。各 Amazon EMR クラスターは、1 つの Amazon EC2 インスタンスを含む 1 つのプライマリインスタンスグループ、1 つ以上の EC2 インスタンスを含むコアインスタンスグループ、および最大 48 オプションのタスクインスタンスグループから成る、最大 50 のインスタンスグループを含む可能性があります。コアインスタンスグループおよびタスクインスタンスグループはそれぞれ、任意の数の Amazon EC2 インスタンスを含むことができます。手動で Amazon EC2 インスタンスを追加または削除して各インスタンスグループをスケールするか、オートスケーリングを設定することもできます。インスタンスの追加および削除の詳細については、「 クラスターのスケーリングを使用する」を参照してください。

ユニフォームインスタンスグループの構成の詳細については、「 ユニフォームインスタンスグループを設定する」を参照してください。

インスタンスフリートとインスタンスグループの操作

トピック

- インスタンスフリートを設定する
- インスタンスフリートでキャパシティ予約を使用する

- ユニフォームインスタンスグループを設定する
- インスタンスとアベイラビリティーゾーンの柔軟性に関するベストプラクティス
- クラスター設定のベストプラクティス

インスタンスフリートを設定する

> **Note**
>
> インスタンスフリート設定は、5.0.0 および 5.0.3 を除く Amazon EMR リリース 4.8.0 以降でのみ使用できます。

Amazon EMR クラスターのインスタンスフリート設定では、Amazon EC2 インスタンスのさまざまなプロビジョニングオプションを選択でき、クラスター内のノードタイプごとに柔軟で伸縮性に富むリソース戦略を開発できます。

インスタンスフリート設定では、各フリート内のオンデマンドインスタンスおよびスポットインスタンスのターゲット容量を指定します。クラスターを起動すると、Amazon EMR は、ターゲットが満たされるまでインスタンスをプロビジョニングします。クラスターの実行中に、料金の値上げまたはインスタンスの失敗のために Amazon EC2 がスポットインスタンスを再利用する場合、Amazon EMR は指定されたインスタンスタイプのいずれかで、そのインスタンスを置き換えようとします。これにより、スポット料金の急激な増加中に容量を再取得することが容易になります。

Amazon EMR がターゲットを達成するときに使用するフリートごとに最大 5 つの Amazon EC2 インスタンスタイプ、または AWS CLI または Amazon EMR API とオンデマンドインスタンスとスポットインスタンスの配分戦略を使用してクラスターを作成するときにフリートごとに最大 30 の Amazon EC2 インスタンスタイプを指定できます。 ???

また、異なるアベイラビリティーゾーンに複数のサブネットを選択することもできます。Amazon EMR は、クラスターを起動するときに、指定されたインスタンスと購入オプションをこれらのサブネットで探します。Amazon EMR が 1 つ以上のアベイラビリティーゾーンで AWS 大規模なイベントを検出すると、Amazon EMR は影響を受けたアベイラビリティーゾーンからトラフィックをルーティングし、選択に応じて代替アベイラビリティーゾーンで作成した新しいクラスターを起動しようとします。クラスターのアベイラビリティーゾーンの選択はクラスターの作成時にのみ行われることに注意してください。アベイラビリティーゾーンが停止しても、既存のクラスターノードは新しいアベイラビリティーゾーンで自動的に再起動されません。

考慮事項

Amazon EMR でインスタンスフリートを使用する場合は、以下の項目について考慮します。

- ノードタイプ (プライマリ、コア、タスク) あたり 1 つのみのインスタンスフリートを使用できます。でフリートごとに最大 5 つの Amazon EC2 インスタンスタイプを指定できます AWS Management Console （ または AWS CLI Amazon EMR API と を使用してクラスターを作成する場合は、インスタンスフリートごとに最大 30 タイプ) インスタンスフリートの配分戦略。
- Amazon EMR は、スポット購入オプションとオンデマンド購入オプションの両方で、プロビジョニングする指定された Amazon EC2 インスタンスタイプのいずれかまたはすべてを選択します。
- コアフリートとタスクフリートのスポットインスタンスおよびオンデマンドインスタンスのターゲット容量を確立できます。ターゲットに対してカウントされる、各 Amazon EC2 インスタンスに割り当てられた vCPU または汎用ユニットを使用します。Amazon EMR は各ターゲット容量が完全に満たされるまでインスタンスをプロビジョニングします。プライマリフリートの場合、ターゲットは常に 1 つです。
- 1 つのサブネット (アベイラビリティーゾーン) または範囲を選択できます。範囲を選択する場合、Amazon EMR はアベイラビリティーゾーンに最適な容量をプロビジョニングします。
- スポットインスタンスのターゲット容量を指定するとき、次のことを実行します。
  - インスタンスタイプごとに、最大スポット料金を指定します。スポット料金が最大スポット料金を下回る場合、Amazon EMR はスポットインスタンスをプロビジョニングします。お客様にご負担いただくのはスポット料金であり、必ずしも最大スポット料金ではありません。
  - フリートごとにスポットインスタンスをプロビジョニングするためのタイムアウト期間を定義します。Amazon EMR がスポット容量をプロビジョニングできない場合は、代わりにクラスターを終了させるか、オンデマンド容量のプロビジョニングに切り替えることができます。これはクラスターのプロビジョニングにのみ適用され、サイズ変更には適用されません。クラスターのサイズ変更プロセス中にタイムアウト期間が終了すると、プロビジョニングされていないスポットリクエストはオンデマンド容量に移行されずに無効になります。
- スポットインスタンスには、フリートごとに、価格と容量の最適化、容量の最適化、最低価格、全プールへの分散のいずれかの配分戦略を指定できます。
- フリートごとに、オンデマンドインスタンスには最低価格の配分戦略を適用できますが、オンデマンドインスタンスの配分戦略をカスタマイズすることはできません。
- オンデマンド `allocation strategy - lowest-price` を使用するフリートごとに、キャパシティ予約オプションを適用することを選択できます。
- クラスターを起動する前に、サブネットのサイズを確認します。タスクフリートを使用してクラスターをプロビジョニングするときに、対応するサブネットに十分な IP アドレスがない場合、フ

リートはクラスターをエラーで終了する代わりに、停止状態になります。この問題を回避するには、サブネット内の IP アドレスの数を増やすことをお勧めします。

インスタンスフリートオプション

次のガイドラインを使用して、インスタンスフリートオプションについて理解してください。

トピック

- ターゲット容量の設定
- 起動オプション
- 複数サブネット (アベイラビリティーゾーン) オプション
- マスターノードの設定

ターゲット容量の設定

コアフリートとタスクフリートのターゲット容量を指定します。この操作を行うと、Amazon EMR がプロビジョニングするオンデマンドインスタンスとスポットインスタンスの数が決まります。インスタンスを指定するときは、ターゲットに対してカウントされる各インスタンスの数を決定します。オンデマンドインスタンスがプロビジョニングされると、オンデマンドターゲットに対してカウントされます。スポットインスタンスの場合も同様です。プライマリフリートはコアフリートやタスクフリートとは異なり、常に 1 つのインスタンスです。したがって、このフリートのターゲット容量は常に 1 になります。

コンソールを使用する場合、デフォルトではターゲット容量のカウントとして、Amazon EC2 インスタンスタイプの vCPU が使用されます。これを [Generic units (汎用ユニット)] に変更し、各 EC2 インスタンスタイプの数を指定できます。を使用する場合は AWS CLI、インスタンスタイプごとに汎用ユニットを手動で割り当てます。

> **Important**
>
> を使用してインスタンスタイプを選択すると AWS Management Console、各インスタンスタイプに表示される vCPU の数は、そのインスタンスタイプの EC2 vCPU の数ではなく、そのインスタンスタイプの YARN vCPUsの数になります。 各インスタンスタイプの vCPU 数の詳細については、「Amazon EC2 インスタンスタイプ」を参照してください。

フリートごとに、最大 5 つの Amazon EC2 インスタンスタイプを指定します。を使用して AWS CLI または Amazon EMR API を使用してクラスターインスタンスフリートの配分戦略を作成する場合、インスタンスフリートごとに最大 30 の EC2 インスタンスタイプを指定できます。Amazon EMR は、これらの EC2 インスタンスタイプの任意の組み合わせを選択して、ターゲット容量を満たします。Amazon EMR は完全にターゲット容量を満たそうとするため、超過が発生する可能性があります。例えば、2 つの満たされていないユニットがある場合、Amazon EMR は 5 ユニットのインスタンスしかプロビジョニングできず、インスタンスは引き続きプロビジョニングされます。つまり、ターゲット容量を 3 ユニット超過します。

ターゲット容量を減らして実行中のクラスターのサイズを変更する場合、Amazon EMR は新しいターゲットに合わせてアプリケーションタスクを完了し、インスタンスを削除しようとします。詳細については、「タスクの完了時に終了」を参照してください。

起動オプション

スポットインスタンスでは、フリート内のインスタンスタイプごとに最大スポット料金を指定できます。この料金は、オンデマンド料金のパーセンテージまたは特定の金額として設定できます。アベイラビリティーゾーンの現在のスポット料金が最大スポット料金を下回っている場合、Amazon EMR はスポットインスタンスをプロビジョニングします。お客様にご負担いただくのはスポット料金であり、必ずしも最大スポット料金ではありません。

> **Note**
>
> 期間が定義されたスポットインスタンス (スポットブロックとも呼ばれます) は、2021 年 7 月 1 日以降の新規のお客様は、ご利用いただけません。既に、期間が指定されたスポットインスタンスを使用した経験をお持ちのお客様については、2022 年 12 月 31 日まで、この機能を引き続きサポートいたします。

Amazon EMR 5.12.1 以降では、容量が最適化された配分を使用してスポットインスタンスフリートおよびオンデマンドインスタンスフリートを起動するオプションを利用できます。この配分戦略オプションは、古い AWS Management Console または API を使用して設定できます`RunJobFlow`。新しいコンソールでは配分戦略をカスタマイズできないことに注意してください。配分戦略オプションを使用するには、追加のサービスロールのアクセス許可が必要です。デフォルトの Amazon EMR サービスロールと管理ポリシー (`EMR_DefaultRole` および `AmazonEMRServicePolicy_v2`) をクラスターに使用する場合、配分戦略のアクセス許可は既に含まれています。デフォルトの Amazon EMR サービスロールと管理ポリシーを使用していない場合、このオプションを使用するには、そ

れらを追加する必要があります。Amazon EMR のサービスロール (EMR ロール) を参照してください。

スポットインスタンスの詳細については、Amazon EC2 ユーザーガイド」の「スポットインスタンス」を参照してください。オンデマンドインスタンスの詳細については、Amazon EC2 ユーザーガイド」の「オンデマンドインスタンス」を参照してください。

最低料金配分戦略を使用してオンデマンドインスタンスフリートを起動する場合は、キャパシティ予約を使用するオプションがあります。キャパシティ予約オプションは、Amazon EMR API RunJobFlow を使用して設定できます。キャパシティ予約では、これらのオプションを使用するためにサービスロールのアクセス許可を追加する必要があります。配分戦略のアクセス許可 を参照してください。新しいコンソールではキャパシティ予約をカスタマイズできないことに注意してください。

複数サブネット (アベイラビリティーゾーン) オプション

インスタンスフリートを使用する場合は、VPC 内で複数の Amazon EC2 サブネットを指定できます。それぞれが異なるアベイラビリティーゾーンに対応しています。EC2-Classic を使用する場合はアベイラビリティーゾーンを明示的に指定します。Amazon EMR は、フリートの指定に従って、インスタンスを起動するために最適なアベイラビリティーゾーンを識別します。インスタンスは、 1 つのアベイラビリティーゾーンのみで常にプロビジョニングされます。プライベートサブネットまたはパブリックサブネットを選択できますが、2 つを組み合わせることはできません。指定するサブネットは同じ VPC 内になければなりません。

マスターノードの設定

プライマリインスタンスフリートは単一インスタンスにすぎないため、その構成はコアインスタンスフリートとタスクインスタンスフリートとは多少異なります。プライマリインスタンスフリートは 1 つのみのインスタンスで構成されているため、プライマリインスタンスフリートにはオンデマンドまたはスポットのいずれかを選択します。コンソールを使用してインスタンスフリートを作成する場合、選択する購入オプションのターゲット容量は 1 に設定されます。を使用する場合は AWS CLI、常に TargetSpotCapacityまたは TargetOnDemandCapacityのいずれかを必要に応じて 1 に設定します。この場合もプライマリインスタンスフリートには、最大 5 つのインスタンスタイプを選択できます (オンデマンドインスタンスまたはスポットインスタンスの配分戦略を使用する場合は、最大 30)。ただし、Amazon EMR が異なるタイプの複数のインスタンスをプロビジョニングする可能性があるコアインスタンスフリートとタスクインスタンスフリートとは異なり、Amazon EMR は、プライマリインスタンスフリートにプロビジョニングするために単一インスタンスタイプを選択します。

インスタンスフリートの配分戦略

Amazon EMR バージョン 5.12.1 以降では、各クラスターノードのオンデマンドインスタンスとスポットインスタンスで配分戦略オプションを使用できます。 AWS CLI、Amazon EMR API、または Amazon EMR コンソールを使用してクラスターを作成する場合、フリートあたり最大 30 の Amazon EC2 インスタンスタイプを指定できます。デフォルトの Amazon EMR クラスターインスタンスフリート設定では、フリートごとに最大 5 つのインスタンスタイプを使用できます。クラスターのプロビジョニングにかかる時間が短縮され、スポットインスタンスの割り当てがより正確になり、スポットインスタンスの中断が低減するように、配分戦略オプションを使用することをお勧めします。

トピック

- オンデマンドインスタンスの配分戦略
- スポットインスタンスの配分戦略
- 配分戦略のアクセス許可
- 配分戦略に必要な IAM アクセス許可

オンデマンドインスタンスの配分戦略

配分戦略を使用すると、オンデマンドインスタンスは最低価格の戦略を採用します。これにより、最低価格のインスタンスが最初に起動されます。オンデマンドインスタンスを起動すると、アカウント内でオープンまたはターゲットのキャパシティ予約を使用するオプションがあります。プライマリノード、コアノード、およびタスクノードには、オープンキャパシティ予約を使用できます。インスタンスフリートの配分戦略でオンデマンドインスタンスを使用すると、容量が不足することがあります。より多くのインスタンスタイプを指定することで分散させ、容量不足になる可能性を低減させることをお勧めします。詳細については、「インスタンスフリートでキャパシティ予約を使用する」を参照してください。

スポットインスタンスの配分戦略

スポットインスタンスには次のいずれかの配分戦略を指定できます。

`price-capacity-optimized` (推奨)

価格と容量の最適化配分戦略では、起動中のインスタンス数に対して、使用可能な容量が最も高く、価格が最も低いスポットインスタンスプールからスポットインスタンスを起動します。そのため、価格と容量の最適化戦略では通常、スポット容量を獲得できる可能性が高く、中断率も低くなります。

capacity-optimized

容量の最適化配分戦略では、短期的に中断の可能性が最も低い、最も利用可能なプールにスポットインスタンスを起動します。これは、作業の再開に関連する中断に伴うコストが高くなる可能性があるワークロードに適しています。これは Amazon EMR リリース 6.9.0 以前のデフォルト戦略です。

diversified

分散型の配分戦略により、Amazon EC2 はすべてのスポット容量プールにスポットインスタンスを分散します。

lowest-price

最低価格配分戦略では、使用可能な容量を持つ最低価格のプールからスポットインスタンスを起動します。最低価格のプールに使用可能な容量がない場合、スポットインスタンスは使用可能な容量のある 2 番目に低価格のプールから取得されます。希望する容量を満たす前にプールの容量が不足した場合、Amazon EC2 Fleet は 2 番目に低い価格のプールから容量を引き出し、引き続きリクエストを満たします。希望する容量を確実に満たすために、複数のプールからスポットインスタンスを受け取る場合があります。この戦略では、インスタンスの価格のみが考慮され、キャパシティの可用性は考慮されないため、中断率が高くなる可能性があります。

配分戦略のアクセス許可

配分戦略オプションでは、デフォルトの Amazon EMR サービスロールと Amazon EMR 管理ポリシー (EMR_DefaultRole と AmazonEMRServicePolicy_v2) に自動的に含まれるいくつかの IAM アクセス許可が必要です。クラスターにカスタムサービスロールまたは管理ポリシーを使用する場合は、クラスターを作成する前にこれらのアクセス許可を追加する必要があります。詳細については、「配分戦略のアクセス許可」を参照してください。

オンデマンド配分戦略オプションを使用すると、オプションのオンデマンドキャパシティ予約 (ODCR) を利用できます。キャパシティ予約オプションを使用すると、Amazon EMR クラスターに対してリザーブドキャパシティを最初に使用する設定を指定できます。これを使用すると、重要なワークロードには、オープンまたはターゲットの ODCR を使用して既に予約済みの容量を使用できます。重要でないワークロードの場合、キャパシティ予約設定を使用して、リザーブドキャパシティを消費するかどうかを指定できます。

キャパシティ予約は、属性 (インスタンスタイプ、プラットフォーム、アベイラビリティーゾーン) が一致するインスタンスのみ使用することができます。デフォルトでは、インスタンス属性と一致するオンデマンドインスタンスをプロビジョニングするときに、Amazon EMR によってオープンキャパシティ予約が自動的に使用されます。キャパシティの予約の属性と一致する実行中のインスタンス

がない場合は、一致する属性を持つインスタンスを起動するまでは使用されません。クラスターの起動時にキャパシティ予約を使用しない場合は、起動オプションでキャパシティ予約設定を [なし] に設定する必要があります。

ただし、特定のワークロードに対してキャパシティ予約を指定することもできます。これにより、リザーブドキャパシティーで実行できるインスタンスを明示的に制御できます。オンデマンドキャパシティ予約については、「インスタンスフリートでキャパシティ予約を使用する」を参照してください。

配分戦略に必要な IAM アクセス許可

Amazon EMR のサービスロール (EMR ロール) では、オンデマンドインスタンスフリートまたはスポットインスタンスフリートの配分戦略オプションを使用するクラスターを作成するには、追加のアクセス許可が必要です。

これらのアクセス権限は、デフォルトの Amazon EMR サービスロール EMR_DefaultRole と Amazon EMR 管理ポリシー AmazonEMRServicePolicy_v2 に自動的に含まれます。

クラスターにカスタムサービスロールまたは管理ポリシーを使用する場合は、次のアクセス許可を追加する必要があります。

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "ec2:DeleteLaunchTemplate",
    "ec2:CreateLaunchTemplate",
    "ec2:DescribeLaunchTemplates",
    "ec2:CreateLaunchTemplateVersion",
    "ec2:CreateFleet"
   ],
   "Resource": "*"
  }
}
```

オープンまたはターゲットのキャパシティ予約を使用するクラスターを作成するために必要なサービスロールのアクセス許可を次に示します。配分戦略オプションの使用に必要なアクセス許可に加えて、これらのアクセス許可を含める必要があります。

Example サービスロールキャパシティ予約に関するポリシードキュメント

オープンキャパシティ予約を使用するには、次の追加のアクセス許可を含める必要があります。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeCapacityReservations",
                "ec2:DescribeLaunchTemplateVersions",
                "ec2:DeleteLaunchTemplateVersions"
            ],
            "Resource": "*"
        }
    ]
}
```

Example

ターゲットキャパシティ予約を使用するには、次の追加のアクセス許可を含める必要があります。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeCapacityReservations",
                "ec2:DescribeLaunchTemplateVersions",
                "ec2:DeleteLaunchTemplateVersions",
                "resource-groups:ListGroupResources"
            ],
            "Resource": "*"
        }
    ]
}
```

クラスターのインスタンスフリートを設定する

> **Note**
>
> Amazon EMR コンソールは、再設計され、使いやすくなりました。古いコンソールと新しいコンソールエクスペリエンスの違いについては、「Amazon EMR コンソール」を参照してください。

New console

新しいコンソールを使用してインスタンスフリートでクラスターを作成するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で、[クラスター] を選択し、[クラスターの作成] を選択します。
3. [クラスターの設定] で [インスタンスフリート] を選択します。
4. [ノードグループ] ごとに、[インスタンスタイプの追加] を選択し、プライマリインスタンスフリートとコアインスタンスフリートには最大 5 個のインスタンスタイプを選択し、タスクインスタンスフリートには最大 15 個のインスタンスタイプを選択します。Amazon EMR は、クラスターを起動するときに、これらのインスタンスタイプの任意の組み合わせをプロビジョニングする場合があります。
5. 各ノードグループタイプで、各インスタンスの横にある [アクション] ドロップダウンメニューを選択し、これらの設定を変更します。

   EBS ボリュームの追加

   Amazon EMR がプロビジョニングした後に、インスタンスタイプにアタッチする EBS ボリュームを指定します。

   加重容量の編集

   コアノードグループでは、この値をアプリケーションに合わせて任意のユニット数に変更します。各フリートインスタンスタイプの YARN vCore の数がデフォルトの加重容量単位として使用されます。プライマリノードの加重容量は編集できません。

最大スポット価格の編集

フリートのインスタンスタイプごとに、最大スポット料金を指定します。この料金は、オンデマンド料金のパーセンテージまたは特定の金額として設定できます。アベイラビリティーゾーンの現在のスポット料金が最大スポット料金を下回っている場合、Amazon EMR はスポットインスタンスをプロビジョニングします。お客様にご負担いただくのはスポット料金であり、必ずしも最大スポット料金ではありません。

6. オプションで、ノードにセキュリティグループを追加するには、[ネットワーク] セクションで [EC2 セキュリティグループ (ファイアウォール)] を展開し、ノードタイプごとにセキュリティグループを選択します。
7. オプションで、配分戦略オプションを使用する場合は [配分戦略を適用] の横にあるチェックボックスをオンにし、スポットインスタンスに指定する配分戦略を選択します。Amazon EMR サービスロールに必要なアクセス許可がない場合は、このオプションを選択しないでください。詳細については、「インスタンスフリートの配分戦略」を参照してください。
8. クラスターに適用するその他のオプションを選択します。
9. クラスターを起動するには、[クラスターの作成] を選択します。

Old console

古いコンソールを使用してインスタンスフリートでクラスターを作成するには

1. 新しい Amazon EMR コンソールに移動し、サイドナビゲーションから [古いコンソールに切り替え] を選択します。古いコンソールに切り替えたときの動作の詳細については、「Using the old console」を参照してください。
2. [クラスターを作成] を選択します。
3. コンソールウィンドウの上部で、[詳細オプションに移動する] を選択し、[ソフトウェア設定] オプションを入力してから、[次へ] を選択します。
4. [Cluster Composition] (クラスター構成) で、[インスタンスフリート] を選択します。インスタンスフリートオプションを選択すると、オンデマンドインスタンスとスポットインスタンスの [ターゲット容量] を指定するオプションが [Cluster Nodes and Instances] (クラスターノードとインスタンス) テーブルに表示されます。
5. [Network] に値を入力します。[ネットワーク] に VPC を選択する場合、[EC2 サブネット] を 1 つ選択するか、CTRL を押しながら複数の Amazon EC2 サブネットを選択します。選択するサブネットは同じ種類 (パブリックまたはプライベート) である必要があります。1 つのみ

選択した場合、クラスターはそのサブネット内で起動します。グループを選択した場合は、クラスターの起動時にグループから最適なサブネットが選択されます。

> **Note**
>
> アカウントおよびリージョンによっては、[ネットワーク] に [EC2-Classic に起動] を選択するオプションが提供されます。そのオプションを選択する場合、[EC2 Subnets (EC2 サブネット)] ではなく、[EC2 Availability Zones (EC2 アベイラビリティーゾーン)] から 1 つ以上を選択します。詳細については、Amazon EC2 ユーザーガイド」の「Amazon EC2 と Amazon VPCAmazon EC2」を参照してください。

6. 配分戦略を使用する場合、[配分戦略] で、チェックボックスをオンにして、配分戦略を適用します。詳細については、「インスタンスフリートの配分戦略」を参照してください。
7. [Node type] (ノードタイプ) の下で、インスタンスフリートのデフォルト名を変更する場合、鉛筆アイコンを選択してわかりやすい名前を入力します。[タスク] インスタンスフリートを削除する場合は、[タスク] 行の右側にある X 印のアイコンを選択します。
8. [フリート対象のインスタンスタイプの追加/削除] を選択し、プライマリインスタンスフリートとコアインスタンスフリートのリストから最大 5 つのインスタンスタイプを選択します。タスクインスタンスフリートには最大 15 個のインスタンスタイプを追加します。Amazon EMR は、クラスターを起動するときに、これらのインスタンスタイプの任意の組み合わせをプロビジョニングすることを選択する場合があります。
9. コアインスタンスタイプおよびタスクインスタンスタイプごとに、そのインスタンスに加重容量を定義する方法を選択します (各インスタンスは X ユニットとしてカウントされます)。各フリートインスタンスタイプの YARN vCore の数は、デフォルトの加重容量ユニットとして使用されますが、この値はアプリケーションに適した任意のユニットに変更できます。
10. [ターゲット容量] で、フリートごとに必要なオンデマンドインスタンスとスポットインスタンスの合計数を定義します。EMR は、フリート内のインスタンスが、オンデマンドターゲット容量とスポットターゲット容量のリクエストされたユニットを確実に満たすようにします。フリートにオンデマンドユニットまたはスポットユニットが指定されていない場合、そのフリートの容量はプロビジョニングされません。
11. フリートがスポット用のターゲット容量で設定される場合、[オンデマンドの %] 料金として最大スポット料金を入力するか、ドル ($) 金額を米ドルで入力できます。
12. インスタンスタイプがプロビジョニングされるときに EBS ボリュームをアタッチするには、[EBS ストレージ] の隣にある鉛筆を選択してから、EBS 設定オプションを入力します。

13. [スポットユニット数] のインスタント数を設定した場合、以下のガイドラインに従って、[詳細スポットオプション] を設定します。

    - プロビジョニングのタイムアウト — これらの設定を使用して、指定された [フリートインスタンスタイプ] の中からスポットインスタンスをプロビジョニングできないときに、Amazon EMR が実行することを制御します。分数でタイムアウト期間を入力し、[Terminate the cluster (クラスターを終了する)] か、[Switch to provisioning On-Demand Instances (オンデマンドインスタンスのプロビジョニングに切り替える)] かを選択します。オンデマンドインスタンスに切り替える選択をする場合、オンデマンドインスタンスの割り当てられた容量はスポットインスタンスのターゲット容量に含められ、Amazon EMR は、スポットインスタンスのターゲット容量が満たされるまでオンデマンドインスタンスをプロビジョニングします。

14. [次へ] を選択してクラスター設定を変更し、[次へ] を選択します。
15. 新しい配分戦略オプションを適用するように選択した場合は、[セキュリティオプション] 設定で、配分戦略オプションに必要なアクセス許可が含まれる [EMR ロール] と [EC2 インスタンスプロファイル] を選択します。そうしないと、クラスターの作成は失敗します。
16. [クラスターの作成] を選択します。

AWS CLI

を使用してインスタンスフリートでクラスターを作成して起動するには AWS CLI、次のガイドラインに従います。

- インスタンスフリートでクラスターを作成して起動するには、 `create-cluster` パラメータとともに`--instance-fleet` コマンドを使用します。
- クラスター内のインスタンスフリートの設定情報を得るには、`list-instance-fleets` コマンドを使用します。
- 作成するクラスターに複数のカスタム Amazon Linux AMI を追加するには、それぞれの `InstanceType` を指定して `CustomAmiId` オプションを使用します。要件に合わせて、複数のインスタンスタイプと複数のカスタム AMI を持つインスタンスフリートノードを設定できます。例: インスタンスフリート設定でクラスターを作成する を参照してください。
- インスタンスフリートのターゲット容量に変更を加えるには、 `modify-instance-fleet` コマンドを使用します。
- タスクインスタンスがないクラスターにタスクインスタンスを追加するには、 `add-instance-fleet`コマンドを使用します。

- add-instance-fleet コマンドで CustomAmiId AMIs をタスクインスタンスフリートに追加できます。例: インスタンスフリート設定でクラスターを作成する を参照してください。
- インスタンスフリートを作成するときに配分戦略オプションを使用するには、サービスロールを更新して、次のセクションのポリシードキュメントの例を含めます。
- オンデマンド配分戦略でインスタンスフリートを作成するときにキャパシティ予約オプションを使用するには、サービスロールを更新して、次のセクションのポリシードキュメントの例を含めます。
- インスタンスフリートは、デフォルトの EMR サービスロールと Amazon EMR 管理ポリシー (EMR_DefaultRole と AmazonEMRServicePolicy_v2) に自動的に組み込まれています。クラスターにカスタムサービスロールまたは管理ポリシーを使用する場合は、次のセクションの配分戦略の新しいアクセス許可を追加する必要があります。

例: インスタンスフリート設定でクラスターを作成する

次の例は、 create-cluster コマンドを、組み合わせ可能なさまざまなオプションを使って実行する方法を示しています。

> **Note**
>
> デフォルト Amazon EMR サービスロールと EC2 インスタンスプロファイルを作成したことがない場合、create-cluster コマンドを使用する前に、aws emr create-default-roles を使って作成します。

Example 例: オンデマンドプライマリ、単一インスタンスタイプのオンデマンドコア、デフォルト VPC

```
aws emr create-cluster --release-label emr-5.3.1 --service-role EMR_DefaultRole \
  --ec2-attributes InstanceProfile=EMR_EC2_DefaultRole \
  --instance-fleets \

 InstanceFleetType=MASTER,TargetOnDemandCapacity=1,InstanceTypeConfigs=['{InstanceType=m5.xlarg
 \

 InstanceFleetType=CORE,TargetOnDemandCapacity=1,InstanceTypeConfigs=['{InstanceType=m5.xlarge}
```

Example 例: スポットプライマリ、単一インスタンスタイプのスポットコア、デフォルト VPC

```
aws emr create-cluster --release-label emr-5.3.1 --service-role EMR_DefaultRole \
  --ec2-attributes InstanceProfile=EMR_EC2_DefaultRole \
  --instance-fleets \
    InstanceFleetType=MASTER,TargetSpotCapacity=1,\
InstanceTypeConfigs=['{InstanceType=m5.xlarge,BidPrice=0.5}'] \
    InstanceFleetType=CORE,TargetSpotCapacity=1,\
InstanceTypeConfigs=['{InstanceType=m5.xlarge,BidPrice=0.5}']
```

Example 例: オンデマンドプライマリ、単一インスタンスタイプの混合コア、単一 EC2 サブネット

```
aws emr create-cluster --release-label emr-5.3.1 --service-role EMR_DefaultRole \
  --ec2-attributes InstanceProfile=EMR_EC2_DefaultRole,SubnetIds=['subnet-ab12345c'] \
  --instance-fleets \
    InstanceFleetType=MASTER,TargetOnDemandCapacity=1,\
InstanceTypeConfigs=['{InstanceType=m5.xlarge}'] \
    InstanceFleetType=CORE,TargetOnDemandCapacity=2,TargetSpotCapacity=6,\
InstanceTypeConfigs=['{InstanceType=m5.xlarge,BidPrice=0.5,WeightedCapacity=2}']
```

Example 例: オンデマンドプライマリ、複数の加重インスタンスタイプのスポットコア、スポットのタイムアウト、EC2 サブネットの範囲

```
aws emr create-cluster --release-label emr-5.3.1 --service-role EMR_DefaultRole \
  --ec2-attributes InstanceProfile=EMR_EC2_DefaultRole,SubnetIds=['subnet-
ab12345c','subnet-de67890f'] \
  --instance-fleets \
    InstanceFleetType=MASTER,TargetOnDemandCapacity=1,\
InstanceTypeConfigs=['{InstanceType=m5.xlarge}'] \
    InstanceFleetType=CORE,TargetSpotCapacity=11,\
InstanceTypeConfigs=['{InstanceType=m5.xlarge,BidPrice=0.5,WeightedCapacity=3}',\
'{InstanceType=m4.2xlarge,BidPrice=0.9,WeightedCapacity=5}'],\
LaunchSpecifications={SpotSpecification='{TimeoutDurationMinutes=120,TimeoutAction=SWITCH_TO_ON
```

Example 例: オンデマンドプライマリ、複数の加重インスタンスタイプの混合コアおよびタスク、コアスポットインスタンスのタイムアウト、EC2 サブネットの範囲

```
aws emr create-cluster --release-label emr-5.3.1 --service-role EMR_DefaultRole \
  --ec2-attributes InstanceProfile=EMR_EC2_DefaultRole,SubnetIds=['subnet-
ab12345c','subnet-de67890f'] \
  --instance-fleets \
```

```
 InstanceFleetType=MASTER,TargetOnDemandCapacity=1,InstanceTypeConfigs=['{InstanceType=m5.xlarg
 \
    InstanceFleetType=CORE,TargetOnDemandCapacity=8,TargetSpotCapacity=6,\
InstanceTypeConfigs=['{InstanceType=m5.xlarge,BidPrice=0.5,WeightedCapacity=3}',\
'{InstanceType=m4.2xlarge,BidPrice=0.9,WeightedCapacity=5}'],\
LaunchSpecifications={SpotSpecification='{TimeoutDurationMinutes=120,TimeoutAction=SWITCH_TO_ON
 \
    InstanceFleetType=TASK,TargetOnDemandCapacity=3,TargetSpotCapacity=3,\
InstanceTypeConfigs=['{InstanceType=m5.xlarge,BidPrice=0.5,WeightedCapacity=3}']
```

Example 例: スポットプライマリ、コアまたはタスクなし、Amazon EBS 設定、デフォルト VPC

```
aws emr create-cluster --release-label Amazon EMR 5.3.1 --service-role EMR_DefaultRole
 \
  --ec2-attributes InstanceProfile=EMR_EC2_DefaultRole \
  --instance-fleets \
    InstanceFleetType=MASTER,TargetSpotCapacity=1,\
LaunchSpecifications={SpotSpecification='{TimeoutDurationMinutes=60,TimeoutAction=TERMINATE_CLU
\
InstanceTypeConfigs=['{InstanceType=m5.xlarge,BidPrice=0.5,\
EbsConfiguration={EbsOptimized=true,EbsBlockDeviceConfigs=[{VolumeSpecification={VolumeType=gp2
\
SizeIn GB=100}},{VolumeSpecification={VolumeType=io1,SizeInGB=100,Iop
 s=100},VolumesPerInstance=4}]}}']
```

Example 例: 複数のカスタム AMI、複数のインスタンスタイプ、オンデマンドプライマリ、オンデマンドコア

```
aws emr create-cluster --release-label Amazon EMR 5.3.1 --service-role EMR_DefaultRole
 \
  --ec2-attributes InstanceProfile=EMR_EC2_DefaultRole \
  --instance-fleets \
    InstanceFleetType=MASTER,TargetOnDemandCapacity=1,\
InstanceTypeConfigs=['{InstanceType=m5.xlarge,CustomAmiId=ami-123456},
{InstanceType=m6g.xlarge, CustomAmiId=ami-234567}'] \
    InstanceFleetType=CORE,TargetOnDemandCapacity=1,\
InstanceTypeConfigs=['{InstanceType=m5.xlarge,CustomAmiId=ami-123456},
{InstanceType=m6g.xlarge, CustomAmiId=ami-234567}']
```

Example 例: 複数のインスタンスタイプと複数のカスタム AMI を持つ実行中のクラスターにタスクノードを追加する

```
aws emr add-instance-fleet --cluster-id j-123456 --release-label Amazon EMR 5.3.1 \
  --service-role EMR_DefaultRole \
  --ec2-attributes InstanceProfile=EMR_EC2_DefaultRole \
  --instance-fleet \
    InstanceFleetType=Task,TargetSpotCapacity=1,\
InstanceTypeConfigs=['{InstanceType=m5.xlarge,CustomAmiId=ami-123456}',\
'{InstanceType=m6g.xlarge,CustomAmiId=ami-234567}']
```

Example 例: JSON 設定ファイルを使用する

JSON ファイルでインスタンスフリートパラメータを構成してから、インスタンスフリートの唯一のパラメータとして JSON ファイルを参照できます 例えば、次のコマンドは JSON 設定ファイル *my-fleet-config.json* を参照します。

```
aws emr create-cluster --release-label emr-5.30.0 --service-role EMR_DefaultRole \
--ec2-attributes InstanceProfile=EMR_EC2_DefaultRole \
--instance-fleets file://my-fleet-config.json
```

my-fleet-config.json ファイルは、次の例に示すように、プライマリ、コア、およびタスクインスタンスフリートを指定します。コアインスタンスフリートはオンデマンドの割合として最大スポット料金 (BidPrice) を使用しますが、タスクおよびプライマリインスタンスフリートは米ドルの文字列として最大スポット料金 (BidPriceAsPercentageofOnDemandPrice) を使用します。

```
[
    {
        "Name": "Masterfleet",
        "InstanceFleetType": "MASTER",
        "TargetSpotCapacity": 1,
        "LaunchSpecifications": {
            "SpotSpecification": {
                "TimeoutDurationMinutes": 120,
                "TimeoutAction": "SWITCH_TO_ON_DEMAND"
            }
        },
        "InstanceTypeConfigs": [
            {
                "InstanceType": "m5.xlarge",
                "BidPrice": "0.89"
            }
```

```
        ]
    },
    {
        "Name": "Corefleet",
        "InstanceFleetType": "CORE",
        "TargetSpotCapacity": 1,
        "TargetOnDemandCapacity": 1,
        "LaunchSpecifications": {
          "OnDemandSpecification": {
            "AllocationStrategy": "lowest-price",
            "CapacityReservationOptions":
            {
                "UsageStrategy": "use-capacity-reservations-first",
                "CapacityReservationResourceGroupArn": "String"
            }
        },
            "SpotSpecification": {
                "AllocationStrategy": "capacity-optimized",
                "TimeoutDurationMinutes": 120,
                "TimeoutAction": "TERMINATE_CLUSTER"
            }
        },
        "InstanceTypeConfigs": [
            {
                "InstanceType": "m5.xlarge",
                "BidPriceAsPercentageOfOnDemandPrice": 100
            }
        ]
    },
    {
        "Name": "Taskfleet",
        "InstanceFleetType": "TASK",
        "TargetSpotCapacity": 1,
        "LaunchSpecifications": {
          "OnDemandSpecification": {
            "AllocationStrategy": "lowest-price",
            "CapacityReservationOptions":
            {
                "CapacityReservationPreference": "none"
            }
        },
            "SpotSpecification": {
                "TimeoutDurationMinutes": 120,
                "TimeoutAction": "TERMINATE_CLUSTER"
```

```
            }
        },
        "InstanceTypeConfigs": [
            {
                "InstanceType": "m5.xlarge",
                "BidPrice": "0.89"
            }
        ]
    }
]
```

インスタンスフリートのターゲット容量を変更する

コマンドを使用して、インスタンスフリートの新しいターゲット容量を指定します modify-instance-fleet 。クラスター ID とインスタンスフリート ID を指定しなければなりません。コマンドを使用してインスタンスフリート ID を取得しますlist-instance-fleets 。

```
aws emr modify-instance-fleet --cluster-id <cluster-id> \
  --instance-fleet \
    InstanceFleetId='<instance-fleet-id>',TargetOnDemandCapacity=1,TargetSpotCapacity=1
```

タスクインスタンスフリートをクラスターに追加する

クラスターにプライマリインスタンスフリートおよびコアインスタンスフリートしかない場合、add-instance-fleet コマンドを使用してタスクインスタンスフリートを追加できます。このコマンドは、タスクインスタンスフリートを追加するためだけに使用できます。

```
aws emr add-instance-fleet --cluster-id <cluster-id>
  --instance-fleet \
    InstanceFleetType=TASK,TargetSpotCapacity=1,\
LaunchSpecifications={SpotSpecification='{TimeoutDurationMinutes=20,TimeoutAction=TERMINATE_CLU
\
InstanceTypeConfigs=['{InstanceType=m5.xlarge,BidPrice=0.5}']
```

クラスター内のインスタンスフリートの設定情報を取得する

クラスター内のインスタンスフリートの構成情報を得るには、 list-instance-fleets コマンドを使用します。コマンドは、クラスター ID を入力として受け入れます。次の例では、このコマンドと、それによるプライマリタスクインスタンスグループとコアタスクインスタンスグループを含むク

ラスターの出力を示しています。完全なレスポンス構文については、「Amazon EMR API リファレンスListInstanceFleets」の「」を参照してください。

```
list-instance-fleets --cluster-id <cluster-id>
```

```
{
    "InstanceFleets": [
        {
            "Status": {
                "Timeline": {
                    "ReadyDateTime": 1488759094.637,
                    "CreationDateTime": 1488758719.817
                },
                "State": "RUNNING",
                "StateChangeReason": {
                    "Message": ""
                }
            },
            "ProvisionedSpotCapacity": 6,
            "Name": "CORE",
            "InstanceFleetType": "CORE",
            "LaunchSpecifications": {
                "SpotSpecification": {
                    "TimeoutDurationMinutes": 60,
                    "TimeoutAction": "TERMINATE_CLUSTER"
                }
            },
            "ProvisionedOnDemandCapacity": 2,
            "InstanceTypeSpecifications": [
                {
                    "BidPrice": "0.5",
                    "InstanceType": "m5.xlarge",
                    "WeightedCapacity": 2
                }
            ],
            "Id": "if-1ABC2DEFGHIJ3"
        },
        {
            "Status": {
                "Timeline": {
                    "ReadyDateTime": 1488759058.598,
                    "CreationDateTime": 1488758719.811
                },
```

```
                "State": "RUNNING",
                "StateChangeReason": {
                    "Message": ""
                }
            },
            "ProvisionedSpotCapacity": 0,
            "Name": "MASTER",
            "InstanceFleetType": "MASTER",
            "ProvisionedOnDemandCapacity": 1,
            "InstanceTypeSpecifications": [
                {
                    "BidPriceAsPercentageOfOnDemandPrice": 100.0,
                    "InstanceType": "m5.xlarge",
                    "WeightedCapacity": 1
                }
            ],
           "Id": "if-2ABC4DEFGHIJ4"
        }
    ]
}
```

インスタンスフリートでキャパシティ予約を使用する

キャパシティ予約オプションを使用してオンデマンドインスタンスフリートを起動するには、キャパシティ予約オプションを使用するために必要な追加のサービスロールのアクセス許可をアタッチします。キャパシティ予約オプションはオンデマンド配分戦略と一緒に使用する必要があるため、配分戦略に必要なアクセス許可をサービスロールと管理ポリシーに含める必要もあります。詳細については、「配分戦略のアクセス許可」を参照してください。

Amazon EMR は、オープンキャパシティ予約とターゲットキャパシティ予約の両方をサポートしています。以下のトピックでは、オンデマンドキャパシティ予約を使用してインスタンスフリートを起動するために RunJobFlow アクションまたは `create-cluster` コマンドとともに使用できるインスタンスフリート設定について説明します。

ベストエフォートベースでオープンキャパシティ予約を使用する

クラスターのオンデマンドインスタンスが、アカウントで使用可能なオープンキャパシティ予約の属性 (インスタンスタイプ、プラットフォーム、テナンシー、アベイラビリティーゾーン) と一致する場合、キャパシティ予約は自動的に適用されます。ただし、キャパシティ予約が使用されるかどうかは保証されません。クラスターをプロビジョニングするために、Amazon EMR は起動リクエストで

指定されたすべてのインスタンスプールを評価し、リクエストされたすべてのコアノードを起動するのに十分な容量を持つ最低料金のインスタンスプールを使用します。インスタンスプールに一致する使用可能なオープンキャパシティ予約が自動的に適用されます。利用可能なオープンキャパシティ予約がインスタンスプールと一致しない場合、それらは未使用のままになります。

コアノードがプロビジョニングされると、アベイラビリティーゾーンが選択され、修正されます。Amazon EMR は、すべてのタスクノードがプロビジョニングされるまで、選択したアベイラビリティーゾーン内の最低料金のものからインスタンスプールにタスクノードをプロビジョニングします。インスタンスプールに一致する使用可能なオープンキャパシティ予約が自動的に適用されます。

ベストエフォートベースでオープンキャパシティ予約を使用するための Amazon EMR 容量の割り当てロジックのユースケースを次に示します。

例 1: 起動リクエストで最低料金のインスタンスプールに、使用可能なオープンキャパシティ予約がある

この場合、Amazon EMR はオンデマンドインスタンスを使用して最低料金のインスタンスプールで容量を起動します。そのインスタンスプール内の使用可能なオープンキャパシティ予約が自動的に使用されます。

オンデマンド戦略	lowest-price		
リクエストされたキャパシティ	100		
インスタンスタイプ	c5.xlarge	m5.xlarge	r5.xlarge
使用可能なオープンキャパシティ予約	150	100	100
オンデマンド料金	$	$$	$$$
プロビジョニングされたインスタンス	100	-	-
使用されたオープンキャパシティ予約	100	-	-
使用可能なオープンキャパシティ予約	50	100	100

インスタンスフリートの起動後、describe-capacity-reservations を実行して、未使用のままのキャパシティ予約の数を確認できます。

例 2: 起動リクエストで最低料金のインスタンスプールに、使用可能なオープンキャパシティ予約がない

この場合、Amazon EMR はオンデマンドインスタンスを使用して最低料金のインスタンスプールで容量を起動します。ただし、オープンキャパシティ予約は未使用のままです。

オンデマンド戦略	lowest-price		
リクエストされたキャパシティ	100		
インスタンスタイプ	c5.xlarge	m5.xlarge	r5.xlarge
使用可能なオープンキャパシティ予約	-	-	100
オンデマンド料金	$	$$	$$$
プロビジョニングされたインスタンス	100	-	-
使用されたオープンキャパシティ予約	-	-	-
使用可能なオープンキャパシティ予約	-	-	100

ベストエフォートベースでオープンキャパシティ予約を使用するようにインスタンスフリートを設定する

RunJobFlow アクションを使用して、インスタンスフリートベースのクラスターを作成する場合、オンデマンド配分戦略を lowest-price に設定し、キャパシティ予約オプションの CapacityReservationPreference を open に設定します。または、このフィールドを空白のままにすると、Amazon EMR はオンデマンドインスタンスのキャパシティ予約設定をデフォルトの open に設定します。

```
"LaunchSpecifications":
    {"OnDemandSpecification": {
        "AllocationStrategy": "lowest-price",
        "CapacityReservationOptions":
         {
            "CapacityReservationPreference": "open"
         }
        }
    }
```

また、Amazon EMR CLI により、オープンキャパシティ予約を使用してインスタンスフリートベースのクラスターを作成することもできます。

```
aws emr create-cluster \
 --name 'open-ODCR-cluster' \
 --release-label emr-5.30.0 \
 --service-role EMR_DefaultRole \
 --ec2-attributes SubnetId=subnet-22XXXX01,InstanceProfile=EMR_EC2_DefaultRole \
 --instance-fleets
 InstanceFleetType=MASTER,TargetOnDemandCapacity=1,InstanceTypeConfigs=['{InstanceType=c4.xlarg
 \

 InstanceFleetType=CORE,TargetOnDemandCapacity=100,InstanceTypeConfigs=['{InstanceType=c5.xlarg
{InstanceType=m5.xlarge},{InstanceType=r5.xlarge}'],\
   LaunchSpecifications={OnDemandSpecification='{AllocationStrategy=lowest-
price,CapacityReservationOptions={CapacityReservationPreference=open}}'}
```

各パラメータの意味は次のとおりです。

- `open-ODCR-cluster` は、オープンキャパシティ予約を使用するクラスターの名前に置き換えられます。
- `subnet-22XXXX01` は、サブネット ID に置き換えられます。

オープンキャパシティ予約を最初に使用する

Amazon EMR クラスターのプロビジョニング中に、最低料金の配分戦略をオーバーライドし、使用可能なオープンキャパシティ予約を最初に使用することを優先するように選択できます。この場合、Amazon EMR は起動リクエストで指定された、キャパシティ予約を使用するすべてのインスタンスプールを評価し、リクエストされたすべてのコアノードを起動するのに十分な容量を持つ最低料金のインスタンスプールを使用します。キャパシティ予約を使用するインスタンスプールに、リクエ

ストされたコアノードのための十分な容量がない場合、Amazon EMR は前のトピックで説明したベストエフォートのケースにフォールバックします。つまり、Amazon EMR は起動リクエストで指定されたすべてのインスタンスプールを評価し直し、リクエストされたすべてのコアノードを起動するのに十分な容量を持つ最低料金のインスタンスプールを使用します。インスタンスプールに一致する使用可能なオープンキャパシティ予約が自動的に適用されます。利用可能なオープンキャパシティ予約がインスタンスプールと一致しない場合、それらは未使用のままになります。

コアノードがプロビジョニングされると、アベイラビリティーゾーンが選択され、修正されます。Amazon EMR は、すべてのタスクノードがプロビジョニングされるまで、選択したアベイラビリティーゾーン内の最低料金のものから、キャパシティ予約を使用するインスタンスプールにタスクノードをプロビジョニングします。Amazon EMR は、選択したアベイラビリティーゾーン内の各インスタンスプールで使用可能なオープンキャパシティ予約を最初に使用し、必要な場合のみ、最低料金戦略を使用して残りのタスクノードをプロビジョニングします。

オープンキャパシティ予約を最初に使用するための Amazon EMR 容量の割り当てロジックのユースケースを次に示します。

例 1: 起動リクエストで使用可能なオープンキャパシティ予約があるインスタンスプールに、コアノードのための十分な容量がある

この場合、Amazon EMR は、インスタンスプールの料金に関係なく、使用可能なオープンキャパシティ予約があるインスタンスプールでキャパシティを起動します。その結果、すべてのコアノードがプロビジョニングされるまで、可能な限りオープンキャパシティ予約が使用されます。

オンデマンド戦略	lowest-price		
リクエストされたキャパシティ	100		
使用戦略	use-capacity-reservations-first		
インスタンスタイプ	c5.xlarge	m5.xlarge	r5.xlarge
使用可能なオープンキャパシティ予約	-	-	150
オンデマンド料金	$	$$	$$$
プロビジョニングされたインスタンス	-	-	100

使用されたオープンキャパシティ予約	-	-	100
使用可能なオープンキャパシティ予約	-	-	50

例 2: 起動リクエストで使用可能なオープンキャパシティ予約があるインスタンスプールに、コアノードのための十分な容量がない

この場合、Amazon EMR は、最低料金戦略とキャパシティ予約を使用するベストエフォートを使用して、起動コアノードにフォールバックします。

オンデマンド戦略	lowest-price		
リクエストされたキャパシティ	100		
使用戦略	use-capacity-reservations-first		
インスタンスタイプ	c5.xlarge	m5.xlarge	r5.xlarge
使用可能なオープンキャパシティ予約	10	50	50
オンデマンド料金	$	$$	$$$
プロビジョニングされたインスタンス	100	-	-
使用されたオープンキャパシティ予約	10	-	-
使用可能なオープンキャパシティ予約	-	50	50

インスタンスフリートの起動後、`describe-capacity-reservations` を実行して、未使用のままのキャパシティ予約の数を確認できます。

オープンキャパシティ予約を最初に使用するようにインスタンスフリートを設定する

RunJobFlow アクションを使用して、インスタンスフリートベースのクラスターを作成する場合、オンデマンド配分戦略を lowest-price に設定し、CapacityReservationOptions の UsageStrategy を use-capacity-reservations-first に設定します。

```
"LaunchSpecifications":
    {"OnDemandSpecification": {
        "AllocationStrategy": "lowest-price",
        "CapacityReservationOptions":
         {
            "UsageStrategy": "use-capacity-reservations-first"
         }
       }
    }
```

また、Amazon EMR CLI により、キャパシティ予約を最初に使用してインスタンスフリートベースのクラスターを作成することもできます。

```
aws emr create-cluster \
  --name 'use-CR-first-cluster' \
  --release-label emr-5.30.0 \
  --service-role EMR_DefaultRole \
  --ec2-attributes SubnetId=subnet-22XXXX01,InstanceProfile=EMR_EC2_DefaultRole \
  --instance-fleets \

 InstanceFleetType=MASTER,TargetOnDemandCapacity=1,InstanceTypeConfigs=['{InstanceType=c4.xlarg
 \

 InstanceFleetType=CORE,TargetOnDemandCapacity=100,InstanceTypeConfigs=['{InstanceType=c5.xlarg
{InstanceType=m5.xlarge},{InstanceType=r5.xlarge}'],\
LaunchSpecifications={OnDemandSpecification='{AllocationStrategy=lowest-
price,CapacityReservationOptions={UsageStrategy=use-capacity-reservations-first}}'}
```

各パラメータの意味は次のとおりです。

- use-CR-first-cluster は、オープンキャパシティ予約を使用するクラスターの名前に置き換えられます。
- subnet-22XXXX01 は、サブネット ID に置き換えられます。

ターゲットキャパシティ予約を最初に使用する

Amazon EMR クラスターをプロビジョニングするときに、最低料金の配分戦略をオーバーライドし、使用可能なターゲットキャパシティ予約を最初に使用することを優先するように選択できます。この場合、Amazon EMR は起動リクエストで指定された、ターゲットキャパシティ予約を使用するすべてのインスタンスプールを評価し、リクエストされたすべてのコアノードを起動するのに十分な容量を持つ最低料金のインスタンスプールを選択します。ターゲットキャパシティ予約を使用するインスタンスプールに、コアノードのための十分な容量がない場合、Amazon EMR は前のトピックで説明したベストエフォートのケースにフォールバックします。つまり、Amazon EMR は起動リクエストで指定されたすべてのインスタンスプールを評価し直し、リクエストされたすべてのコアノードを起動するのに十分な容量を持つ最低料金のインスタンスプールを選択します。インスタンスプールに一致する使用可能なオープンキャパシティ予約が自動的に適用されます。ただし、ターゲットキャパシティ予約は未使用のままです。

コアノードがプロビジョニングされると、アベイラビリティーゾーンが選択され、修正されます。Amazon EMR は、すべてのタスクノードがプロビジョニングされるまで、選択したアベイラビリティーゾーン内の最低料金のものから、ターゲットキャパシティ予約を使用するインスタンスプールにタスクノードをプロビジョニングします。Amazon EMR は、選択したアベイラビリティーゾーン内の各インスタンスプールで使用可能なターゲットキャパシティ予約を最初に使用しようとします。次に、必要な場合にのみ、Amazon EMR は最低料金戦略を使用して、残りのタスクノードをプロビジョニングします。

ターゲットキャパシティ予約を最初に使用するための Amazon EMR 容量の割り当てロジックのユースケースを次に示します。

例 1: 起動リクエストで使用可能なターゲットキャパシティ予約があるインスタンスプールに、コアノードのための十分な容量がある

この場合、Amazon EMR は、インスタンスプールの料金に関係なく、使用可能なターゲットキャパシティ予約があるインスタンスプールでキャパシティを起動します。その結果、すべてのコアノードがプロビジョニングされるまで、可能な限りターゲットキャパシティ予約が使用されます。

オンデマンド戦略	lowest-price
使用戦略	use-capacity-reservations-first
リクエストされたキャパシティ	100

インスタンスタイプ	c5.xlarge	m5.xlarge	r5.xlarge
使用可能なターゲットキャパシティ予約	-	-	150
オンデマンド料金	$	$$	$$$
プロビジョニングされたインスタンス	-	-	100
使用されたターゲットキャパシティ予約	-	-	100
使用可能なターゲットキャパシティ予約	-	-	50

Example 例 2: 起動リクエストで使用可能なターゲットキャパシティ予約があるインスタンスプールに、コアノードのための十分な容量がない

オンデマンド戦略	lowest-price		
リクエストされたキャパシティ	100		
使用戦略	use-capacity-reservations-first		
インスタンスタイプ	c5.xlarge	m5.xlarge	r5.xlarge
使用可能なターゲットキャパシティ予約	10	50	50
オンデマンド料金	$	$$	$$$
プロビジョニングされたインスタンス	100	-	-
使用するキャパシティ予約の目標	10	-	-

使用可能なターゲットキャパシティ予約	-	50	50

インスタンスフリートの起動後、describe-capacity-reservations を実行して、未使用のままのキャパシティ予約の数を確認できます。

ターゲットキャパシティ予約を最初に使用するようにインスタンスフリートを設定する

RunJobFlow アクションを使用して、インスタンスフリートベースのクラスターを作成する場合、オンデマンド配分戦略を lowest-price に、CapacityReservationOptions の UsageStrategy を use-capacity-reservations-first に、そして CapacityReservationOptions のCapacityReservationResourceGroupArn を <your resource group ARN> に設定します。詳細については、「Amazon EC2 ユーザーガイド」の「キャパシティ予約の操作」を参照してください。 Amazon EC2

```
"LaunchSpecifications":
    {"OnDemandSpecification": {
        "AllocationStrategy": "lowest-price",
        "CapacityReservationOptions":
         {
            "UsageStrategy": "use-capacity-reservations-first",
            "CapacityReservationResourceGroupArn": "arn:aws:resource-groups:sa-
east-1:123456789012:group/MyCRGroup"
         }
       }
    }
```

arn:aws:resource-groups:sa-east-1:123456789012:group/MyCRGroup は、リソースグループ ARN に置き換えられます。

また、Amazon EMR CLI により、ターゲットキャパシティ予約を使用してインスタンスフリートベースのクラスターを作成することもできます。

```
aws emr create-cluster \
  --name 'targeted-CR-cluster' \
  --release-label emr-5.30.0 \
  --service-role EMR_DefaultRole \
  --ec2-attributes SubnetId=subnet-22XXXX01,InstanceProfile=EMR_EC2_DefaultRole \
```

```
  --instance-fleets
 InstanceFleetType=MASTER,TargetOnDemandCapacity=1,InstanceTypeConfigs=['{InstanceType=c4.xlarg
 \
    InstanceFleetType=CORE,TargetOnDemandCapacity=100,\
InstanceTypeConfigs=['{InstanceType=c5.xlarge},{InstanceType=m5.xlarge},
{InstanceType=r5.xlarge}'],\
LaunchSpecifications={OnDemandSpecification='{AllocationStrategy=lowest-
price,CapacityReservationOptions={UsageStrategy=use-capacity-reservations-
first,CapacityReservationResourceGroupArn=arn:aws:resource-groups:sa-
east-1:123456789012:group/MyCRGroup}}'}
```

各パラメータの意味は次のとおりです。

- `targeted-CR-cluster` は、ターゲットキャパシティ予約を使用するクラスターの名前に置き換えられます。
- `subnet-22XXXX01` は、サブネット ID に置き換えられます。
- `arn:aws:resource-groups:sa-east-1:123456789012:group/MyCRGroup` は、リソースグループ ARN に置き換えられます。

使用可能なオープンキャパシティ予約を使用しないようにする

Example

Amazon EMR クラスターの起動時にオープンキャパシティの予約のいずれかが予期せず使用されないようにするには、オンデマンド配分戦略を `lowest-price` に、`CapacityReservationOptions` の `CapacityReservationPreference` を`none` に設定します。そうしないと、Amazon EMR はオンデマンドインスタンスのキャパシティ予約設定をデフォルトの `open` に設定して、使用可能なオープンキャパシティ予約をベストエフォートベースで使用しようとします。

```
"LaunchSpecifications":
    {"OnDemandSpecification": {
        "AllocationStrategy": "lowest-price",
        "CapacityReservationOptions":
         {
            "CapacityReservationPreference": "none"
         }
       }
    }
```

また、Amazon EMR CLI により、オープンキャパシティ予約を使用しないでインスタンスフリートベースのクラスターを作成することもできます。

```
aws emr create-cluster \
  --name 'none-CR-cluster' \
  --release-label emr-5.30.0 \
  --service-role EMR_DefaultRole \
  --ec2-attributes SubnetId=subnet-22XXXX01,InstanceProfile=EMR_EC2_DefaultRole \
  --instance-fleets \

 InstanceFleetType=MASTER,TargetOnDemandCapacity=1,InstanceTypeConfigs=['{InstanceType=c4.xlarg
 \

 InstanceFleetType=CORE,TargetOnDemandCapacity=100,InstanceTypeConfigs=['{InstanceType=c5.xlarg
{InstanceType=m5.xlarge},{InstanceType=r5.xlarge}'],\
LaunchSpecifications={OnDemandSpecification='{AllocationStrategy=lowest-
price,CapacityReservationOptions={CapacityReservationPreference=none}}'}
```

各パラメータの意味は次のとおりです。

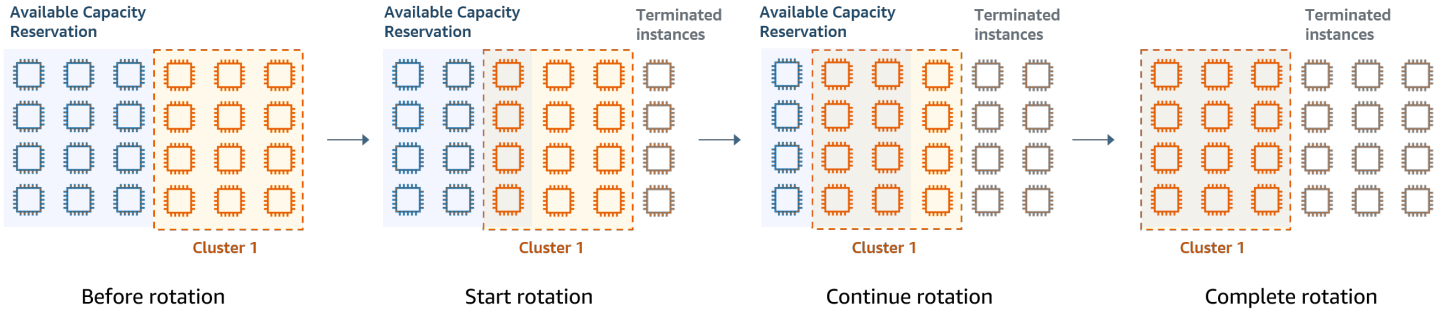
- none-CR-cluster は、オープンキャパシティ予約を使用していないクラスターの名前に置き換えられます。
- subnet-22XXXX01 は、サブネット ID に置き換えられます。

キャパシティ予約を使用するシナリオ

次のシナリオでキャパシティ予約を使用すると利点があります。

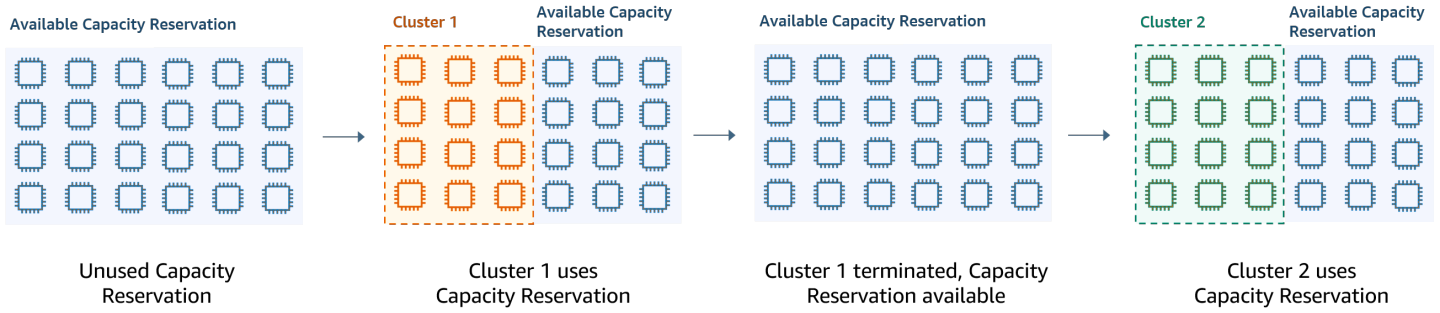
シナリオ 1: キャパシティ予約を使用して長時間稼働するクラスターをローテーションする

長時間稼働するクラスターをローテーションする場合、プロビジョニングする新しいインスタンスのインスタンスタイプとアベイラビリティーゾーンに厳格な要件がある場合があります。キャパシティ予約では、キャパシティ保証を使用して、中断することなくクラスターのローテーションを完了できます。



シナリオ 2: キャパシティ予約を使用して存続期間の短い連続したクラスターをプロビジョニングする

また、キャパシティ予約を使用して、個々のワークロードに対して存続期間の短い連続するクラスターのグループをプロビジョニングして、あるクラスターを終了するときに次のクラスターがキャパシティ予約を使用できるようにすることもできます。ターゲットキャパシティ予約を使用して、目的のクラスターだけがキャパシティ予約を使用するようにできます。



ユニフォームインスタンスグループを設定する

インスタンスグループの構成では、各ノードタイプ (マスター、コア、またはタスク) は、同じインスタンスタイプで構成されており、オンデマンドまたはスポットインスタンスで、同じ購入オプションが使用されます。これらの設定は、インスタンスグループを作成するときに指定します。この価格を後で変更することはできません。ただし、同じタイプと同じ購入オプションのインスタンスを、コアインスタンスグループおよびタスクインスタンスグループに追加できます。インスタンスを削除することもできます。

クラスターのオンデマンドインスタンスが、アカウントで使用可能なオープンキャパシティ予約の属性 (インスタンスタイプ、プラットフォーム、テナンシー、アベイラビリティーゾーン) と一致する場合、キャパシティ予約は自動的に適用されます。プライマリノード、コアノード、およびタスクノードには、オープンキャパシティ予約を使用できます。ただし、インスタンスグループを使用してクラスターをプロビジョニングする場合、ターゲットキャパシティ予約を使用したり、一致する属性を持つオープンキャパシティ予約でインスタンスが起動しないようにしたりすることはできません。

ターゲットキャパシティ予約を使用するか、またはインスタンスがオープンキャパシティ予約で起動しないようにする場合、代わりにインスタンスフリートを使用します。詳細については、「インスタンスフリートでキャパシティ予約を使用する」を参照してください。

クラスターの作成後に異なるインスタンスタイプを追加するには、タスクインスタンスグループを追加することができます。インスタンスグループごとに、異なるインスタンスタイプと購入オプションを選択できます。詳細については、「クラスターのスケーリングを使用する」を参照してください。

インスタンスを起動するときに、オンデマンドインスタンスのキャパシティ予約設定がデフォルトの open に設定されるため、一致する属性 (インスタンスタイプ、プラットフォーム、アベイラビリティーゾーン) を持つすべてのオープンキャパシティ予約で実行できます。オンデマンドキャパシティ予約の共有については、「インスタンスフリートでキャパシティ予約を使用する」を参照してください 。

このセクションでは、ユニフォームインスタンスグループでクラスターを作成する手順を説明します手動でのインスタンスグループの追加または削除、あるいは移動スケーリングによって既存のインスタンスグループを変更する詳細については、「 クラスターを管理する」を参照してください。

コンソールを使用してユニフォームインスタンスグループを設定する

> **Note**
>
> Amazon EMR コンソールは、再設計され、使いやすくなりました。古いコンソールと新しいコンソールエクスペリエンスの違いについては、「Amazon EMR コンソール」を参照してください。

New console

新しいコンソールを使用してインスタンスグループでクラスターを作成するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で、[クラスター] を選択し、[クラスターの作成] を選択します。
3. [クラスターの設定] で [インスタンスグループ] を選択します。
4. [ノードグループ] には、ノードグループのタイプごとにセクションがあります。プライマリノードグループで 3 つのプライマリノードを作成する場合は、[複数のプライマリノードを

使用] チェックボックスを選択します。スポット購入を使用する場合は、[スポット購入オプションを使用] チェックボックスを選択します。

5. プライマリノードグループとコアノードグループでは、[インスタンスタイプの追加] を選択し、最大 5 つのインスタンスタイプを選択します。タスクグループでは、[インスタンスタイプの追加] を選択し、最大 15 個のインスタンスタイプを選択します。Amazon EMR は、クラスターを起動するときに、これらのインスタンスタイプの任意の組み合わせをプロビジョニングする場合があります。
6. 各ノードグループタイプで、各インスタンスの横にある [アクション] ドロップダウンメニューを選択し、これらの設定を変更します。

   EBS ボリュームの追加

   Amazon EMR がプロビジョニングした後に、インスタンスタイプにアタッチする EBS ボリュームを指定します。

   最大スポット価格の編集

   フリートのインスタンスタイプごとに、最大スポット料金を指定します。この料金は、オンデマンド料金のパーセンテージまたは特定の金額として設定できます。アベイラビリティーゾーンの現在のスポット料金が最大スポット料金を下回っている場合、Amazon EMR はスポットインスタンスをプロビジョニングします。お客様にご負担いただくのはスポット料金であり、必ずしも最大スポット料金ではありません。
7. オプションで、[ノード設定] を展開して JSON 設定を入力するか、Amazon S3 から JSON をロードします。
8. クラスターに適用するその他のオプションを選択します。
9. クラスターを起動するには、[クラスターの作成] を選択します。

Old console

次の手順は、クラスターを作成するときの [Advanced Options (詳細オプション)] を示しています。[Quick options (クイックオプション)] によっても、インスタンスグループ構成でクラスターを作成できます。

古いコンソールを使用してユニフォームインスタンスグループでクラスターを作成するには

1. 新しい Amazon EMR コンソールに移動し、サイドナビゲーションから [古いコンソールに切り替え] を選択します。古いコンソールに切り替えたときの動作の詳細については、「Using the old console」を参照してください。

2. [クラスターを作成] を選択します。
3. [Go to advanced options (詳細オプションに移動する)] を選択し、[ソフトウェアの構成] オプションに入り、[次へ] を選択します。
4. [Hardware Configuration (ハードウェア構成)] 画面で、[Uniform instance groups (ユニフォームインスタンスグループ)] を選択した状態のままにします。
5. [Network (ネットワーク)] を選択し、クラスター用の [EC2 Subnet (EC2 サブネット)] を選択します。選択したサブネットが、各サブネットにリストされたアベイラビリティーゾーンと関連付けられます。詳細については、「ネットワークを設定する」を参照してください。

> **Note**
>
> アカウントおよびリージョンによっては、[ネットワーク] に [EC2-Classic に起動] を選択するオプションが提供されます。そのオプションを選択する場合、[EC2 サブネット] ではなく、[EC2 アベイラビリティーゾーン] を選択します。詳細については、Amazon EC2 ユーザーガイド」の「Amazon EC2 と Amazon VPCAmazon EC2」を参照してください。

6. 各 [Node type (ノードタイプ)] 行で次の手順に従います。

- [ノードタイプ] の下で、インスタンスグループのデフォルト名を変更する場合、鉛筆アイコンを選択してわかりやすい名前を入力します。[タスク] インスタンスグループを削除する場合、X アイコンを選択してください。追加の [タスク] インスタンスグループを追加するには、[タスクインスタンスグループを追加] を選択します。
- [インスタンスタイプ] の下で、鉛筆アイコンを選択し、そのノードタイプに使用するインスタンスタイプを選択します。

> **Important**
>
> を使用してインスタンスタイプを選択すると AWS Management Console、各インスタンスタイプに表示される vCPU の数は、そのインスタンスタイプの EC2 vCPU の数ではなく、そのインスタンスタイプの YARN vCPUsの数になります。各インスタンスタイプの vCPU 数の詳細については、「Amazon EC2 インスタンスタイプ」を参照してください。

- [インスタンスタイプ] で、[設定] の鉛筆アイコンを選択し、インスタンスグループごとにアプリケーションの設定を編集します。

- [Instance count (インスタンス数)] に、各ノードタイプに使用するインスタンス数を入力します。
- [Purchasing option] (購入のオプション) で、[オンデマンド] または [スポット] を選択します。[スポット] を選択した場合は、スポットインスタンスの上限価格のオプションを選択します。デフォルトでは、[最高価格としてオンデマンドを使用] が選択されます。[最高価格 ¥/時の設定] を選択し、上限価格を入力できます。選択した [EC2 Subnet (EC2 サブネット)] のアベイラビリティーゾーンは、[Maximum Spot price (最大スポット料金)] を下回っています。

  > **Tip**
  >
  > [スポット] の情報ツールヒントの上にカーソルを置き、現在のリージョンのアベイラビリティーゾーンに対する現在のスポット料金を確認します。最低のスポット価格は緑色で表示されます。[EC2 Subnet (EC2 サブネット)] の選択の変更に、この情報を使用できます。

- [Auto Scaling for Core and Task node types (コアノードタイプとタスクノードタイプの自動スケーリング)] で、鉛筆アイコンをクリックして、自動スケーリングオプションを設定します。詳細については、「カスタムポリシーによる自動スケーリングをインスタンスグループに使用する」を参照してください。

7. 必要に応じて [Add task instance group (タスクインスタンスグループの追加)] を選択して、前の手順で説明したように設定します。
8. [次へ] を選択してクラスター設定を変更し、続いてクラスターを起動します。

を使用して AWS CLI ユニフォームインスタンスグループを持つクラスターを作成する

AWS CLIを使用してクラスターのインスタンスグループ設定を指定するには、`create-cluster` コマンドと共に `--instance-groups` パラメータを使用します。インスタンスグループに `BidPrice` 引数を指定しない限り、Amazon EMR はオンデマンドインスタンスオプションを想定します。オンデマンドインスタンスとさまざまなクラスターオプションで、ユニフォームインスタンスグループを起動する `create-cluster` コマンドの例では、コマンドラインに `aws emr create-cluster help` と入力するか、「AWS CLI コマンドリファレンス」の「create-cluster」を参照してください。

を使用して AWS CLI 、スポットインスタンスを使用するクラスターにユニフォームインスタンスグループを作成できます。提供されるスポット価格はアベイラビリティーゾーンによって異なります。CLI または API を使用する場合、`AvailabilityZone` 引数 (EC2-classic ネットワークを使

用している場合)、または --ec2-attributes パラメータの SubnetID 引数でアベイラビリティーゾーンを指定することができます。選択するアベイラビリティーゾーンまたはサブネットはクラスターに適用されるため、すべてのインスタンスグループに使用されます。アベイラビリティーゾーンまたはサブネットを明示的に指定しない場合、Amazon EMR はクラスターの起動時に最低のスポット料金のアベイラビリティーゾーンを選択します。

次の例は、すべてがスポットインスタンスを使用しているプライマリ、コア、および 2 つのタスクインスタンスグループを作成する、create-cluster コマンドを示しています。*myKey* を Amazon EC2 キーペアの名前に置き換えます。

> **Note**
>
> 読みやすくするために、Linux 行連続文字 (\) が含まれています。Linux コマンドでは、これらは削除することも、使用することもできます。Windows の場合、削除するか、キャレット (^) に置き換えてください。

```
aws emr create-cluster --name "MySpotCluster" \
  --release-label emr-7.1.0 \
  --use-default-roles \
  --ec2-attributes KeyName=myKey \
  --instance-groups \
    InstanceGroupType=MASTER,InstanceType=m5.xlarge,InstanceCount=1,BidPrice=0.25 \
    InstanceGroupType=CORE,InstanceType=m5.xlarge,InstanceCount=2,BidPrice=0.03 \
    InstanceGroupType=TASK,InstanceType=m5.xlarge,InstanceCount=4,BidPrice=0.03 \
    InstanceGroupType=TASK,InstanceType=m5.xlarge,InstanceCount=2,BidPrice=0.04
```

CLI を使用して、インスタンスグループ内のインスタンスタイプごとに一意のカスタム AMI を指定するユニフォームインスタンスグループクラスターを作成できます。これにより、同じインスタンスグループ内で異なるインスタンスアーキテクチャを使用できます。各インスタンスタイプは、アーキテクチャが一致するカスタム AMI を使用する必要があります。例えば、m5.xlarge インスタンスタイプは x86_64 アーキテクチャのカスタム AMI を使用して設定し、m6g.xlarge インスタンスタイプは対応する AWS AARCH64 (ARM) アーキテクチャのカスタム AMI を使用して設定します。

次の例は、2 つのインスタンスタイプで作成され、それぞれ独自のカスタム AMI を持つユニフォームインスタンスグループクラスターを示しています。カスタム AMI は、クラスターレベルではなく、インスタンスタイプレベルでのみ指定されることに注意してください。これは、インスタンスタイプ AMI とクラスターレベルの AMI の間の競合を回避するためです。そうしないと、クラスターの起動が失敗する原因となります。

```
aws emr create-cluster
  --release-label emr-5.30.0 \
  --service-role EMR_DefaultRole \
  --ec2-attributes SubnetId=subnet-22XXXX01,InstanceProfile=EMR_EC2_DefaultRole \
  --instance-groups \

 InstanceGroupType=MASTER,InstanceType=m5.xlarge,InstanceCount=1,CustomAmiId=ami-123456
 \

 InstanceGroupType=CORE,InstanceType=m6g.xlarge,InstanceCount=1,CustomAmiId=ami-234567
```

実行中のクラスターに追加するインスタンスグループに複数のカスタム AMI を追加できます。次の例に示すように、CustomAmiId 引数を add-instance-groups コマンドとともに使用できます。

```
aws emr add-instance-groups --cluster-id j-123456 \
  --instance-groups \

 InstanceGroupType=Task,InstanceType=m5.xlarge,InstanceCount=1,CustomAmiId=ami-123456
```

Java SDK を使用してインスタンスグループを作成する

クラスターのインスタンスグループの構成を指定する InstanceGroupConfig オブジェクトをインスタンス化します。スポットインスタンスを使用するには、 withBidPrice オブジェクトで、withMarket および InstanceGroupConfig プロパティを設定します。次のコードは、スポットインスタンスを実行するプライマリ、コア、およびタスクインスタンスグループを定義する方法を示します。

```
InstanceGroupConfig instanceGroupConfigMaster = new InstanceGroupConfig()
 .withInstanceCount(1)
 .withInstanceRole("MASTER")
 .withInstanceType("m4.large")
 .withMarket("SPOT")
 .withBidPrice("0.25");

InstanceGroupConfig instanceGroupConfigCore = new InstanceGroupConfig()
 .withInstanceCount(4)
 .withInstanceRole("CORE")
 .withInstanceType("m4.large")
 .withMarket("SPOT")
 .withBidPrice("0.03");
```

```
InstanceGroupConfig instanceGroupConfigTask = new InstanceGroupConfig()
 .withInstanceCount(2)
 .withInstanceRole("TASK")
 .withInstanceType("m4.large")
 .withMarket("SPOT")
 .withBidPrice("0.10");
```

インスタンスとアベイラビリティーゾーンの柔軟性に関するベストプラクティス

各 AWS リージョン には、アベイラビリティーゾーンと呼ばれる複数の独立した場所があります。インスタンスを起動するときは、必要に応じて、使用している AWS リージョン でアベイラビリティーゾーン (AZ) を指定できます。アベイラビリティーゾーンの柔軟性とは、インスタンスを複数の AZ に分散させることです。あるインスタンスで障害が発生しても、別の AZ のインスタンスでリクエストを処理できるようにアプリケーションを設計できます。詳細については、「Amazon EC2 ユーザーガイド」の「リージョンとゾーン」のドキュメントを参照してください。

インスタンスの柔軟性とは、容量要件を満たすために複数のインスタンスタイプを使用することです。インスタンスに柔軟性を持たせると、インスタンスのサイズ、ファミリー、世代をまたいで集計された容量を使用できます。柔軟性が高いほど、単一のインスタンスタイプを使用するクラスターと比較して、必要な量のコンピューティング容量を見つけて割り当てられる可能性が高くなります。

インスタンスとアベイラビリティーゾーンの柔軟性により、インスタンスタイプまたは AZ が 1 つのクラスターに比べて、容量不足エラー (ICE) やスポットの中断が少なくなります。最初のインスタンスファミリーとサイズがわかったら、ここで説明するベストプラクティスを参考にして、どのインスタンスを分散させるかを判断してください。このアプローチは、パフォーマンスとコストの変動を最小限に抑えながら、Amazon EC2 容量プールの可用性を最大化します。

アベイラビリティーゾーンに関する柔軟性

すべてのアベイラビリティーゾーンを仮想プライベートクラウド (VPC) で使用するように設定し、EMR クラスター用に選択することをお勧めします。クラスターは 1 つのアベイラビリティーゾーンにのみ存在する必要がありますが、Amazon EMR インスタンスフリートでは、アベイラビリティーゾーンごとに複数のサブネットを選択できます。Amazon EMR は、クラスターを起動するときに、指定されたインスタンスと購入オプションをこれらのサブネットで探します。複数のサブネットの EMR クラスターをプロビジョニングすると、クラスターは、単一のサブネット内のクラスターと比較して、より深い Amazon EC2 キャパシティプールにアクセスできます。

EMR クラスターの仮想プライベートクラウド (VPC) で特定の数のアベイラビリティーゾーンを優先的に使用する必要がある場合は、Amazon EC2 のスポットプレイスメントスコア機能を活用できま

す。スポットプレイスメントスコアリングでは、スポットインスタンスのコンピューティング要件を指定し、EC2 は 1～10 のスケールでスコア付けされた上位 10 個の AWS リージョン またはアベイラビリティーゾーンを返します。スコア 10 はスポットリクエストが成功する可能性が高いことを示し、スコア 1 はスポットリクエストが成功する可能性が低いことを示します。スポットプレイスメントスコアリングの使用方法の詳細については、「Amazon EC2 ユーザーガイド」の「スポットプレイスメントスコア」を参照してください。 Amazon EC2

インスタンスタイプに関する柔軟性

インスタンスの柔軟性とは、容量要件を満たすために複数のインスタンスタイプを使用することです。インスタンスの柔軟性は Amazon EC2 スポットインスタンスとオンデマンドインスタンスの使用の両方にメリットをもたらします。スポットインスタンスでは、インスタンスの柔軟性により、Amazon EC2 はリアルタイムの容量データを使用して、より深い容量プールからインスタンスを起動できます。また、どのインスタンスが最も可用性が高いかを予測します。これにより、中断が少なくなり、ワークロード全体のコストを削減できます。オンデマンドインスタンスを使用すると、インスタンスが柔軟になり、より多くのインスタンスプールに合計キャパシティをプロビジョニングする場合の、キャパシティ不足エラー (ICE) が減少します。

インスタンスグループクラスターでは、最大 50 の EC2 インスタンスタイプを指定できます。配分戦略付きのインスタンスフリートでは、プライマリ、コア、タスクノードグループごとに最大 30 の EC2 インスタンスタイプを指定できます。インスタンスの範囲が広いほど、インスタンスの柔軟性のメリットも高まります。

インスタンスに柔軟性を持たせる

アプリケーションのインスタンスに柔軟性を持たせるには、次のベストプラクティスを考慮してください。

トピック

- インスタンスファミリーとサイズを決定する
- 追加のインスタンスを含める

インスタンスファミリーとサイズを決定する

Amazon EMR は、異なるユースケース向けに複数のインスタンスタイプをサポートします。これらのインスタンスタイプは「サポートされるインスタンスタイプ」ドキュメントに記載されています。各インスタンスタイプは、そのタイプがどのアプリケーションに最適化されているかを説明するインスタンスファミリーに属します。

新しいワークロードでは、m5 や c5 などの汎用ファミリーのインスタンスタイプでベンチマークする必要があります。次に、Ganglia および Amazon CloudWatch の OS と YARN のメトリクスを監視し、ピーク負荷時のシステムボトルネックを特定します。ボトルネックには、CPU、メモリ、ストレージ、I/O 操作が含まれます。ボトルネックを特定したら、コンピューティング最適化、メモリ最適化、ストレージ最適化、またはインスタンスタイプに適した他のインスタンスファミリーを選択します。詳細については、「」の「Amazon EMR ベストプラクティスガイド」の「Spark ワークロードに適したインフラストラクチャを決定する」ページを参照してください GitHub。

次に、アプリケーションに必要な最小の YARN コンテナまたは Spark エグゼキューターを特定します。これはコンテナに収まる最小のインスタンスサイズであり、クラスターの最小インスタンスサイズでもあります。この指標を使用して、さらに分散できるインスタンスを特定してください。インスタンスが小さいほど、インスタンスの柔軟性が高まります。

インスタンスの柔軟性を最大限に高めるには、できるだけ多くのインスタンスを活用する必要があります。ハードウェア仕様が似ているインスタンスを使用して分散することをおすすめします。これにより、コストとパフォーマンスの変動を最小限に抑えながら EC2 容量プールに最大限にアクセスできるようになります。規模を問わず分散できます。そのためには、 AWS Graviton とそれ以前の世代を優先してください。一般的に、ワークロードごとに少なくとも 15 種類のインスタンスタイプの間で柔軟に対応してみてください。汎用インスタンス、コンピューティング最適化インスタンス、またはメモリ最適化インスタンスから開始することをお勧めします。これらのインスタンスタイプは最大の柔軟性を提供します。

追加のインスタンスを含める

多様性を最大限に高めるには、追加のインスタンスタイプを含めます。まず、インスタンスサイズ、Graviton、および生成の柔軟性を優先します。これにより、コストとパフォーマンスのプロファイルが似ている追加の EC2 キャパシティプールにアクセスできるようになります。ICE やスポットでの中断によりさらに柔軟性が必要な場合は、バリアントとファミリーの柔軟性を検討してください。それぞれのアプローチには、ユースケースと要件に応じたトレードオフがあります。

- サイズの柔軟性 — まず、同じファミリー内でサイズの異なるインスタンスを分散させます。同じファミリー内のインスタンスはもコストとパフォーマンスは同じですが、ホストごとに異なる数のコンテナを起動できます。例えば、必要なエグゼキューターの最小サイズが 2vCPU と 8Gb メモリの場合、最小インスタンスサイズは `m5.xlarge` です。サイズを柔軟にするために、`m5.xlarge`、`m5.2xlarge`、`m5.4xlarge`、`m5.8xlarge`、`m5.12xlarge`、`m5.16xlarge` および `m5.24xlarge` を含めてください。
- Graviton の柔軟性 — Graviton インスタンスでは、サイズだけでなく分散も可能です。Graviton インスタンスは、Amazon EC AWS 2 のクラウドワークロードに最適

な価格パフォーマンスを提供する Graviton2 プロセッサを搭載しています。 Amazon EC2 例えば、m5.xlarge の最小インスタンスサイズに、Graviton の柔軟性のために、m6g.xlarge、m6g.2xlarge、m6g.4xlarge、m6g.8xlarge および m6g.16xlarge を含めることができます。

- 生成の柔軟性 — Graviton やサイズの柔軟性と同様に、前世代のファミリーのインスタンスは同じハードウェア仕様を共有しています。これにより、アクセス可能な Amazon EC2 プールの合計が増加し、同様のコストとパフォーマンスのプロファイルになります。生成の柔軟性を高めるため、m4.xlarge、m4.2xlarge、m4.10xlarge、および m4.16xlarge を含めます。
- ファミリーとバリアントの柔軟性
  - キャパシティ — キャパシティを最適化するには、インスタンスファミリー全体でインスタンスに柔軟性を持たせることをお勧めします。異なるインスタンスファミリーの一般的なインスタンスには、容量要件を満たすのに役立つより深いインスタンスプールがあります。ただし、ファミリーのインスタンスが異なれば、vCPU とメモリの比率は異なります。このため、想定されるアプリケーションコンテナのサイズが別のインスタンス用であれば、十分に利用できないことになります。例えば m5.xlarge には、インスタンスファミリーの柔軟性のために、コンピューティング最適化インスタンス (c5 など) やメモリ最適化インスタンス (r5 など) が含まれます。
  - コスト — コストを最適化するには、バリアント全体でインスタンスに柔軟性を持たせることをお勧めします。これらのインスタンスのメモリと vCPU の比率は初期インスタンスと同じです。バリアントの柔軟性とのトレードオフは、これらのインスタンスの容量プールが小さいため、追加の容量が制限されたり、スポットの中断が増えたりする可能性があることです。例えば m5.xlarge では、インスタンスバリアントの柔軟性のために、AMD ベースのインスタンス (m5a)、SSD ベースのインスタンス (m5d)、またはネットワーク最適化インスタンス (m5n) を含めます。

クラスター設定のベストプラクティス

このセクションのガイドラインを使用して、インスタンスタイプ、購入オプション、EMR クラスター内の各ノードタイプをプロビジョニングするためのストレージの容量を決定することができます。

使用すべきインスタンスタイプ

Amazon EC2 インスタンスをクラスターに追加する方法はいくつかあります。選択の必要がある方法は、クラスターにインスタンスグループ設定を使用しているか、またはインスタンスフリート設定を使用しているかによって異なります。

- インスタンスグループ

  - 既存のコアおよびタスクインスタンスグループと同じタイプのインスタンスを手動で追加します。
  - 手動でタスクインスタンスグループを追加します。これには、別のインスタンスタイプを使用できます。
  - インスタンスグループの Amazon EMR で自動スケーリングを設定し、指定した Amazon CloudWatch メトリクスの値に基づいてインスタンスを自動的に追加および削除します。詳細については、「クラスターのスケーリングを使用する」を参照してください。
- インスタンスフリート
  - 単一のタスクインスタンスフリートを追加します。
  - 既存のコアおよびタスクインスタンスフリートにオンデマンドおよびスポットインスタンスのターゲット容量を変更します。詳細については、「インスタンスフリートを設定する」を参照してください。

クラスターのインスタンスを計画する 1 つの方法は、代表的なデータのサンプルセットで、テストクラスターを実行し、クラスター内のノードの使用状況を監視することです。詳細については、「クラスターを表示し、モニタリングする」を参照してください。別の方法は、考慮するインスタンスの容量を計算し、その値をデータのサイズに対して比較することです。

一般的に、タスクを割り当てるプライマリノードタイプでは、処理能力の大きい EC2 インスタンスは必要ありません。タスクを処理して HDFS にデータを保存するコアノードタイプの Amazon EC2 インスタンスは、処理能力とストレージ容量の両方が必要であり、データを保存しないタスクノードタイプの Amazon EC2 インスタンスは、処理能力のみが必要です。利用可能な Amazon EC2 インスタンスとその設定のガイドラインについては、「Amazon EC2 インスタンスを設定する」を参照してください。

ほとんどの Amazon EMR クラスターには次のガイドラインがあてはまります。

- AWS ごとにアカウントで実行するオンデマンド Amazon EC2 インスタンスの合計数には vCPU 制限があります AWS リージョン。vCPU の制限とアカウントでの制限の引き上げをリクエストする方法の詳細については、「Linux インスタンス用 Amazon EC2 ユーザーガイド」の「オンデマンドインスタンス」を参照してください。
- プライマリノードには通常、大量の計算要件がありません。多数のノードを持つクラスター、または特にプライマリノード (、Hue など) にデプロイされたアプリケーションを持つクラスターの場合JupyterHub、より大きなプライマリノードが必要になる場合があり、クラスターのパフォーマンスを向上させるのに役立ちます。例えば、小さなクラスター (50 以下のノード) には m5.xlarge

インスタンスを使用し、より大きなクラスターではより大きなインスタンスタイプに増やすことを検討します。

- コアノードとタスクノードの計算ニーズは、アプリケーションが実行する処理のタイプによって異なります。汎用のインスタンスタイプでは多くのジョブを実行できます。これは、CPU、ディスク領域、入出力に関して、バランスの取れたパフォーマンスを発揮します。計算集約的なクラスターは、比率的に RAM より CPU が多い、High CPU インスタンス上で実行するとメリットがあります。データベースおよびメモリキャッシュアプリケーションは、大きいメモリインスタンスで実行するとメリットがあります。解析、NLP、機械学習のようなネットワークと CPU を多く使用するアプリケーションは、クラスターコンピューティングインスタンスで実行すると利点があります。この場合、比例的に CPU リソースが大きくなり、ネットワークパフォーマンスが向上します。
- クラスターの段階によって必要な容量が異なる場合は、少ない数のコアノードから始め、タスクノードの数を増減してジョブフローでの容量要件の変化に合わせます。
- 処理可能なデータの量は、コアノードの容量と、入力、処理時、および出力のデータのサイズによって異なります。処理中、入力、中間、および出力データセットはすべてクラスターに存在します。

スポットインスタンスを使用すべき場合

Amazon EMR でクラスターを起動するとき、プライマリ、コア、またはタスクのインスタンスを選んでスポットインスタンス上で起動できます。各インスタンスグループタイプがクラスターではそれぞれ異なる役割を果たすので、スポットインスタンス上で各ノードタイプを起動したときの意味合いもそれぞれ異なります。クラスターの実行中にインスタンス購入オプションを変更することはできません。プライマリノードおよびコアノードのオンデマンドインスタンスをスポットインスタンスに変更する、またはその逆に変更するには、クラスターを終了して新しいクラスターを起動する必要があります。タスクノードについては、新しいタスクインスタンスグループまたはインスタンスフリートを開始し、古いものを削除できます。

トピック

- タスクノードのスポットインスタンスの終了によるジョブの失敗を防ぐ Amazon EMR 設定
- スポットインスタンス上のプライマリノード
- スポットインスタンス上のコアノード
- スポットインスタンス上のタスクノード
- アプリケーション向けのインスタンスの設定シナリオ

タスクノードのスポットインスタンスの終了によるジョブの失敗を防ぐ Amazon EMR 設定

スポットインスタンスはタスクノードの実行に使用されることが多いため、Amazon EMR には、タスクノードが終了しても実行中のジョブが失敗しないように YARN ジョブをスケジュールするための機能がデフォルトで備えられています。Amazon EMR は、アプリケーションマスタープロセスをコアノードでのみ実行できるようにすることで、これを実現しています。アプリケーションマスタープロセスは実行中のジョブを制御し、ジョブが有効である間は存続する必要があります。

Amazon EMR リリース 5.19.0 以降では、組み込みの YARN ノードラベル機能を使用して、これを実現しています。(以前のバージョンではコードパッチを使用していました)。`yarn-site` と `capacity-scheduler` の設定分類のプロパティは、YARN capacity-scheduler と fair-scheduler がノードラベルを利用できるように、デフォルトで設定されます。Amazon EMR は、`CORE` ラベルでコアノードに自動的にラベルを付け、アプリケーションマスターが CORE ラベルを持つノードでのみスケジュールされるようにプロパティを設定します。yarn-site および capacity-scheduler 設定分類の関連プロパティを手動で変更したり、関連する XML ファイルで直接変更したりすると、この機能が停止したり、この機能が変更されたりする可能性があります。

デフォルトでは、Amazon EMR は次のプロパティと値を設定します。これらのプロパティを設定するときは注意が必要です。

> **Note**
>
> Amazon EMR 6.x リリースシリーズ以降では、YARN ノードラベル機能はデフォルトで無効になっています。アプリケーションプライマリプロセスは、デフォルトでコアノードとタスクノードの両方で実行できます。次のプロパティを設定することで、YARN ノードラベル機能を有効にできます。
>
> - `yarn.node-labels.enabled: true`
> - `yarn.node-labels.am.default-node-label-expression: 'CORE'`

- 全ノード上の yarn-site (yarn-site.xml)
  - `yarn.node-labels.enabled: true`
  - `yarn.node-labels.am.default-node-label-expression: 'CORE'`
  - `yarn.node-labels.fs-store.root-dir: '/apps/yarn/nodelabels'`
  - `yarn.node-labels.configuration-type: 'distributed'`
- プライマリおよびコアノード上の yarn-site (yarn-site.xml)

  - yarn.nodemanager.node-labels.provider: 'config'
  - yarn.nodemanager.node-labels.provider.configured-node-partition: 'CORE'
- 全ノード上の capacity-scheduler (capacity-scheduler.xml)
  - yarn.scheduler.capacity.root.accessible-node-labels: '*'
  - yarn.scheduler.capacity.root.accessible-node-labels.CORE.capacity: 100
  - yarn.scheduler.capacity.root.default.accessible-node-labels: '*'
  - yarn.scheduler.capacity.root.default.accessible-node-labels.CORE.capacity: 100

スポットインスタンス上のプライマリノード

プライマリノードは、クラスターを制御し、指示を与えます。プライマリノードが終了するとクラスターも終了するので、プライマリノードは、突然終了しても問題が発生しないクラスターを実行している場合にのみ、スポットインスタンスとして起動するようにします。例えば、新しいアプリケーションをテストしている場合、Simple Storage Service (Amazon S3) などの外部ストアにクラスターのデータを定期的に保持している場合、またはクラスターが確実に完了することよりもコストの方が重要なクラスターを実行している場合などがこれに当てはまります。

プライマリインスタンスグループをスポットインスタンスとして起動する場合、クラスターは、そのスポットインスタンスリクエストが満たされるまで開始されません。最大スポット料金を選択する場合は、この点を考慮に入れてください。

スポットインスタンスプライマリノードを追加できるのは、クラスターの起動時のみです。実行中のクラスターに対してプライマリノードを追加したり削除したりすることはできません。

通常は、クラスター全体 (すべてのインスタンスグループ) をスポットインスタンスとして実行している場合にのみ、プライマリノードをスポットインスタンスとして実行します。

スポットインスタンス上のコアノード

コアノードはデータを処理して、HDFS を使用して情報を格納します。コアインスタンスを終了すると、データ損失のリスクがあります。このため、スポットインスタンス上でコアノードを実行するのは、HDFS での一部のデータ損失を許容できる場合に限る必要があります。

コアインスタンスグループをスポットインスタンスとして起動した場合、Amazon EMR がインスタンスグループを起動するのは、リクエストされたすべてのコアインスタンスのプロビジョニングが完了してからです。つまり、6 個の Amazon EC2 インスタンスをリクエストしても、最大スポット料金以下で使用できるものが 5 個しかない場合、インスタンスグループは起動しません。Amazon

EMR は、6 個の Amazon EC2 インスタンスがすべて使用可能になるまで、またはクラスターが終了するまで待機し続けます。1 つのコアインスタンスグループに含まれるスポットインスタンスの数を変更して、実行中のクラスターに容量を追加することができます。インスタンスグループの操作方法およびインスタンスフリートでのスポットインスタンスの動作の詳細については、「the section called “インスタンスフリートまたはインスタンスグループを設定する”」を参照してください。

スポットインスタンス上のタスクノード

タスクノードはデータを処理しますが、HDFS に永続的データを保持しません。スポット価格が最大スポット料金を上回ったためにクラスターが終了した場合、データは失われず、クラスターに対する影響も最小限に抑えられます。

1 つ以上のタスクインスタンスグループをスポットインスタンスとして起動すると、Amazon EMR は、最大スポット料金を使用して、可能な数だけタスクノードをプロビジョニングします。つまり、6 個のノードを持つタスクインスタンスグループをリクエストした場合、最大スポット料金以下で使用できるスポットインスタンスノードが 5 個しかないと、インスタンスグループは Amazon EMR によって 5 個のノードで起動され、使用できるようになると 6 個目が追加されます。

スポットインスタンスとしてタスクインスタンスグループを起動すると、最小限のコストで戦略的にクラスターの容量を拡大できます。プライマリインスタンスグループとコアインスタンスグループをオンデマンドインスタンスとして起動する場合、その容量はクラスターを実行するために確保されます。必要に応じて、タスクインスタンスグループにタスクインスタンスを追加し、ピークトラフィックの負荷に対応するか、データ処理の速度を上げます。

タスクノードを追加または削除するには、コンソール AWS CLI、または API を使用します。また、タスクグループを追加することもできますが、作成後にタスクグループを削除することはできません。

アプリケーション向けのインスタンスの設定シナリオ

次の表は、通常さまざまなアプリケーションシナリオに適しているノードタイプ購入オプションと構成のクイックリファレンスです。各シナリオタイプの詳細情報を表示するには、リンクを選択します。

アプリケーションシナリオ	プライマリノードの購入オプション	コアノード購入オプション	タスクノード購入オプション
長時間稼働クラスターおよびデータウェアハウス	オンデマンド	オンデマンドまたはインスタンスフ	スポットまたはインスタンスフリートの組み合わせ

アプリケーションシナリオ	プライマリノードの購入オプション	コアノード購入オプション	タスクノード購入オプション
		リートの組み合わせ	
コスト主導のワークロード	スポット	スポット	スポット
データクリティカルなワークロード	オンデマンド	オンデマンド	スポットまたはインスタンスフリートの組み合わせ
アプリケーションのテスト	スポット	スポット	スポット

スポットインスタンスを使って Amazon EMR クラスターを実行すると便利なシナリオがいくつかあります。

長時間稼働クラスターおよびデータウェアハウス

コンピュータの計算処理機能で変動を予測できる永続的な Amazon EMR クラスター (データウェアハウスなど) を実行している場合は、スポットインスタンスによって低コストでピーク時の需要に対応できます。プライマリインスタンスグループおよびコアインスタンスグループをオンデマンドインスタンスとして通常の容量に対応するように起動し、タスクインスタンスグループをスポットインスタンスとしてピーク負荷の要件に対応するように起動することができます。

コスト主導のワークロード

完了までの時間よりも、コストをかけないことの方が重要な一時クラスターを実行している場合、一部の作業が失われてもよいときは、クラスター全体 (プライマリ、コア、およびタスクインスタンスグループ) をスポットインスタンスとして実行して、最大限のコスト削減を実現できます。

データクリティカルなワークロード

完了までの時間よりも、コストをかけないことの方が重要な一時クラスターを実行している場合、すべての作業を保持する必要があるときは、プライマリインスタンスグループとコアインスタンスグループをオンデマンドインスタンスとして起動し、スポットインスタンスの 1 つ以上のタスクインスタンスグループで補完します。プライマリインスタンスグループとコアインスタンスグループをオンデマンドインスタンスとして実行すると、データが HDFS に確実に保持されるので、スポット市場の変動による終了からクラスターが保護されます。同時にスポットインスタンスとしてタスクインスタンスグループを実行することで、コストを削減できます。

アプリケーションのテスト

実稼働環境で起動できるよう準備する目的で新しいアプリケーションをテストする場合は、クラスター全体 (プライマリ、コア、およびタスクインスタンスグループ) をスポットインスタンスとして実行し、テストコストを削減できます。

クラスターの必要な HDFS 容量の計算

クラスターで利用できる HDFS ストレージの容量は、次の要因に応じて異なります。

- コアノードに使用する Amazon EC2 インスタンスの数。
- 使用するインスタンスタイプの Amazon EC2 インスタンスストアの容量。インスタンスストアボリュームの詳細については、Amazon EC2 ユーザーガイド」の「Amazon EC2 インスタンスストアAmazon EC2」を参照してください。
- コアノードにアタッチされている Amazon EBS ボリュームの数とサイズ。
- 各データブロックが RAID のような冗長性を実現するために　HDFS に保存されている方法を説明する、レプリケーションの要因。デフォルトで、レプリケーション係数は、10 個以上のノードのクラスターで 3、4 ~ 9 個のノードのクラスターで 2、3 個以下のノードのクラスターで 1 です。

クラスターの HDFS 容量を計算するには、各コアノードで、インスタンスストアボリュームの容量を Amazon EBS ストレージ容量 (使用している場合) に追加します。計算結果にコアノードの数をかけて、その合計をコアノードの数に基づいてレプリケーション係数で割ります。例えば、10 個のタイプ i2.xlarge のコアノードを持ち、800 GB のインスタンスストレージがあり、Amazon EBS ボリュームがアタッチされていないクラスターでは、HDFS に使用できるのは合計で約 2,666 GB です (10 ノード x 800 GB ÷ レプリケーション係数 3)。

計算された HDFS 容量の値がデータより小さい場合、次の方法で HDFS ストレージの容量を増やすことができます:

- 追加の Amazon EBS ボリュームでクラスターを作成する、または Amazon EBS ボリュームを既存のクラスターにアタッチしたインスタンスグループを追加する
- より多くのコアノードを追加する
- より大きいストレージ容量で、Amazon EC2 インスタンスタイプを選択する
- データ圧縮を使用する
- レプリケーション要素を減らすためにHadoop 構成設定を変更する

レプリケーション係数を減らすことは、HDFS データの冗長性や、クラスターの HDFS ブロックの損失や破損から復元する能力が低下するため、注意して使用する必要があります。

クラスターのログ記録とデバッグを設定する

クラスターの計画時に決定すべき事項の 1 つは、どの程度のデバッグサポートを必要とするかです。データ処理アプリケーションを初めて導入するお客様に対しては、小規模で典型的なデータサブセットを処理するアプリケーションをクラスターでテストすることをお勧めします。これを行う場合は、Amazon EMR が提供するあらゆるデバッグツール (Simple Storage Service (Amazon S3) へのログファイルのアーカイブなど) を利用する可能性があります。

導入を終了し、データ処理アプリケーションをフル稼働状態に移行させたら、デバッグの規模を縮小してもかまいません。デバッグの規模を縮小すると、Simple Storage Service (Amazon S3) にログファイルアーカイブを格納するのにかかるコストを節約できるほか、Simple Storage Service (Amazon S3) への状態の書き込みが必要でなくなるので、クラスターでの処理負荷を軽減することができます。もちろん、その反面、何かうまくいかないことがあっても問題を調査するのに使用できるツールは少なくなります。

デフォルトログファイル

デフォルトで各クラスターはプライマリノードにログファイルを書き込みます。その書き込み先は、`/mnt/var/log/` ディレクトリです。書き込まれたログファイルにアクセスするには、SSH を使用してプライマリノードに接続します (「SSH を使用してプライマリノードに接続する」を参照)。

> **Note**
>
> Amazon EMR リリース 6.8.0 以前を使用している場合、ログファイルはクラスターの終了時に Amazon S3 に保存されるため、プライマリノードが終了するとログファイルにアクセスできなくなります。Amazon EMR リリース 6.9.0 以降は、クラスターのスケールダウン中にログを Amazon S3 にアーカイブするため、クラスターで生成されたログファイルは、ノードが終了した後も保持されます。

何かの設定を有効にしなくても、ログファイルはプライマリノードに書き込まれます。これは、Amazon EMR および Hadoop のデフォルト動作です。

クラスターでは、以下のような数種類のログファイルが生成されます。

- ステップログ — これは Amazon EMR サービスによって生成されるログファイルであり、クラスターに関する情報と各ステップの結果を含みます。このログファイルは、プライマリノードの /mnt/var/log/hadoop/steps/ ディレクトリに格納されます。各ステップは個別に番号が振られたサブディレクトリにそれぞれの結果を記録します。すなわち、1 番目のステップの場合は /mnt/var/log/hadoop/steps/s-*stepId1*/ に記録され、2 番目のステップの場合は /mnt/var/log/hadoop/steps/s-*stepId2*/ に記録される、といった具合です。13 文字のステップ識別子 (stepId1、stepId2 など) は、クラスターに固有です。
- Hadoop および YARN コンポーネントログ — 例えば MapReduce、Apache YARN と の両方に関連付けられているコンポーネントのログは、 の個別のフォルダに含まれています/mnt/var/log。/mnt/var/log にある Hadoop コンポーネントのログファイルの場所は、hadoop-hdfs、hadoop-mapreduce、hadoop-httpfs、hadoop-yarn です。hadoop-state-pusher ディレクトリは Hadoop 状態プッシャープロセスの出力用です。
- ブートストラップアクションログ — ブートストラップアクションがジョブで使用された場合に、そのアクションの結果がログに記録されます。このログファイルは、プライマリノードの /mnt/var/log/bootstrap-actions/ ディレクトリに格納されます。各ブートストラップアクションは個別に番号が振られたサブディレクトリにそれぞれの結果を記録します。すなわち、1 番目のブートストラップアクションの場合は /mnt/var/log/bootstrap-actions/1/ に記録され、2 番目のブートストラップアクションの場合は /mnt/var/log/bootstrap-actions/2/ に記録される、といった具合です。
- インスタンス状態ログ — このログは、CPU に関する情報、メモリの状態、およびノードのガベージコレクタースレッドです。このログファイルは、プライマリノードの /mnt/var/log/instance-state/ に格納されます。

Simple Storage Service (Amazon S3) にログファイルをアーカイブする

> **Note**
>
> 現在、yarn logs ユーティリティを使って Simple Storage Service (Amazon S3) にログを集計することはできません。

Amazon EMR リリース 6.9.0 以降は、クラスターのスケールダウン中にログを Amazon S3 にアーカイブするため、クラスターで生成されたログファイルは、ノードが終了した後も保持されます。この動作は自動的に有効になるため、有効にするために何もする必要はありません。Amazon EMR リリース 6.8.0 以前は、クラスターを設定することにより、プライマリノードに格納されているログファイルを Simple Storage Service (Amazon S3) に定期的にアーカイブすることができます。こ

うすれば、クラスターの終了の理由が通常のシャットダウンかエラーのいずれであっても、クラスターの終了後にログファイルを確実に利用できます。Amazon EMR は、5 分間隔でログファイルを Simple Storage Service (Amazon S3) にアーカイブします。

Amazon EMR リリース 6.8.0 以前でログファイルが Simple Storage Service (Amazon S3) にアーカイブされるようにするには、クラスターの起動時に、この機能を有効にする必要があります。これは、コンソール、CLI または API を使用すれば可能です。デフォルトでは、コンソールを使用して起動したクラスターは、ログのアーカイブが有効になってます。CLI または API を使用して起動したクラスターは、Simple Storage Service (Amazon S3) へのログ記録を手動で有効にする必要があります。

> **Note**
>
> Amazon EMR コンソールは、再設計され、使いやすくなりました。古いコンソールと新しいコンソールエクスペリエンスの違いについては、「Amazon EMR コンソール」を参照してください。

New console

新しいコンソールを使用して Simple Storage Service (Amazon S3) にログファイルをアーカイブするには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で、[クラスター] を選択し、[クラスターの作成] を選択します
3. [クラスターログ] で、[クラスター固有のログを Amazon S3 に公開] チェックボックスを選択します。
4. [Amazon S3 のロケーション] フィールドに、ログを格納する Simple Storage Service (Amazon S3) のパスを入力 (または参照) します。バケットに存在しないフォルダの名前を入力した場合、Amazon S3 によりそのフォルダが作成されます。

   この値が設定されると、Amazon EMR はクラスターの EC2 インスタンスからのログファイルを Simple Storage Service (Amazon S3) にコピーします。これにより、クラスターの終了時およびクラスターをホストしているインスタンスを EC2 が終了してもログファイルが失われるのを回避できます。これらのログは、トラブルシューティングに役立ちます。詳細については、「ログファイルを表示する」を参照してください。

5. オプションで、[クラスター固有のログを暗号化] チェックボックスを選択します。次に、リストから AWS KMS キーを選択するか、キー ARN を入力するか、新しいキーを作成します。このオプションは、Amazon EMR バージョン 5.30.0 以降 (バージョン 6.0.0 は除く) でのみ使用できます。このオプションを使用するには、EC2 インスタンスプロファイルと Amazon EMR ロール AWS KMS の アクセス許可を に追加します。詳細については、「AWS KMS カスタマーマネージドキーを使用して Simple Storage Service (Amazon S3) に格納されたログファイルを暗号化するには」を参照してください。
6. クラスターに適用するその他のオプションを選択します。
7. クラスターを起動するには、[クラスターの作成] を選択します。

Old console

古いコンソールを使用して Simple Storage Service (Amazon S3) にログファイルをアーカイブするには

1. 新しい Amazon EMR コンソールに移動し、サイドナビゲーションから [古いコンソールに切り替え] を選択します。古いコンソールに切り替えたときの動作の詳細については、「Using the old console」を参照してください。
2. [クラスターを作成] を選択します。
3. [Go to advanced options] を選択します。
4. [全般オプション] セクションの [ログ記録] フィールドで、デフォルトのオプションの [有効] をそのまま使用します。

   これにより、Amazon EMR が詳細なログデータを Simple Storage Service (Amazon S3) にキャプチャするかどうかが決まります。これを設定できるのは、クラスターを作成するときのみです。詳細については、「 ログファイルを表示する」を参照してください。
5. [S3 folder] (S3 フォルダ) フィールドに、ログを格納する Simple Storage Service (Amazon S3) のパスを入力 (または参照) します。コンソールを使用して、Simple Storage Service (Amazon S3) を生成することもできます。バケットに存在しないフォルダの名前を入力した場合、そのフォルダは自動的に作成されます。

   この値が設定されると、Amazon EMR はクラスターの EC2 インスタンスからのログファイルを Simple Storage Service (Amazon S3) にコピーします。これにより、クラスターの終了時およびクラスターをホストしている EC2 が終了してもログファイルが失われるのを回避できます。これらのログは、トラブルシューティングに役立ちます。

   詳細については、「ログファイルを表示する」を参照してください。

6. ログ暗号化 フィールドで、 AWS KMS カスタマーマネージドキー を使用して S3 に保存されているログを暗号化 を選択します。次に、リストから AWS KMS キーを選択するか、キー ARN を入力します。新しい AWS KMS キーを作成することもできます。

   このオプションは、Amazon EMR バージョン 5.30.0 以降 (バージョン 6.0.0 は除く) でのみ使用できます。このオプションを使用するには、EC2 インスタンスプロファイルと Amazon EMR ロールに AWS KMS へのアクセス許可を追加します。詳細については、「AWS KMS カスタマーマネージドキーを使用して Simple Storage Service (Amazon S3) に格納されたログファイルを暗号化するには」を参照してください。

7. 「クラスターの計画と設定」で説明されているように、クラスターの作成に進みます。

CLI

を使用してログファイルを Amazon S3 にアーカイブするには AWS CLI

を使用してログファイルを Amazon S3 にアーカイブするには AWS CLI、 `create-cluster` コマンドを入力し、 `--log-uri`パラメータを使用して Amazon S3 ログパスを指定します。

1. Simple Storage Service (Amazon S3) にログファイルをアーカイブするには、次のコマンドを入力し、*myKey* を EC2 キーペアの名前に置き換えます。

```
aws emr create-cluster --name "Test cluster" --release-label emr-7.1.0 --log-
uri s3://DOC-EXAMPLE-BUCKET/logs --applications Name=Hadoop Name=Hive Name=Pig
 --use-default-roles --ec2-attributes KeyName=myKey --instance-type m5.xlarge --
instance-count 3
```

2. `--instance-groups` パラメータを使用せずにインスタンス数を指定すると、1 つのプライマリノードが起動され、残りのインスタンスはコアノードとして起動されます。すべてのノードで、コマンドで指定したインスタンスタイプが使用されます。

> **Note**
>
> 以前にデフォルトの Amazon EMR サービスロールと EC2 インスタンスプロファイルを作成していない場合は、「`aws emr create-default-roles`」と入力してそれらを作成してから、`create-cluster` サブコマンドを入力します。

AWS KMS カスタマーマネージドキーを使用して Simple Storage Service (Amazon S3) に格納されたログファイルを暗号化するには

Amazon EMR バージョン 5.30.0 以降 (Amazon EMR 6.0.0 を除く) では、Amazon S3 に保存されているログファイルを AWS KMS カスタマーマネージドキーで暗号化できます。コンソールでこのオプションを有効にするには、「Simple Storage Service (Amazon S3) にログファイルをアーカイブする」の手順に従います。Amazon EC2 インスタンスプロファイルと Amazon EMR ロールは、以下の前提条件を満たしている必要があります。

- クラスターで使用する Amazon EC2 インスタンスプロファイルには、`kms:GenerateDataKey` を使用するアクセス許可が必要です。
- クラスターで使用する Amazon EMR ロールには、`kms:DescribeKey` を使用するアクセス許可が必要です。
- 次の手順で示すように、Amazon EC2 インスタンスプロファイルと Amazon EMR ロールを、指定された AWS KMS カスタマーマネージドキーのキーユーザーのリストに追加する必要があります。
  1. https://console.aws.amazon.com/kms で AWS Key Management Service （ AWS KMS) コンソールを開きます。
  2. AWS リージョンを変更するには、ページの右上隅にあるリージョンセレクターを使用します。
  3. 変更する KMS キーのエイリアスを選択します。
  4. [Key Users] のキーの詳細ページで、[Add] を選択します。
  5. [キーユーザーの追加] ダイアログボックスで、Amazon EC2 インスタンスプロファイルと Amazon EMR ロールを選択します。
  6. 追加を選択します。

詳細については、「 Key Management Service デベロッパーガイド」の「Amazon EMR で使用される IAM サービスロール」および「キーポリシーの使用」を参照してください。 AWS

AWS CLIを使用して Simple Storage Service (Amazon S3) でログを集計するには

> **Note**
>
> 現在、`yarn logs` ユーティリティを使ってログを集計することはできません。この手順でサポートされる集計のみ使用できます。

ログ集計 (Hadoop 2.x) では、個々のアプリケーションのすべてのコンテナのログが 1 つのファイルにコンパイルされます。を使用して Amazon S3 へのログ集約を有効にするには AWS CLI、クラスター起動時にブートストラップアクションを使用してログ集約を有効にし、ログを保存するバケットを指定します。

- ログ集計を有効にするには、次の内容が含まれる myConfig.json と呼ばれる設定ファイルを作成します。

```
[
  {
    "Classification": "yarn-site",
    "Properties": {
      "yarn.log-aggregation-enable": "true",
      "yarn.log-aggregation.retain-seconds": "-1",
      "yarn.nodemanager.remote-app-log-dir": "s3:\/\/DOC-EXAMPLE-BUCKET\/logs"
    }
  }
]
```

次のコマンドを入力し、*myKey* を EC2 キーペアの名前に置き換えます。さらに、赤色のテキストはいずれも独自の設定に置き換えることができます。

```
aws emr create-cluster --name "Test cluster" \
--release-label emr-7.1.0 \
--applications Name=Hadoop \
--use-default-roles \
--ec2-attributes KeyName=myKey \
--instance-type m5.xlarge \
--instance-count 3 \
--configurations file://./myConfig.json
```

--instance-groups パラメータを使用せずにインスタンス数を指定すると、1 つのプライマリノードが起動され、残りのインスタンスはコアノードとして起動されます。すべてのノードで、コマンドで指定したインスタンスタイプが使用されます。

> **Note**
>
> 以前にデフォルトの EMR サービスロールと EC2 インスタンスプロファイルを作成していない場合は、「`aws emr create-default-roles`」を実行してそれらを作成してから、`create-cluster` サブコマンドを入力します。

での Amazon EMR コマンドの使用の詳細については AWS CLI、AWS CLI「 コマンドリファレンス」を参照してください。

ログの場所

次のリストに、Amazon S3 のすべてのログタイプとそれらの場所を示します。Amazon EMR の問題のトラブルシューティングにこれらを使用できます。

ステップログ

`s3://DOC-EXAMPLE-LOG-BUCKET/<cluster-id>/steps/<step-id>/`

アプリケーションログ

`s3://DOC-EXAMPLE-LOG-BUCKET/<cluster-id>/containers/`

この場所には、`stderr` コンテナと `stdout`、`directory.info`、`prelaunch.out`、および `launch_container.sh` ログが含まれます。

リソースマネージャーログ

`s3://DOC-EXAMPLE-LOG-BUCKET/<cluster-id>/node/<leader-instance-id>/applications/hadoop-yarn/`

Hadoop HDFS

`s3://DOC-EXAMPLE-LOG-BUCKET/<cluster-id>/node/<all-instance-id>/applications/hadoop-hdfs/`

この場所には NameNode、、 DataNode、および YARN TimelineServer ログが含まれます。

ノードマネージャーログ

`s3://DOC-EXAMPLE-LOG-BUCKET/<cluster-id>/node/<all-instance-id>/applications/hadoop-yarn/`

インスタンス状態ログ

```
s3://DOC-EXAMPLE-LOG-BUCKET/<cluster-id>/node/<all-instance-id>/daemons/
instance-state/
```

Amazon EMR プロビジョニングログ

```
s3://DOC-EXAMPLE-LOG-BUCKET/<cluster-id>/node/<leader-instance-id>/
provision-node/*
```

Hive ログ

```
s3://DOC-EXAMPLE-LOG-BUCKET/<cluster-id>/node/<leader-instance-id>/
applications/hive/*
```

- クラスター上の Hive ログを見つけるには、アスタリスク (*) を削除して、/var/log/hive/ を上記のリンクに追加します。
- HiveServer2 つのログを検索するには、アスタリスク (*) を削除し、上記のリンクvar/log/hive/hiveserver2.logに を追加します。
- HiveCLI ログを見つけるには、アスタリスク (*) を削除して、/var/log/hive/user/hadoop/hive.log を上記のリンクに追加します。
- Hive メタストアサーバーログを見つけるには、アスタリスク (*) を削除して、/var/log/hive/user/hive/hive.log を上記のリンクに追加します。

Tez アプリケーションのプライマリノードまたはタスクノードで障害が発生している場合は、適切な Hadoop コンテナのログを提供してください。

デバッグツールを有効にする

デバッグツールを使うと、Amazon EMR コンソールからログファイルを、より容易に参照できます。詳細については、「デバッグツールでログファイルを表示する」を参照してください。クラスターでデバッグを有効にすると、Amazon EMR によってログファイルが Simple Storage Service (Amazon S3) にアーカイブされ、そのログファイルにインデックスが付けられます。これで、コンソールを使用して、クラスター関するステップ、ジョブ、タスク、およびタスク試行のログを、わかりやすい方法で参照できます。

Amazon EMR コンソールでデバッグツールを使用するには、コンソール、CLI、または API を使用してクラスターを起動するときに、デバッグを有効にする必要があります。新しい Amazon EMR コンソールにはデバッグツールがないことに注意してください。

Old console

古いコンソールでデバッグツールを有効にするには

1. 新しい Amazon EMR コンソールに移動し、サイドナビゲーションから [古いコンソールに切り替え] を選択します。古いコンソールに切り替えたときの動作の詳細については、「Using the old console」を参照してください。
2. [クラスターを作成] を選択します。
3. [Go to advanced options] を選択します。
4. [Cluster Configuration (クラスターの設定)] セクションの [Logging (ログ記録)] フィールドで、[Enabled (有効)] を選択します。ログ記録を有効にしなければ、デバッグを有効にすることはできません。
5. [ログフォルダ S3 の場所] フィールドに、ログを格納する Simple Storage Service (Amazon S3) パスを入力します。
6. [Debugging (デバッグ)] フィールドで [有効] を選択します。デバッグオプションにより Amazon SQS 交換が作成され、Amazon EMR サービスバックエンドにデバッグのメッセージが発行されます。交換にメッセージを発行する際、料金が発生する場合があります。詳細については、「Amazon SQS 製品ページ」を参照してください。
7. 「クラスターの計画と設定」で説明されているように、クラスターの作成に進みます。

AWS CLI

を使用してデバッグツールを有効にするには AWS CLI

を使用してデバッグを有効にするには AWS CLI、 `--enable-debugging`パラメータを指定して `create-cluster`サブコマンドを入力します。デバッグを有効にするには、`--log-uri` パラメータも指定する必要があります。

- を使用してデバッグを有効にするには AWS CLI、次のコマンドを入力し、*myKey* を EC2 キーペアの名前に置き換えます。

```
aws emr create-cluster --name "Test cluster" \
--release-label emr-7.1.0 \
--log-uri s3://DOC-EXAMPLE-BUCKET/logs \
--enable-debugging \
--applications Name=Hadoop Name=Hive Name=Pig \
--use-default-roles \
--ec2-attributes KeyName=myKey \
```

```
--instance-type m5.xlarge \
--instance-count 3
```

--instance-groups パラメータを使用せずにインスタンス数を指定すると、1 つのプライマリノードが起動され、残りのインスタンスはコアノードとして起動されます。すべてのノードで、コマンドで指定したインスタンスタイプが使用されます。

> **Note**
>
> 以前にデフォルトの EMR サービスロールと EC2 インスタンスプロファイルを作成していない場合は、「aws emr create-default-roles」と入力してそれらを作成してから、create-cluster サブコマンドを入力します。

API

Amazon EMR API でデバッグツールを有効にするには

- 次の Java SDK 設定を使用してデバッグを有効にします。

```
    StepFactory stepFactory = new StepFactory();
StepConfig enabledebugging = new StepConfig()
    .withName("Enable debugging")
    .withActionOnFailure("TERMINATE_JOB_FLOW")
    .withHadoopJarStep(stepFactory.newEnableDebuggingStep());
```

この例では、new StepFactory() はデフォルトのリージョンとして us-east-1 を使用しています。クラスターが別のリージョンで起動されている場合、new StepFactory("*region*.elasticmapreduce") を使用してリージョンを指定する必要があります (new StepFactory("ap-northeast-2.elasticmapreduce") など)。

デバッグオプション情報

Amazon EMR リリース 4.1.0 ~ 5.27.0 は、すべてのリージョンでデバッグをサポートしています。他の Amazon EMR バージョンでは、デバッグオプションをサポートしていません。2023 年 1 月 23 日をもって、Amazon EMR はすべてのバージョンのデバッグツールを終了します。

Amazon EMR により Amazon SQS キューが作成され、デバッグデータが処理されます。メッセージ料金が発生する場合があります。ただし、Amazon SQS で使用できるリクエストには、1,000,000

件までの無料利用枠があります。詳細については、「https://aws.amazon.com/sqs」を参照してください。

デバッグにはロールの使用が必要です。サービスロールおよびインスタンスプロファイルですべての Amazon SQS API オペレーションができるようになっている必要があります。Amazon EMR 管理ポリシーにロールがアタッチされている場合、ロールに対する変更は必要ありません。カスタムロールの場合、`sqs:*` アクセス権限を与える必要があります。詳細については、「AWS のサービスおよびリソースへのアクセス許可を Amazon EMR に付与する IAM サービスロールの設定」を参照してください。

クラスターのタグ付け

AWS リソースを目的、所有者、環境などさまざまな方法で分類すると便利です。Amazon EMR でこれを達成するには、タグを使用して、カスタムメタデータを Amazon EMR クラスターに割り当てます。タグは、1 つのキーと 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。Amazon EMR では、クラスターはタグ付け可能なリソースレベルです。たとえば、各クラスターの所有者の追跡または実稼働用クラスターとテスト用クラスターの識別に役立つ、アカウントのクラスターに一連のタグを定義できます。組織の要件に適合する一連の一貫したタグを作成することをお勧めします。

Amazon EMR クラスターにタグを追加するとき、タグはクラスターに関連するアクティブな Amazon EC2 インスタンスそれぞれに伝達されます。同様に、Amazon EMR クラスターからタグを削除すると、そのタグは関連するアクティブな Amazon EC2 インスタンスそれぞれから削除されます。

> **Important**
>
> Amazon EMR コンソールまたは CLI を使用して、Amazon EC2 コンソールまたは CLI ではなく、クラスターの一部である Amazon EC2 インスタンス上のタグを管理します。これは、Amazon EC2 で加える変更は Amazon EMR のタグ付けシステムと同期しないためです。

次のシステムタグを探すことにより、Amazon EMR クラスターの一部である Amazon EC2 インスタンスを識別できます。この例では、*CORE* はインスタンスグループロールの値であり、*j-12345678* はサンプルジョブフロー（クラスター）識別子の値です。

- aws:elasticmapreduce:instance-group-role=*CORE*

- aws:elasticmapreduce:job-flow-id=*j-12345678*

> **Note**
>
> Amazon EMR および Amazon EC2 は、意味論的意味のない文字の文字列としてタグを解釈します。

タグは、、 CLI AWS Management Console、および API を使用して操作できます。

新しい Amazon EMR クラスターを作成するときタグを追加でき、実行中の Amazon EMR クラスターを対象に、タグを追加、編集、または削除できます。タグの編集は、Amazon EMR コンソールに適用される概念であり、CLI と API を使用する場合、タグを編集するには、古いタグを削除して新しいタグを追加します。タグのキーと値は編集でき、クラスター実行中であればいつでもタグをリソースから削除できます。ただし、まだアクティブなクラスターと以前に関連付けられていた、終了したクラスターまたは終了したインスタンスを対象にタグの追加、編集、または削除はできません。また、タグの値を空の文字列に設定することはできますが、null に設定することはできません。

タグによるリソースベースのアクセス許可のために Amazon EC2 インスタンスで AWS Identity and Access Management (IAM) を使用している場合、IAM ポリシーは Amazon EMR がクラスターの Amazon EC2 インスタンスに伝達するタグに適用されます。Amazon EMR タグを Amazon EC2 インスタンスに伝達するには、Amazon EC2 の IAM ポリシーで Amazon EC2 CreateTags および DeleteTags APIsを呼び出すアクセス許可を許可する必要があります。また、伝達されたタグは、Amazon EC2 のリソースベースのアクセス許可に影響を及ぼす可能性があります。Amazon EC2 に伝達されたタグは、その他の Amazon EC2 タグと同じように IAM ポリシーでの条件として読み取られます。ユーザーがクラスターに対して正しくないアクセス許可を持つことがないように、Amazon EMR クラスターにタグを追加するときは、IAM ポリシーに留意してください。問題を避けるために、Amazon EMR クラスターでも使用する予定のタグの条件が IAM ポリシーに含まれないようにしてください。詳細については、「Amazon EC2 のリソースに対するアクセスの制御」を参照してください。

タグの制限

タグには以下のベーシックな制限があります。

- Amazon EC2 リソースに適用される制限は、Amazon EMR にも適用されます。詳細については、「https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/Using_Tags.html#tag-restrictions」を参照してください。

- タグ名と値にはaws:プレフィックスを使用しないでください。プレフィックスは AWS 用に予約されています。また、このプレフィックスが含まれるタグの名前または値は編集または削除できません。
- 終了したクラスターでタグの変更または編集はできません。
- タグの値を空の文字列にすることはできますが、null にすることはできません。また、タグキーを空の文字列にすることはできません。
- キーと値には、任意の言語のアルファベット文字、数字、空白、表示されない区切り文字、記号 (_ . : / = + - @) を使用できます。

を使用したタグ付けの詳細については AWS Management Console、Amazon EC2 ユーザーガイド」の「コンソールでのタグの使用」を参照してください。Amazon EC2API またはコマンドラインを使用したタグ付けの詳細については、「Amazon EC2 ユーザーガイド」の「API と CLI の概要」を参照してください。 Amazon EC2

請求用のタグリソース

タグを使用して AWS 請求書を整理し、独自のコスト構造を反映することができます。これを行うには、サインアップして、タグキー値を含む AWS アカウント請求書を取得します。その後で、タグキーの値により課金情報を整理して、リソースを合わせたコストを確認できます。Amazon EMR および Amazon EC2 では請求明細書が異なりますが、各クラスターのタグが関連付けられているインスタンスそれぞれにも付けられているため、タグを使用して関連する Amazon EMR のコストと Amazon EC2 のコストをリンクすることができます。

例えば、複数のリソースに特定のアプリケーション名のタグを付け、請求情報を整理することで、複数のサービスを利用しているアプリケーションの合計コストを確認することができます。詳細については、「AWS Billing ユーザーガイド」の「コスト配分とタグ付け」を参照してください。

クラスターにタグを追加する

タグの作成時にクラスターにタグを追加することもできます。

> **Note**
>
> Amazon EMR コンソールは、再設計され、使いやすくなりました。古いコンソールと新しいコンソールエクスペリエンスの違いについては、「Amazon EMR コンソール」を参照してください。

New console

新しいコンソールでクラスター作成時にタグを追加するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で、[クラスター] を選択し、[クラスターの作成] を選択します
3. [タグ] で [タグを追加] を選択します。Key フィールドにタグを指定します。オプションで、[値] フィールドにタグを指定します。
4. クラスターに適用するその他のオプションを選択します。
5. クラスターを起動するには、[クラスターの作成] を選択します。

Old console

古いコンソールでクラスター作成時にタグを追加するには

1. 新しい Amazon EMR コンソールに移動し、サイドナビゲーションから [古いコンソールに切り替え] を選択します。古いコンソールに切り替えたときの動作の詳細については、「Using the old console」を参照してください。
2. [Create cluster (クラスターの作成)]、[Go to advanced options (詳細オプションに移動する)] の順に選択します。
3. [Step 3: General Cluster Settings (ステップ 3: 一般的なクラスター設定)] ページの [タグ] セクションで、タグの [キー] を入力します。

   [キー] を入力を開始すると、新しい行が自動的に追加され、次の新しいタグを入力できるようになります。
4. 必要に応じてタグの [値] を入力します。
5. クラスターに追加するタグキー/値のペアについて上記のステップを繰り返します。クラスターを起動すると、入力したタグが自動的にクラスターと関連付けられます。

AWS CLI

を使用してクラスターを作成するときにタグを追加するには AWS CLI

次の例は、 AWS CLIを使用して新しいクラスターにタグを追加する方法を示しています。クラスターの作成時にタグを追加するには、`create-cluster` サブコマンドを入力して `--tags` パラメータを指定します。

- クラスターの作成時に、*costCenter* という名前のタグをキー値 *marketing* と共に追加するには、次のコマンドを入力し、*myKey* を EC2 キーペアの名前に置き換えます。

```
aws emr create-cluster --name "Test cluster" --release-label emr-4.0.0 --
applications Name=Hadoop Name=Hive Name=Pig --tags "costCenter=marketing" --
use-default-roles --ec2-attributes KeyName=myKey --instance-type m5.xlarge --
instance-count 3
```

`--instance-groups` パラメータを使用せずにインスタンス数を指定すると、1 つのマスターノードが起動され、残りのインスタンスはコアノードとして起動されます。すべてのノードで、コマンドで指定したインスタンスタイプが使用されます。

> **Note**
>
> 以前にデフォルトの EMR サービスロールと EC2 インスタンスプロファイルを作成していない場合は、「aws emr create-default-roles」と入力してそれらを作成してから、`create-cluster` サブコマンドを入力します。

での Amazon EMR コマンドの使用の詳細については AWS CLI、「」を参照してくださいhttps://docs.aws.amazon.com/cli/latest/reference/emr。

既存のクラスターにタグを追加することもできます。

New console

新しいコンソールで既存のクラスターにタグを追加するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。

2. 左側のナビゲーションペインの [EMR on EC2] で [クラスター] を選択し、更新するクラスターを選択します。
3. クラスターの詳細ページの [タグ] タブで、[タグ管理] を選択します。Key フィールドにタグを指定します。オプションで、[値] フィールドにタグを指定します。
4. [変更を保存] を選択します。[タグ] タブは、クラスターに追加されたタグの新しい数で更新されます。例えば、タグが 2 つになった場合、タブのラベルは [タグ (2)] になります。

Old console

古いコンソールで既存のクラスターにタグを追加するには

1. Amazon EMR コンソールで、[Cluster List] (クラスターリスト) ページを選択し、タグを追加するクラスターをクリックします。
2. [Cluster Details] ページの [タグ] フィールドで、[View All/Edit] をクリックします。
3. [View All/Edit] ページで、[Add] をクリックします。
4. [Key] 列にある空のフィールドをクリックし、キーの名前を入力します。
5. 必要に応じて、[Value] 列にある空のフィールドをクリックし、値の名前を入力します。
6. 入力を開始したタグごとに、編集中のタグの下に空のタグの行が表示されます。追加するタグごとに新しいタグの行で上記のステップを繰り返します。

AWS CLI

を使用して実行中のクラスターにタグを追加するには AWS CLI

- クラスター ID にタグを割り当てるには、`add-tags` サブコマンドを入力して `--tag` パラメータを指定します。コンソールまたは `list-clusters` コマンドを使用してクラスター ID を見つけることができます。現時点では、`add-tags` サブコマンドで使用できるリソース ID は 1 つだけです。

  例えば、実行中のクラスターに 2 つのタグ (1 つはキーの名前が *`costCenter`* で値が *`marketing`* のタグ、もう 1 つはキーの名前が *`other`* で値が *`accounting`* のタグ) を追加するには、次のコマンドを入力して、*`j-KT4XXXXXXXX1NM`* をクラスター ID に置き換えます。

  ```
  aws emr add-tags --resource-id j-KT4XXXXXXXX1NM --tag "costCenter=marketing" --
  tag "other=accounting"
  ```

AWS CLI を使用してタグを追加すると、 コマンドからの出力は行われないことに注意してください。での Amazon EMR コマンドの使用の詳細については AWS CLI、「」を参照してくださいhttps://docs.aws.amazon.com/cli/latest/reference/emr。

クラスター上でタグを表示する

クラスターに関連付けられたすべてのタグを確認する場合は、コンソールまたは AWS CLIでそれらのタグを表示できます。

> **Note**
>
> Amazon EMR コンソールは、再設計され、使いやすくなりました。古いコンソールと新しいコンソールエクスペリエンスの違いについては、「Amazon EMR コンソール」を参照してください。

New console

新しいコンソールでクラスター上のタグを表示するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で [クラスター] を選択し、更新するクラスターを選択します。
3. すべてのタグを表示するには、クラスターの詳細ページの [タグ] タブを選択します。

Old console

古いコンソールでクラスター上のタグを表示するには

1. Amazon EMR コンソールで、[Cluster List] (クラスターリスト) ページを選択し、タグを表示するクラスターをクリックします。
2. [Cluster Details] ページの [Tags] フィールドでは、いくつかのタグがここに表示されます。クラスター上の利用できるすべてのタグを表示するには、[View All/Edit] をクリックします。

AWS CLI

を使用してクラスターのタグを表示するには AWS CLI

を使用してクラスターのタグを表示するには AWS CLI、 `--query`パラメータを指定して`describe-cluster`サブコマンドを入力します。

- クラスターのタグを表示するには、次のコマンドを入力し、*j-KT4XXXXXXXX1NM* をクラスター ID に置き換えます。

```
aws emr describe-cluster --cluster-id j-KT4XXXXXX1NM --query Cluster.Tags
```

出力では、次のように、クラスターに関するすべてのタグ情報が表示されます。

```
Value: accounting     Value: marketing
Key: other            Key: costCenter
```

での Amazon EMR コマンドの使用の詳細については AWS CLI、「」を参照してくださいhttps://docs.aws.amazon.com/cli/latest/reference/emr。

クラスターからタグを削除する

タグが必要でなくなったら、クラスターから削除できます。

> **Note**
>
> Amazon EMR コンソールは、再設計され、使いやすくなりました。古いコンソールと新しいコンソールエクスペリエンスの違いについては、「Amazon EMR コンソール」を参照してください。

New console

新しいコンソールでクラスター上のタグを削除するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で [クラスター] を選択し、更新するクラスターを選択します。

3. クラスターの詳細ページの [タグ] タブで、[タグ管理] を選択します。
4. 削除するキーと値のペアごとに [削除] を選択します。
5. [変更の保存] を選択します。

Old console

古いコンソールでクラスター上のタグを削除するには

1. Amazon EMR コンソールで、[Cluster List] (クラスターリスト) ページを選択し、タグを削除するクラスターをクリックします。
2. [Cluster Details] ページの [タグ] フィールドで、[View All/Edit] をクリックします。
3. [View All/Edit] ダイアログボックスで、削除するタグの横にある [X] アイコンをクリックし、[Save] をクリックします。
4. (オプション) クラスターから削除するタグのキーと値のペアごとに上記のステップを繰り返します。

AWS CLI

を使用してクラスターのタグを削除するには AWS CLI

`remove-tags` サブコマンドを入力し、`--tag-keys` パラメータを指定します。タグを削除する場合に必要なのは、キー名のみです。

- クラスターからタグを削除するには、次のコマンドを入力し、*`j-KT4XXXXXXXX1NM`* をクラスター ID に置き換えます。

```
aws emr remove-tags --resource-id j-KT4XXXXXX1NM --tag-keys "costCenter"
```

> **Note**
>
> 現在、1 つのコマンドで複数のタグを削除することはできません。

での Amazon EMR コマンドの使用の詳細については AWS CLI、「」を参照してくださいhttps://docs.aws.amazon.com/cli/latest/reference/emr。

ドライバーとサードパーティーアプリケーション統合

ユーティリティ料金で、いくつかの一般的なッグデータアプリケーションを Amazon EMR 上で実行できます。つまり、クラスターの実行中、サードパーティアプリケーションに対して支払う追加時間料金を最小限に抑えることができます。これにより年間ライセンスを購入せずに、アプリケーションを使用することができます。以降のセクションでは、EMR で利用できるツールの一部を紹介します。

トピック

Amazon EMR でのビジネスインテリジェンスツールの使用

Amazon EMR で Microsoft Excel、 MicroStrategy、、Tableau などの一般的なビジネスインテリジェンスツールを使用してQlikView、データを探索および視覚化できます。この種のツールの多くは ODBC（Open Database Connectivity）または JDBC（Java Database Connectivity）ドライバを必要とします。最新のドライバーをダウンロードしてインストールするには、「http://awssupportdatasvcs.com/bootstrap-actions/Simba/latest/」を参照してください。

古いバージョンのドライバーを検索するには、「http://awssupportdatasvcs.com/bootstrap-actions/Simba/」を参照してください。

Amazon EMR でのセキュリティ

セキュリティとコンプライアンスは、 と共有する責任です AWS。この責任共有モデルは、ホストオペレーティングシステムや仮想化レイヤーから EMR クラスターが動作する施設の物理的なセキュリティまで、コンポーネントを AWS 運用、管理、制御する際の運用上の負担を軽減します。Amazon EMR クラスターの責任、管理、更新、およびアプリケーションソフトウェアの設定と AWS セキュリティコントロールの提供は、お客様が引き受けます。この責任の区別は、一般的にクラウドのセキュリティとクラウドのセキュリティと呼ばれます。

- クラウドのセキュリティ — AWS は、 AWS のサービス で実行されるインフラストラクチャを保護する責任を担います AWS。また、 は、安全に使用できるサービス AWS も提供します。AWS コンプライアンスプログラムの一環として、サードパーティーの監査が定期的にセキュリティの有効性をテストおよび検証しています。Amazon EMR に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、「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 」を参照してください。
- クラウド内のセキュリティ — また、Amazon EMR クラスターのセキュリティを保護するために必要なセキュリティ設定および管理タスクをすべて実行する責任もあります。Amazon EMR クラスターをデプロイするお客様は、インスタンスにインストールされたアプリケーションソフトウェアの管理、およびセキュリティグループ、要件に応じた暗号化とアクセス制御、適用される法律、規制などの AWSが提供する機能の設定に責任を負います。

このドキュメントは、Amazon EMR を使用する際の責任共有モデルの適用について理解するのに役立ちます。この章のトピックでは、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ために Amazon EMR を設定し AWS のサービス 、他の を使用する方法を示します。

ネットワークとインフラストラクチャのセキュリティ

マネージドサービスである Amazon EMR は、ホワイトペーパー「Amazon Web Services: セキュリティプロセスの概要」に記載されている AWS グローバルネットワークセキュリティの手順で保護されています。 AWS ネットワークおよびインフラストラクチャ保護サービスでは、ホストレベルとネットワークレベルの境界の両方できめ細かな保護を提供します。Amazon EMR は AWS のサービス 、ネットワーク保護とコンプライアンス要件に対応する およびアプリケーション機能をサポートしています。

- Amazon EC2 セキュリティグループは、Amazon EMR クラスターインスタンスの仮想ファイアウォールとして機能し、インバウンドおよびアウトバウンドのネットワークトラフィックを制限し

ます。詳細については、「セキュリティグループによるネットワークトラフィックの制御」を参照してください。

- Amazon EMR Block Public Access (BPA) では、クラスターにポート上のパブリック IP アドレスからのインバウンドトラフィックを許可するセキュリティ設定がある場合、パブリックサブネットでクラスターを起動できません。詳細については、「Amazon EMR のパブリックアクセスブロックの使用」を参照してください。
- Secure Shell (SSH) は、ユーザーがクラスターインスタンスのコマンドラインに接続するための安全な方法を提供します。SSH を使用して、アプリケーションがクラスターのマスターノードでホストするウェブインターフェイスを表示することもできます。詳細については、「SSH 認証情報に EC2 キーペアを使用する」および「クラスターに接続する」を参照してください。

Amazon EMR のデフォルトの Amazon Linux AMI の更新

> **Important**
>
> Amazon Linux または Amazon Linux 2 Amazon マシンイメージ (AMI) を実行する EMR クラスターは、デフォルトの Amazon Linux 動作を使用します。再起動が必要な重要かつクリティカルなカーネル更新が自動的にダウンロードされてインストールされることはありません。これは、デフォルトの Amazon Linux AMI を実行している他の Amazon EC2 インスタンスと同じ動作です。Amazon EMR リリースが利用可能になった後に、再起動が必要な新しい Amazon Linux ソフトウェアアップデート (カーネル、NVIDIA、CUDA のアップデートなど) が使用可能になった場合、デフォルトの AMI を実行する EMR クラスターインスタンスで、それらの更新が自動的にダウンロードされてインストールされることはありません。カーネルの更新を取得するには、Amazon EMR AMI をカスタマイズして、最新の Amazon Linux AMI を使用できるようにします。

アプリケーションのセキュリティ体制やクラスターが実行される時間に応じて、クラスターを定期的に再起動してセキュリティ更新を適用したり、ブートストラップアクションを作成してパッケージのインストールと更新をカスタマイズすることもできます。実行中のクラスターインスタンスで、選択したセキュリティ更新をテストしてインストールすることもできます。詳細については、「Amazon EMR にデフォルトの Amazon Linux AMI を使用する」を参照してください。ネットワーク設定では、Amazon S3 の Linux リポジトリへの HTTP および HTTPS 出力を許可する必要があります。許可しないと、セキュリティ更新は成功しません。

AWS Identity and Access Management Amazon EMR を使用した

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御するのに役立つ AWS サービスです。IAM 管理者は、誰を認証 (サインイン) し、誰に Amazon EMR リソースの使用を承認する (アクセス許可を付与する) かを制御します。IAM アイデンティティには、ユーザー、グループ、およびロールが含まれます。IAM ロールは IAM ユーザーと似ていますが、特定のユーザーに関連付けられておらず、アクセス許可を必要とするすべてのユーザーが引き受けることを意図しています。詳細については、AWS Identity and Access Management Amazon EMR の「」を参照してください。Amazon EMR は、複数の IAM ロールを使用して、Amazon EMR クラスターのアクセスコントロールを実装するのに役立ちます。IAM は、追加料金なしで使用できる AWS のサービスです。

- Amazon EMR の IAM ロール (EMR ロール) — Amazon EMR クラスターの起動時に Amazon EC2 インスタンスをプロビジョニングするなど、Amazon EMR サービスが AWS のサービス ユーザーに代わって他の にアクセスする方法を制御します。詳細については、「 AWS のサービス および リソースへの Amazon EMR アクセス許可の IAM サービスロールを設定する」を参照してください。
- クラスター EC2 インスタンスの IAM ロール (EC2 インスタンスプロファイル) — インスタンスの起動時に Amazon EMR クラスター内のすべての EC2 インスタンスに割り当てられるロール。クラスターで実行されるアプリケーションプロセスは、このロールを使用して Amazon S3 AWS のサービスなどの他の とやり取りします。 Amazon S3 詳細については、「クラスターの EC2 インスタンスの IAM ロール」を参照してください。
- アプリケーションの IAM ロール (ランタイムロール) — Amazon EMR クラスターにジョブまたはクエリを送信するときに指定できる IAM ロール。Amazon EMR クラスターに送信するジョブまたはクエリは、ランタイムロールを使用して Amazon S3 のオブジェクトなどの AWS リソースにアクセスします。Amazon EMR では、Spark ジョブと Hive ジョブ用のランタイムロールを指定できます。ランタイムロールを使用して Bu すると、異なる IAM ロールを使用して、同じクラスターで実行されているジョブを分離できます。詳細については、「Amazon EMR でのランタイムロールとしての IAM ロールの使用」を参照してください。

ワークフォース ID とは、 でワークロードを構築または運用するユーザーを指します AWS。Amazon EMR は、以下を使用してワークフォース ID をサポートします。

- AWS IAM アイデンティティセンター (Idc) は、 AWS リソースへのユーザーアクセスを管理する AWS のサービス ために推奨されます。これは、ワークフォース ID を割り当てて、複数の AWS アカウントやアプリケーションに一貫してアクセスできる単一の場所です。Amazon EMR は、信

頼できる ID の伝播を通じてワークフォース ID をサポートします。信頼できる ID 伝達機能を使用すると、ユーザーはアプリケーションにサインインでき、そのアプリケーションはユーザーの ID AWS のサービス を他の に渡して、データまたはリソースへのアクセスを許可できます。詳細については、AWS 「Amazon EMR で IAM アイデンティティセンターのサポートを有効にする」を参照してください。

Lightweight Directory Access Protocol (LDAP) は、ネットワーク経由でユーザー、システム、サービス、アプリケーションに関する情報にアクセスして維持するための、オープンでベンダーに依存しない業界標準のアプリケーションプロトコルです。LDAP は、Active Directory (AD) や OpenLDAP などの企業 ID サーバーに対するユーザー認証によく使用されます。EMR クラスターで LDAP を有効にすると、ユーザーは既存の認証情報を使用してクラスターを認証し、アクセスできるようになります。詳細については、「Amazon EMR で LDAP のサポートを有効にする」を参照してください。

Kerberos は、シークレットキー暗号化を使用してクライアント/サーバーアプリケーションに強力な認証を提供するように設計されたネットワーク認証プロトコルです。Kerberos を使用すると、Amazon EMR はクラスターにインストールするアプリケーション、コンポーネント、サブシステムに対して Kerberos を設定し、相互に認証されるようにします。Kerberos が設定されたクラスターにアクセスするには、Kerberos ドメインコントローラー (KDC) に kerberos プリンシパルが存在する必要があります。詳細については、「Amazon EMR で Kerberos のサポートを有効にする」を参照してください。

シングルテナントクラスターとマルチテナントクラスター

デフォルトでは、クラスターは EC2 インスタンスプロファイルを IAM ID とする単一のテナンシーに設定されています。シングルテナントクラスターでは、すべてのジョブがクラスターへの完全かつ完全なアクセス権を持ち、すべての AWS のサービス およびリソースへのアクセスは EC2 インスタンスプロファイルに基づいて行われます。マルチテナントクラスターでは、テナントは互いに分離され、テナントはクラスターとクラスターの EC2 インスタンスへの完全かつ完全なアクセス権を持ちません。マルチテナントクラスターのアイデンティティは、ランタイムロールまたはワークフォースが識別します。マルチテナントクラスターでは、 AWS Lake Formation または Apache Ranger を介したきめ細かなアクセスコントロール (FGAC) のサポートを有効にすることもできます。ランタイムロールまたは FGAC が有効になっているクラスターでは、EC2 インスタンスプロファイルへのアクセスも iptables 経由で無効になります。

> **Important**
>
> シングルテナントクラスターにアクセスできるユーザーは、Linux オペレーティングシステム (OS) に任意のソフトウェアをインストールしたり、Amazon EMR によってインストールされたソフトウェアコンポーネントを変更または削除したりして、クラスターの一部である EC2 インスタンスに影響を与えることができます。ユーザーが Amazon EMR クラスターをインストールまたは変更できないようにするには、クラスターのマルチテナンシーを有効にすることをお勧めします。ランタイムロール、 AWS IAM アイデンティティセンター、Kerberos、または LDAP のサポートを有効にすることで、クラスターでマルチテナンシーを有効にできます。

データ保護

では AWS、 AWS のサービス および ツールを使用して、データがどのように保護され、誰がアクセスできるかを判断することで、データを制御できます。 AWS Identity and Access Management (IAM) などのサービスを使用すると、 AWS のサービス および リソースへのアクセスを安全に管理できます。 は検出と監査 AWS CloudTrail を可能にします。Amazon EMR では、 によって管理されているか、 AWS ユーザーが完全に管理しているキーを使用して、Amazon S3 に保管中のデータを簡単に暗号化できます。Amazon EMR は、転送中のデータの暗号化の有効化もサポートしています。詳細については、「保管中および転送中のデータの暗号化」を参照してください。

データアクセスコントロール

データアクセスコントロールを使用すると、IAM アイデンティティまたはワークフォースアイデンティティがアクセスできるデータを制御できます。Amazon EMR では、次のアクセスコントロールがサポートされています。

- IAM アイデンティティベースのポリシー – Amazon EMR で使用する IAM ロールのアクセス許可を管理します。IAM ポリシーをタグ付けと組み合わせて、アクセスを cluster-by-cluster ベースで制御できます。詳細については、AWS Identity and Access Management Amazon EMR の「」を参照してください。
- AWS Lake Formation は、データのアクセス許可管理を一元化し、組織全体および外部での共有を容易にします。Lake Formation を使用して、 AWS Glue データカタログ内のデータベースとテーブルへのきめ細かな列レベルのアクセスを有効にできます。詳細については、「Amazon EMR AWS Lake Formation での の使用」を参照してください。

- Amazon S3 アクセス許可は、Active Directory や AWS Identity and Access Management (IAM) プリンシパルなどのディレクトリのアイデンティティを S3 のデータセットにマッピングします。さらに、S3 アクセスは、ログのエンドユーザー ID と、 の S3 データへのアクセスに使用されるアプリケーションを許可します AWS CloudTrail。詳細については、「Amazon EMR での Amazon S3 アクセス許可の使用」を参照してください。
- Apache Ranger は、Hadoop プラットフォーム全体で包括的なデータセキュリティを有効化、監視、管理するフレームワークです。Amazon EMR は、Apache Hive メタストアと Amazon S3 の Apache Ranger ベースのきめ細かなアクセスコントロールをサポートしています。詳細については、「Apache Ranger と Amazon EMR の統合」を参照してください。

セキュリティ設定を使用してクラスターセキュリティをセットアップする

Amazon EMR セキュリティ設定を使用して、お使いのクラスターのデータ暗号化、Kerberos 認証、および EMRFS 用 Amazon S3 認証を設定できます。まずセキュリティ設定を作成してください。一度作成したセキュリティ設定は、クラスター作成時に繰り返し使用できます。

AWS Management Console、 AWS Command Line Interface（ AWS CLI ）、または AWS SDKsを使用してセキュリティ設定を作成できます。テンプレートを使用してセキュリティ設定 AWS CloudFormation を作成することもできます。詳細については、AWS CloudFormation 「 ユーザーガイド」および「 のテンプレートリファレンス」を参照してくださいAWS::EMR::SecurityConfiguration。

トピック

- セキュリティ設定を作成する
- クラスターのセキュリティ設定を指定する

セキュリティ設定を作成する

このトピックでは、Amazon EMR コンソールと を使用してセキュリティ設定を作成し AWS CLI、EMRFS の暗号化、認証、IAM ロールを構成するパラメータのリファレンスを作成する一般的な手順について説明します。これらの機能の詳細については、次のトピックを参照してください。

- 保管中と転送中のデータの暗号化
- Amazon EMR での認証に Kerberos を使用する

- Amazon S3 への EMRFS リクエストの IAM ロールを設定する

コンソールを使用してセキュリティ設定を作成するには

1. https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. ナビゲーションペインで [Security Configurations (セキュリティ設定)] を選択して、[Create security configuration (セキュリティ設定の作成)] を選択します。
3. セキュリティ設定の [Name (名前)] を入力します。
4. 以下のセクションで説明するように、[Encryption] (暗号化) および [Authentication] (認証) のオプションを選択し、[Create] (作成) を選択します。

を使用してセキュリティ設定を作成するには AWS CLI

- 次の例に示すように `create-security-configuration` コマンドを使用します。
  - には*SecConfigName*、セキュリティ設定の名前を指定します。この名前は、セキュリティ設定を使用するクラスターを作成する際に指定します。
  - *SecConfigDef* で、インライン JSON 構造またはローカル JSON ファイルへのパス (例: *file://MySecConfig.json*) を指定します。JSON パラメータは、以下のセクションで説明するように、[Encryption] (暗号化)、[IAM Roles for EMRFS access to Amazon S3] (Amazon S3 にアクセスするための EMRFS の IAM ロール)、および [Authentication] (認証) のオプションを定義します。

```
aws emr create-security-configuration --name "SecConfigName" --security-
configuration SecConfigDef
```

データ暗号化の設定

セキュリティ設定で暗号化を設定する前に、暗号化に使用するキーと証明書を作成します。詳細については、「Amazon EMR を使用した保管中のデータを暗号化するためのキーの提供」および「Amazon EMR 暗号化を使用して転送中のデータを暗号化するための証明書の提供」を参照してください。

セキュリティ設定を作成する場合、保管時のデータ暗号化および転送時のデータ暗号化の 2 種類の暗号化オプションを指定します。保管時のデータの暗号化オプションには、EMRFS での

Amazon S3 暗号化とローカルディスク暗号化の両方が含まれます。転送時の暗号化オプションでは、Transport Layer Security (TLS) をサポートする特定のアプリケーションで、オープンソースの暗号化機能を有効にします。保管時と転送時のオプションは、一緒にまたは個別に有効にできます。詳細については、「保管中と転送中のデータの暗号化」を参照してください。

> **Note**
>
> を使用する場合 AWS KMS、暗号化キーのストレージと使用に対して料金が適用されます。詳細については、「AWS KMS の料金」を参照してください。

コンソールを使用して暗号化オプションを指定する

次のガイドラインに従って、[暗号化] の下のオプションを選択します。

- ファイルシステム内に保存されたデータを暗号化するには、[At rest encryption (保管時の暗号化)] の下のオプションを選択します。

  Amazon S3、ローカルディスク、またはその両方でデータを暗号化するように選択できます。
- [S3 data encryption] (S3 データ暗号化) の [Encryption mode] (暗号化モード) で、Amazon EMR での EMRFS による Amazon S3 データの暗号化方法を指定する値を選択します。

  次に実行する操作は、選択する暗号化モードによって異なります。

  - SSE-S3

    Amazon S3 が管理する暗号化キーによるサーバー側の暗号化を指定します。Amazon S3 によってキーが自動的に処理されるため、それ以上何もすることはありません。
  - SSE-KMS または CSE-KMS

    マネージドキーによるサーバー側の暗号化 (SSE AWS KMS-KMS) または AWS KMSマネージドキーによるクライアント側の暗号化 (CSE-KMS) を指定します。[AWS KMS key] で、キーを選択します。キーは EMR クラスターと同じリージョンに存在する必要があります。キーの要件については、暗号化 AWS KMS keys に を使用するを参照してください。
  - CSE-Custom

    クライアント側のカスタムルートキーを使用したクライアント側の暗号化 (CSE-Custom) を指定します。[S3 object] (S3 オブジェクト) で、カスタムキープロバイダー JAR ファイルの Amazon S3 の場所を入力するか、Amazon S3 ARN を入力します。次に、キープロバイダーク

ラス に、EncryptionMaterialsProvider インターフェイスを実装するアプリケーションで宣言されたクラスの完全なクラス名を入力します。

- [Local disk encryption (ローカルディスク暗号化)] で、[Key provider type (キープロバイダーのタイプ)] の値を選択します。
  - AWS KMS key

    このオプションを選択して AWS KMS keyを指定します。[AWS KMS key] で、キーを選択します。キーは EMR クラスターと同じリージョンに存在する必要があります。キーの要件の詳細については、暗号化 AWS KMS keys に を使用するを参照してください。

    EBS 暗号化

    をキープロバイダー AWS KMS として指定すると、EBS 暗号化を有効にして EBS ルートデバイスとストレージボリュームを暗号化できます。このようなオプションを有効にするには、Amazon EMR サービスロール `EMR_DefaultRole` に、指定した AWS KMS key を使用するアクセス許可を付与する必要があります。キーの要件の詳細については、KMS キーに追加のアクセス許可を提供して EBS 暗号化を有効にするを参照してください。
  - カスタム

    カスタムキープロバイダーを指定するには、このオプションを選択します。[S3 object] (S3 オブジェクト) で、カスタムキープロバイダー JAR ファイルの Amazon S3 の場所を入力するか、Amazon S3 ARN を入力します。キープロバイダークラス には、EncryptionMaterialsProvider インターフェイスを実装するアプリケーションで宣言されたクラスの完全なクラス名を入力します。ここで提供するクラス名は、CSE カスタムで提供したクラス名とは異なる必要があります。
- 転送時の暗号化でオープンソース TLS 暗号化機能を有効にするには、[In-transit encryption (転送時の暗号化)] を選択します。以下のガイドラインに従って [Certificate provider type (証明書プロバイダーのタイプ)] を選択します。
  - PEM

    このオプションを選択すると、zip ファイル内で提供する PEM ファイルを使用できます。zip ファイル内では、privateKey.pem と certificateChain.pem の 2 つのアーティファクトが必要です。3 つ目の trustedCertificates.pem ファイルはオプションです。詳細については、「Amazon EMR 暗号化を使用して転送中のデータを暗号化するための証明書の提供」を参照してください。[S3 object] (S3 オブジェクト) で、zip ファイルフィールドの Amazon S3 の場所または Amazon S3 ARN を指定します。
  - カスタム

このオプションを選択してカスタム証明書プロバイダーを指定し、[S3 object] (S3 オブジェクト) で、カスタム証明書プロバイダー JAR ファイルの Amazon S3 の場所、または Amazon S3 ARN を指定します。キープロバイダークラス には、TLS ArtifactsProvider インターフェイスを実装するアプリケーションで宣言されたクラスの完全なクラス名を入力します。

を使用した暗号化オプションの指定 AWS CLI

以下のセクションでは、サンプルシナリオを使用して、設定、キープロバイダー、JSON パラメータの参照、および適切な値ごとに正しい形式の --security-configuration JSON を示します。

伝送中のデータ暗号化オプションの例

次の例は、以下のシナリオについて示しています。

- 伝送中のデータ暗号化が有効で、保管時のデータ暗号化が無効。
- Amazon S3 の証明書を含む zip ファイルがキープロバイダーとして使用されます (証明書の要件については、「Amazon EMR 暗号化を使用して転送中のデータを暗号化するための証明書の提供」を参照してください)

```
aws emr create-security-configuration --name "MySecConfig" --security-configuration '{
 "EncryptionConfiguration": {
  "EnableInTransitEncryption": true,
  "EnableAtRestEncryption": false,
  "InTransitEncryptionConfiguration": {
   "TLSCertificateConfiguration": {
    "CertificateProviderType": "PEM",
    "S3Object": "s3://MyConfigStore/artifacts/MyCerts.zip"
   }
  }
 }
}'
```

次の例は、以下のシナリオについて示しています。

- 伝送中のデータ暗号化が有効で、保管時のデータ暗号化が無効。
- カスタムキープロバイダーが使用されます (証明書の要件については、「Amazon EMR 暗号化を使用して転送中のデータを暗号化するための証明書の提供」を参照してください)。

```
aws emr create-security-configuration --name "MySecConfig" --security-configuration '{
 "EncryptionConfiguration": {
  "EnableInTransitEncryption": true,
  "EnableAtRestEncryption": false,
  "InTransitEncryptionConfiguration": {
   "TLSCertificateConfiguration": {
    "CertificateProviderType": "Custom",
    "S3Object": "s3://MyConfig/artifacts/MyCerts.jar",
    "CertificateProviderClass": "com.mycompany.MyCertProvider"
   }
  }
  }
}'
```

保管時のデータ暗号化オプションの例

次の例は、以下のシナリオについて示しています。

- 伝送中のデータ暗号化が無効で、保管時のデータ暗号化が有効です。
- SSE-S3 が Amazon S3 の暗号化に使用されます。
- ローカルディスク暗号化では、 をキープロバイダー AWS KMS として使用します。

```
aws emr create-security-configuration --name "MySecConfig" --security-configuration '{
 "EncryptionConfiguration": {
  "EnableInTransitEncryption": false,
  "EnableAtRestEncryption": true,
  "AtRestEncryptionConfiguration": {
   "S3EncryptionConfiguration": {
    "EncryptionMode": "SSE-S3"
   },
   "LocalDiskEncryptionConfiguration": {
    "EncryptionKeyProviderType": "AwsKms",
    "AwsKmsKey": "arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012"
   }
  }
  }
}'
```

次の例は、以下のシナリオについて示しています。

- 伝送中のデータ暗号化が有効で、ARN を使用して Amazon S3 の PEM 証明書を含む zip ファイルを参照します。
- SSE-KMS が Amazon S3 の暗号化に使用されます。
- ローカルディスク暗号化では、 をキープロバイダー AWS KMS として使用します。

```
aws emr create-security-configuration --name "MySecConfig" --security-configuration '{
 "EncryptionConfiguration": {
  "EnableInTransitEncryption": true,
  "EnableAtRestEncryption": true,
  "InTransitEncryptionConfiguration": {
   "TLSCertificateConfiguration": {
    "CertificateProviderType": "PEM",
    "S3Object": "arn:aws:s3:::MyConfigStore/artifacts/MyCerts.zip"
   }
  },
  "AtRestEncryptionConfiguration": {
   "S3EncryptionConfiguration": {
    "EncryptionMode": "SSE-KMS",
    "AwsKmsKey": "arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012"
   },
   "LocalDiskEncryptionConfiguration": {
    "EncryptionKeyProviderType": "AwsKms",
    "AwsKmsKey": "arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012"
   }
  }
 }
}'
```

次の例は、以下のシナリオについて示しています。

- 伝送中のデータ暗号化が有効で、Amazon S3 の PEM 証明書を含む zip ファイルを参照します。
- CSE-KMS が Amazon S3 の暗号化に使用されます。
- ローカルディスク暗号化では、ARN で参照されたカスタムキープロバイダーが使用されます。

```
aws emr create-security-configuration --name "MySecConfig" --security-configuration '{
```

```
 "EncryptionConfiguration": {
  "EnableInTransitEncryption": true,
  "EnableAtRestEncryption": true,
  "InTransitEncryptionConfiguration": {
   "TLSCertificateConfiguration": {
    "CertificateProviderType": "PEM",
    "S3Object": "s3://MyConfigStore/artifacts/MyCerts.zip"
   }
  },
  "AtRestEncryptionConfiguration": {
   "S3EncryptionConfiguration": {
    "EncryptionMode": "CSE-KMS",
    "AwsKmsKey": "arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012"
   },
   "LocalDiskEncryptionConfiguration": {
    "EncryptionKeyProviderType": "Custom",
    "S3Object": "arn:aws:s3:::artifacts/MyKeyProvider.jar",
    "EncryptionKeyProviderClass": "com.mycompany.MyKeyProvider"
   }
  }
 }
}'
```

次の例は、以下のシナリオについて示しています。

- 伝送中のデータ暗号化は、カスタムキープロバイダーで有効になります。
- Amazon S3 データに CSE-Custom が使用されます。
- ローカルディスク暗号化ではカスタムキープロバイダーが使用されます。

```
aws emr create-security-configuration --name "MySecConfig" --security-configuration '{
 "EncryptionConfiguration": {
  "EnableInTransitEncryption": "true",
  "EnableAtRestEncryption": "true",
  "InTransitEncryptionConfiguration": {
   "TLSCertificateConfiguration": {
    "CertificateProviderType": "Custom",
    "S3Object": "s3://MyConfig/artifacts/MyCerts.jar",
    "CertificateProviderClass": "com.mycompany.MyCertProvider"
   }
```

```
  },
  "AtRestEncryptionConfiguration": {
   "S3EncryptionConfiguration": {
    "EncryptionMode": "CSE-Custom",
    "S3Object": "s3://MyConfig/artifacts/MyCerts.jar",
    "EncryptionKeyProviderClass": "com.mycompany.MyKeyProvider"
    },
   "LocalDiskEncryptionConfiguration": {
    "EncryptionKeyProviderType": "Custom",
    "S3Object": "s3://MyConfig/artifacts/MyCerts.jar",
    "EncryptionKeyProviderClass": "com.mycompany.MyKeyProvider"
   }
  }
 }
}'
```

次の例は、以下のシナリオについて示しています。

- 伝送中のデータ暗号化が無効で、保管時のデータ暗号化が有効です。
- Amazon S3 の暗号化は SSE-KMS で有効になっています。
- S3 バケットごとに 1 つずつ複数の AWS KMS キーが使用され、暗号化例外がこれらの個々の S3 バケットに適用されます。
- ローカルディスク暗号化が無効になっています。

```
aws emr create-security-configuration --name "MySecConfig" --security-configuration '{
   "EncryptionConfiguration": {
     "AtRestEncryptionConfiguration": {
             "S3EncryptionConfiguration": {
           "EncryptionMode": "SSE-KMS",
           "AwsKmsKey": "arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012",
           "Overrides": [
              {
                "BucketName": "sse-s3-bucket-name",
                "EncryptionMode": "SSE-S3"
              },
              {
                "BucketName": "cse-kms-bucket-name",
                "EncryptionMode": "CSE-KMS",
                "AwsKmsKey": "arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012"
```

```
              },
              {
                "BucketName": "sse-kms-bucket-name",
                "EncryptionMode": "SSE-KMS",
                "AwsKmsKey": "arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012"
              }
             ]
             }
           },
     "EnableInTransitEncryption": false,
      "EnableAtRestEncryption": true
  }
}'
```

次の例は、以下のシナリオについて示しています。

- 伝送中のデータ暗号化が無効で、保管時のデータ暗号化が有効です。
- Amazon S3 暗号化は SSE-S3 で有効で、ローカルディスク暗号化は無効になっています。

```
aws emr create-security-configuration --name "MyS3EncryptionConfig" --security-
configuration '{
    "EncryptionConfiguration": {
        "EnableInTransitEncryption": false,
        "EnableAtRestEncryption": true,
        "AtRestEncryptionConfiguration": {
            "S3EncryptionConfiguration": {
                "EncryptionMode": "SSE-S3"
            }
        }
     }
}'
```

次の例は、以下のシナリオについて示しています。

- 伝送中のデータ暗号化が無効で、保管時のデータ暗号化が有効です。
- ローカルディスク暗号化は、キープロバイダー AWS KMS として で有効になり、Amazon S3 暗号化は無効になります。

```
aws emr create-security-configuration --name "MyLocalDiskEncryptionConfig" --security-
configuration '{
    "EncryptionConfiguration": {
        "EnableInTransitEncryption": false,
        "EnableAtRestEncryption": true,
        "AtRestEncryptionConfiguration": {
            "LocalDiskEncryptionConfiguration": {
                "EncryptionKeyProviderType": "AwsKms",
                "AwsKmsKey": "arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012"
            }
        }
     }
}'
```

次の例は、以下のシナリオについて示しています。

- 伝送中のデータ暗号化が無効で、保管時のデータ暗号化が有効です。
- ローカルディスク暗号化は、キープロバイダー AWS KMS として で有効になり、Amazon S3 暗号化は無効になります。
- EBS 暗号化が有効になっています。

```
aws emr create-security-configuration --name "MyLocalDiskEncryptionConfig" --security-
configuration '{
    "EncryptionConfiguration": {
        "EnableInTransitEncryption": false,
        "EnableAtRestEncryption": true,
        "AtRestEncryptionConfiguration": {
            "LocalDiskEncryptionConfiguration": {
                "EnableEbsEncryption": true,
                "EncryptionKeyProviderType": "AwsKms",
                "AwsKmsKey": "arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012"
            }
        }
     }
}'
```

次の例は、以下のシナリオについて示しています。

SSE-EMR-WAL は EMR WAL 暗号化に使用されます

```
aws emr create-security-configuration --name "MySecConfig" \
    --security-configuration '{
        "EncryptionConfiguration": {
            "EMRWALEncryptionConfiguration":{ },
            "EnableInTransitEncryption":false, "EnableAtRestEncryption":false
        }
    }'
```

EnableInTransitEncryption 関連する暗号化を有効にする場合、 とはEnableAtRestEncryption引き続き true になる可能性があります。

次の例は、以下のシナリオについて示しています。

- SSE-KMS-WAL は EMR WAL 暗号化に使用されます
- サーバー側の暗号化でキープロバイダー AWS Key Management Service として を使用する

```
aws emr create-security-configuration --name "MySecConfig" \
    --security-configuration '{
        "EncryptionConfiguration": {
            "EMRWALEncryptionConfiguration":{
                "AwsKmsKey":"arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012"
                },
            "EnableInTransitEncryption":false, "EnableAtRestEncryption":false
        }
    }'
```

EnableInTransitEncryption 関連する暗号化を有効にする場合、 とはEnableAtRestEncryption引き続き true になる可能性があります。

暗号化設定の JSON 参照

次の表には、暗号化設定用の JSON パラメータが一覧表示されており、各パラメータで許可される値について説明しています。

パラメータ	説明
"EnableInTransitEncryption" : true | false	true を指定すると転送時の暗号化が有効になり、false を指定すると無効になります。省略すると、false とみなされ、転送時の暗号化は無効になります。
"EnableAtRestEncryption": true | false	true を指定すると保管時の暗号化が有効になり、false を指定すると無効になります。省略すると、false とみなされ、保管時の暗号化は無効になります。
伝送中の暗号化パラメータ	
"InTransitEncryptionConfiguration" :	EnableInTransitEncryption が true の場合に、転送時の暗号化を設定するのに使用される値のコレクションを指定します。
"CertificateProviderType": "PEM" | "Custom"	圧縮ファイルで参照されている PEM 証明書を使用するか、Custom プロバイダーを使用するかを指定します。PEM を指定する場合、 は証明書を含む zip ファイルの Amazon S3 内の場所への参照S3Objectである必要があります。Custom を指定する場合、 は JAR ファイルの Amazon S3 内の場所への参照で、その後にCertificateProviderClass エントリが続くS3Object必要があります。
"S3Object" : " *ZipLocation* " | "*JarLocation* "	が指定されている場合は zip ファイルに、PEMが指定されている場合は JAR ファイルに、Amazon S3 Custom 内の場所を提供します。形式は、パス (例: s3://MyConfig/artifacts/CertFiles.zip) または ARN (例: arn:aws:s3:::Code/MyCertProvider.jar)) にできます。zip ファイルを指定した場合、privateKey.pem と certificateChain.pem という正確な名前のファイルを含める必要がありま

パラメータ	説明
	す。trustedCertificates.pem という名前のファイルはオプションです。
"CertificateProviderClass" : "*MyClassID* "	Custom が に指定されている場合にのみ必要ですCertificateProviderType 。は、TLS ArtifactsProvider インターフェイスを実装する JAR ファイルで宣言された完全なクラス名*MyClassID* を指定します。例えば com.mycompany.MyCertProvider です。
保管時の暗号化パラメータ	
"AtRestEncryptionConfiguration" :	が の場合、Amazon S3 EnableAtRestEncryption 暗号化やローカルディスク暗号化などtrue、保管時の暗号化の値のコレクションを指定します。
Amazon S3 暗号化パラメータ	
"S3EncryptionConfiguration" :	Amazon EMR File System (EMRFS) による Amazon S3 暗号化に使用される値のコレクションを指定します。
"EncryptionMode" : "SSE-S3" | "SSE-KMS" | "CSE-KMS" | "CSE-Custom"	使用する Amazon S3 暗号化のタイプを指定します。SSE-S3 を指定した場合、それ以上 Amazon S3 暗号化値は必要ありません。SSE-KMS または のいずれかCSE-KMSを指定する場合は、ARN AWS KMS key をAwsKmsKey 値として指定する必要があります。CSE-Custom を指定した場合、S3Object と EncryptionKeyProviderClass の値を指定する必要があります。

パラメータ	説明
"AwsKmsKey" : " *MyKeyARN*"	SSE-KMS または CSE-KMS が EncryptionMode で指定された場合にのみ必要です。*MyKeyARN* はキーへの完全修飾 ARN にする必要があります (例: arn:aws:kms:us-east-1:123456789012:key/12345678-1234-1234-1234-123456789012)。
"S3Object" : "*JarLocation* "	CSE-Custom が に指定されている場合にのみ必要ですCertificateProviderType 。*JarLocation* は Amazon S3 内の場所を JAR ファイルに提供します。形式は、パス (例: s3://MyConfig/artifacts/MyKeyProvider.jar) または ARN (例: arn:aws:s3:::Code/MyKeyProvider.jar) にできます。
"EncryptionKeyProviderClass" : "*MyS3KeyClassID* "	CSE-Custom が に指定されている場合にのみ必要ですEncryptionMode 。 は、インターフェイスを実装 EncryptionMaterialsProviderするアプリケーションで宣言されたクラスの完全なクラス名*MyS3KeyClassID* を指定しますcom.mycompany.MyS3KeyProvider 。例えば、。
ローカルディスクの暗号化パラメータ	
"LocalDiskEncryptionConfiguration"	ローカルディスクの暗号化に使用するキープロバイダーと対応する値を指定します。
"EnableEbsEncryption": true | false	EBS 暗号化を有効にするtrueには、 を指定します。EBS 暗号化は、EBS ルートデバイスボリュームとアタッチされたストレージボリュームを暗号化します。EBS 暗号化を使用するには、 を AwsKmsとして指定する必要がありますEncryptionKeyProviderType 。

パラメータ	説明
"EncryptionKeyProviderType": "AwsKms" | "Custom"	キープロバイダーを指定します。AwsKms が指定されている場合、KMS キー ARN をAwsKmsKey 値として指定する必要があります。**Custom** を指定した場合、S3Object と EncryptionKeyProviderClass の値を指定する必要があります。
"AwsKmsKey : " *MyKeyARN*"	AwsKms が に指定されている場合にのみ必要ですType。 はキーに対して完全に指定された ARN *MyKeyARN*である必要があります (例: arn:aws:kms:us-east-1:12345 6789012:key/12345678-1234-1 234-1234-456789012123)。
"S3Object" : "*JarLocation* "	CSE-Custom が に指定されている場合にのみ必要ですCertificateProviderType 。*JarLocation* は Amazon S3 内の場所を JAR ファイルに提供します。形式は、パス (例: s3://MyConfig/artifacts/ MyKeyProvider.jar) または ARN (例: arn:aws:s3:::Code/MyKeyProv ider.jar) にできます。
"EncryptionKeyProviderClass" : "*MyLocalDiskKeyClassID* "	Custom が に指定されている場合にのみ必要ですType。 は、インターフェイスを実装 EncryptionMaterialsProviderするアプリケーションで宣言されたクラスの完全なクラス名*MyLocalDiskKeyClassID* を指定します。例えば、 です*com.mycompany.MyLo calDiskKeyProvider* 。
EMR WAL 暗号化パラメータ	
"EMRWALEncryptionConfiguration"	EMR WAL 暗号化の値を指定します。

パラメータ	説明
"AwsKmsKey"	CMK キー ID Arn を指定します。

Kerberos 認証の設定

Kerberos 設定を使用したセキュリティ設定は、Kerberos 属性を使用して作成されているクラスターでのみ使用できます。それ以外の場合はエラーが発生します。詳細については、「Amazon EMR での認証に Kerberos を使用する」を参照してください。Kerberos は Amazon EMR リリースバージョン 5.10.0 以降でのみ使用できます。

コンソールを使用して Kerberos 設定を指定する

次のガイドラインに従って、[Kerberos authentication (Kerberos 認証)] のオプションを選択します。

パラメータ		説明
Kerberos		このセキュリティ設定を使用するクラスターで Kerberos を有効にすることを指定します。クラスターがこのセキュリティ設定を使用する場合、クラスターで Kerberos 設定も指定する必要があります。そうしないと、エラーが発生します。
プロバイダー	クラスター専用 KDC	Amazon EMR が、このセキュリティ設定を使用するクラスターのプライマリノードに KDC を作成することを指定します。領域名と KDC 管理者パスワードは、クラスターの作成時に指定します。 必要に応じて、この KDC を他のクラスターから参照できます。異なるセキュリティ設定を使用してこれらのクラスターを作成し、外部 KDC を指定し、クラスター専用 KDC に指定した領域名と KDC 管理者パスワードを使用します。
	外部 KDC	Amazon EMR 5.20.0 以降でのみ利用できます。このセキュリティ設定を使用するクラスターが、クラスター外の KDC サーバーを使用して Kerberos プリンシパルを認証するように指定します。KDC はクラス

パラメータ		説明
		ター上に作成されません。クラスターの作成時に、外部 KDC の領域名と KDC 管理者パスワードを指定します。
チケットのライフタイム		オプション。このセキュリティ設定を使用するクラスターで KDC によって発行された Kerberos チケットが有効である期間を指定します。 チケットの有効期間は、セキュリティ上の理由により制限されます。クラスターアプリケーションとサービスでは、期限が切れるとチケットを自動更新します。Kerberos 認証情報を使用して SSH 経由でクラスターに接続する場合は、チケットの有効期限が切れたら、プライマリノードのコマンドラインから `kinit` を実行して更新する必要があります。
クロス領域信頼		このセキュリティ設定を使用するクラスター上のクラスター専用 KDC と、異なる Kerberos 領域内の KDC との間のクロス領域信頼を指定します。 別の領域のプリンシパル (通常はユーザー) は、この設定を使用するクラスターに対して認証されます。他の Kerberos 領域での追加設定が必要です。詳細については、「チュートリアル: Active Directory ドメインを使用したクロス領域信頼の設定」を参照してください。
クロス領域信頼プロパティ	領域	信頼関係の他の領域の Kerberos 領域名を指定します。慣例により、Kerberos 領域名はドメイン名と同じにします。ただし、すべて大文字にします。
	[ドメイン]	信頼関係の他の領域のドメイン名を指定します。

パラメータ		説明
	[Admin server] (管理者サーバー)	信頼関係の他の領域の管理サーバーの完全修飾ドメイン名 (FQDN) または IP アドレスを指定します。通常、管理サーバーと KDC サーバーは同じ FQDN を持つ同じマシン上で実行されますが、通信には別のポートを使用します。 ポートを指定しない場合、Kerberos のデフォルトであるポート 749 が使用されます。オプションで、ポート (`domain.example.com` *`:749`* など) を指定できます。
	[KDC server] (KDC サーバー)	信頼関係の他の領域の KDC サーバーの完全修飾ドメイン名 (FQDN) または IP アドレスを指定します。通常、KDC サーバーと管理サーバーは同じ FQDN を持つ同じマシン上で実行されますが、別のポートを使用します。 ポートを指定しない場合、Kerberos のデフォルトであるポート 88 が使用されます。オプションで、ポート (`domain.example.com` *`:88`* など) を指定できます。
外部 KDC		クラスター外部 KDC がクラスターで使用されることを指定します。
外部 KDC プロパティ	[Admin server] (管理者サーバー)	外部管理サーバーの完全修飾ドメイン名 (FQDN) または IP アドレスを指定します。通常、管理サーバーと KDC サーバーは同じ FQDN を持つ同じマシン上で実行されますが、通信には別のポートを使用します。 ポートを指定しない場合、Kerberos のデフォルトであるポート 749 が使用されます。オプションで、ポート (`domain.example.com` *`:749`* など) を指定できます。

パラメータ			説明
	[KDC server] (KDC サーバー)		外部 KDC サーバーの完全修飾ドメイン名 (FQDN) を指定します。通常、KDC サーバーと管理サーバーは同じ FQDN を持つ同じマシン上で実行されますが、別のポートを使用します。 ポートを指定しない場合、Kerberos のデフォルトであるポート 88 が使用されます。オプションで、ポート (`domain.example.com :88` など) を指定できます。
	[Active Directory Integration] (Active Directory の統合)		Kerberos プリンシパル認証が Microsoft Active Directory ドメインに統合されることを指定します。
	Active Directory 統合プロパティ	[Active Directory realm] (Active Directory 領域)	Active Directory ドメインの Kerberos 領域名を指定します。慣例により、Kerberos 領域名は通常、ドメイン名と同じにします。ただし、すべて大文字にします。
		[Active Directory domain] (Active Directory ドメイン)	Active Directory ドメイン名を指定します。
		[Active Directory server] (Active Directory サーバー)	Microsoft Active Directory ドメインコントローラーの完全修飾ドメイン名 (FQDN) を指定します。

を使用した Kerberos 設定の指定 AWS CLI

次のリファレンス表では、セキュリティ設定における Kerberos 設定の JSON パラメータを示しています。設定例については、「設定例」を参照してください。

パラメータ	説明
`"AuthenticationConfiguration": {`	Kerberos の場合は必須です。認証設定がこのセキュリティ設定の一部であることを指定します。
`"KerberosConfiguration": {`	Kerberos の場合は必須です。Kerberos 設定プロパティを指定します。

`"Provider": "ClusterDedicatedKdc",`
- または -
`"Provider": "ExternalKdc",`
	ClusterDedicatedKdc は、Amazon EMR が、このセキュリティ設定を使用するクラスターのプライマリノードに KDC を作成することを指定します。領域名と KDC 管理者パスワードは、クラスターの作成時に指定します。必要に応じて、この KDC を他のクラスターから参照できます。異なるセキュリティ設定を使用してこれらのクラスターを作成し、外部 KDC を指定し、クラスター専用 KDC を使用してクラスターを作成したときに指定した領域名と KDC 管理者パスワードを使用します。
	
	ExternalKdc は、クラスターが外部 KDC を使用することを指定します。Amazon EMR は、プライマリノードに KDC を作成しません。このセキュリティ設定を使用するクラスターでは、外部 KDC の領域名と KDC 管理者パスワードを指定する必要があります。

パラメータ	説明
"ClusterDedicatedKdcConfiguration": {	*ClusterDedicatedKdc* が指定されている場合は必須です。
"TicketLifetimeInHours": *24*,	オプション。このセキュリティ設定を使用するクラスターで KDC によって発行された Kerberos チケットが有効である期間を指定します。 チケットの有効期間は、セキュリティ上の理由により制限されます。クラスターアプリケーションとサービスでは、期限が切れるとチケットを自動更新します。Kerberos 認証情報を使用して SSH 経由でクラスターに接続する場合は、チケットの有効期限が切れたら、プライマリノードのコマンドラインから `kinit` を実行して更新する必要があります。
"CrossRealmTrustConfiguration": {	このセキュリティ設定を使用するクラスター上のクラスター専用 KDC と、異なる Kerberos 領域内の KDC との間のクロス領域信頼を指定します。 別の領域のプリンシパル (通常はユーザー) は、この設定を使用するクラスターに対して認証されます。他の Kerberos 領域での追加設定が必要です。詳細については、「チュートリアル: Active Directory ドメインを使用したクロス領域信頼の設定」を参照してください。

パラメータ			説明
		"Realm": "*KDC2.COM*",	信頼関係の他の領域の Kerberos 領域名を指定します。慣例により、Kerberos 領域名はドメイン名と同じにします。ただし、すべて大文字にします。
		"Domain": "*kdc2.com*",	信頼関係の他の領域のドメイン名を指定します。
		"AdminServer": "*kdc.com:749* ",	信頼関係の他の領域の管理サーバーの完全修飾ドメイン名 (FQDN) または IP アドレスを指定します。通常、管理サーバーと KDC サーバーは同じ FQDN を持つ同じマシン上で実行されますが、通信には別のポートを使用します。 ポートを指定しない場合、Kerberos のデフォルトであるポート 749 が使用されます。オプションで、ポート (domain.example.com *:749* など) を指定できます。
		"KdcServer": "*kdc.com:88* "	信頼関係の他の領域の KDC サーバーの完全修飾ドメイン名 (FQDN) または IP アドレスを指定します。通常、KDC サーバーと管理サーバーは同じ FQDN を持つ同じマシン上で実行されますが、別のポートを使用します。 ポートを指定しない場合、Kerberos のデフォルトであるポート 88 が使用されます。オプションで、ポート (domain.example.com *:88* など) を指定できます。

パラメータ			説明
		}	
		}	
		"ExternalKdcConfiguration": {	*ExternalKdc* が指定されている場合は必須です。
		"TicketLifetimeInHours": *24*,	オプション。このセキュリティ設定を使用するクラスターで KDC によって発行された Kerberos チケットが有効である期間を指定します。 チケットの有効期間は、セキュリティ上の理由により制限されます。クラスターアプリケーションとサービスでは、期限が切れるとチケットを自動更新します。Kerberos 認証情報を使用して SSH 経由でクラスターに接続する場合は、チケットの有効期限が切れたら、プライマリノードのコマンドラインから kinit を実行して更新する必要があります。
		"KdcServerType": "Single",	単一の KDC サーバーを参照するように指定します。Single は、現在サポートされている唯一の値です。

パラメータ				説明
			AdminServer「」：*kdc.com:749* 」、	外部管理サーバーの完全修飾ドメイン名 (FQDN) または IP アドレスを指定します。通常、管理サーバーと KDC サーバーは同じ FQDN を持つ同じマシン上で実行されますが、通信には別のポートを使用します。 ポートを指定しない場合、Kerberos のデフォルトであるポート 749 が使用されます。オプションで、ポート (`domain.example.com` *:749* など) を指定できます。
			KdcServer「」：*kdc.com:88* 」、	外部 KDC サーバーの完全修飾ドメイン名 (FQDN) を指定します。通常、KDC サーバーと管理サーバーは同じ FQDN を持つ同じマシン上で実行されますが、別のポートを使用します。 ポートを指定しない場合、Kerberos のデフォルトであるポート 88 が使用されます。オプションで、ポート (`domain.example.com` *:88* など) を指定できます。
			`"AdIntegrationConfiguration": {`	Kerberos プリンシパル認証が Microsoft Active Directory ドメインに統合されることを指定します。
			`"AdRealm": "`*AD.DOMAIN.COM* `",`	Active Directory ドメインの Kerberos 領域名を指定します。慣例により、Kerberos 領域名は通常、ドメイン名と同じにします。ただし、すべて大文字にします。

パラメータ	説明
"AdDomain": "*ad.domain.com* "	Active Directory ドメイン名を指定します。
"AdServer": "*ad.domain.com* "	Microsoft Active Directory ドメインコントローラーの完全修飾ドメイン名 (FQDN) を指定します。

```
        }
      }
    }
}
```

Amazon S3 への EMRFS リクエストの IAM ロールを設定する

EMRFS の IAM ロールを使用すると、Amazon S3 の EMRFS データに異なるアクセス権限を付与することができます。指定した ID がアクセスリクエストに含まれている場合は、アクセス権限で使用する IAM ロールを指定するロールマッピングを作成します。識別子は、Hadoop ユーザーかロール、または Amazon S3 プレフィックスです。

詳細については、「Amazon S3 への EMRFS リクエストの IAM ロールを設定する」を参照してください。

を使用した EMRFS の IAM ロールの指定 AWS CLI

以下は、セキュリティ設定内の EMRFS のカスタム IAM ロールを指定するための JSON スニペットの例です。3 つの異なる識別子タイプのロールマッピングと、パラメータリファレンスを示します。

```
{
  "AuthorizationConfiguration": {
    "EmrFsConfiguration": {
      "RoleMappings": [{
        "Role": "arn:aws:iam::123456789101:role/allow_EMRFS_access_for_user1",
        "IdentifierType": "User",
        "Identifiers": [ "user1" ]
```

```
      },{
        "Role": "arn:aws:iam::123456789101:role/allow_EMRFS_access_to_MyBuckets",
        "IdentifierType": "Prefix",
        "Identifiers": [ "s3://MyBucket/","s3://MyOtherBucket/" ]
      },{
        "Role": "arn:aws:iam::123456789101:role/allow_EMRFS_access_for_AdminGroup",
        "IdentifierType": "Group",
        "Identifiers": [ "AdminGroup" ]
      }]
    }
  }
}
```

パラメータ	説明
"AuthorizationConfiguration":	必須。
"EmrFsConfiguration":	必須。ロールマッピングが含まれます。
"RoleMappings":	必須。1 つ以上のロールマッピング定義が含まれます。ロールのマッピングは、表示順に上から下に評価されます。Amazon S3 のデータに対する EMRFS 呼び出しでロールマッピングが true と評価された場合、それ以上のロールマッピングは評価されず、EMRFS はリクエストに指定された IAM ロールを使用します。ロールマッピングは、以下の必須パラメータで構成されます。
"Role":	IAM ロールの ARN 識別子を形式 arn:aws:iam:: *account-id* :role/*role-name* で指定します。これは、Amazon S3 への EMRFS リクエストが、指定された Identifiers のいずれかに一致した場合に Amazon EMR が引き受ける IAM ロールです。
"IdentifierType":	次のいずれかを指定できます。 • "User" は、識別子が 1 人以上の Hadoop ユーザーであることを指定します。これは

パラメータ	説明
	Linux アカウントユーザーまたは Kerberos プリンシパルです。EMRFS リクエストが指定のユーザーから送信されると、IAM ロールが引き受けられます。 • "Prefix" は、識別子が Amazon S3 の場所であることを指定します。IAM ロールは、指定されたプレフィックスを持つ場所への呼び出しに対して引き受けられます。例えば、プレフィックス s3://mybucket/ は、s3://mybucket/mydir および s3://mybucket/yetanotherdir に一致します。 • "Group" は、識別子が 1 つ以上の Hadoop グループであることを指定します。IAM ロールは、リクエストが指定されたグループのユーザーから発信された場合に引き受けられます。
"Identifiers":	適切な識別子タイプの 1 つ以上の識別子を指定します。複数の識別子は、スペースを入れずにカンマで区切ります。

Amazon EC2 インスタンスへのメタデータサービスリクエストを設定する

インスタンスメタデータは、インスタンスに関するデータで、実行中のインスタンスを設定または管理するために使用します。次のいずれかのメソッドを使って、実行中のインスタンスからインスタンスメタデータにアクセスできます。

- インスタンスメタデータサービスバージョン 1 (IMDSv1) – リクエスト/レスポンスメソッド
- インスタンスメタデータサービスバージョン 2 (IMDSv2) – セッション指向メソッド

Amazon EC2 は IMDSv1 と IMDSv2 の両方をサポートしていますが、Amazon EMR 5.23.1、5.27.1、5.32 以降、6.2 以降では IMDSv2 をサポートしています。これらのリリースでは、Amazon EMR コンポーネントはすべての IMDS 呼び出しに IMDSv2 を使用します。アプリケー

ションコードでの IMDS 呼び出しでは、IMDSv1 と IMDSv2 の両方を使用するか、または、セキュリティを強化するために IMDSv2 のみを使用するように IMDS を設定できます。IMDSv2を使用しなければならないように指定すると、IMDSv1はもう機能しなくなります。

詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスメタデータサービスの設定」を参照してください。 Amazon EC2

> **Note**
>
> 以前の Amazon EMR 5.x または 6.x リリースでは、IMDSv1 をオフにすると、Amazon EMR コンポーネントがすべての IMDS 呼び出しに IMDSv1 を使用するため、クラスターの起動に失敗します。IMDSv1 をオフにするときは、IMDSv1 を使用するカスタムソフトウェアが IMDSv2 に更新されていることを確認してください。

AWS CLIを使用してインスタンスメタデータサービス設定を指定する

以下は、セキュリティ設定内で Amazon EC2 インスタンスメタデータサービス (IMDS) を指定するための JSON スニペットの例です。カスタムセキュリティ設定の使用はオプションです。

```
{
  "InstanceMetadataServiceConfiguration" : {
      "MinimumInstanceMetadataServiceVersion": integer,
      "HttpPutResponseHopLimit": integer
   }
}
```

パラメータ	説明
`"InstanceMetadataServiceConfiguration":`	セキュリティ設定内で IMDS を指定せず、IMDSv1 を必要とする Amazon EMR リリースを使用する場合、Amazon EMR はデフォルトで IMDSv1 を最小インスタンスメタデータサービスバージョンとして使用します。独自の設定を使用する場合は、次のパラメータの両方が必要です。

パラメータ	説明
`"MinimumInstanceMetadataServiceVersion":`	必須。1 または 2 を指定します。値 1 では、IMDSv1 と IMDSv2 が許可されます。値 2 では、IMDSv2 のみが許可されます。
`"HttpPutResponseHopLimit":`	必須。インスタンスメタデータリクエストで必要な HTTP PUT レスポンスのホップ制限。数値が大きいほど、インスタンスメタデータリクエストの転送距離を伸ばすことができます。デフォルト: 1。1 ~ 64 の整数を指定します。

コンソールを使用してインスタンスメタデータサービス設定を指定する

Amazon EMR コンソールから起動するときに、クラスターの IMDS の使用を設定できます。Amazon EMR コンソールでの IMDS セキュリティ設定コントロール

コンソールを使用して IMDS の使用を設定するには

1. [Security configurations] (セキュリティ設定) ページで新しいセキュリティ設定を作成する場合、[EC2 Instance Metadata Service] (EC2 インスタンスメタデータサービス) 設定で [Configure EC2 Instance metadata service] (EC2 インスタンスメタデータサービスの設定) を選択します。この設定は、Amazon EMR 5.23.1、5.27.1、5.32 以降、および 6.2 以降でのみサポートされます。
2. [Minimum Instance Metadata Service Version] (インスタンスメタデータサービスの最小バージョン) オプションで、次のいずれかを選択します。
   - [Turn off IMDSv1 and only allow IMDSv2] (IMDSv1 をオフにして IMDSv2 のみを許可する)。これは、このクラスターで IMDSv2 のみを許可する場合に選択します。「Amazon EC2 ユーザーガイド」の「インスタンスメタデータサービスバージョン 2 の使用への移行」を参照してください。 Amazon EC2
   - [Allow both IMDSv1 and IMDSv2 on cluster] (クラスターで IMDSv1 と IMDSv2 の両方を許可する)。これは、このクラスターで IMDSv1 およびセッション指向の IMDSv2 を許可する場合に選択します。
3. IMDSv2 では、[HTTP put response hop limit] (HTTP PUT レスポンスのホップ制限) を 1 ~ 64 の整数に設定して、メタデータトークンに許容されるネットワークホップ数を設定することもできます。

詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスメタデータサービスの設定」を参照してください。 Amazon EC2

「Amazon EC2 ユーザーガイド」の「インスタンスの詳細を設定する」および「インスタンスメタデータサービスを設定するAmazon EC2」を参照してください。

クラスターのセキュリティ設定を指定する

セキュリティ設定を指定して、クラスターの作成時に暗号化の設定を指定できます。 AWS Management Console または を使用できます AWS CLI。

> **Note**
>
> Amazon EMR コンソールは、再設計され、使いやすくなりました。古いコンソールと新しいコンソールエクスペリエンスの違いについては、「Amazon EMR コンソール」を参照してください。

New console

新しいコンソールを使用してセキュリティ設定を指定する方法

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で、[クラスター] を選択し、[クラスターの作成] を選択します
3. [セキュリティの設定とアクセス許可] で [セキュリティ設定] フィールドを見つけます。ドロップダウンメニューを選択するか、[参照] を選択して、以前に作成したセキュリティ設定の名前を選択します。または、[セキュリティ設定の作成] を選択して、クラスターに使用できる設定を作成します。
4. クラスターに適用するその他のオプションを選択します。
5. クラスターを起動するには、[クラスターの作成] を選択します。

Old console

古いコンソールを使用してセキュリティ設定を指定する方法

1. https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。

2. [Create cluster (クラスターの作成)]、[Go to advanced options (詳細オプションに移動する)] の順に選択します。
3. [Step 1: Software and Steps] (ステップ 1: ソフトウェアとステップ) 画面で、[Release] (リリース) リストから、[emr-4.8.0] またはそれ以降のリリースを選択します。該当する設定を選択し、[Next (次へ)] を選択します。
4. [Step 2: Hardware (ステップ 2: ハードウェア)] 画面で、該当する設定を選択して、[Next (次へ)] を選択します。[Step 3: General Cluster Settings (ステップ 3: クラスターの全般設定)] でも同様にします。
5. [Step 4: Security (ステップ 4: セキュリティ)] 画面で、[Encryption Options (暗号化オプション)] の一覧から [Security configuration (セキュリティ設定)] を選択します。
6. 他のセキュリティオプションを必要に応じて設定し、[Create cluster (クラスターの作成)] を選択します。

CLI

を使用してセキュリティ設定を指定するには AWS CLI

- オプションでセキュリティ設定を適用するには、`aws emr create-cluster` を使用して `--security-configuration` *`MySecConfig`* を指定します。*`MySecConfig`* は、次の例に示すようにセキュリティ設定の名前です。`--release-label` には 4.8.0 以降を指定する必要があり、`--instance-type` には任意のものを指定することができます。

```
aws emr create-cluster --instance-type m5.xlarge --release-label emr-5.0.0 --
security-configuration mySecConfig
```

Amazon EMR でのデータ保護

AWS 責任共有モデルは、Amazon EMR でのデータ保護に適用されます。このモデルで説明されているように、 AWS はすべての AWS クラウドを実行するグローバルインフラストラクチャを保護する責任があります。お客様は、このインフラストラクチャでホストされているコンテンツに対する管理を維持する責任があります。このコンテンツには、 AWS 使用する のセキュリティ設定および管理タスクが含まれます。データプライバシーの詳細については、「データプライバシーのよくある質問」を参照してください。欧州でのデータ保護の詳細については、 AWS セキュリティブログの「Amazon 責任共有モデルと GDPR」ブログ記事を参照してください。

データ保護の目的で、 AWS アカウントの認証情報を保護し、 で個々のアカウントを設定することをお勧めします AWS Identity and Access Management。この方法により、それぞれのジョブを遂行するために必要な許可のみを各ユーザーに付与できます。また、次の方法でデータを保護することをお勧めします。

- 各アカウントで多要素認証 (MFA) を使用します。
- TLS を使用して AWS リソースと通信します。TLS 1.2 が必要です。
- で API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。
- AWS 暗号化ソリューションと、 AWS サービス内のすべてのデフォルトのセキュリティコントロールを使用します。
- Amazon Macie などのアドバンストマネージドセキュリティサービスを使用します。これは、Amazon S3 に保存されている個人データの検出と保護を支援します。
- コマンドラインインターフェースまたは API を使用して AWS にアクセスするときに FIPS 140-2 検証済みの暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-2」 を参照してください。

顧客のアカウント番号などの機密の識別情報は、[Name] (名前)フィールドなどの自由形式のフィールドに配置しないことを強くお勧めします。これは、コンソール、API、、または SDK を使用して Amazon EMR AWS CLIまたは他の AWS のサービスを使用する場合も同様です。 AWS SDKs Amazon EMR や他のサービスに入力したすべてのデータは、診断ログに取り込まれる可能性があります。外部サーバーへの URL を指定するときは、そのサーバーへのリクエストを検証するための認証情報を URL に含めないでください。

保管中と転送中のデータの暗号化

データの暗号化は、承認されていないユーザーがクラスターおよび関連するデータストレージシステムのデータを読み取れないようにするのに役立ちます。このデータには、保管中のデータと呼ばれる、永続的なメディアに保存されているデータや、転送中のデータと呼ばれる、ネットワークを介した転送の間に傍受される可能性のあるデータが含まれます。

Amazon EMR バージョン 4.8.0 から、Amazon EMR セキュリティ設定を使用して、クラスターのデータ暗号化設定をより簡単に行うことができます。セキュリティ設定は、Amazon Elastic Block Store (Amazon EBS) ボリュームおよび Amazon S3 の EMRFS で伝送時のデータと保管時のデータに対するセキュリティを有効にするオプションを提供します。

必要に応じて、Amazon EMR リリースバージョン 4.1.0 以降では、セキュリティ設定を使用して設定されていない、HDFS での透過的暗号化を設定できます。詳細については、「Amazon EMR リリースガイド」の「Transparent encryption in HDFS on Amazon EMR」を参照してください。

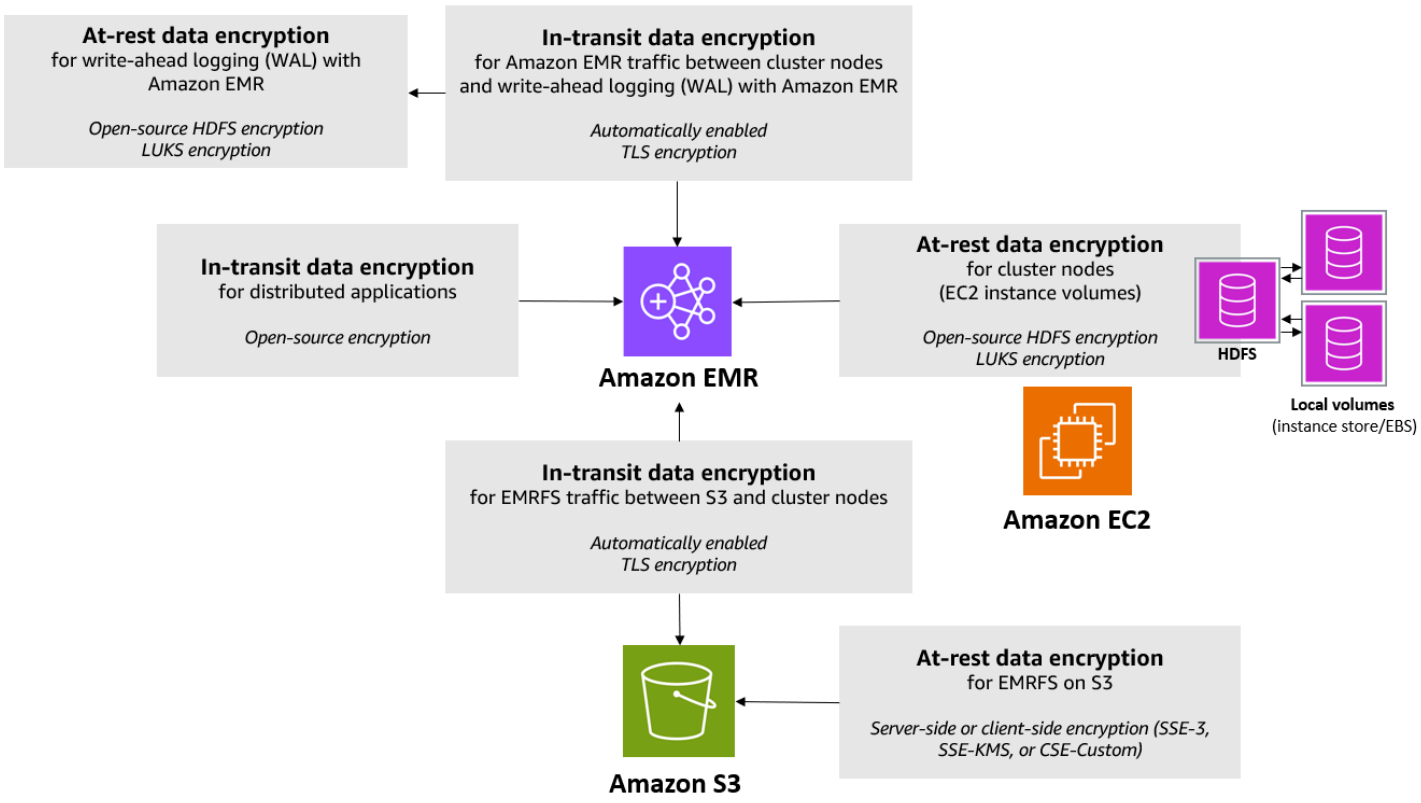
トピック

- 暗号化オプション
- データの暗号化に必要なキーと証明書を作成する

暗号化オプション

Amazon EMR リリース 4.8.0 以降では、セキュリティ設定を使用して、保管中のデータ、転送中のデータ、またはその両方を暗号化するための設定を指定できます。保管中のデータの暗号化を有効にすると、Amazon S3 内にある EMRFS データもしくはローカルディスク内にあるデータのいずれかまたはその両方の暗号化を選択できるようになります。作成する各セキュリティ設定は、クラスター設定ではなく Amazon EMR に保存されるため、設定を簡単に再利用して、クラスターの作成時にいつでもデータ暗号化設定を指定できます。詳細については、「セキュリティ設定を作成する」を参照してください。

次の図は、セキュリティ設定で使用できるさまざまなデータ暗号化オプションを示しています。



以下の暗号オプションも使用できます。また、セキュリティ設定では設定できません。

- Amazon EMR のバージョン 4.1.0 以降では、必要に応じて、HDFS で透過的データ暗号化の設定を選択することもできます。詳細については、「Amazon EMR リリースガイド」の「Transparent encryption in HDFS on Amazon EMR」を参照してください。
- セキュリティ設定をサポートしていない Amazon EMR のリリースバージョンを使用している場合は、手動で Amazon S3 の EMRFS データの暗号化を設定できます。詳細については、「EMRFS プロパティを使用して Amazon S3 の暗号化を指定する」を参照してください。
- 5.24.0 より前の Amazon EMR バージョンを使用している場合、暗号化された EBS ルートデバイスボリュームは、カスタム AMI を使用する場合にのみサポートされます。詳細については、「Amazon EMR 管理ガイド」の「暗号化された Amazon EBS ルートデバイスボリュームを使用したカスタム AMI の作成」を参照してください。

> **Note**
>
> Amazon EMR バージョン 5.24.0 以降では、 をキープロバイダー AWS KMS として指定すると、セキュリティ設定オプションを使用して EBS ルートデバイスとストレージボリュームを暗号化できます。詳細については、「ローカルディスク暗号化」を参照してください。

データの暗号化には、キーと証明書が必要です。セキュリティ設定では、 によって管理されるキー、Amazon S3 によって管理されるキー AWS Key Management Service、提供するカスタムプロバイダーのキーと証明書など、複数のオプションから柔軟に選択できます。をキープロバイダー AWS KMS として使用する場合、暗号化キーの保存と使用には料金が適用されます。詳細については、「AWS KMS 料金表」を参照してください。

暗号化オプションを指定する前に、使用するキーと証明書の管理システムを決定します。これにより、暗号化設定の一部として指定するキーと証明書、またはカスタムプロバイダーをまず作成できます。

Amazon S3 内に保管中の EMRFS データの暗号化

Amazon S3 暗号化は、Amazon S3 との間で読み書きされる Amazon S3 EMR File System (EMRFS) オブジェクトで機能します。保管中のデータの暗号化を有効にする場合は、デフォルトの暗号化モードとして、Amazon S3 サーバー側での暗号化 (SSE) またはクライアント側での暗号化 (CSE) を指定します。オプションで、[Per bucket encryption overrides (バケットごとの暗号化オーバーライド)] を使用して、バケットごとに異なる暗号化方法を指定できます。Amazon S3 の暗号化が有効かどうか

にかかわらず、Transport Layer Security (TLS) は、EMR クラスターノードと Amazon S3 の間で転送される EMRFS オブジェクトを暗号化します。Amazon S3 暗号化の詳細については、「Amazon Simple Storage Service ユーザーガイド」の「暗号化を使用したデータの保護」を参照してください。

> **Note**
>
> を使用する場合 AWS KMS、暗号化キーのストレージと使用に対して料金が適用されます。詳細については、「AWS KMS の料金」を参照してください。

Amazon S3 のサーバー側の暗号化

Amazon S3 のサーバー側の暗号化をセットアップすると、Amazon S3 はデータをディスクに書き込むときにオブジェクトレベルで暗号化し、アクセスするときに復号します。SSE に関する詳細は、「Amazon Simple Storage Service ユーザーガイド」の「サーバー側の暗号化を使用したデータの保護」を参照してください。

Amazon EMR で SSE を指定するときに、次の 2 つの異なるキー管理システムから選択できます。

- SSE-S3 — Amazon S3 がキーを管理します。
- SSE-KMS – を使用して AWS KMS key 、Amazon EMR に適したポリシーをセットアップします。Amazon EMR のキー要件の詳細については、「暗号化に を使用する AWS KMS keys」を参照してください。

SSE とお客様が用意したキーとの組み合わせ (SSE-C) は、Amazon EMR では使用できません。

Amazon S3 クライアント側の暗号化

Amazon S3 のクライアント側の暗号化を使用すると、Amazon S3 の暗号化と復号はクラスターの EMRFS クライアントで行われます。オブジェクトは Amazon S3 にアップロードされる前に暗号化され、ダウンロード後に復号化されます。指定するプロバイダーが、クライアントが使用する暗号化キーを提供します。クライアントは、 AWS KMS によって提供されるキー (CSE-KMS) か、クライアント側のルートキーを提供するカスタム Java クラス (CSE-C) を使用できます。CSE-KMS と CSE-C では、指定するプロバイダーと、復号化または暗号化されるオブジェクトのメタデータに応じて、暗号化の仕様が少し異なります。これらの差異に関する詳細は、「Amazon Simple Storage Service ユーザーガイド」の「クライアント側の暗号化を使用したデータの保護」を参照してください。

> **Note**
>
> Amazon S3 CSE では、Amazon S3 と交換される EMRFS データのみが暗号化されます。クラスターインスタンスボリュームのすべてのデータが暗号化されるわけではありません。さらに、Hue は EMRFS を使用しないため、Hue S3 ファイルブラウザが Amazon S3 に書き込むオブジェクトは暗号化されません。

Amazon EMR WAL のデータの保存時の暗号化

先書きログ (WAL) 用にサーバー側の暗号化 (SSE) を設定すると、Amazon EMR は保管中のデータを暗号化します。Amazon EMR で SSE を指定する場合、2 つの異なるキー管理システムから選択できます。

SSE-EMR-WAL

Amazon EMR がキーを管理します。デフォルトでは、Amazon EMR は Amazon EMR WAL に保存したデータを で暗号化しますSSE-EMR-WAL。

SSE-KMS-WAL

AWS KMS キーを使用して、Amazon EMR WAL に適用されるポリシーを設定します。Amazon EMR の主要な要件の詳細については、「」を参照してください暗号化 AWS KMS keys に を使用する。

Amazon EMR で WAL を有効にする場合、SSE で独自のキーを使用することはできません。詳細については、「Amazon EMR のログ先行書き込み (WAL)」を参照してください。

ローカルディスク暗号化

Amazon EMR セキュリティ設定を使用してローカルディスク暗号化を有効にすると、次のメカニズムが連携してローカルディスクを暗号化します。

オープンソースの HDFS 暗号化

HDFS は、分散処理中にクラスターインスタンス間でデータを交換します。また、インスタンスストアボリュームと、インスタンスにアタッチされた EBS ボリュームとの間でデータを読み書きします。ローカルディスク暗号化を有効にすると、次のオープンソース Hadoop 暗号化オプションがアクティブになります。

- セキュア Hadoop RPC は `Privacy` に設定され、単純認証とセキュリティ層 (SASL、Simple Authentication Security Layer) が使用されます。
- HDFS ブロックデータ転送時のデータ暗号化は `true` に設定され、AES 256 暗号化を使用するように設定されます。

> **Note**
>
> 追加の Apache Hadoop 暗号化をアクティブ化するには、転送時の暗号化を有効にします。詳細については、「転送中の暗号化」を参照してください。これらの暗号化設定では、HDFS 透過的暗号化はアクティブにされず、手動で設定することができます。詳細については、「Amazon EMR リリースガイド」の「Transparent encryption in HDFS on Amazon EMR」を参照してください。

インスタンスストアの暗号化

NVMe ベースの SSD をインスタンスストアボリュームとして使用する EC2 インスタンスタイプでは、Amazon EMR 暗号化設定に関係なく NVMe 暗号化が使用されます。詳細については、「Amazon EC2 ユーザーガイド」のNVMe SSD ボリューム」を参照してください。 Amazon EC2 他のインスタンスストアボリュームの場合、Amazon EMR は、EBS ボリュームが EBS 暗号化と LUKS のどちらを使用して暗号化されているかにかかわらず、ローカルディスクの暗号化が有効になると、LUKS を使用してインスタンスストアボリュームを暗号化します。

EBS ボリュームの暗号化

アカウントで EBS ボリュームの Amazon EC2 暗号化がデフォルトで有効になっているリージョンにクラスターを作成する場合、ローカルディスクの暗号化が有効になっていなくても EBS ボリュームは暗号化されます。詳細については、「Amazon EC2 ユーザーガイド」の「デフォルトで暗号化」を参照してください。セキュリティ設定でローカルディスク暗号化を有効にすると、Amazon EMR 設定がクラスター Amazon EC2 encryption-by-default EC2 設定よりも優先されます。

セキュリティ設定を使用して EBS ボリュームを暗号化するには、以下のオプションを使用できます。

- EBS 暗号化 - Amazon EMR バージョン 5.24.0 以降では、EBS 暗号化を有効にすることを選択できます。EBS 暗号化オプションは、EBS ルートデバイスボリュームとアタッチされたストレージボリュームを暗号化します。EBS 暗号化オプションは、 をキープロバイダー AWS Key

Management Service として指定した場合にのみ使用できます。EBS 暗号化を使用することをお勧めします。

- LUKS 暗号化 - Amazon EBS ボリュームに LUKS 暗号化を使用することを選択した場合、LUKS 暗号化はルートデバイスボリュームではなく、アタッチされたストレージボリュームにのみ適用されます。LUKS の暗号化の詳細については、「LUKS on-disk specification」を参照してください。

  キープロバイダーの場合、Amazon EMR に適したポリシー、または暗号化アーティファクトを提供するカスタム Java クラス AWS KMS key を使用して を設定できます。を使用する場合 AWS KMS、暗号化キーのストレージと使用に対して料金が適用されます。詳細については、「AWS KMS 料金表」を参照してください。

> **Note**
>
> クラスターで EBS 暗号化が有効になっているかどうかを確認するには、`DescribeVolumes` API コールを使用することをお勧めします。詳細については、「」を参照してくださいDescribeVolumes。クラスターで `lsblk` を実行すると、EBS 暗号化ではなく LUKS 暗号化のステータスのみが確認されます。

転送中の暗号化

転送時の暗号化では、複数の暗号化メカニズムが有効になります。これらはオープンソース機能であり、アプリケーション固有のもので、Amazon EMR リリースによって異なる可能性があります。次のアプリケーション固有の暗号化機能は、Apache アプリケーション設定を使用して有効にすることができます。詳細については、「アプリケーションの設定」を参照してください。

Hadoop

- Hadoop MapReduce 暗号化シャッフルは TLS を使用します。
- セキュア Hadoop RPC が「プライバシー」に設定され、SASL (保管時の暗号化が有効な場合に Amazon EMR でアクティブ化) を使用します。
- HDFS ブロックのデータ転送でのデータの暗号化では、AES 256 が使用されます (セキュリティ設定で保管時の暗号化を有効にしたときに Amazon EMR でアクティブ化されます)。
- 詳細については、Apache Hadoop ドキュメントの「Hadoop in secure mode」を参照してください。

HBase

- Kerberos が有効になっている場合、`hbase.rpc.protection` プロパティは暗号化された通信のために `privacy` に設定されます。
- 詳細については、Apache HBase ドキュメントの「Client-side configuration for secure operation」を参照してください。
- Amazon EMR を使用した Kerberos の詳細については、「Amazon EMR での認証に Kerberos を使用する」を参照してください。

Hive

- HiveServer2 (HS2) との JDBC/ODBC クライアント通信は、Amazon EMR リリース 6.9.0 以降の SSL 設定を使用して暗号化されます。
- 詳細については、Apache Hive ドキュメントの「SSL encryption」セクションを参照してください。

Spark

- Spark コンポーネント間 (ブロック転送サービスと外部シャッフルサービスなど) での内部 RPC 通信は、Amazon EMR のバージョン 5.9.0 以降では AES-256 暗号を使用して暗号化されます。以前のリリースでは、内部 RPC 通信は SASL と、暗号として DIGEST-MD5 を使用して暗号化されます。
- Spark 履歴サーバーや HTTPS 対応ファイルサーバーなどのユーザーインターフェイスを使用した HTTP プロトコル通信は、Spark の SSL 設定を使用して暗号化されます。詳細については、Spark ドキュメントの「SSL Configuration」を参照してください。
- 詳細については、Apache Spark ドキュメントの「Spark security settings」セクションを参照してください。

Tez

- Tez Shuffle Handler は TLS を使用します (`tez.runtime.ssl.enable`)。

Presto

- Presto のノード間の内部通信は SSL/TLS を使用します (Amazon EMR バージョン 5.6.0 以降のみ)。

転送時の暗号化で使用する暗号化アーティファクトを指定するには、Amazon S3 にアップロードする証明書の圧縮ファイルを提供するか、暗号化アーティファクトを提供するカスタム Java クラスを参照するかのいずれかを行います。詳細については、「Amazon EMR 暗号化を使用して転送中のデータを暗号化するための証明書の提供」を参照してください。

データの暗号化に必要なキーと証明書を作成する

セキュリティ設定を使用して暗号化オプションを指定する場合は、その前に、キーや暗号化アーティファクトの提供元として使用したいプロバイダーを決定します。例えば、 AWS KMS または作成したカスタムプロバイダーを使用できます。次に、このセクションで説明する方法に沿ってキーまたはキープロバイダーを作成します。

Amazon EMR を使用した保管中のデータを暗号化するためのキーの提供

Amazon EMR の保管時のデータ暗号化には、 AWS Key Management Service （ AWS KMS) またはカスタムキープロバイダーを使用できます。を使用する場合 AWS KMS、暗号化キーのストレージと使用に対して料金が適用されます。詳細については、「AWS KMS 料金表」を参照してください。

このトピックでは、Amazon EMR で使用する KMS キーのキーポリシーの詳細と、Amazon S3 暗号化のカスタムキープロバイダークラスを作成するためのガイドラインおよびコードサンプルを示します。キーの作成の詳細については、「AWS Key Management Service デベロッパーガイド」の「キーの作成」を参照してください。

暗号化 AWS KMS keys に を使用する

AWS KMS 暗号化キーは、Amazon EMR クラスターインスタンスおよび EMRFS で使用される Amazon S3 バケットと同じ リージョンに作成する必要があります。指定するキーが、クラスターの設定に使用するアカウントとは異なるアカウントにある場合は、その ARN を使用してキーを指定する必要があります。

Amazon EC2 インスタンスプロファイルのロールには、指定した KMS キーを使用するためのアクセス許可が必要です。Amazon EMR 内のインスタンスプロファイルのデフォルトのロールは `EMR_EC2_DefaultRole` です。インスタンスプロファイルに別のロールを使用する場合、または Amazon S3 への EMRFS リクエストに IAM ロールを使用する場合は、必要に応じて各ロールがキーユーザーとして追加されていることを確認してください。これにより、KMS キーを使用するアクセス許可がロールに付与されます。詳細については、「AWS Key Management Service デベロッパーガイド」の「キーポリシーの使用」と、「Amazon S3 への EMRFS リクエストの IAM ロールの設定」を参照してください。

を使用して AWS Management Console 、指定した KMS キーのキーユーザーのリストにインスタンスプロファイルまたは EC2 インスタンスプロファイルを追加するか、 AWS CLI または AWS SDK を使用して適切なキーポリシーをアタッチできます。

Amazon EMR は、対称 KMS キーのみをサポートします。非対称 KMS キーを使用して、Amazon EMR クラスター内の保管中のデータを暗号化することはできません。KMS キーが対称か非対称かを判別するには、「対称および非対称 KMS キーを識別する」を参照してください。

以下の手順では、 AWS Management Consoleを使用して、デフォルトの Amazon EMR インスタンスプロファイル `EMR_EC2_DefaultRole` をキーユーザーとして追加する方法について説明します。既に KMS キーが作成されていることを前提としています。新しい KMS キーを作成するには、「AWS Key Management Service デベロッパーガイド」の「キーの作成」を参照してください。

暗号化キーユーザーのリストに Amazon EMR の EC2 インスタンスプロファイルを追加するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/kms で AWS Key Management Service （ AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセレクターを使用します。
3. 変更する KMS キーのエイリアスを選択します。
4. [Key Users] のキーの詳細ページで、[Add] を選択します。
5. [Add key users] ダイアログボックスで、適切なロールを選択します。デフォルトロールの名前は `EMR_EC2_DefaultRole` です。
6. 追加を選択します。

KMS キーに追加のアクセス許可を提供して EBS 暗号化を有効にする

Amazon EMR バージョン 5.24.0 から、セキュリティ設定オプションを使用して EBS ルートデバイスとストレージボリュームを暗号化できます。このようなオプションを有効にするには、キープロバイダー AWS KMS として を指定する必要があります。さらに、 AWS KMS key 指定した を使用するためのアクセス許可`EMR_DefaultRole`をサービスロールに付与する必要があります。

を使用して、指定された KMS キーのキーユーザーのリストにサービスロール AWS Management Console を追加するか、 AWS CLI または AWS SDK を使用して適切なキーポリシーをアタッチできます。

次の手順では、 を使用して、デフォルトの Amazon EMR サービスロールをキーユーザー`EMR_DefaultRole`として AWS Management Console 追加する方法について説明します。既に

KMS キーが作成されていることを前提としています。新しい KMS キーを作成するには、「AWS Key Management Service デベロッパーガイド」の「キーの作成」を参照してください。

Amazon EMR サービスロールを暗号化キーユーザーのリストに追加するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/kms で AWS Key Management Service（ AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセレクターを使用します。
3. 左サイドバーで [Customer managed keys] (カスタマー管理型のキー) を選択します。
4. 変更する KMS キーのエイリアスを選択します。
5. [Key Users] のキーの詳細ページで、[Add] を選択します。
6. 「キーユーザーの追加」セクションで、適切なロールを選択します。Amazon EMR のデフォルトのサービスロールの名前は ですEMR_DefaultRole。
7. 追加を選択します。

カスタムキープロバイダーの作成

セキュリティ設定を使用する場合は、ローカルディスク暗号化と Amazon S3 暗号化用に異なるプロバイダークラス名を指定する必要があります。カスタムキープロバイダーの要件は、ローカルディスク暗号化と Amazon S3 暗号化のどちらを使用するか、および Amazon EMR リリースバージョンによって異なります。

カスタムキープロバイダーの作成時に使用する暗号化のタイプに応じて、アプリケーションは異なる EncryptionMaterialsProvider インターフェイスも実装する必要があります。どちらのインターフェイスも AWS SDK for Java バージョン 1.11.0 以降で使用できます。

- Amazon S3 暗号化を実装するには、com.amazonaws.services.s3.model.EncryptionMaterialsProvider interface を使用します。
- ローカルディスク暗号化を実装するには、com.amazonaws.services.elasticmapreduce.spi.security.EncryptionMaterialsProvider interface を使用します。

実装用の暗号化マテリアルを提供するには、任意の戦略を使用できます。例えば、静的暗号化マテリアルを提供するか、より複雑なキー管理システムと統合するかを選択できます。

Amazon S3 暗号化を使用している場合は、カスタム暗号化マテリアルに暗号化アルゴリズム AES/GCM/NoPadding を使用する必要があります。

ローカルディスク暗号化を使用している場合、カスタム暗号化マテリアルに使用する暗号化アルゴリズムは EMR リリースによって異なります。Amazon EMR 7.0.0 以前では、AES/GCM/NoPadding を使用する必要があります。Amazon EMR 7.1.0 以降では、AES を使用する必要があります。

EncryptionMaterialsProvider クラスは、暗号化コンテキストによって暗号化マテリアルを取得します。Amazon EMR は、呼び出し元が返す正しい暗号化マテリアルを判別しやすいように、実行時に暗号化コンテキスト情報を設定します。

Example 例: EMRFS での Amazon S3 の暗号化にカスタムキープロバイダを使用する

Amazon EMR がEncryptionMaterialsProvider クラスから暗号化マテリアルを取得して暗号化を実行すると、EMRFS はオプションで materialsDescription 引数に、オブジェクトの Amazon S3 URI とクラスター JobFlowId の の 2 つのフィールドを入力します。これは、EncryptionMaterialsProvider クラスが暗号化マテリアルを選択的に返すために使用できます。

たとえば、プロバイダは Amazon S3 URI プレフィックスごとに異なるキーを返すことができます。最終的に Amazon S3 オブジェクトに保存されるのは、EMRFS によって生成され、プロバイダに渡される materialsDescription 値ではなく、返された暗号化マテリアルの記述です。Amazon S3 オブジェクトの復号中に、暗号化マテリアルの説明が EncryptionMaterialsProvider クラスに渡されるため、オブジェクトを復号するために一致するキーを選択的に返すことができます。

EncryptionMaterialsProvider リファレンス実装を以下に示します。別のカスタムプロバイダーである EMRFSRSA EncryptionMaterialsProviderは、 から入手できます GitHub。

```
import com.amazonaws.services.s3.model.EncryptionMaterials;
import com.amazonaws.services.s3.model.EncryptionMaterialsProvider;
import com.amazonaws.services.s3.model.KMSEncryptionMaterials;
import org.apache.hadoop.conf.Configurable;
import org.apache.hadoop.conf.Configuration;

import java.util.Map;

/**
 * Provides KMSEncryptionMaterials according to Configuration
 */
public class MyEncryptionMaterialsProviders implements EncryptionMaterialsProvider,
 Configurable{
  private Configuration conf;
  private String kmsKeyId;
  private EncryptionMaterials encryptionMaterials;

  private void init() {
```

```
    this.kmsKeyId = conf.get("my.kms.key.id");
    this.encryptionMaterials = new KMSEncryptionMaterials(kmsKeyId);
  }

  @Override
  public void setConf(Configuration conf) {
    this.conf = conf;
    init();
  }

  @Override
  public Configuration getConf() {
    return this.conf;
  }

  @Override
  public void refresh() {

  }

  @Override
  public EncryptionMaterials getEncryptionMaterials(Map<String, String>
 materialsDescription) {
    return this.encryptionMaterials;
  }

  @Override
  public EncryptionMaterials getEncryptionMaterials() {
    return this.encryptionMaterials;
  }
}
```

Amazon EMR 暗号化を使用して転送中のデータを暗号化するための証明書の提供

Amazon EMR リリースバージョン 4.8.0 以降では、下記の 2 つのオプションのいずれかにより、セキュリティ設定を使用して転送中のデータを暗号化するのに必要なアーティファクトを指定できます。

- 手動で PEM 証明書を作成し、zip ファイルに含め、Amazon S3 から zip ファイルを参照できます。
- Java クラスとしてカスタム証明書プロバイダーを実装できます。Amazon S3 でアプリケーションの JAR ファイルを指定し、アプリケーションで宣言したプロバイダーの完全なクラス名を提供

します。クラスは、 AWS SDK for Java バージョン 1.11.0 以降で利用可能な TLSArtifactsProvider インターフェイスを実装する必要があります。

Amazon EMR はクラスターの各ノードに自動的にアーティファクトをダウンロードし、その後、それらのアーティファクトを使用してオープンソースの伝送中の暗号化機能を実装します。使用できるオプションの詳細については、転送中の暗号化を参照してください。

PEM 証明書の使用

zip ファイルを指定して転送中のデータを暗号化する場合、セキュリティ設定は、zip ファイル内の PEM ファイルに下記の名前が正確に付されていることを要求します。

伝送中の暗号化証明書

ファイル名	必須/オプション	詳細
privateKey.pem	必須	プライベートキー
certificateChain.pem	必須	証明書チェーン
trustedCertificates.pem	オプションです。	提供された証明書が、Java のデフォルトの信頼されたルート証明機関 (CA) によって証明されていない場合、または Java のデフォルトの信頼されたルート CA にリンクできる中間 CA によって署名されていない場合は必須です。Java のデフォルトの信頼されたルート CA は、`jre/lib/security/cacerts` にあります。

プライベートキー PEM ファイルは、クラスターインスタンスが存在する Amazon VPC ドメインへのアクセスを有効にするワイルドカード証明書として設定するようにします。たとえば、クラスターが us-east-1 (N. Virginia) に存在する場合、証明書件名定義で `CN=*.ec2.internal` を指定して、クラスターへのアクセスが可能になるよう、証明書設定で共通の名前を指定することができます。ク

ラスターが us-west-2 (オレゴン) に存在する場合、CN=*.us-west-2.compute.internal を指定できます。

暗号化アーティファクト内の提供された PEM ファイルのドメインの CN にワイルドカード文字が含まれていない場合は、hadoop.ssl.hostname.verifier の値を ALLOW_ALL に変更する必要があります。これを行うには、クラスターに設定を送信するときに core-site 分類を使用するか、core-site.xml ファイルでこの値を追加します。デフォルトのホスト名検証でワイルドカードなしのホスト名が受け入れられず、エラーになるため、この変更が必要です。Amazon VPC 内の EMR クラスター設定の詳細については、「ネットワークを設定する」を参照してください。

次の例は、OpenSSL を使用して、1024-bit RSA プライベートキーの自己署名 X.509 証明書を生成する方法を示しています。キーにより、共通名として *.us-west-2.compute.internal ドメイン名で指定された us-west-2 (オレゴン) リージョンにある発行者の Amazon EMR クラスターインスタンスへのアクセスが可能になります。

国 (C)、州 (S) および口ケール (L) といった他のオプション項目も指定されます。自己署名証明書が生成されるため、例の 2 番目のコマンドでは、certificateChain.pem ファイルを trustedCertificates.pem ファイルにコピーします。3 番目のコマンドでは、zip を使って証明書を含む my-certs.zip ファイルを作成します。

> **Important**
>
> この例は proof-of-concept デモンストレーションのみです。自己署名証明書の使用は推奨されておらず、セキュリティリスクが生じる可能性があります。本番システムでは、証明書の発行で信頼できる認証機関 (CA) を使用してください。

```
$ openssl req -x509 -newkey rsa:1024 -keyout privateKey.pem -out certificateChain.pem
 -days 365 -nodes -subj '/C=US/ST=Washington/L=Seattle/O=MyOrg/OU=MyDept/CN=*.us-
west-2.compute.internal'
$ cp certificateChain.pem trustedCertificates.pem
$ zip -r -X my-certs.zip certificateChain.pem privateKey.pem trustedCertificates.pem
```

AWS Identity and Access Management Amazon EMR 用の

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 Amazon

EMR リソースの使用を承認する (アクセス許可を付与する) かを制御します。IAM は、追加料金なしで AWS のサービス 使用できる です。

トピック

対象者

AWS Identity and Access Management (IAM) の使用方法は、Amazon EMR で行う作業によって異なります。

サービスユーザー – ジョブを実行するために Amazon EMR サービスを使用する場合は、管理者から必要なアクセス許可と認証情報が与えられます。さらに多くの Amazon EMR 機能を使用して作業を行う場合は、追加のアクセス許可が必要になることがあります。アクセスの管理方法を理解しておくと、管理者に適切な許可をリクエストするうえで役立ちます。Amazon EMR の機能にアクセスできない場合は、「Amazon EMR の ID とアクセスのトラブルシューティング」を参照してください。

サービス管理者 – 社内の Amazon EMR リソースを担当している場合は、通常、Amazon EMR へのフルアクセスがあります。サービスユーザーがアクセスする必要がある Amazon EMR 機能やリソースを決定するのは、このユーザーの仕事です。その後、IAM 管理者にリクエストを送信して、サービスユーザーの権限を変更する必要があります。このページの情報を点検して、IAM の基本概念を理解してください。会社で Amazon EMR と IAM を併用する方法の詳細については、「Amazon EMR で IAM が機能する仕組み」を参照してください。

IAM 管理者 – 管理者は、Amazon EMR へのアクセスを管理するポリシーの作成方法の詳細について確認する場合があります。IAM で使用できる Amazon EMR 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、「Amazon EMR のアイデンティティベースポリシーの例」を参照してください。

アイデンティティを使用した認証

認証とは、ID 認証情報 AWS を使用して にサインインする方法です。として、IAM ユーザーとして AWS アカウントのルートユーザー、または IAM ロールを引き受けて認証 (にサインイン AWS) される必要があります。

ID ソースを介して提供された認証情報を使用して、フェデレーティッド ID AWS として にサインインできます。 AWS IAM Identity Center （ IAM Identity Center) ユーザー、会社のシングルサインオン認証、Google または Facebook の認証情報は、フェデレーション ID の例です。フェデレーティッド ID としてサインインする場合、IAM ロールを使用して、前もって管理者により ID フェデレーションが設定されています。フェデレーション AWS を使用して にアクセスすると、間接的にロールを引き受けることになります。

ユーザーのタイプに応じて、 AWS Management Console または AWS アクセスポータルにサインインできます。へのサインインの詳細については AWS、「 ユーザーガイド」の「 へのサインイン AWS アカウント方法AWS サインイン 」を参照してください。

AWS プログラムで にアクセスする場合、 は Software Development Kit (SDK) とコマンドラインインターフェイス (CLI) AWS を提供し、認証情報を使用してリクエストに暗号で署名します。 AWS ツールを使用しない場合は、リクエストに自分で署名する必要があります。推奨される方法を使用してリクエストを自分で署名する方法の詳細については、IAM ユーザーガイドの API AWS リクエストの署名を参照してください。

使用する認証方法を問わず、追加セキュリティ情報の提供をリクエストされる場合もあります。例えば、 AWS では、多要素認証 (MFA) を使用してアカウントのセキュリティを向上させることをお勧めします。詳細については、「AWS IAM Identity Center ユーザーガイド」の「Multi-factor authentication」(多要素認証) および「IAM ユーザーガイド」の「AWSでの多要素認証 (MFA) の使用」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、アカウント内のすべての およびリソースへの AWS のサービス 完全なアクセス権を持つ 1 つのサインインアイデンティティから始めます。この ID は AWS アカウント ルートユーザーと呼ばれ、アカウントの作成に使用した E メールアドレスとパスワードでサインインすることでアクセスできます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザーの認証情報は保護し、ルートユーザーでしか実行できないタスクを実行するときに使用します。ルートユーザーとしてサインインする必要があるタスクの完全なリストについては、IAM ユーザーガイドのルートユーザー認証情報が必要なタスクを参照してください。

フェデレーティッドアイデンティティ

ベストプラクティスとして、管理者アクセスを必要とするユーザーを含む人間のユーザーに、一時的な認証情報を使用して にアクセスするための ID プロバイダーとのフェデレーションの使用を要求 AWS のサービス します。

フェデレーティッド ID は、エンタープライズユーザーディレクトリ、ウェブ ID プロバイダー、AWS Directory Service、アイデンティティセンターディレクトリ、または ID ソースを通じて提供された認証情報 AWS のサービス を使用して にアクセスするユーザーです。フェデレーティッド ID がにアクセスすると AWS アカウント、ロールを引き受け、ロールは一時的な認証情報を提供します。

アクセスを一元管理する場合は、 AWS IAM Identity Centerを使用することをお勧めします。IAM Identity Center でユーザーとグループを作成することも、独自の ID ソース内のユーザーとグループのセットに接続して同期して、すべての AWS アカウント とアプリケーションで使用することもできます。IAM Identity Center の詳細については、「AWS IAM Identity Center ユーザーガイド」の「What is IAM Identity Center?」(IAM Identity Center 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、単一のユーザーまたはアプリケーションに対して特定のアクセス許可 AWS アカウント を持つ 内のアイデンティティです。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を保有する IAM ユーザーを作成する代わりに、一時的な認証情報を使用することをお勧めします。ただし、IAM ユーザーでの長期的な認証情報が必要な特定のユースケースがある場合は、アクセスキーをローテーションすることをお勧めします。詳細については、IAM ユーザーガイドの長期的な認証情報を必要とするユースケースのためにアクセスキーを定期的にローテーションするを参照してください。

IAM グループは、IAM ユーザーの集団を指定するアイデンティティです。グループとしてサインインすることはできません。グループを使用して、複数のユーザーに対して一度に権限を指定できます。多数のユーザーグループがある場合、グループを使用することで権限の管理が容易になります。例えば、IAMAdminsという名前のグループを設定して、そのグループにIAM リソースを管理する許可を与えることができます。

ユーザーは、ロールとは異なります。ユーザーは 1 人の人または 1 つのアプリケーションに一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユーザーには永続的な長期の認証情報がありますが、ロールでは一時的な認証情報が提供されます。詳細については、「IAM ユーザーガイド」の「IAM ユーザー (ロールではなく) の作成が適している場合」を参照してください。

IAM ロール

IAM ロールは、特定のアクセス許可 AWS アカウント を持つ 内のアイデンティティです。これは IAM ユーザーに似ていますが、特定のユーザーには関連付けられていません。ロール を切り替える AWS Management Console ことで、 で IAM ロールを一時的に引き受けることができます。ロールを引き受けるには、 または AWS API AWS CLI オペレーションを呼び出すか、カスタム URL を使用します。ロールを使用する方法の詳細については、「IAM ユーザーガイド」の「IAM ロールの使用」を参照してください。

IAM ロールと一時的な認証情報は、次の状況で役立ちます:

- フェデレーションユーザーアクセス – フェデレーティッド ID に許可を割り当てるには、ロールを作成してそのロールの許可を定義します。フェデレーティッド ID が認証されると、その ID はロールに関連付けられ、ロールで定義されている許可が付与されます。フェデレーションの詳細については、「IAM ユーザーガイド」の「Creating a role for a third-party Identity Provider」(サードパーティーアイデンティティプロバイダー向けロールの作成) を参照してください。IAM Identity Center を使用する場合は、許可セットを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM Identity Center は、権限セットを IAM のロールに関連付けます。アクセス許可セットの詳細については、「AWS IAM Identity Center ユーザーガイド」の「アクセス許可セット」を参照してください。
- 一時的な IAM ユーザー権限 - IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる権限を一時的に IAM ロールで引き受けることができます。
- クロスアカウントアクセス - IAM ロールを使用して、自分のアカウントのリソースにアクセスすることを、別のアカウントの人物 (信頼済みプリンシパル) に許可できます。クロスアカウントアクセス権を付与する主な方法は、ロールを使用することです。ただし、一部の では AWS のサービス、(ロールをプロキシとして使用する代わりに) ポリシーをリソースに直接アタッチできます。クロスアカウントアクセスにおけるロールとリソースベースのポリシーの違いについては、「IAM ユーザーガイド」の「IAM 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。
- クロスサービスアクセス — 一部の は、他の の機能 AWS のサービス を使用します AWS のサービス。例えば、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。サービスでは、呼び出し元プリンシパルの許可、サービスロール、またはサービスリンクロールを使用してこれを行う場合があります。
  - 転送アクセスセッション (FAS) – IAM ユーザーまたはロールを使用して でアクションを実行する場合 AWS、ユーザーはプリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがありま

す。FAS は、 を呼び出すプリンシパルのアクセス許可を AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストのリクエストと組み合わせて使用します。FAS リクエストは、サービスが他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。

- サービスロール - サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、「IAM ユーザーガイド」の「AWS のサービスに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。
- サービスにリンクされたロール – 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。

- Amazon EC2 で実行されているアプリケーション – IAM ロールを使用して、EC2 インスタンスで実行され、 AWS CLI または AWS API リクエストを行うアプリケーションの一時的な認証情報を管理できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。 AWS ロールを EC2 インスタンスに割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスにアタッチされたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、IAM ユーザーガイドのAmazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用して許可を付与するを参照してください。

IAM ロールと IAM ユーザーのどちらを使用するかについては、IAM ユーザーガイドの(IAM ユーザーではなく) IAM ロールをいつ作成したら良いのか?を参照してください。

ポリシーを使用したアクセスの管理

でアクセスを制御する AWS には、ポリシーを作成し、 AWS ID またはリソースにアタッチします。ポリシーは AWS 、アイデンティティまたはリソースに関連付けられているときにアクセス許可を定義する のオブジェクトです。 は、プリンシパル (ユーザー、ルートユーザー、またはロールセッション) がリクエストを行うときに、これらのポリシー AWS を評価します。ポリシーでの権限により、リクエストが許可されるか拒否されるかが決まります。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構造と内容の詳細については、IAM ユーザーガイドのJSON ポリシー概要を参照してください。

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

デフォルトでは、ユーザーやロールに権限はありません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き継ぐことができます。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションの許可を定義します。例えば、`iam:GetRole`アクションを許可するポリシーがあるとします。そのポリシーを持つユーザーは、 AWS Management Console、、 AWS CLIまたは AWS API からロール情報を取得できます。

アイデンティティベースのポリシー

アイデンティティベースポリシーは、IAM ユーザーグループ、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 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のポリシーを作成する方法については、IAM ユーザーガイドのIAM ポリシーの作成を参照してください。

アイデンティティベースのポリシーは、さらにインラインポリシーまたはマネージドポリシーに分類できます。インラインポリシーは、単一のユーザー、グループ、またはロールに直接埋め込まれています。管理ポリシーは、 内の複数のユーザー、グループ、ロールにアタッチできるスタンドアロンポリシーです AWS アカウント。管理ポリシーには、 AWS 管理ポリシーとカスタマー管理ポリシーが含まれます。マネージドポリシーまたはインラインポリシーのいずれかを選択する方法については、IAM ユーザーガイドのマネージドポリシーとインラインポリシーの比較を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシー や Amazon S3 バケットポリシー があげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

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。

アクセスコントロールリスト (ACL)

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための許可を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Amazon S3、 AWS WAF、および Amazon VPC は、ACLs。ACL の詳細については、Amazon Simple Storage Service デベロッパーガイドのアクセスコントロールリスト (ACL) の概要を参照してください。

その他のポリシータイプ

AWS は、一般的ではない追加のポリシータイプをサポートします。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の権限を設定できます。

- アクセス許可の境界 - アクセス許可の境界は、アイデンティティベースのポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる権限の上限を設定する高度な機能です。エンティティにアクセス許可の境界を設定できます。結果として得られる権限は、エンティティのアイデンティティベースポリシーとそのアクセス許可の境界の共通部分になります。`Principal` フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、アクセス許可の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。アクセス許可の境界の詳細については、IAM ユーザーガイドのIAM エンティティのアクセス許可の境界を参照してください。
- サービスコントロールポリシー (SCPs) – SCPs は、 の組織または組織単位 (OU) に対する最大アクセス許可を指定する JSON ポリシーです AWS Organizations。 AWS Organizations は、 AWS アカウント ビジネスが所有する複数の をグループ化して一元管理するサービスです。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP は、各 を含むメンバーアカウントのエンティティのアクセス許可を制限します AWS アカウントのルートユーザー。Organizations と SCP の詳細については、AWS Organizations ユーザーガイドの「SCP の仕組み」を参照してください。
- セッションポリシー - セッションポリシーは、ロールまたはフェデレーションユーザーの一時的なセッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。結果としてセッションの権限は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポリシーの共通部分になります。また、リソースベースのポリシーから権限が派生する場合もあります。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。詳細については、IAM ユーザーガイドのセッションポリシーを参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成される権限を理解するのがさらに難しくなります。複数のポリシータイプが関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。

Amazon EMR で IAM が機能する仕組み

IAM を使用して Amazon EMR へのアクセスを管理する前に、Amazon EMR で使用できる IAM 機能について理解しておく必要があります。

Amazon EMR で使用できる IAM の機能

IAM 機能	Amazon EMR 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	あり
ポリシーアクション	あり
ポリシーリソース	Yes
ポリシー条件キー	Yes
ACL	なし
ABAC (ポリシー内のタグ)	はい
一時的な認証情報	あり
プリンシパル権限	あり
サービスロール	いいえ
サービスリンクロール	あり

Amazon EMR およびその他の AWS のサービスがほとんどの IAM 機能と連携する方法の概要を把握するには、「IAM ユーザーガイド」のAWS 「IAM と連携する のサービス」を参照してください。

Amazon EMR の ID ベースのポリシー

アイデンティティベースポリシーをサポートする	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のポリシーを作成する方法については、IAM ユーザーガイドのIAM ポリシーの作成を参照してください。

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。プリンシパルは、それが添付されているユーザーまたはロールに適用されるため、アイデンティティベースのポリシーでは指定できません。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。

Amazon EMR の ID ベースのポリシーの例

Amazon EMR のアイデンティティベースポリシーの例を確認するには、「Amazon EMR のアイデンティティベースポリシーの例」を参照してください。

Amazon EMR 内のリソースベースのポリシー

リソースベースのポリシーのサポート	あり

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシー や Amazon S3 バケットポリシー があげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

クロスアカウントアクセスを有効にするには、アカウント全体、または別のアカウントの IAM エンティティをリソースベースのポリシーのプリンシパルとして指定します。リソースベースのポリシーにクロスアカウントのプリンシパルを追加しても、信頼関係は半分しか確立されない点に注意してください。プリンシパルとリソースが異なる にある場合 AWS アカウント、信頼されたアカウントの IAM 管理者は、プリンシパルエンティティ (ユーザーまたはロール) にリソースへのアクセス許可も付与する必要があります。IAM 管理者は、アイデンティティベースのポリシーをエンティティにアタッチすることで権限を付与します。ただし、リソースベースのポリシーで、同じアカウントのプリンシパルへのアクセス権が付与されている場合は、アイデンティティベースのポリシーをさらに付与する必要はありません。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。

Amazon EMR のポリシーアクション

ポリシーアクションに対するサポート	あり

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

JSON ポリシーの`Action`要素には、ポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。ポリシーアクションの名前は通常、関連付けられた AWS API オペレーションと同じです。一致する API オペレーションのない許可のみのアクションなど、いくつかの例外があります。また、ポリシーに複数のアクションが必要なオペレーションもあります。これらの追加アクションは、依存アクションと呼ばれます。

このアクションは、関連付けられたオペレーションを実行するための権限を付与するポリシーで使用されます。

Amazon EMR のアクションのリストについては、「サービス認証リファレンス」の「Amazon EMR 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。

Amazon EMR 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を使用します。

```
EMR
```

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": [
      "EMR:action1",
```

```
      "EMR:action2"
         ]
```

Amazon EMR のアイデンティティベースポリシーの例を確認するには、「Amazon EMR のアイデンティティベースポリシーの例」を参照してください。

Amazon EMR のポリシーリソース

ポリシーリソースに対するサポート	あり

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースにどのような条件でアクションを実行できるかということです。

`Resource` JSON ポリシー要素は、アクションが適用されるオブジェクトを指定します。ステートメントには、`Resource` または `NotResource` 要素を含める必要があります。ベストプラクティスとして、Amazon リソースネーム (ARN) を使用してリソースを指定します。これは、リソースレベルの許可と呼ばれる特定のリソースタイプをサポートするアクションに対して実行できます。

オペレーションのリスト化など、リソースレベルの権限をサポートしないアクションの場合は、ステートメントがすべてのリソースに適用されることを示すために、ワイルドカード (*) を使用します。

```
"Resource": "*"
```

Amazon EMR リソースのタイプとその ARN のリストについては、「サービス認証リファレンス」の「Resources Defined by Amazon EMR」を参照してください。各リソースの ARN を指定できるアクションについては、「Actions, resources, and condition keys for Amazon EMR」を参照してください。

Amazon EMR のアイデンティティベースポリシーの例を確認するには、「Amazon EMR のアイデンティティベースポリシーの例」を参照してください。

Amazon EMR 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 AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

Condition 要素 (または Condition ブロック) を使用すると、ステートメントが有効な条件を指定できます。Condition 要素はオプションです。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することで、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。

1 つのステートメントに複数の Condition 要素を指定する場合、または 1 つの Condition 要素に複数のキーを指定する場合、 AWS では AND 論理演算子を使用してそれらを評価します。1 つの条件キーに複数の値を指定すると、 は論理ORオペレーションを使用して条件 AWS を評価します。ステートメントの権限が付与される前にすべての条件が満たされる必要があります。

条件を指定する際にプレースホルダー変数も使用できます。例えば IAM ユーザーに、IAM ユーザー名がタグ付けされている場合のみリソースにアクセスできる権限を付与することができます。詳細については、IAM ユーザーガイドのIAM ポリシーの要素: 変数およびタグを参照してください。

AWS は、グローバル条件キーとサービス固有の条件キーをサポートします。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。

Amazon EMR の条件キーのリスト、および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、「サービス認証リファレンス」の「Actions, resources, and condition keys for Amazon EMR」を参照してください。

Amazon EMR のアイデンティティベースポリシーの例を確認するには、「Amazon EMR のアイデンティティベースポリシーの例」を参照してください。

Amazon EMR のアクセスコントロールリスト (ACL)

ACL のサポート	なし

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための許可を持つかをコントロールします。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Amazon EMR での属性ベースのアクセスコントロール (ABAC)

ABAC のサポート (ポリシー内のタグ)	はい

属性ベースのアクセス制御 (ABAC) は、属性に基づいてアクセス許可を定義するアクセス許可戦略です。では AWS、これらの属性はタグ と呼ばれます。タグは、IAM エンティティ (ユーザーまたはロール) および多くの AWS リソースにアタッチできます。エンティティとリソースのタグ付けは、ABAC の最初の手順です。その後、プリンシパルのタグがアクセスしようとしているリソースのタグと一致した場合にオペレーションを許可するように ABAC ポリシーをします。

ABAC は、急成長する環境やポリシー管理が煩雑になる状況で役立ちます。

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 とは?を参照してください。ABAC をセットアップするステップ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に基づく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。

Amazon EMR での一時認証情報の使用

一時的な認証情報のサポート	あり

一部の は、一時的な認証情報を使用してサインインすると機能 AWS のサービス しません。一時的な認証情報 AWS のサービス を使用する などの詳細については、IAM ユーザーガイドのAWS のサービス「IAM と連携する 」を参照してください。

ユーザー名とパスワード以外の AWS Management Console 方法で にサインインする場合、一時的な認証情報を使用します。例えば、会社の Single Sign-On (SSO) リンク AWS を使用して にアクセスすると、そのプロセスによって一時的な認証情報が自動的に作成されます。また、ユーザーとしてコンソールにサインインしてからロールを切り替える場合も、一時的な認証情報が自動的に作成されます。ロールの切り替えに関する詳細については、IAM ユーザーガイドのロールへの切り替え (コンソール)を参照してください。

一時的な認証情報は、 AWS CLI または AWS API を使用して手動で作成できます。その後、これらの一時的な認証情報を使用して . AWS recommends にアクセスできます AWS。これは、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することを推奨しています。詳細については、IAM の一時的セキュリティ認証情報を参照してください。

Amazon EMR のクロスサービスプリンシパルのアクセス許可

フォワードアクセスセッション (FAS) をサポート	あり

IAM ユーザーまたはロールを使用して でアクションを実行すると AWS、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、 を呼び出すプリンシパルのアクセス許可を AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストのリクエストと組み合わせて使用します。FAS リクエストは、サービスが他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。

Amazon EMR のサービスロール

サービスロールのサポート	なし

Amazon EMR のサービスリンクロール

サービスリンクロールのサポート	あり

サービスリンク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。表の中から、[Service-linked role] (サービスにリンクされたロール) 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、はい リンクを選択します。

IAM ポリシーを持つクラスターとノートブックのタグをアクセスコントロールに使用する

EMR Notebooks および EMR クラスターに関連した Amazon EMR アクションに対する許可は、アイデンティティベースの IAM ポリシーを用いたタグベースのアクセスコントロールを使用して調整できます。ノートブック、クラスター、またはその両方に特定のタグキーまたはキーと値の組み合わせがある場合にのみ特定のアクションを許可するには、`Condition` 要素 (`Condition` ブロックとも呼ばれる) 内で条件キーを使用できます。`CreateEditor` アクション (EMR Notebooks が作成され

る)、RunJobFlow アクション (クラスターが作成される) を制限することもできます。これにより、タグに対するリクエストが、リソースの作成時に送信されなければならないようにすることができます。

Amazon EMR では、Condition 要素で使用できる条件キーは、リクエストパラメータとして ClusterID または NotebookID を必要とする Amazon EMR API アクションにのみ適用されます。例えば、 ModifyInstanceGroupsアクションは、 ClusterIDがオプションのパラメータであるため、コンテキストキーをサポートしていません。

EMR ノートブックを作成すると、デフォルトのタグが適用されます。このタグのキー文字列は creatorUserId で、ノートブックを作成した IAM ユーザー ID の値に設定されています。ノートブックに対して許可されるアクションを作成者のみに制限するのに便利です。

Amazon EMR で次の条件キーを使用できます。

- elasticmapreduce:ResourceTag/*TagKeyString* 条件コンテキストキーを使用して、指定した *TagKeyString* のあるタグを持つクラスターまたはノートブックのユーザーアクションを許可または拒否します。ClusterID と NotebookID の両方を渡す場合、条件がクラスターとノートブッククラスターの両方に適用されます。つまり、両方のリソースに、タグキー文字または指定するキーと値の組み合わせが必要です。Resource 要素を使用してステートメントを制限できるため、必要に応じてクラスターまたはノートブックにのみ適用されます。詳細については、「Amazon EMR のアイデンティティベースポリシーの例」を参照してください。
- アクション/API コールを持つ特定のタグを必要としている elasticmapreduce:RequestTag/*TagKeyString* 条件コンテキストキーを使用します。たとえば、この条件コンテキストキーとともに CreateEditor アクションを使用して、*TagKeyString* を持つキーがノートブック作成時に適用されるよう要求することができます。

例

Amazon EMR アクションのリストを確認するには、「IAM ユーザーガイド」の「Amazon EMR で定義されるアクション」を参照してください。

Amazon EMR ステップのランタイムロール

ランタイムロールは、Amazon EMR クラスターにジョブまたはクエリを送信するときに指定できる AWS Identity and Access Management (IAM) ロールです。Amazon EMR クラスターに送信するジョブまたはクエリは、ランタイムロールを使用して Amazon S3 のオブジェクトなどの AWS リソース

にアクセスします。Amazon EMR では、Spark ジョブと Hive ジョブ用のランタイムロールを指定できます。

Amazon SageMaker で Amazon EMR クラスターに接続するとき、および Amazon EMR Studio ワークスペースを EMR クラスターにアタッチするときに、ランタイムロールを指定することもできます。詳細については、「Studio から Amazon EMR クラスターに接続する」および「ランタイムロールを使用して EMR Studio Workspace を実行する」を参照してください。

以前は、Amazon EMR クラスターは、クラスターの起動に使用したインスタンスプロファイルにアタッチされた IAM ポリシーに基づくアクセス許可で、Amazon EMR ジョブまたはクエリを実行していました。つまり、ポリシーには、Amazon EMR クラスターで実行されるすべてのジョブとクエリに対するすべてのアクセス許可の統合を含める必要がありました。ランタイムロールを使用すると、クラスターの Amazon EMR インスタンスプロファイルを共有する代わりに、各ジョブまたはクエリのアクセスコントロールを個別に管理できるようになりました。

ランタイムロールを持つ Amazon EMR クラスターでは、データレイクに対して Spark、Hive、Presto のジョブとクエリに AWS Lake Formation ベースのアクセスコントロールを適用することもできます。との統合方法の詳細については、 AWS Lake Formation「」を参照してくださいAmazon EMR を と統合する AWS Lake Formation。

> **Note**
>
> Amazon EMR ステップのランタイムロールを指定すると、送信するジョブまたはクエリは、ランタイムロールにアタッチされたポリシーで許可されている AWS リソースにのみアクセスできます。これらのジョブやクエリは、クラスターの EC2 インスタンス上のインスタンスメタデータサービスにアクセスしたり、クラスターの EC2 インスタンスプロファイルを使用して AWS リソースにアクセスしたりすることはできません。

ランタイムロールで Amazon EMR クラスターを起動するための前提条件

トピック

- ステップ 1: Amazon EMR でセキュリティ設定を設定する
- ステップ 2: Amazon EMR クラスターの EC2 インスタンスプロファイルを設定する
- ステップ 3: 信頼ポリシーを設定する

ステップ 1: Amazon EMR でセキュリティ設定を設定する

次の JSON 構造を使用して、 AWS Command Line Interface （ AWS CLI) でセキュリティ設定を作成し、 EnableApplicationScopedIAMRoleを に設定しますtrue。セキュリティ設定の詳細については、「セキュリティ設定を使用してクラスターセキュリティをセットアップする」を参照してください。

```
{
    "AuthorizationConfiguration":{
        "IAMConfiguration":{
            "EnableApplicationScopedIAMRole":true
        }
    }
}
```

インターネット経由で転送されるデータがプレーンテキストではなく暗号化されるように、セキュリティ設定では転送時の暗号化オプションを常に有効にすることが推奨されます。Runtime Studio または EMR Studio のランタイムロールを使用して Amazon EMR SageMaker クラスターに接続しない場合は、これらのオプションをスキップできます。データ暗号化を設定するには、「データ暗号化の設定」を参照してください。

または、AWS Management Consoleを使用して、カスタム設定でセキュリティ設定を作成することもできます。

ステップ 2: Amazon EMR クラスターの EC2 インスタンスプロファイルを設定する

Amazon EMR クラスターは、Amazon EC2 インスタンスプロファイルロールを使用してランタイムロールを引き受けます。Amazon EMR ステップでランタイムロールを使用するには、インスタンスプロファイルロールとして使用する予定の IAM ロールに次のポリシーを追加します。IAM ロールにポリシーを追加する、または既存のインラインポリシーや管理ポリシーを編集するには、「IAM ID のアクセス許可の追加および削除」を参照してください。

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"AllowRuntimeRoleUsage",
            "Effect":"Allow",
            "Action":[
                "sts:AssumeRole",
```

```
                "sts:TagSession"
            ],
            "Resource":[
                <runtime-role-ARN>
            ]
        }
    ]
}
```

ステップ 3: 信頼ポリシーを設定する

ランタイムロールとして使用する予定の各 IAM ロールについて、以下の信頼ポリシーを設定します。EMR_EC2_DefaultRole は、ご使用のインスタンスプロファイルロールに置き換えてください。IAM ロールの信頼ポリシーを変更するには、「ロールの信頼ポリシーの変更」を参照してください。

```
{
    "Sid":"AllowAssumeRole",
    "Effect":"Allow",
    "Principal":{
        "AWS":"arn:aws:iam::<AWS_ACCOUNT_ID>:role/EMR_EC2_DefaultRole"
    },
    "Action":"sts:AssumeRole"
}
```

ロールベースのアクセスコントロールで Amazon EMR クラスターを起動する

設定が完了したら、「ステップ 1: Amazon EMR でセキュリティ設定を設定する」のセキュリティ設定を使用して Amazon EMR クラスターを起動できます。Amazon EMR ステップでランタイムロールを使用するには、リリースラベル emr-6.7.0 以降を使用し、クラスターアプリケーションとして Hive、Spark、またはその両方を選択します。 SageMaker Studio から接続するには、リリース emr-6.9.0以降を使用し、クラスターアプリケーションとして Livy、Spark、Hive、Presto を選択します。クラスターを起動する手順については、「クラスターのセキュリティ設定を指定する」を参照してください。

Amazon EMR ステップを使用して Spark ジョブを送信する

Apache Spark に含まれている HdfsTest 例を実行する方法の例を次に示します。この API 呼び出しは、指定された Amazon EMR ランタイムロールが S3_LOCATION にアクセスできる場合にのみ成功します。

```
RUNTIME_ROLE_ARN=<runtime-role-arn>
S3_LOCATION=<s3-path>
REGION=<aws-region>
CLUSTER_ID=<cluster-id>

aws emr add-steps --cluster-id $CLUSTER_ID \
--steps '[{ "Name": "Spark Example", "ActionOnFailure": "CONTINUE","HadoopJarStep":
 { "Jar":"command-runner.jar","Args" : ["spark-example","HdfsTest",
 "$S3_LOCATION"] } }]' \
--execution-role-arn $RUNTIME_ROLE_ARN \
--region $REGION
```

> **Note**
>
> Amazon EMR クラスターへの SSH アクセスを無効にし、Amazon EMR AddJobFlowSteps API のみがクラスターにアクセスできるようにすることが推奨されます。

Amazon EMR ステップを使用して Hive ジョブを送信する

次の例では、Apache Hive と Amazon EMR のステップを使用して、QUERY_FILE.hql ファイルを実行するジョブを送信します。このクエリは、指定されたランタイムロールがクエリファイルの Amazon S3 パスにアクセスできる場合にのみ成功します。

```
RUNTIME_ROLE_ARN=<runtime-role-arn>
REGION=<aws-region>
CLUSTER_ID=<cluster-id>

aws emr add-steps --cluster-id $CLUSTER_ID \
--steps '[{ "Name": "Run hive query using command-runner.jar - simple
 select","ActionOnFailure":"CONTINUE","HadoopJarStep": { "Jar": "command-
runner.jar","Args" :["hive -
f","s3://DOC_EXAMPLE_BUCKET/QUERY_FILE.hql"] } }]' \
--execution-role-arn $RUNTIME_ROLE_ARN \
--region $REGION
```

Studio ノートブックからランタイムロールを使用して Amazon EMR SageMakerクラスターに接続する

SageMaker Studio から Amazon EMR クラスターで実行するクエリに Amazon EMR ランタイムロールを適用できます。これを行うには、次のステップを実行します。

1. 「Amazon SageMaker Studio を起動する」の指示に従って SageMaker Studio を作成します。
2. SageMaker Studio UI で、サポートされているカーネルでノートブックを起動します。例えば、PySpark カーネルを使用して SparkMagic イメージを開始します。
3. SageMaker Studio で Amazon EMR クラスターを選択し、Connect を選択します。
4. ランタイムロールを選択し、[接続] を選択します。

これにより、選択した Amazon EMR ランタイムロールを使用して Amazon EMR クラスターに接続するためのマジックコマンドを含む SageMaker ノートブックセルが作成されます。ノートブックセルでは、ランタイムロールと Lake Formation ベースのアクセスコントロールを使用して、クエリを入力および実行できます。より詳細な例については、「Amazon Studio の AWS Lake Formation と Amazon EMR SageMaker を使用してきめ細かなデータアクセスコントロールを適用する」を参照してください。

Amazon EMR ランタイムロールへのアクセスを制御する

条件キー `elasticmapreduce:ExecutionRoleArn` を使用して、ランタイムロールへのアクセスを制御できます。次のポリシーでは、IAM プリンシパルが `Caller` という名前の IAM ロール、または文字列 `CallerTeamRole` で始まる任意の IAM ロールをランタイムロールとして使用することを許可します。

> **Important**
>
> 次の例に示すように、`AddJobFlowSteps` または `GetClusterSessionCredentials` API を呼び出すためのアクセス権限を発信者に付与する場合は、`elasticmapreduce:ExecutionRoleArn` コンテキストキーに基づいて条件を作成する必要があります。

```
{
    "Sid":"AddStepsWithSpecificExecRoleArn",
    "Effect":"Allow",
    "Action":[
        "elasticmapreduce:AddJobFlowSteps"
    ],
    "Resource":"*",
    "Condition":{
        "StringEquals":{
            "elasticmapreduce:ExecutionRoleArn":[
```

```
                "arn:aws:iam::<AWS_ACCOUNT_ID>:role/Caller"
            ]
        },
        "StringLike":{
            "elasticmapreduce:ExecutionRoleArn":[
                "arn:aws:iam::<AWS_ACCOUNT_ID>:role/CallerTeamRole*"
            ]
        }
    }
}
```

ランタイムロールと Amazon EMR クラスターの間に信頼を確立する

Amazon EMR は、ランタイムロール認証が有効になっている各セキュリティ設定に対して、一意の識別子 `ExternalId` を生成します。この認証により、すべてのユーザーが自分に属するクラスターで使用できる一連のランタイムロールを所有できるようになります。例えば、企業では、各部門がそれぞれの外部 ID を使用して、所有する一連のランタイムロールの信頼ポリシーを更新できます。

外部 ID は、次の例に示すように Amazon EMR の `DescribeSecurityConfiguration` API を使用して確認できます。

```
aws emr describe-security-configuration --name 'iamconfig-with-lf'{"Name": "iamconfig-
with-lf",
    "SecurityConfiguration":
        "{\"AuthorizationConfiguration\":{\"IAMConfiguration\":
{\"EnableApplicationScopedIAMRole\
        ":true,\"ApplicationScopedIAMRoleConfiguration\":{\"PropagateSourceIdentity
\":true,\"Exter
        nalId\":\"FXH5TSACFDWUCDSR3YQE2O7ETPUSM4OBCGLYWODSCUZDNZ4Y\"}},\"Lake
        FormationConfiguration\":{\"AuthorizedSessionTagValue\":\"Amazon EMR\"}}}",
    "CreationDateTime": "2022-06-03T12:52:35.308000-07:00"
}
```

外部 ID の使用方法については、AWS 「 リソースへのアクセスをサードパーティに付与するときに外部 ID を使用する方法」を参照してください。

監査

IAM ロールを使用してエンドユーザーが実行するアクションをモニタリングおよび制御するには、ソース ID 機能を有効にします。ソース ID の詳細については、「引き受けたロールで実行されるアクションのモニタリングと制御」を参照してください。

ソース ID を追跡するには、次のように、セキュリティ設定で ApplicationScopedIAMRoleConfiguration/PropagateSourceIdentity を true に設定します。

```
{
    "AuthorizationConfiguration":{
        "IAMConfiguration":{
            "EnableApplicationScopedIAMRole":true,
            "ApplicationScopedIAMRoleConfiguration":{
                "PropagateSourceIdentity":true
            }
        }
    }
}
```

PropagateSourceIdentity を true に設定すると、Amazon EMR は呼び出し側の認証情報からのソース ID を、ランタイムロールで作成したジョブまたはクエリのセッションに適用します。呼び出し側の認証情報にソース ID が存在しない場合、Amazon EMR はソース ID を設定しません。

このプロパティを使用するには、次のように、インスタンスプロファイルに sts:SetSourceIdentity アクセス許可を付与します。

```
{ // PropagateSourceIdentity statement
    "Sid":"PropagateSourceIdentity",
    "Effect":"Allow",
    "Action":"sts:SetSourceIdentity",
    "Resource":[
        <runtime-role-ARN>
    ],
    "Condition":{
        "StringEquals":{
            "sts:SourceIdentity":<source-identity>
        }
    }
}
```

また、AllowSetSourceIdentity ステートメントをランタイムロールの信頼ポリシーに追加する必要があります。

```
{ // AllowSetSourceIdentity statement
    "Sid":"AllowSetSourceIdentity",
    "Effect":"Allow",
```

```
    "Principal":{
        "AWS":"arn:aws:iam::<AWS_ACCOUNT_ID>:role/EMR_EC2_DefaultRole"
    },
    "Action":[
        "sts:SetSourceIdentity",
        "sts:AssumeRole"
    ],
    "Condition":{
        "StringEquals":{
            "sts:SourceIdentity":<source-identity>
        }
    }
}
```

追加の考慮事項

> **Note**
>
> Amazon EMR リリース ではemr-6.9.0、 SageMaker Studio から Amazon EMR クラスターに接続すると、断続的な障害が発生する可能性があります。この問題に対処するには、クラスターの起動時にブートストラップアクションを使用してパッチをインストールします。パッチの詳細については、「Amazon EMR release 6.9.0 known issues」を参照してください。

さらに、Amazon EMR のランタイムロールを設定する際には、次の点を考慮してください。

- Amazon EMR は、すべての商用 AWS リージョンでランタイムロールをサポートしています。
- Amazon EMR ステップでは、リリース emr-6.7.0 以降を使用する場合、ランタイムロールを使用した Apache Spark および Apache Hive のジョブをサポートします。
- SageMaker Studio は、リリース emr-6.9.0以降を使用する場合、ランタイムロールを持つ Spark、Hive、Presto クエリをサポートします。
- の次のノートブックカーネルはランタイムロール SageMaker をサポートしています。
  - DataScience – Python 3 カーネル
  - DataScience 2.0 – Python 3 カーネル
  - DataScience 3.0 – Python 3 カーネル
  - SparkAnalytics 1.0 – SparkMagic および PySpark カーネル
  - SparkAnalytics 2.0 – SparkMagic および PySpark カーネル

  - SparkMagic – PySpark カーネル
- Amazon EMR では、クラスター作成時にのみ RunJobFlow を使用するステップをサポートします。この API はランタイムロールをサポートしていません。
- Amazon EMR では、高可用性用に設定したクラスターでのランタイムロールをサポートしていません。
- command-runner.jar JAR ファイルでコマンドを実行するときは、Bash コマンド引数をエスケープする必要があります。

```
aws emr add-steps --cluster-id <cluster-id> --steps '[{"Name":"sample-
step","ActionOnFailure":"CONTINUE","Jar":"command-runner.jar","Properties":"","Args":
["bash","-c","\"aws s3 ls\""],"Type":"CUSTOM_JAR"}]' --execution-role-
arn <IAM_ROLE_ARN>
```

- ランタイムロールは、HDFS や HMS などのクラスター上のリソースへのアクセスの制御をサポートしていません。

AWS のサービスおよびリソースへのアクセス許可を Amazon EMR に付与する IAM サービスロールの設定

Amazon EMR およびアプリケーション (Hadoop や Spark など) は、実行時に AWS の他のリソースにアクセスしてアクションを実行するための権限が必要です。Amazon EMR 内の各クラスターは、Amazon EC2 インスタンスプロファイル向けのサービスロールおよびロールが必要です。詳細については、「IAM ユーザーガイド」の「IAMロール」および「インスタンスプロファイルの使用」を参照してください。これらのロールにアタッチされている IAM ポリシーにより、クラスターはユーザーに代わって AWS の他のサービスとやり取りできます。

クラスターで Amazon EMR の自動スケーリングを使用する場合は、追加のロール (Auto Scaling ロール) が必要になります。EMR Notebooks を使用する場合は、EMR Notebooks AWS のサービスロールが必要です。

Amazon EMR は、各ロールのアクセス許可を決定するデフォルトのロールとデフォルトの管理ポリシーを提供します。管理ポリシーは によって作成および管理されるため AWS、サービス要件が変更されると自動的に更新されます。「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。

アカウントで初めてクラスターやノートブックを作成する場合、Amazon EMR のロールはまだありません。ロールを作成すると、IAM コンソール (https://console.aws.amazon.com/iam/) でロール、ロールにアタッチされたポリシー、およびポリシーで許可または拒否されるアクセス許可を確認でき

ます。Amazon EMR で作成および使用するデフォルトのロールを指定できます。また、独自のロールを作成して、これをクラスターの作成時に個別に指定してアクセス許可をカスタマイズできます。さらに、 AWS CLIを使用してクラスターを作成するときに使用するデフォルトのロールを指定できます。詳細については、「IAM ロールのカスタマイズ」を参照してください。

Amazon EMR にサービスロールを渡すアクセス許可のアイデンティティベースのポリシーの変更

Amazon EMR のフルアクセス許可のデフォルトの管理ポリシーには、次を含む `iam:PassRole` セキュリティ設定が組み込まれています。

- 特定のデフォルトの Amazon EMR ロール専用の `iam:PassRole` アクセス許可。
- `iam:PassedToService` や などの指定された AWS サービスでのみポリシーを使用できるようにする `elasticmapreduce.amazonaws.com` 条件`ec2.amazonaws.com`。

AmazonEMRFullAccessPolicy AmazonEMRServicePolicy_v2 ポリシーの JSON バージョンは、IAM コンソールで表示できます。v2 管理ポリシーを使用して新しいクラスターを作成することが推奨されます。

サービスロールの概要

次の表は、Amazon EMR に関連する IAM サービスロールを簡単に参照できる一覧です。

機能	デフォルトロール	説明	デフォルト管理ポリシー
Amazon EMR のサービスロール (EMR ロール)	`EMR_DefaultRole_V2`	リソースをプロビジョニングし、 AWS サービスレベルのアクションを実行するときに、Amazon EMR がユーザーに代わって他の サービスを呼び出すことを許可します。このロールは、すべてのクラスターに必須です。	`AmazonEMRServicePolicy_v2` **Important** スポットインスタンスをリクエストするには、サービスリンクロールが必要です

機能	デフォルトロール	説明	デフォルト管理ポリシー
			。このロールが存在しない場合、Amazon EMR サービスロールには、作成するためのアクセス許可が必要です。ない場合はアクセス許可エラーが発生します。スポットインスタンスをリクエストする場合は、このポリシーを更新して、このサービスにリンクされたロールの作成を許可するステートメントを含める必要があります。詳細については、Amazon EMR のサービスロール (EMR ロール)「」および「Amazon EC2 ユーザーガイド」の

機能	デフォルトロール	説明	デフォルト管理ポリシー
			「スポットインスタンスリクエストのサービスにリンクされたロール」を参照してください。 Amazon EC2

機能	デフォルトロール	説明	デフォルト管理ポリシー
クラスター EC2 インスタンスのサービスロール (EC2 インスタンスプロファイル)	EMR_EC2_DefaultRole	クラスターインスタンスの Hadoop エコシステム上で実行されるアプリケーションプロセスは、他の AWS サービスを呼び出すときにこのロールを使用します。EMRFS を使用して Amazon S3 のデータにアクセスする場合は、Amazon S3 のデータの場所に応じて引き受ける各種ロールを指定できます。例えば、複数のチームが単一の Amazon S3 データ「ストレージアカウント」にアクセスできます。詳細については、「Amazon S3 への EMRFS リクエストの IAM ロールを設定する」を参照してください。このロールは、すべてのクラスターに必須です。	AmazonElasticMapReduceforEC2Role 。詳細については、「クラスター EC2 インスタンスのサービスロール (EC2 インスタンスプロファイル)」を参照してください。

機能	デフォルトロール	説明	デフォルト管理ポリシー
Amazon EMR の自動スケーリングのサービスロール (Auto Scaling ロール)	EMR_AutoScaling_DefaultRole	動的なスケーリング環境用の追加のアクションを許可します。Amazon EMR でオートスケーリングを使用するクラスターでのみ必須です。詳細については、「カスタムポリシーによる自動スケーリングをインスタンスグループに使用する」を参照してください。	AmazonElasticMapReduceforAutoScalingRole 。詳細については、「Amazon EMR の自動スケーリングのサービスロール (Auto Scaling ロール)」を参照してください。

機能	デフォルトロール	説明	デフォルト管理ポリシー

EMR Notebooks のサービスロール	EMR_Notebooks_DefaultRole	EMR Notebooks が他の AWS リソースにアクセスしてアクションを実行するために必要なアクセス許可を提供します。EMR Notebooks を使用する場合にのみ必須です。	AmazonElasticMapReduceEditorsRole 。詳細については、「EMR Notebooks のサービスロール」を参照してください。
			
			S3FullAccessPolicy もデフォルトでアタッチされます。このポリシーの内容は次のとおりです。
			
			{
			"Version": "2012-10-17",
			"Statement": [
			{
			"Effect": "Allow",
			"Action": "s3:*",
			"Resource": "*"
			}
			]
			}

機能	デフォルトロール	説明	デフォルト管理ポリシー
サービスにリンクされたロール	AWSServiceRoleForEMRCleanup	Amazon EMR では、サービスにリンクされたロールが自動的に作成されます。Amazon EMR のサービスが Amazon EC2 リソースをクリーンアップできなくなった場合、Amazon EMR はこのロールを使用してクリーンアップできます。クラスターでスポットインスタンスを使用している場合、Amazon EMR のサービスロール (EMR ロール) にアタッチされているアクセス許可ポリシーは、サービスにリンクされたロールの作成を許可する必要があります。詳細については、「Amazon EMR 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。	AmazonEMRCleanupPolicy

トピック

- Amazon EMR で使用される IAM サービスロール

- IAM ロールのカスタマイズ
- Amazon S3 への EMRFS リクエストの IAM ロールを設定する
- AWS Glue Data Catalog への Amazon EMR アクセスのリソースベースのポリシーを使用する
- AWS のサービスを直接呼び出すアプリケーションで IAM ロールを使用する
- ユーザーおよびグループによるロールの作成および変更を許可する

Amazon EMR で使用される IAM サービスロール

Amazon EMR は、クラスターリソースのプロビジョニング、アプリケーションの実行、リソースの動的スケーリング、および EMR Notebooks の作成と実行を行う際に、IAM サービスロールを使用してユーザーに代わってアクションを実行します。Amazon EMR は、他の AWS のサービスとやり取りする際に以下のロールを使用します。各ロールは、Amazon EMR 内で固有の機能を果たします。このセクションのトピックでは、各ロールの機能を説明し、ロール別のデフォルトのロールおよびアクセス許可ポリシーを示します。

AWS サービスを直接呼び出すアプリケーションコードがクラスターにある場合は、 SDK を使用してロールを指定する必要がある場合があります。詳細については、「AWS のサービスを直接呼び出すアプリケーションで IAM ロールを使用する」を参照してください。

トピック

- Amazon EMR のサービスロール (EMR ロール)
- クラスター EC2 インスタンスのサービスロール (EC2 インスタンスプロファイル)
- Amazon EMR の自動スケーリングのサービスロール (Auto Scaling ロール)
- EMR Notebooks のサービスロール
- Amazon EMR のサービスにリンクされたロールの使用

Amazon EMR のサービスロール (EMR ロール)

Amazon EMR ロールは、Amazon EMR がリソースをプロビジョニングし、クラスター内で実行されている Amazon EC2 インスタンスのコンテキストでは実行されないサービスレベルのタスクを実行するときに Amazon EMR に対して許可されるアクションを定義します。たとえば、このサービスロールを使用してクラスターの起動時に EC2 インスタンスをプロビジョニングします。

- デフォルトのロール名は `EMR_DefaultRole_V2` です。

- EMR_DefaultRole_V2 にアタッチされる、Amazon EMR 範囲のデフォルトの管理ポリシーは AmazonEMRServicePolicy_v2 です。この v2 ポリシーは、廃止されたデフォルトの管理ポリシー AmazonElasticMapReduceRole を置き換えるものです。

AmazonEMRServicePolicy_v2 は、Amazon EMR がプロビジョニングまたは使用するリソースへの、範囲が制限されたアクセス権限に依存します。このポリシーを使用する場合は、クラスターのプロビジョニング時にユーザータグ for-use-with-amazon-emr-managed-policies = true を渡す必要があります。Amazon EMR はこれらのタグを自動的に伝播します。さらに、Amazon EMR によって作成されていない EC2 セキュリティグループなど、特定のタイプのリソースにユーザータグを手動で追加する必要がある場合があります。管理ポリシーを使用するためにリソースにタグを付ける を参照してください。

> **Important**
>
> Amazon EMR は、この Amazon EMR サービスロールと AWSServiceRoleForEMRCleanup ロールを使用して、アカウント内で使用しなくなったクラスターリソース (Amazon EC2 インスタンスなど) をクリーンアップします。ロールポリシーにはリソースを削除または終了するアクションを含める必要があります。そうでない場合、Amazon EMR はこれらのクリーンアップアクションを実行できず、クラスターに残っている未使用のリソースに対して料金が発生する可能性があります。

以下は、現在の AmazonEMRServicePolicy_v2 ポリシーの内容を示しています。IAM コンソールで AmazonEMRServicePolicy_v2 管理ポリシーの現在の内容を確認することもできます。

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "CreateInTaggedNetwork",
   "Effect": "Allow",
   "Action": [
    "ec2:CreateNetworkInterface",
    "ec2:RunInstances",
    "ec2:CreateFleet",
    "ec2:CreateLaunchTemplate",
    "ec2:CreateLaunchTemplateVersion"
   ],
   "Resource": [
```

```
    "arn:aws:ec2:*:*:subnet/*",
    "arn:aws:ec2:*:*:security-group/*"
   ],
   "Condition": {
    "StringEquals": {
     "aws:ResourceTag/for-use-with-amazon-emr-managed-policies": "true"
    }
   }
  },
  {
   "Sid": "CreateWithEMRTaggedLaunchTemplate",
   "Effect": "Allow",
   "Action": [
    "ec2:CreateFleet",
    "ec2:RunInstances",
    "ec2:CreateLaunchTemplateVersion"
   ],
   "Resource": "arn:aws:ec2:*:*:launch-template/*",
   "Condition": {
    "StringEquals": {
     "aws:ResourceTag/for-use-with-amazon-emr-managed-policies": "true"
    }
   }
  },
  {
   "Sid": "CreateEMRTaggedLaunchTemplate",
   "Effect": "Allow",
   "Action": "ec2:CreateLaunchTemplate",
   "Resource": "arn:aws:ec2:*:*:launch-template/*",
   "Condition": {
    "StringEquals": {
     "aws:RequestTag/for-use-with-amazon-emr-managed-policies": "true"
    }
   }
  },
  {
   "Sid": "CreateEMRTaggedInstancesAndVolumes",
   "Effect": "Allow",
   "Action": [
    "ec2:RunInstances",
    "ec2:CreateFleet"
   ],
   "Resource": [
    "arn:aws:ec2:*:*:instance/*",
```

```
    "arn:aws:ec2:*:*:volume/*"
   ],
   "Condition": {
    "StringEquals": {
     "aws:RequestTag/for-use-with-amazon-emr-managed-policies": "true"
    }
   }
  },
  {
   "Sid": "ResourcesToLaunchEC2",
   "Effect": "Allow",
   "Action": [
    "ec2:RunInstances",
    "ec2:CreateFleet",
    "ec2:CreateLaunchTemplate",
    "ec2:CreateLaunchTemplateVersion"
   ],
   "Resource": [
    "arn:aws:ec2:*:*:network-interface/*",
    "arn:aws:ec2:*::image/ami-*",
    "arn:aws:ec2:*:*:key-pair/*",
    "arn:aws:ec2:*:*:capacity-reservation/*",
    "arn:aws:ec2:*:*:placement-group/pg-*",
    "arn:aws:ec2:*:*:fleet/*",
    "arn:aws:ec2:*:*:dedicated-host/*",
    "arn:aws:resource-groups:*:*:group/*"
   ]
  },
  {
   "Sid": "ManageEMRTaggedResources",
   "Effect": "Allow",
   "Action": [
    "ec2:CreateLaunchTemplateVersion",
    "ec2:DeleteLaunchTemplate",
    "ec2:DeleteNetworkInterface",
    "ec2:ModifyInstanceAttribute",
    "ec2:TerminateInstances"
   ],
   "Resource": "*",
   "Condition": {
    "StringEquals": {
     "aws:ResourceTag/for-use-with-amazon-emr-managed-policies": "true"
    }
   }
```

```
  },
  {
   "Sid": "ManageTagsOnEMRTaggedResources",
   "Effect": "Allow",
   "Action": [
    "ec2:CreateTags",
    "ec2:DeleteTags"
   ],
   "Resource": [
    "arn:aws:ec2:*:*:instance/*",
    "arn:aws:ec2:*:*:volume/*",
    "arn:aws:ec2:*:*:network-interface/*",
    "arn:aws:ec2:*:*:launch-template/*"
   ],
   "Condition": {
    "StringEquals": {
     "aws:ResourceTag/for-use-with-amazon-emr-managed-policies": "true"
    }
   }
  },
  {
   "Sid": "CreateNetworkInterfaceNeededForPrivateSubnet",
   "Effect": "Allow",
   "Action": [
    "ec2:CreateNetworkInterface"
   ],
   "Resource": [
    "arn:aws:ec2:*:*:network-interface/*"
   ],
   "Condition": {
    "StringEquals": {
     "aws:RequestTag/for-use-with-amazon-emr-managed-policies": "true"
    }
   }
  },
  {
   "Sid": "TagOnCreateTaggedEMRResources",
   "Effect": "Allow",
   "Action": [
    "ec2:CreateTags"
   ],
   "Resource": [
    "arn:aws:ec2:*:*:network-interface/*",
    "arn:aws:ec2:*:*:instance/*",
```

```
    "arn:aws:ec2:*:*:volume/*",
    "arn:aws:ec2:*:*:launch-template/*"
   ],
   "Condition": {
    "StringEquals": {
     "ec2:CreateAction": [
      "RunInstances",
      "CreateFleet",
      "CreateLaunchTemplate",
      "CreateNetworkInterface"
     ]
    }
   }
  },
  {
   "Sid": "TagPlacementGroups",
   "Effect": "Allow",
   "Action": [
    "ec2:CreateTags",
    "ec2:DeleteTags"
   ],
   "Resource": [
    "arn:aws:ec2:*:*:placement-group/pg-*"
   ]
  },
  {
   "Sid": "ListActionsForEC2Resources",
   "Effect": "Allow",
   "Action": [
    "ec2:DescribeAccountAttributes",
    "ec2:DescribeCapacityReservations",
    "ec2:DescribeDhcpOptions",
    "ec2:DescribeImages",
    "ec2:DescribeInstances",
    "ec2:DescribeInstanceTypeOfferings",
    "ec2:DescribeLaunchTemplates",
    "ec2:DescribeNetworkAcls",
    "ec2:DescribeNetworkInterfaces",
    "ec2:DescribePlacementGroups",
    "ec2:DescribeRouteTables",
    "ec2:DescribeSecurityGroups",
    "ec2:DescribeSubnets",
    "ec2:DescribeVolumes",
    "ec2:DescribeVolumeStatus",
```

```
    "ec2:DescribeVpcAttribute",
    "ec2:DescribeVpcEndpoints",
    "ec2:DescribeVpcs"
   ],
   "Resource": "*"
  },
  {
   "Sid": "CreateDefaultSecurityGroupWithEMRTags",
   "Effect": "Allow",
   "Action": [
    "ec2:CreateSecurityGroup"
   ],
   "Resource": [
    "arn:aws:ec2:*:*:security-group/*"
   ],
   "Condition": {
    "StringEquals": {
     "aws:RequestTag/for-use-with-amazon-emr-managed-policies": "true"
    }
   }
  },
  {
   "Sid": "CreateDefaultSecurityGroupInVPCWithEMRTags",
   "Effect": "Allow",
   "Action": [
    "ec2:CreateSecurityGroup"
   ],
   "Resource": [
    "arn:aws:ec2:*:*:vpc/*"
   ],
   "Condition": {
    "StringEquals": {
     "aws:ResourceTag/for-use-with-amazon-emr-managed-policies": "true"
    }
   }
  },
  {
   "Sid": "TagOnCreateDefaultSecurityGroupWithEMRTags",
   "Effect": "Allow",
   "Action": [
    "ec2:CreateTags"
   ],
   "Resource": "arn:aws:ec2:*:*:security-group/*",
   "Condition": {
```

```
    "StringEquals": {
     "aws:RequestTag/for-use-with-amazon-emr-managed-policies": "true",
     "ec2:CreateAction": "CreateSecurityGroup"
    }
   }
  },
  {
   "Sid": "ManageSecurityGroups",
   "Effect": "Allow",
   "Action": [
    "ec2:AuthorizeSecurityGroupEgress",
    "ec2:AuthorizeSecurityGroupIngress",
    "ec2:RevokeSecurityGroupEgress",
    "ec2:RevokeSecurityGroupIngress"
   ],
   "Resource": "*",
   "Condition": {
    "StringEquals": {
     "aws:ResourceTag/for-use-with-amazon-emr-managed-policies": "true"
    }
   }
  },
  {
   "Sid": "CreateEMRPlacementGroups",
   "Effect": "Allow",
   "Action": [
    "ec2:CreatePlacementGroup"
   ],
   "Resource": "arn:aws:ec2:*:*:placement-group/pg-*"
  },
  {
   "Sid": "DeletePlacementGroups",
   "Effect": "Allow",
   "Action": [
    "ec2:DeletePlacementGroup"
   ],
   "Resource": "*"
  },
  {
   "Sid": "AutoScaling",
   "Effect": "Allow",
   "Action": [
    "application-autoscaling:DeleteScalingPolicy",
    "application-autoscaling:DeregisterScalableTarget",
```

```
    "application-autoscaling:DescribeScalableTargets",
    "application-autoscaling:DescribeScalingPolicies",
    "application-autoscaling:PutScalingPolicy",
    "application-autoscaling:RegisterScalableTarget"
   ],
   "Resource": "*"
  },
  {
   "Sid": "ResourceGroupsForCapacityReservations",
   "Effect": "Allow",
   "Action": [
    "resource-groups:ListGroupResources"
   ],
   "Resource": "*"
  },
  {
   "Sid": "AutoScalingCloudWatch",
   "Effect": "Allow",
   "Action": [
    "cloudwatch:PutMetricAlarm",
    "cloudwatch:DeleteAlarms",
    "cloudwatch:DescribeAlarms"
   ],
   "Resource": "arn:aws:cloudwatch:*:*:alarm:*_EMR_Auto_Scaling"
  },
  {
   "Sid": "PassRoleForAutoScaling",
   "Effect": "Allow",
   "Action": "iam:PassRole",
   "Resource": "arn:aws:iam::*:role/EMR_AutoScaling_DefaultRole",
   "Condition": {
    "StringLike": {
     "iam:PassedToService": "application-autoscaling.amazonaws.com*"
    }
   }
  },
  {
   "Sid": "PassRoleForEC2",
   "Effect": "Allow",
   "Action": "iam:PassRole",
   "Resource": "arn:aws:iam::*:role/EMR_EC2_DefaultRole",
   "Condition": {
    "StringLike": {
     "iam:PassedToService": "ec2.amazonaws.com*"
```

```
    }
   }
  }
 ]
}
```

サービスロールでは、次の信頼ポリシーを使用する必要があります。

> **Important**
>
> 次の信頼ポリシーには、Amazon EMR に付与するアクセス許可をアカウント内の特定のリソースに制限するためのグローバル条件キー aws:SourceArn および aws:SourceAccount が含まれています。これらを使用することで、「混乱した代理」問題から保護できます。

```
{
    "Version": "2008-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "elasticmapreduce.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "<account-id>"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:elasticmapreduce:<region>:<account-id>:*"
                }
            }
        }
    ]
}
```

クラスター EC2 インスタンスのサービスロール (EC2 インスタンスプロファイル)

クラスター EC2 インスタンスのサービスロール (Amazon EMR の EC2 インスタンスプロファイルとも呼ばれます) は、インスタンスの起動時に Amazon EMR クラスター内のすべての EC2 インスタンスに割り当てられる特殊なサービスロールです。Hadoop エコシステム上で実行されるアプリケーションプロセスは、このロールを引き受けることで、 AWS の他のサービスとやり取りするアクセス許可を取得します。

EC2 インスタンスのサービスロールの詳細については、IAM ユーザーガイドの「Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用してアクセス許可を付与する」を参照してください。

> **Important**
>
> クラスター EC2 インスタンスのデフォルトのサービスロールとそれに関連する AWS デフォルトの管理ポリシー`AmazonElasticMapReduceforEC2Role`は、代替 AWS の管理ポリシーが提供されず、非推奨になる予定です。廃止されるロールとデフォルトポリシーを置き換えるには、インスタンスプロファイルを作成して指定する必要があります。

デフォルトのロールとデフォルトの管理ポリシー

- デフォルトのロール名は `EMR_EC2_DefaultRole` です。
- `EMR_EC2_DefaultRole` のデフォルトの管理ポリシー `AmazonElasticMapReduceforEC2Role` のサポートはまもなく終了します。EC2 インスタンスプロファイルのデフォルトの管理ポリシーを使用する代わりに、Amazon EMR が必要とする S3 バケットおよびその他のリソースにリソースベースのポリシーを適用するか、インスタンスプロファイルとして IAM ロールを使用して独自のカスタマー管理ポリシーを使用します。詳細については、「最低限のアクセス権限を持つクラスター EC2 インスタンスのサービスロールの作成」を参照してください。

以下は、`AmazonElasticMapReduceforEC2Role` のバージョン 3 の内容を示しています。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Resource": "*",
```

```
            "Action": [
                "cloudwatch:*",
                "dynamodb:*",
                "ec2:Describe*",
                "elasticmapreduce:Describe*",
                "elasticmapreduce:ListBootstrapActions",
                "elasticmapreduce:ListClusters",
                "elasticmapreduce:ListInstanceGroups",
                "elasticmapreduce:ListInstances",
                "elasticmapreduce:ListSteps",
                "kinesis:CreateStream",
                "kinesis:DeleteStream",
                "kinesis:DescribeStream",
                "kinesis:GetRecords",
                "kinesis:GetShardIterator",
                "kinesis:MergeShards",
                "kinesis:PutRecord",
                "kinesis:SplitShard",
                "rds:Describe*",
                "s3:*",
                "sdb:*",
                "sns:*",
                "sqs:*",
                "glue:CreateDatabase",
                "glue:UpdateDatabase",
                "glue:DeleteDatabase",
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:CreateTable",
                "glue:UpdateTable",
                "glue:DeleteTable",
                "glue:GetTable",
                "glue:GetTables",
                "glue:GetTableVersions",
                "glue:CreatePartition",
                "glue:BatchCreatePartition",
                "glue:UpdatePartition",
                "glue:DeletePartition",
                "glue:BatchDeletePartition",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:BatchGetPartition",
                "glue:CreateUserDefinedFunction",
                "glue:UpdateUserDefinedFunction",
```

```
                "glue:DeleteUserDefinedFunction",
                "glue:GetUserDefinedFunction",
                "glue:GetUserDefinedFunctions"
            ]
        }
    ]
}
```

サービスロールでは、次の信頼ポリシーを使用する必要があります。

```
{
    "Version": "2008-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "ec2.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

最低限のアクセス権限を持つクラスター EC2 インスタンスのサービスロールの作成

ベストプラクティスとして、クラスター EC2 インスタンスのサービスロールと、アプリケーションに必要な他の AWS サービスへの最小限のアクセス許可を持つアクセス許可ポリシーを作成することを強くお勧めします。

デフォルトの管理ポリシー (AmazonElasticMapReduceforEC2Role) が提供するアクセス許可を使用すると、最初のクラスターを簡単に起動できます。ただし、AmazonElasticMapReduceforEC2Roleは非推奨になる予定であり、Amazon EMR は非推奨ロールの代替 AWS マネージドデフォルトポリシーを提供しません。初期クラスターを起動するには、カスタマー管理リソースベースまたは ID ベースのポリシーを提供する必要があります。

以下のポリシーステートメントは、Amazon EMR の機能別に必要なアクセス許可の例を示しています。これらのアクセス許可を使用することで、クラスターが必要とする機能やリソースにのみアクセスを制限するアクセス許可ポリシーを作成することをお勧めします。すべてのポリシーステートメントの例では、 *us-west-2*リージョンと架空の AWS アカウント ID を使用します*123456789012*。実際のクラスターに応じて置き換えてください。

カスタムロールの作成と指定の詳細については、「IAM ロールのカスタマイズ」を参照してください。

> **Note**
>
> EC2 のカスタム EMR ロールを作成する場合は、基本的なワークフローに従います。これにより、同じ名前のインスタンスプロファイルが自動的に作成されます。Amazon EC2 では、異なる名前のインスタンスプロファイルとロールを作成できますが、Amazon EMR はこの設定をサポートしていないため、クラスターの作成時に「無効なインスタンスプロファイル」エラーが発生します。

EMRFS を使用した Amazon S3 に対するデータの読み書き

Amazon EMR クラスターで実行されているアプリケーションが `s3://mydata` 形式を使用してデータを参照する場合、Amazon EMR は EC2 インスタンスプロファイルを使用してリクエストを行います。通常、クラスターはこの方法で Amazon S3 に対するデータの読み書きを行います。Amazon EMR は、デフォルトでクラスター EC2 インスタンスのサービスロールにアタッチされているアクセス許可を使用します。詳細については、「Amazon S3 への EMRFS リクエストの IAM ロールを設定する」を参照してください。

EMRFS の IAM ロールは、クラスター EC2 インスタンスのサービスロールにアタッチされているアクセス許可にフォールバックするため、ベストプラクティスとして、EMRFS の IAM ロールを使用し、クラスター EC2 インスタンスのサービスロールにアタッチされている EMRFS および Amazon S3 アクセス許可に制限することをお勧めします。

次のステートメントの例は、EMRFS が Amazon S3 に対してリクエストを行うために必要なアクセス許可を示しています。

- *my-data-bucket-in-s3-for-emrfs-reads-and-writes* は、クラスターがデータを読み書きする Amazon S3 内のバケットを指定します。また、*/** を使用してすべてのサブフォルダを指定します。アプリケーションが必要とするバケットおよびフォルダのみを追加します。
- dynamodb アクションを許可するポリシーステートメントは、EMRFS の整合性のあるビューが有効になっている場合にのみ必要です。*EmrFSMetadata* は、EMRFS の整合性のあるビューのデフォルトフォルダを指定します。

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:CreateBucket",
                "s3:DeleteObject",
                "s3:GetBucketVersioning",
                "s3:GetObject",
                "s3:GetObjectTagging",
                "s3:GetObjectVersion",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:ListBucketVersions",
                "s3:ListMultipartUploadParts",
                "s3:PutBucketVersioning",
                "s3:PutObject",
                "s3:PutObjectTagging"
            ],
            "Resource": [
                "arn:aws:s3:::my-data-bucket-in-s3-for-emrfs-reads-and-writes",
                "arn:aws:s3:::my-data-bucket-in-s3-for-emrfs-reads-and-writes/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:CreateTable",
                "dynamodb:BatchGetItem",
                "dynamodb:BatchWriteItem",
                "dynamodb:PutItem",
                "dynamodb:DescribeTable",
                "dynamodb:DeleteItem",
                "dynamodb:GetItem",
                "dynamodb:Scan",
                "dynamodb:Query",
                "dynamodb:UpdateItem",
                "dynamodb:DeleteTable",
                "dynamodb:UpdateTable"
            ],
            "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/EmrFSMetadata"
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "cloudwatch:PutMetricData",
                "dynamodb:ListTables",
                "s3:ListBucket"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "sqs:GetQueueUrl",
                "sqs:ReceiveMessage",
                "sqs:DeleteQueue",
                "sqs:SendMessage",
                "sqs:CreateQueue"
            ],
            "Resource": "arn:aws:sqs:us-west-2:123456789012:EMRFS-Inconsistency-*"
        }
    ]
}
```

Amazon S3 へのログファイルのアーカイブ

次のポリシーステートメントでは、指定された Amazon S3 の場所にログファイルをアーカイブすることを Amazon EMR クラスターに許可します。以下の例では、クラスターの作成時に、コンソールのログフォルダ S3 の場所、 `--log-uri`のオプション、または `RunJobFlow` コマンドの `LogUri`パラメータを使用して AWS CLIが指定され*s3://MyLoggingBucket/MyEMRClusterLogs*ました。詳細については、「Simple Storage Service (Amazon S3) にログファイルをアーカイブする」を参照してください。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::MyLoggingBucket/MyEMRClusterLogs/*"
        }
    ]
}
```

デバッグツールの使用

次のポリシーステートメントでは、Amazon EMR デバッグツールを有効にする場合に必要となるアクションを許可します。デバッグには、Amazon S3 へのログファイルのアーカイブと、上の例に示した関連するアクセス許可が必要です。詳細については、「デバッグツールを有効にする」を参照してください。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sqs:GetQueueUrl",
                "sqs:SendMessage"
            ],
            "Resource": "arn:aws:sqs:us-west-2:123456789012:AWS-ElasticMapReduce-*"
        }
    ]
}
```

AWS Glue データカタログの使用

次のポリシーステートメントでは、アプリケーションのメタストアとして AWS Glue Data Catalog を使用する場合に必要なアクションを許可します。詳細については、「Amazon EMR リリースガイド」の「Spark SQL のメタストア AWS としての AWS Glue データカタログの使用」、「Hive のメタストアとしての Glue データカタログの使用」、および「AWS Glue データカタログでの Presto の使用」を参照してください。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:CreateDatabase",
                "glue:UpdateDatabase",
                "glue:DeleteDatabase",
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:CreateTable",
                "glue:UpdateTable",
```

```
                "glue:DeleteTable",
                "glue:GetTable",
                "glue:GetTables",
                "glue:GetTableVersions",
                "glue:CreatePartition",
                "glue:BatchCreatePartition",
                "glue:UpdatePartition",
                "glue:DeletePartition",
                "glue:BatchDeletePartition",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:BatchGetPartition",
                "glue:CreateUserDefinedFunction",
                "glue:UpdateUserDefinedFunction",
                "glue:DeleteUserDefinedFunction",
                "glue:GetUserDefinedFunction",
                "glue:GetUserDefinedFunctions"
            ],
            "Resource": "*",
        }
    ]
}
```

Amazon EMR の自動スケーリングのサービスロール (Auto Scaling ロール)

Amazon EMR の Auto Scaling ロールは、サービスロールと同様の機能を実行しますが、環境を動的にスケーリングするための追加のアクションを実行できます。

- デフォルトのロール名は EMR_AutoScaling_DefaultRole です。
- EMR_AutoScaling_DefaultRole にアタッチされるデフォルトの管理ポリシーは AmazonElasticMapReduceforAutoScalingRole です。

AmazonElasticMapReduceforAutoScalingRole のバージョン 1 の内容は次のとおりです。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "cloudwatch:DescribeAlarms",
                "elasticmapreduce:ListInstanceGroups",
                "elasticmapreduce:ModifyInstanceGroups"
```

```
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

サービスロールでは、次の信頼ポリシーを使用する必要があります。

> **Important**
>
> 次の信頼ポリシーには、Amazon EMR に付与するアクセス許可をアカウント内の特定のリソースに制限するためのグローバル条件キー aws:SourceArn および aws:SourceAccount が含まれています。これらを使用することで、「混乱した代理」問題から保護できます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "application-autoscaling.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "<account-id>"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:application-
autoscaling:<region>:<account-id>:scalable-target/*"
                }
            }
        }
    ]
}
```

EMR Notebooks のサービスロール

各 EMR Notebooks には、他の AWS リソースにアクセスしてアクションを実行するためのアクセス許可が必要です。このサービスロールにアタッチされた IAM ポリシーは、ノートブックが他の AWS サービスと相互運用するためのアクセス許可を提供します。を使用してノートブックを作成するときは AWS Management Console、 AWS サービスロール を指定します。デフォルトのロール (EMR_Notebooks_DefaultRole) を使用するか、独自に作成したロールを指定できます。ノートブックを以前に作成していない場合は、デフォルトのロールを作成することを選択できます。

- デフォルトのロール名は EMR_Notebooks_DefaultRole です。
- EMR_Notebooks_DefaultRole にアタッチされるデフォルト管理ポリシーは、AmazonElasticMapReduceEditorsRole および S3FullAccessPolicy です。

サービスロールでは、次の信頼ポリシーを使用する必要があります。

> **Important**
>
> 次の信頼ポリシーには、Amazon EMR に付与するアクセス許可をアカウント内の特定のリソースに制限するためのグローバル条件キー aws:SourceArn および aws:SourceAccount が含まれています。これらを使用することで、「混乱した代理」問題から保護できます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "elasticmapreduce.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "<account-id>"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:elasticmapreduce:<region>:<account-id>:*"
                }
            }
```

```
        }
    ]
}
```

AmazonElasticMapReduceEditorsRole のバージョン 1 の内容は以下のとおりです。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:CreateSecurityGroup",
                "ec2:DescribeSecurityGroups",
                "ec2:RevokeSecurityGroupEgress",
                "ec2:CreateNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DescribeNetworkInterfaces",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:DescribeTags",
                "ec2:DescribeInstances",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "elasticmapreduce:ListInstances",
                "elasticmapreduce:DescribeCluster",
                "elasticmapreduce:ListSteps"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateTags",
            "Resource": "arn:aws:ec2:*:*:network-interface/*",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": [
                        "aws:elasticmapreduce:editor-id",
                        "aws:elasticmapreduce:job-flow-id"
                    ]
```

```
                }
            }
        }
    ]
}
```

S3FullAccessPolicy の内容は次のとおりです。S3FullAccessPolicy により、EMR Notebooks のサービスロールが、 AWS アカウントのオブジェクトに対してすべての Amazon S3 アクションを実行できます。EMR Notebooks のカスタムサービスロールを作成するときは、サービスロールに Amazon S3 アクセス許可を付与する必要があります。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": "*"
        }
    ]
}
```

サービスロールの読み取りおよび書き込みアクセスの範囲を、ノートブックファイルを保存する Amazon S3 の場所に制限できます。次の最小 Amazon S3 アクセス許可セットを使用します。

```
"s3:PutObject",
"s3:GetObject",
"s3:GetEncryptionConfiguration",
"s3:ListBucket",
"s3:DeleteObject"
```

Amazon S3 バケットが暗号化されている場合は、 AWS Key Management Serviceの以下のアクセス許可を含める必要があります。

```
"kms:Decrypt",
"kms:GenerateDataKey",
"kms:ReEncryptFrom",
"kms:ReEncryptTo",
"kms:DescribeKey"
```

Git リポジトリをノートブックにリンクし、リポジトリのシークレットを作成する必要がある場合は、Amazon EMR Notebooks のサービスロールにアタッチされた IAM ポリシーに `secretsmanager:GetSecretValue` アクセス許可を追加する必要があります。ポリシーの例を以下に示します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "*"
        }
    ]
}
```

EMR Notebooks サービスロールのアクセス許可

次の表は、EMR Notebooks がサービスロールを使用して実行するアクションと、各アクションに必要なアクセス許可をリストしています。

アクション	アクセス許可

ノートブックと Amazon EMR クラスターの間にセキュアネットワークチャネルを確立し、必要なクリーンアップアクションを実行する。	`"ec2:CreateNetworkInterface",`
	`"ec2:CreateNetworkInterfacePermission",`
	`"ec2:DeleteNetworkInterface",`
	`"ec2:DeleteNetworkInterfacePermission",`
	`"ec2:DescribeNetworkInterfaces",`
	`"ec2:ModifyNetworkInterfaceAttribute",`
	`"ec2:AuthorizeSecurityGroupEgress",`
	`"ec2:AuthorizeSecurityGroupIngress",`
	`"ec2:CreateSecurityGroup",`
	`"ec2:DescribeSecurityGroups",`
	`"ec2:RevokeSecurityGroupEgress",`
	`"ec2:DescribeTags",`
	`"ec2:DescribeInstances",`
	`"ec2:DescribeSubnets",`
	`"ec2:DescribeVpcs",`
	`"elasticmapreduce:ListInstances",`
	`"elasticmapreduce:DescribeCluster",`

アクション	アクセス許可
	`"elasticmapreduce:ListSteps"`
AWS Secrets Manager に保存されている Git 認証情報を使用して Git リポジトリをノートブックにリンクする。	`"secretsmanager:GetSecretValue"`
セキュアネットワークチャネルの設定中に EMR Notebooks が作成するネットワークインターフェイスとデフォルトのセキュリティグループに AWS タグを適用します。詳細については、「AWS リソースのタグ付け」を参照してください。	`"ec2:CreateTags"`

ノートブックファイルとメタデータにアクセスする、それらを Amazon S3 にアップロードする。	`"s3:PutObject",`
	`"s3:GetObject",`
	`"s3:GetEncryptionConfiguration",`
	`"s3:ListBucket",`
	`"s3:DeleteObject"`
	
	以下のアクセス許可は、暗号化された Amazon S3 バケットを使用する場合にのみ必要です。
	
	`"kms:Decrypt",`
	`"kms:GenerateDataKey",`
	`"kms:ReEncryptFrom",`
	`"kms:ReEncryptTo",`
	`"kms:DescribeKey"`

AWS マネージドポリシーに対する EMR Notebooks の更新

2021 年 3 月 1 日以降の EMR Notebooks の AWS マネージドポリシーの更新に関する詳細を表示します。

変更	説明	日付
AmazonElasticMapReduceEditorsRole - Added permissions	EMR Notebooks で ec2:describeVPCs および elastmicmapreduce:ListSteps アクセス許可が AmazonElasticMapReduceEditorsRole に追加されました。	2023 年 2 月 8 日
EMR Notebooks が変更の追跡を開始	EMR Notebooks が AWS マネージドポリシーの変更の追跡を開始しました。	2023 年 2 月 8 日

Amazon EMR のサービスにリンクされたロールの使用

Amazon EMR 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、Amazon EMR に直接リンクされた特殊な IAM ロールです。サービスにリンクされたロールは Amazon EMR によって事前定義されており、ユーザーに代わってサービスから他の AWS のサービスを呼び出すために必要なすべてのアクセス許可が含まれています。

トピック

- クリーンアップにサービスにリンクされたロールを使用する
- 先行書き込みログ記録にサービスにリンクされたロールを使用する

サービスリンクロールをサポートする他のサービスについては、「IAM と連動するAWS のサービス」を参照し、[Service-linked role (サービスリンクロール)] の列内で [Yes (はい)] と表記されたサービスを確認してください。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[はい] リンクを選択します。

クリーンアップにサービスにリンクされたロールを使用する

Amazon EMR 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、Amazon EMR に直接リンクされた特殊な IAM ロールです。サービスにリンクされたロールは Amazon EMR によって事前定義されており、ユーザーに

代わってサービスから他の AWS のサービスを呼び出すために必要なすべてのアクセス許可が含まれています。

サービスにリンクされたロールは、Amazon EMR のサービスロールおよび Amazon EMR の Amazon EC2 インスタンスプロファイルと連携します。サービスロールとインスタンスプロファイルに関する詳細については、「AWS のサービスおよびリソースへのアクセス許可を Amazon EMR に付与する IAM サービスロールの設定」を参照してください。

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、Amazon EMR の設定が簡単になります。Amazon EMR は、サービスにリンクされたロールのアクセス許可を定義します。特に定義されている場合を除き、Amazon EMR のみがそのロールを引き受けることができます。定義したアクセス許可には、信頼ポリシーと許可ポリシーが含まれます。こ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。

Amazon EMR のサービスにリンクされたロールは、関連するリソースを削除し、アカウント内のすべての EMR クラスターを終了した後にのみ削除できます。これにより、Amazon EMR リソースが保護されるため、リソースへのアクセス許可が誤って削除されることはありません。

クリーンアップにサービスにリンクされたロールを使用する

Amazon EMR は、サービスベースのAWSServiceRoleForEMRCleanupロールを使用して、Amazon EMR サービスにリンクされたロールがその機能を失った場合に、ユーザーに代わって Amazon EC2 リソースを終了および削除するアクセス許可を Amazon EMR に付与します。Amazon EMR は、サービスにリンクされたロールが存在しない場合、クラスターの作成時に自動的に作成します。

AWSServiceRoleForEMRCleanup サービスにリンクされたロールは、次のサービスを信頼してロールを引き受けます。

- `elasticmapreduce.amazonaws.com`

AWSServiceRoleForEMRCleanup サービスにリンクされたロールのアクセス許可ポリシーにより、Amazon EMR 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。

- アクション: `ec2` 上で `DescribeInstances`
- アクション: `ec2` 上で `DescribeSpotInstanceRequests`
- アクション: `ec2` 上で `ModifyInstanceAttribute`
- アクション: `ec2` 上で `TerminateInstances`
- アクション: `ec2` 上で `CancelSpotInstanceRequests`
- アクション: `ec2` 上で `DeleteNetworkInterface`

- アクション: ec2 上で DescribeInstanceAttribute
- アクション: ec2 上で DescribeVolumeStatus
- アクション: ec2 上で DescribeVolumes
- アクション: ec2 上で DetachVolume
- アクション: DeleteVolume 上で ec2

サービスにリンクされたロールの作成、編集、削除を IAM エンティティ (ユーザー、グループ、ロールなど) に許可するには、権限を設定する必要があります。

Amazon EMR のサービスリンクロールの作成

AWSServiceRoleForEMRCleanup ロールを手動で作成する必要はありません。クラスターを初めて起動する場合、または AWSServiceRoleForEMRCleanup サービスにリンクされたロールが存在しない場合、Amazon EMR は AWSServiceRoleForEMRCleanup サービスにリンクされたロールを作成します。サービスにリンクされたロールを作成するには、アクセス許可が必要です。この機能を (ユーザー、グループ、ロールなどの) IAM エンティティのアクセス許可ポリシーに追加するステートメントの例については、「クリーンアップにサービスにリンクされたロールを使用する」を参照してください。

> **Important**
>
> サービスにリンクされたロールがサポートされていない 2017 年 10 月 24 日より前に Amazon EMR を使用した場合、Amazon EMR はアカウントに AWSServiceRoleForEMRCleanup サービスにリンクされたロールを作成しました。詳細については、「IAM アカウントに新しいロールが表示される」を参照してください。

Amazon EMR のサービスリンクロールの編集

Amazon EMR では、 AWSServiceRoleForEMRCleanup サービスにリンクされたロールを編集することはできません。サービスにリンクされたロールを作成した後は、さまざまなエンティティがサービスにリンクされたロールを参照する可能性があるため、サービスにリンクされたロールの名前を変更することはできません。ただし、IAM を使用してサービスにリンクされたロールの説明を編集することはできます。

サービスにリンクされたロールの説明の編集 (IAM コンソール)

サービスにリンクされたロールの説明は、IAM コンソールを使用して編集できます。

サービスにリンクされたロールの説明を編集するには (コンソール)

1. 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。
2. 変更するロールの名前を選択します。
3. [Role description] (ロールの説明) の右にある [Edit] (編集) を選択します。
4. ボックスに新しい説明を入力し、[Save changes] (変更の保存) を選択します。

サービスにリンクされたロールの説明の編集 (IAM 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の説明を編集できます。

サービスにリンクされたロールの説明を変更するには (CLI)

1. (オプション) ロールの現在の説明を表示するには、次のコマンドを使用します。

   ```
   $ aws iam get-role --role-name role-name
   ```

   CLI コマンドでは、ARN ではなくロール名を使用してロールを参照します。例えば、ロールの ARN が `arn:aws:iam::123456789012:role/myrole` である場合、そのロールを **myrole** と参照します。
2. サービスにリンクされたロールの説明を更新するには、次のいずれかのコマンドを使用します。

   ```
   $ aws iam update-role-description --role-name role-name --description description
   ```

サービスにリンクされたロールの説明の編集 (IAM API)

サービスにリンクされたロールの説明は、IAM API を使用して編集できます。

サービスにリンクされたロールの説明を変更するには (API)

1. (オプション) ロールの現在の説明を表示するには、次のコマンドを使用します。

   IAM API: GetRole
2. ロールの説明を更新するには、次のコマンドを使用します。

   IAM API: UpdateRoleDescription

Amazon EMR のサービスリンクロールの削除

サービスにリンクされたロールを必要とする機能やサービスを使用する必要がなくなった場合は、そのサービスにリンクされた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、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。

サービスにリンクされた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、まずサービスにリンクされたロールにアクティブなセッションがないことを確認し、サービスにリンクされたロールが使用するリソース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にアクティブなセッションがあるかどうかを、IAM コンソールで確認するには

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] を選択します。 AWSServiceRoleForEMRCleanup サービスにリンクされたロールの名前 (チェックボックスではありません) を選択します。
3. 選択したサービスにリンクされたロールの概要ページで、Access Advisor を選択します。
4. [アクセスアドバイザー] タブで、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**
>
> Amazon EMR が AWSServiceRoleForEMRCleanup サービスにリンクされたロールを使用しているかどうか不明な場合は、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使用している場合、削除は失敗し、サービスにリンクされたロールが使用されているリージョンを表示できます。サービスにリンクされたロールが使用されている場合は、セッションが終了するのを待ってから、サービスにリンクされたロール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のセッションを取り消すことはできません。

が使用する Amazon EMR リソースを削除するには AWSServiceRoleForEMRCleanup

- アカウントのすべてのクラスターを終了します。詳細については、「クラスターを終了する」を参照してください。

サービスにリンクされたロールの削除 (IAM 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. 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。名前や行自体ではなく AWSServiceRoleForEMRCleanup、 の横にあるチェックボックスをオンにします。
3. ページ上部にある ロールのアクション で 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。このデータには、選択した各ロールが最後に AWS サービスにアクセスした日時が表示されます。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。続行するには、[はい、削除します] を選択します。
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監視します。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、削除のためにサービスにリンクされたロールを送信すると、削除タスクが成功または失敗する可能性があります。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。そのロールで使用中のリソースがサービスにあるために削除に失敗した場合、この失敗の理由にはリソースのリストも含まれます。

サービスにリンクされたロールの削除 (IAM 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とき、そのリクエストは拒否される場合があります。

サービスにリンクされたロールを削除するには (CLI)

1. 削除タスクのステータスを確認するには、レスポンスから `deletion-task-id` を取得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 AWSServiceRoleForEMRCleanup
```

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。

サービスにリンクされたロールの削除 (IAM API)

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とき、そのリクエストは拒否される場合があります。

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、 を呼び出しますDeleteServiceLinkedRole。リクエストで、 AWSServiceRoleForEMRCleanup ロール名を指定します。

   削除タスクのステータスを確認するには、レスポンスから DeletionTaskId を取得する必要があります。
2. 削除のステータスを確認するには、 を呼び出しますGetServiceLinkedRoleDeletionStatus。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。

でサポートされているリージョン AWSServiceRoleForEMRCleanup

Amazon EMR は、以下のリージョンで AWSServiceRoleForEMRCleanup 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。

リージョン名	リージョン識別子	Amazon EMR でのサポート
米国東部 (バージニア北部)	us-east-1	はい
米国東部 (オハイオ)	us-east-2	はい
米国西部 (北カリフォルニア)	us-west-1	はい
米国西部 (オレゴン)	us-west-2	はい
アジアパシフィック (ムンバイ)	ap-south-1	はい
アジアパシフィック (大阪)	ap-northeast-3	はい
アジアパシフィック (ソウル)	ap-northeast-2	はい
アジアパシフィック (シンガポール)	ap-southeast-1	はい
アジアパシフィック (シドニー)	ap-southeast-2	はい
アジアパシフィック (東京)	ap-northeast-1	はい
カナダ (中部)	ca-central-1	はい
欧州 (フランクフルト)	eu-central-1	はい
欧州 (アイルランド)	eu-west-1	はい
欧州 (ロンドン)	eu-west-2	はい
欧州 (パリ)	eu-west-3	はい
南米 (サンパウロ)	sa-east-1	はい

先行書き込みログ記録にサービスにリンクされたロールを使用する

Amazon EMR 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、Amazon EMR に直接リンクされた特殊な IAM ロールです。サービスにリンクされたロールは Amazon EMR によって事前定義されており、ユーザーに

代わってサービスから他の AWS のサービスを呼び出すために必要なすべてのアクセス許可が含まれています。

サービスにリンクされたロールは、Amazon EMR のサービスロールおよび Amazon EMR の Amazon EC2 インスタンスプロファイルと連携します。サービスロールとインスタンスプロファイルに関する詳細については、「AWS のサービスおよびリソースへのアクセス許可を Amazon EMR に付与する IAM サービスロールの設定」を参照してください。

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、Amazon EMR の設定が簡単になります。Amazon EMR は、サービスにリンクされたロールのアクセス許可を定義します。特に定義されている場合を除き、Amazon EMR のみがそのロールを引き受けることができます。定義したアクセス許可には、信頼ポリシーと許可ポリシーが含まれます。こ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。

Amazon EMR のサービスにリンクされたロールは、関連するリソースを削除し、アカウント内のすべての EMR クラスターを終了した後にのみ削除できます。これにより、Amazon EMR リソースが保護されるため、リソースへのアクセス許可が誤って削除されることはありません。

先行書き込みログ (WAL) のサービスにリンクされたロールのアクセス許可

Amazon EMR は、サービスにリンクされたロールAWSServiceRoleForEMRWALを使用してクラスターのステータスを取得します。

AWSServiceRoleForEMRWAL サービスにリンクされたロールは、次のサービスを信頼してロールを引き受けます。

- `emrwal.amazonaws.com`

サービスにリンクされたロールのEMRDescribeClusterPolicyForEMRWALアクセス許可ポリシーにより、Amazon EMR 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。

- アクション: * 上で `DescribeCluster`

IAM エンティティ (この場合は Amazon EMR WAL) がサービスにリンクされたロールを作成、編集、または削除できるようにするアクセス許可を設定する必要があります。必要に応じて、インスタンスプロファイルのアクセス許可ポリシーに次のステートメントを追加します。

CreateServiceLinkedRole

IAM エンティティがサービスにリンクされたロールを作成 AWSServiceRoleForEMRWALできるようにするには

サービスにリンクされたロールを作成する必要のある IAM エンティティのアクセス権限ポリシーに、次のステートメントを追加します。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:CreateServiceLinkedRole",
        "iam:PutRolePolicy"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/emrwal.amazonaws.com*/
AWSServiceRoleForEMRWAL*",
    "Condition": {
        "StringLike": {
            "iam:AWSServiceName": [
                "emrwal.amazonaws.com",
                "elasticmapreduce.amazonaws.com.cn"
            ]
        }
    }
}
```

UpdateRoleDescription

IAM エンティティが AWSServiceRoleForEMRWAL 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

サービスにリンクされたロールの説明を編集する必要のある IAM エンティティのアクセス許可ポリシーに次のステートメントを追加します。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:UpdateRoleDescription"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/emrwal.amazonaws.com*/
AWSServiceRoleForEMRWAL*",
    "Condition": {
        "StringLike": {
```

```
            "iam:AWSServiceName": [
                "emrwal.amazonaws.com",
                "elasticmapreduce.amazonaws.com.cn"
            ]
        }
    }
}
```

DeleteServiceLinkedRole

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 AWSServiceRoleForEMRWALできるようにするには

サービスにリンクされたロールを削除する必要のある IAM エンティティのアクセス権限ポリシーに、次のステートメントを追加します。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:DeleteServiceLinkedRole",
        "iam:GetServiceLinkedRoleDeletionStatus"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/elasticmapreduce.amazonaws.com*/
AWSServiceRoleForEMRCleanup*",
    "Condition": {
        "StringLike": {
            "iam:AWSServiceName": [
                "emrwal.amazonaws.com",
                "elasticmapreduce.amazonaws.com.cn"
            ]
        }
    }
}
```

Amazon EMR のサービスリンクロールの作成

AWSServiceRoleForEMRWAL ロールを手動で作成する必要はありません。Amazon EMR は、EMRWAL CLI または で WAL ワークスペースを作成すると、こ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。または AWS CloudFormation、Amazon EMR WAL のワークスペースを設定したときに、サービスにリンクされたロールがまだ存在しない場合、HBase はサービスにリンクされたロールを作成します。サービスにリンクされたロールを作成するには、アクセス許可が必要です。IAM エンティティ (ユーザー、グループ、ロールなど) のアクセス許可ポリシーにこの機能を追

加するステートメントの例については、前のセクション「」を参照してください先行書き込みログ (WAL) のサービスにリンクされたロールのアクセス許可。

Amazon EMR のサービスリンクロールの編集

Amazon EMR では、 AWSServiceRoleForEMRWAL サービスにリンクされたロールを編集することはできません。サービスにリンクされたロールを作成した後は、さまざまなエンティティがサービスにリンクされたロールを参照する可能性があるため、サービスにリンクされたロールの名前を変更することはできません。ただし、IAM を使用してサービスにリンクされたロールの説明を編集することはできます。

サービスにリンクされたロールの説明の編集 (IAM コンソール)

サービスにリンクされたロールの説明は、IAM コンソールを使用して編集できます。

サービスにリンクされたロールの説明を編集するには (コンソール)

1. 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。
2. 変更するロールの名前を選択します。
3. [Role description] (ロールの説明) の右にある [Edit] (編集) を選択します。
4. ボックスに新しい説明を入力し、[Save changes] (変更の保存) を選択します。

サービスにリンクされたロールの説明の編集 (IAM 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の説明を編集できます。

サービスにリンクされたロールの説明を変更するには (CLI)

1. (オプション) ロールの現在の説明を表示するには、次のコマンドを使用します。

   ```
   $ aws iam get-role --role-name role-name
   ```

   CLI コマンドでは、ARN ではなくロール名を使用してロールを参照します。例えば、ロールの ARN が `arn:aws:iam::123456789012:role/myrole` である場合、そのロールを **myrole** と参照します。
2. サービスにリンクされたロールの説明を更新するには、次のいずれかのコマンドを使用します。

   ```
   $ aws iam update-role-description --role-name role-name --description description
   ```

サービスにリンクされたロールの説明の編集 (IAM API)

サービスにリンクされたロールの説明は、IAM API を使用して編集できます。

サービスにリンクされたロールの説明を変更するには (API)

1. (オプション) ロールの現在の説明を表示するには、次のコマンドを使用します。

   IAM API: GetRole
2. ロールの説明を更新するには、次のコマンドを使用します。

   IAM API: UpdateRoleDescription

Amazon EMR のサービスリンクロールの削除

サービスにリンクされたロールを必要とする機能やサービスを使用する必要がなくなった場合は、そのサービスにリンクされた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、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。

> **Note**
>
> AWSServiceRoleForEMRWAL ロールを削除してもログ先行書き込みオペレーションは影響を受けませんが、Amazon EMR は EMR クラスターの終了後に作成したログを自動的に削除しません。したがって、サービスにリンクされたロールを削除する場合は、Amazon EMR WAL ログを手動で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のクリーンアップ

IAM を使用してサービスにリンクされたロールを削除するには、まずそのロールにアクティブなセッションがないことを確認し、そのロールで使用されているリソースをすべて削除する必要があります。

サービスにリンクされたロールにアクティブなセッションがあるかどうかを、IAM コンソールで確認するには

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] を選択します。 AWSServiceRoleForEMRWAL ロールの名前 (チェックボックスではありません) を選択します。

3. 選択したロールの [概要] ページで [アクセスアドバイザー] を選択します。
4. [アクセスアドバイザー] タブで、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**
>
> Amazon EMR が AWSServiceRoleForEMRWAL ロールを使用しているかどうかが不明な場合は、サービスにリンクされたロールを削除できます。サービスがロールを使用している場合、削除は失敗し、サービスにリンクされたロールが使用されているリージョンを表示できます。サービスにリンクされたロールが使用されている場合は、セッションが終了するのを待ってから、サービスにリンクされたロール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のセッションを取り消すことはできません。

で使用される Amazon EMR リソースを削除するには AWSServiceRoleForEMRWAL

- アカウントのすべてのクラスターを終了します。詳細については、「クラスターを終了する」を参照してください。

サービスにリンクされたロールの削除 (IAM 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. 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。名前や行自体ではなく AWSServiceRoleForEMRWAL、 の横にあるチェックボックスをオンにします。
3. ページ上部にある ロールのアクション で 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。このデータには、選択した各ロールが最後に AWS サービスにアクセスした日時が表示されます。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。続行するには、[はい、削除します] を選択します。
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。タスクが失敗した場合は、通知から 詳細を表示または 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。そのロールで使用中のリソースがサービスにあるために削除に失敗した場合、この失敗の理由にはリソースのリストも含まれます。

サービスにリンクされたロールの削除 (IAM 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とき、そのリクエストは拒否される場合があります。

サービスにリンクされたロールを削除するには (CLI)

1. 削除タスクのステータスを確認するには、レスポンスから `deletion-task-id` を取得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 AWSServiceRoleForEMRWAL
```

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。

サービスにリンクされたロールの削除 (IAM API)

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とき、そのリクエストは拒否される場合があります。

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、 を呼び出しますDeleteServiceLinkedRole。リクエストで、 AWSServiceRoleForEMRWAL ロール名を指定します。

削除タスクのステータスを確認するには、レスポンスから DeletionTaskId を取得する必要があります。

2. 削除のステータスを確認するには、 を呼び出しますGetServiceLinkedRoleDeletionStatus。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。

でサポートされているリージョン AWSServiceRoleForEMRWAL

Amazon EMR では、以下のリージョンで AWSServiceRoleForEMRWAL サービスにリンクされたロールの使用がサポートされています。

リージョン名	リージョン識別子	Amazon EMR でのサポート
米国東部 (バージニア北部)	us-east-1	はい
米国東部 (オハイオ)	us-east-2	はい
米国西部 (北カリフォルニア)	us-west-1	はい
米国西部 (オレゴン)	us-west-2	はい
アジアパシフィック (ムンバイ)	ap-south-1	あり
アジアパシフィック (シンガポール)	ap-southeast-1	はい
アジアパシフィック (シドニー)	ap-southeast-2	はい
アジアパシフィック (東京)	ap-northeast-1	あり
欧州 (フランクフルト)	eu-central-1	はい
欧州 (アイルランド)	eu-west-1	あり

IAM ロールのカスタマイズ

IAM サービスロールおよびアクセス許可をカスタマイズして、セキュリティ要件に応じて権限を制限することもできます。アクセス許可をカスタマイズするには、新しいロールおよびポリシーを作成することをお勧めします。まず、デフォルトのロール (`AmazonElasticMapReduceforEC2Role` や `AmazonElasticMapReduceRole`) の管理ポリシーのアクセス許可を確認します。次に、内容をコピーして新しいポリシーステートメントに貼り付け、必要に応じてアクセス許可を変更します。変更したアクセス許可ポリシーを、作成したロールにアタッチします。ロールおよびポリシーを操作するには、IAM の適切なアクセス許可が必要です。詳細については、「ユーザーおよびグループによるロールの作成および変更を許可する」を参照してください。

EC2 のカスタム EMR ロールを作成する場合は、基本的なワークフローに従います。これにより、同じ名前のインスタンスプロファイルが自動的に作成されます。Amazon EC2 では、異なる名前のインスタンスプロファイルとロールを作成できますが、Amazon EMR はこの設定をサポートしていないため、クラスターの作成時に「無効なインスタンスプロファイル」エラーが発生します。

> **Important**
>
> サービス要件が変更されても、インラインポリシーは自動的に更新されません。インラインポリシーを作成およびアタッチすると、サービスの更新によって突然アクセス許可エラーが発生することがあるので注意してください。詳細については、「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」および「クラスター作成時にカスタムの IAM ロールを指定する」を参照してください。

IAM ロールの操作の詳細については、「IAM ユーザーガイド」の以下のトピックを参照してください。

- AWS サービスにアクセス許可を委任するロールの作成
- ロールの修正
- ロールの削除

クラスター作成時にカスタムの IAM ロールを指定する

クラスターを作成するときに、Amazon EMR のサービスロールや Amazon EC2 インスタンスプロファイルのロールを指定します。クラスターを作成するユーザーは、ロールを取得し、Amazon EMR インスタンスおよび EC2 インスタンスに割り当てるためのアクセス許可が必要です。アクセ

ス許可がない場合、[account is not authorized to call EC2] エラーが発生します。詳細については、「ユーザーおよびグループによるロールの作成および変更を許可する」を参照してください。

コンソールを使用してカスタムロールを指定する

クラスターを作成する際、[Advanced options] (詳細オプション) を使用して、Amazon EMR のカスタムサービスロール、EC2 インスタンスプロファイルのカスタムロール、およびカスタムの Auto Scaling ロールを指定することができます。[Quick options] を使用すると、EC2 インスタンスプロファイルのデフォルトのサービスロールおよびデフォルトロールが指定されます。詳細については、「Amazon EMR で使用される IAM サービスロール」を参照してください。

> **Note**
>
> Amazon EMR コンソールは、再設計され、使いやすくなりました。古いコンソールと新しいコンソールエクスペリエンスの違いについては、「Amazon EMR コンソール」を参照してください。

New console

新しいコンソールでカスタム IAM ロールを指定する方法

新しいコンソールでクラスターを作成する際には、Amazon EMR のサービスロールや EC2 インスタンスプロファイルのカスタムロールを指定する必要があります。詳細については、「Amazon EMR で使用される IAM サービスロール」を参照してください。

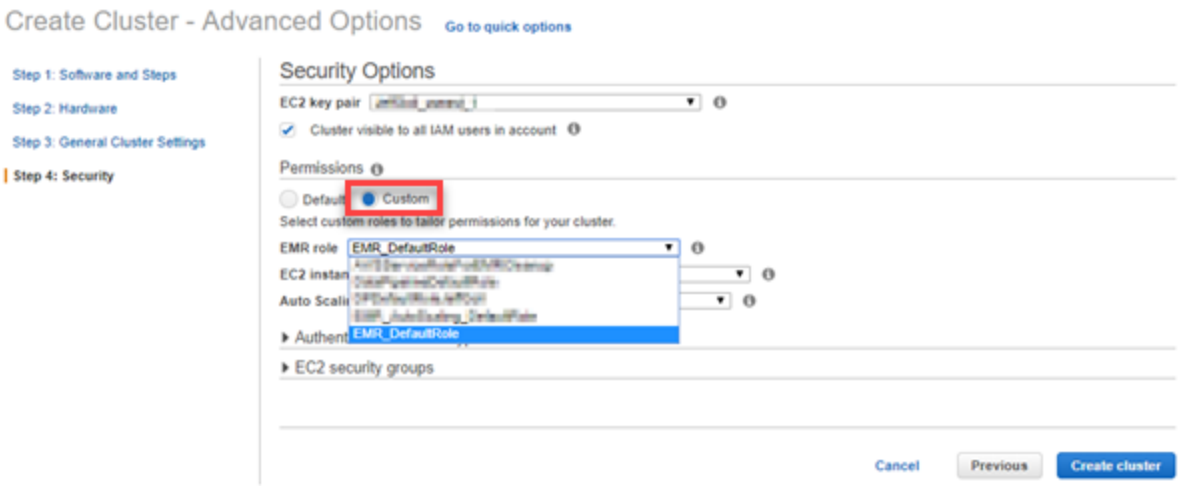
1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で、[クラスター] を選択し、[クラスターの作成] を選択します
3. [セキュリティの設定とアクセス許可] で、[インスタンスプロファイルの IAM ロール] フィールドと [Amazon EMR のサービスロール] フィールドを見つけます。各ロールタイプで、リストからロールを選択します。アカウント内のロールで、そのロールタイプに適切な信頼ポリシーが指定されているもののみ表示されます。
4. クラスターに適用するその他のオプションを選択します。
5. クラスターを起動するには、[クラスターの作成] を選択します。

Old console

古いコンソールでカスタム IAM ロールを指定する方法

古いコンソールでクラスターを作成する際には、[詳細オプション] を使用して、Amazon EMR のカスタムサービスロール、EC2 インスタンスプロファイルのカスタムロール、およびカスタム Auto Scaling ロールを指定することができます。[Quick options] を使用すると、EC2 インスタンスプロファイルのデフォルトのサービスロールおよびデフォルトロールが指定されます。詳細については、「Amazon EMR で使用される IAM サービスロール」を参照してください。

1. 新しい Amazon EMR コンソールに移動し、サイドナビゲーションから [古いコンソールに切り替え] を選択します。古いコンソールに切り替えたときの動作の詳細については、「Using the old console」を参照してください。
2. [Create cluster (クラスターの作成)]、[Go to advanced options (詳細オプションに移動する)] の順に選択します。
3. [Security Options] が表示されるまで、アプリケーションに適切なクラスター設定を選択します。[Permissions] (アクセス許可) で、Amazon EMR の [Default] (デフォルト) ロールが選択されています。
4. [Custom] を選択します。
5. 各ロールタイプで、リストからロールを選択します。アカウント内のロールで、そのロールタイプに適切な信頼ポリシーが指定されているもののみ表示されます。



6. 必要に応じて、クラスターの他のオプションを選択し、[Create Cluster] を選択します。

AWS CLI を使用してカスタムロールを指定する

AWS CLIから `create-cluster` コマンドとオプションを使用して、Amazon EMR のサービスロールとクラスター EC2 インスタンスのサービスロールを明示的に指定できます。`--service-role` オプションを使用してサービスロールを指定します。`--ec2-attributes` オプションの `InstanceProfile` 引数を使用して、EC2 インスタンスプロファイルのロールを指定します。

Auto Scaling ロールは、別のオプション `--auto-scaling-role` を使用して指定されます。詳細については、「カスタムポリシーによる自動スケーリングをインスタンスグループに使用する」を参照してください。

を使用してカスタム IAM ロールを指定するには AWS CLI

- 以下のコマンドは、クラスター起動時に、EC2 インスタンスプロファイルのカスタムのサービスロール (*MyCustomServiceRoleForEMR*) およびカスタムロール (*MyCustomServiceRoleForClusterEC2Instances*) を指定します。この例では、デフォルトの Amazon EMR ロールを使用します。

  > **Note**
  >
  > 読みやすくするために、Linux 行連続文字 (\) が含まれています。Linux コマンドでは、これらは削除することも、使用することもできます。Windows の場合、削除するか、キャレット (^) に置き換えてください。

```
aws emr create-cluster --name "Test cluster" --release-label emr-7.1.0 \
--applications Name=Hive Name=Pig --service-role MyCustomServiceRoleForEMR \
--ec2-attributes InstanceProfile=MyCustomServiceRoleForClusterEC2Instances,\
KeyName=myKey --instance-type m5.xlarge --instance-count 3
```

`--use-default-roles` オプションではなく、これらのオプションを使用して、明示的にデフォルトロールを指定できます。`--use-default-roles` オプションは、サービスロールを指定します。また、 AWS CLIの `config` ファイルに定義されている EC2 インスタンスプロファイルのロールを指定します。

次の例は、Amazon EMR のカスタムロール AWS CLI を指定する の `config` ファイルの内容を示しています。この設定ファイルで `--use-default-roles` オプションを指定すると、*MyCustomServiceRoleForEMR* と *MyCustomServiceRoleForClusterEC2Instances*

を使用してクラスターが作成されます。デフォルトでは、config ファイルはデフォルトの service_role として AmazonElasticMapReduceRole を指定し、デフォルトの instance_profile として EMR_EC2_DefaultRole を指定します。

```
[default]
output = json
region = us-west-1
aws_access_key_id = myAccessKeyID
aws_secret_access_key = mySecretAccessKey
emr =
     service_role = MyCustomServiceRoleForEMR
     instance_profile = MyCustomServiceRoleForClusterEC2Instances
```

Amazon S3 への EMRFS リクエストの IAM ロールを設定する

> **Note**
>
> このページで説明している EMRFS ロールマッピング機能は、Amazon EMR 6.15.0 で Amazon S3 Access Grants が導入されたことで改善されました。Amazon S3 のデータに対するスケーラブルなアクセスコントロールソリューションとして、Amazon EMR での S3 Access Grants の使用をお勧めします。

クラスターで実行されているアプリケーションが s3://*mydata* 形式を使用してデータを参照する場合、Amazon EMR は EMRFS を使用してリクエストを行います。EMRFS は Amazon S3 と対話するために、Amazon EC2 インスタンスプロファイルにアタッチされたアクセス許可ポリシーを引き受けます。アプリケーションを実行するユーザーやグループ、または Amazon S3 内のデータの場所に関係なく、同じ Amazon EC2 インスタンスプロファイルが使用されます。

クラスターに複数のユーザーがあり、各ユーザーに EMRFS を介して Amazon S3 のデータへの異なるレベルのアクセスが必要な場合は、EMRFS の IAM ロールを使用してセキュリティ設定を設定できます。EMRFS は、リクエスト元のユーザーやグループ、または Amazon S3 内のデータの場所に応じて、クラスター EC2 インスタンスの異なるサービスロールを引き受けることができます。EMRFS の IAM ロールごとに、Amazon S3 のデータにアクセスするための異なるアクセス許可を持つことができます。クラスター EC2 インスタンスのサービスロールの使用の詳細については、「クラスター EC2 インスタンスのサービスロール (EC2 インスタンスプロファイル)」を参照してください。

EMRFS のカスタム IAM ロールの使用は、Amazon EMR バージョン 5.10.0 以降でサポートされます。以前のバージョンを使用しているか、EMRFS の IAM ロールが提供するもの以上の要件がある場合は、代わりにカスタム認証情報プロバイダーを作成することができます。詳細については、「Authorizing access to EMRFS data in Amazon S3」を参照してください。

セキュリティ設定を使用して EMRFS の IAM を指定する場合は、ロールマッピングを設定します。各ロールマッピングは、識別子に対応する IAM ロールを指定します。これらの識別子は、EMRFS を介して Amazon S3 にアクセスするための基準を決定します。識別子は、ユーザー、グループ、またはデータの場所を示す Amazon S3 のプレフィックスです。EMRFS から Amazon S3 に行ったリクエストがアクセスの基準に一致すると、EMRFS では、リクエストに対応する IAM ロールをクラスター EC2 インスタンスに引き受けさせます。クラスター EC2 インスタンスのサービスロールにアタッチされた IAM アクセス許可の代わりに、そのロールにアタッチされた IAM アクセス許可が適用されます。

ロールマッピングのユーザーやグループは、クラスターで定義されている Hadoop ユーザーおよびグループです。ユーザーおよびグループは、EMRFS を使用するアプリケーション (例: YARN ユーザーの偽装) のコンテキストで EMRFS に渡されます。Amazon S3 プレフィックスは、任意の深さのバケット指定子 (例: `s3://mybucket`、`s3://mybucket/myproject/mydata`) にすることができます。1 つのロールマッピング内で複数の識別子を指定できますが、すべてを同じタイプにする必要があります。

> **Important**
>
> EMRFS の IAM ロールは、アプリケーションのユーザー間でのアプリケーションレベルの分離をもたらします。ホスト上のユーザー間でのホストレベルの分離は提供されません。クラスターのアクセス権を持つユーザーは、ロールを引き受けるための分離をバイパスできます。

クラスターアプリケーションが EMRFS を介して Amazon S3 にリクエストを行うと、EMRFS は、セキュリティ設定に表示されるトップダウン順序でロールマッピングを評価します。EMRFS を介して行われた要求がどの識別子とも一致しない場合、EMRFS はクラスター EC2 インスタンスのサービスロールを使用するようフォールバックします。このため、このロールにアタッチされたポリシーで Amazon S3 へのアクセス許可を制限することをお勧めします。詳細については、「クラスター EC2 インスタンスのサービスロール (EC2 インスタンスプロファイル)」を参照してください。

ロールを設定する

EMRFS の IAM ロールを使用してセキュリティ設定をセットアップする前に、ロールおよびアクセス許可のポリシーを計画して作成し、ロールにアタッチします。詳細については、「IAM ユーザーガイド」の「EC2 インスタンスのロールの仕組み」を参照してください。アクセス許可ポリシーを作成する際は、まず EC2 のデフォルトの Amazon EMR ロールにアタッチする管理ポリシーを作成し、次にこのポリシーを要件に応じて編集することが推奨されます。デフォルトのロール名は EMR_EC2_DefaultRole で、編集するデフォルトの管理ポリシーは AmazonElasticMapReduceforEC2Role です。詳細については、「クラスター EC2 インスタンスのサービスロール (EC2 インスタンスプロファイル)」を参照してください。

ロールアクセス許可を引き受けるための信頼ポリシーの更新

EMRFS が使用する各ロールには、EC2 に対するクラスターの Amazon EMR ロールがそのロールを引き受けることを許可する信頼ポリシーが必要です。同様に、EC2 に対するクラスターの Amazon EMR ロールには、EMRFS ロールがそのロールを引き受けること許可する信頼ポリシーが必要です。

次の例では、信頼ポリシーが EMRFS のロールにアタッチされています。ステートメントは、EC2 のデフォルトの Amazon EMR ロールがこのロールを引き受けることを許可します。たとえば、2 つの架空の EMRFS ロール (EMRFSRole_First と EMRFSRole_Second) がある場合、このポリシーステートメントは両方の信頼ポリシーにそれぞれ追加されます。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::AWSAcctID:role/EMR_EC2_DefaultRole"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

また、次の信頼ポリシー例は EMR_EC2_DefaultRole に追加されて、2 つの架空の EMRFS ロールを引き受けることを許可します。

```
{
```

```
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "AWS": ["arn:aws:iam::AWSAcctID:role/EMRFSRole_First",
 "arn:aws:iam::AWSAcctID:role/EMRFSRole_Second"]
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

IAM ロールの信頼ポリシーを更新するには

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

1. [ロール] を選択して [検索] にロール名を入力したら、その [ロール名] を選択します。
2. [Trust relationships] (信頼関係)、[Edit trust relationship] (信頼関係の編集) の順に選択します。
3. 上記のガイドラインに従って、[ポリシードキュメント] に応じて信頼ステートメントを追加し、[信頼ポリシーの更新] を選択します。

キーユーザーとしてロールを指定する

AWS KMS keyを使用して暗号化されている Amazon S3 の場所へのアクセスをロールで許可する場合は、そのロールがキーユーザーとして指定されていることを確認します。これにより、KMS キーを使用するアクセス許可がロールに付与されます。詳細については、「AWS Key Management Service デベロッパーガイド」の「AWS KMSでのキーポリシー」を参照してください。

EMRFS の IAM ロールを使用したセキュリティ設定のセットアップ

> **Important**
>
> 指定した EMRFS の IAM ロールのいずれも適用されない場合、EMRFS は EC2 の Amazon EMR ロールにフォールバックします。クラスターを作成するときに、このロールをカスタマイズして、アプリケーションに適切な Amazon S3 へのアクセス権限を制限し、EMR_EC2_DefaultRole の代わりにこのカスタムロールを指定することを検討してください。詳細については、「IAM ロールのカスタマイズ」および「クラスター作成時にカスタムの IAM ロールを指定する」を参照してください。

コンソールを使用して Amazon S3 への EMRFS リクエストの IAM ロールを指定するには

1. ロールのマッピングを指定するセキュリティ設定を作成します。

   a. Amazon EMR コンソールで [Security configurations] (セキュリティ設定) > [Create] (作成) を選択します。

   b. セキュリティ設定の [Name (名前)] を入力します。この名前を使用して、クラスターの作成時に使用するセキュリティ設定を指定します。

   c. [Use IAM roles for EMRFS requests to Amazon S3] (Amazon S3 への EMRFS のリクエストに IAM ロールを使用) を選択します。

   d. 適用する [IAM role] (IAM ロール) を選択し、[Basis for access] (アクセスの基礎) でリストから識別子タイプ ([Users] (ユーザー)、[Groups] (グループ)、または [S3 prefixes] (S3 プレフィックス) を選択して、対応する識別子を入力します。複数の識別子を使用する場合は、カンマでスペースを入れずに区切ります。各識別子の種類の詳細については、下記の「JSON configuration reference」を参照してください。

   e. [Add role (ロールの追加)] を選択して、前の手順で説明したように追加のロールマッピングを設定します。

   f. 必要に応じて他のセキュリティ設定オプションを設定し、[Create (作成)] を選択します。詳細については、「セキュリティ設定を作成する」を参照してください。

2. クラスターの作成時に上記で作成したセキュリティ設定を指定します。詳細については、「クラスターのセキュリティ設定を指定する」を参照してください。

を使用して Amazon S3 への EMRFS リクエストの IAM ロールを指定するには AWS CLI

1. `aws emr create-security-configuration` コマンドを使用して、セキュリティ設定の名前とセキュリティ設定の詳細を JSON 形式で指定します。

   次のコマンド例は、`EMRFS_Roles_Security_Configuration` という名前のセキュリティ設定を作成します。これは、`MyEmrfsSecConfig.json` ファイルの JSON 構造に基づいており、コマンドが実行されるのと同じディレクトリに保存されます。

   ```
   aws emr create-security-configuration --name EMRFS_Roles_Security_Configuration --
   security-configuration file://MyEmrFsSecConfig.json.
   ```

次の MyEmrFsSecConfig.json ファイルの構造のガイドラインを使用します。他のセキュリティ設定オプションの構造と共に、この構造を指定することができます。詳細については、「セキュリティ設定を作成する」を参照してください。

以下は、セキュリティ設定内の EMRFS のカスタム IAM ロールを指定するための JSON スニペットの例です。3 つの異なる識別子タイプのロールマッピングと、パラメータリファレンスを示します。

```
{
  "AuthorizationConfiguration": {
    "EmrFsConfiguration": {
      "RoleMappings": [{
        "Role": "arn:aws:iam::123456789101:role/allow_EMRFS_access_for_user1",
        "IdentifierType": "User",
        "Identifiers": [ "user1" ]
      },{
        "Role": "arn:aws:iam::123456789101:role/allow_EMRFS_access_to_MyBuckets",
        "IdentifierType": "Prefix",
        "Identifiers": [ "s3://MyBucket/","s3://MyOtherBucket/" ]
      },{
        "Role": "arn:aws:iam::123456789101:role/allow_EMRFS_access_for_AdminGroup",
        "IdentifierType": "Group",
        "Identifiers": [ "AdminGroup" ]
      }]
    }
  }
}
```

パラメータ	説明
"AuthorizationConfiguration":	必須。
"EmrFsConfiguration":	必須。ロールマッピングが含まれます。

パラメータ	説明
"RoleMappings":	必須。1 つ以上のロールマッピング定義が含まれます。ロールのマッピングは、表示順に上から下に評価されます。Amazon S3 のデータに対する EMRFS 呼び出しでロールマッピングが true と評価された場合、それ以上のロールマッピングは評価されず、EMRFS はリクエストに指定された IAM ロールを使用します。ロールマッピングは、以下の必須パラメータで構成されます。
"Role":	IAM ロールの ARN 識別子を形式 arn:aws:iam:: *account-id* :role/*role-name* で指定します。これは、Amazon S3 への EMRFS リクエストが、指定された Identifiers のいずれかに一致した場合に Amazon EMR が引き受ける IAM ロールです。

パラメータ	説明
"IdentifierType":	次のいずれかを指定できます。 • "User" は、識別子が 1 人以上の Hadoop ユーザーであることを指定します。これは Linux アカウントユーザーまたは Kerberos プリンシパルです。EMRFS リクエストが指定のユーザーから送信されると、IAM ロールが引き受けられます。 • "Prefix" は、識別子が Amazon S3 の場所であることを指定します。IAM ロールは、指定されたプレフィックスを持つ場所への呼び出しに対して引き受けられます。例えば、プレフィックス s3://mybucket/ は、 s3://mybucket/mydir および s3://mybucket/yetanotherdir に一致します。 • "Group" は、識別子が 1 つ以上の Hadoop グループであることを指定します。IAM ロールは、リクエストが指定されたグループのユーザーから発信された場合に引き受けられます。
"Identifiers":	適切な識別子タイプの 1 つ以上の識別子を指定します。複数の識別子は、スペースを入れずにカンマで区切ります。

2. aws emr create-cluster コマンドを使用してクラスターを作成し、前のステップで作成したセキュリティ設定を指定します。

   次の例では、Hadoop アプリケーションがインストールされているデフォルトのコアでクラスターを作成します。クラスターは、上記で EMRFS_Roles_Security_Configuration として作成されたセキュリティ設定を使用し、さらに EC2 のカスタム Amazon EMR ロール EC2_Role_EMR_Restrict_S3 (--ec2-attributes パラメータの InstanceProfile 引数を使用して指定) も使用します。

> **Note**
>
> 読みやすくするために、Linux 行連続文字 (\) が含まれています。Linux コマンドでは、これらは削除することも、使用することもできます。Windows の場合、削除するか、キャレット (^) に置き換えてください。

```
aws emr create-cluster --name MyEmrFsS3RolesCluster \
--release-label emr-7.1.0 --ec2-attributes
 InstanceProfile=EC2_Role_EMR_Restrict_S3,KeyName=MyKey \
--instance-type m5.xlarge --instance-count 3 \
--security-configuration EMRFS_Roles_Security_Configuration
```

AWS Glue Data Catalog への Amazon EMR アクセスのリソースベースのポリシーを使用する

Amazon EMR で AWS Glue を Hive、Spark、または Presto と組み合わせて使用する場合、 AWS Glue は Data Catalog リソースへのアクセスを制御するためのリソースベースのポリシーをサポートします。これらのリソースには、データベース、テーブル、接続、ユーザー定義関数が含まれます。詳細については、「AWS Glue デベロッパーガイド」の「AWS Glue リソースポリシー」を参照してください。

リソースベースのポリシーを使用して Amazon EMR 内から AWS Glue へのアクセスを制限する場合、アクセス許可ポリシーで指定するプリンシパルは、クラスターの作成時に指定する EC2 インスタンスプロファイルに関連付けられたロール ARN である必要があります。例えば、カタログにアタッチされたリソースベースのポリシーの場合、次の例に示す形式を使用して、クラスター EC2 インスタンスのデフォルトのサービスロールのロール ARNPrincipal、*EMR_EC2_DefaultRole* をとして指定できます。

```
arn:aws:iam::acct-id:role/EMR_EC2_DefaultRole
```

acct-id は AWS Glue アカウント ID とは異なる場合があります。これにより、さまざまなアカウントで EMR クラスターからアクセスできます。異なるアカウントから、複数のプリンシパルを指定できます。

AWS のサービスを直接呼び出すアプリケーションで IAM ロールを使用する

クラスターの EC2 インスタンスで実行されているアプリケーションは、EC2 インスタンスプロファイルを使用して、 AWS サービスを呼び出すときに一時的なセキュリティ認証情報を取得できます。

Amazon EMR のリリース 2.3.0 以降で使用できる Hadoop のバーションは、IAM ロールを利用できるように更新されています。アプリケーションが Hadoop アーキテクチャ上で厳密に実行され、 のどのサービスも直接呼び出さない場合は AWS、変更を加えずに IAM ロールと連携する必要があります。

アプリケーションで のサービス AWS を直接呼び出す場合は、IAM ロールを利用するためにサービスを更新する必要があります。これにより、アプリケーションは、クラスターの EC2 インスタンスの `/etc/hadoop/conf/core-site.xml` からアカウント認証情報を取得する代わりに、SDK を利用して IAM ロールを使うリソースにアクセスしたり、EC2 インスタンスメタデータを呼び出して一時的な認証情報を取得したりできます。

SDK を使用して IAM ロールで AWS リソースにアクセスするには

- 以下のトピックでは、いくつかの AWS SDKs IAM ロールを使用して一時的な認証情報にアクセスする方法を示します。各トピックは IAM ロールを使用しないアプリケーションのバージョンから始まり、IAM ロールを使用するためにアプリケーションを変換する処理を説明します。
  - 「AWS SDK for Java デベロッパーガイド」の「Java 用 SDK を使用した Amazon EC2 インスタンスの IAM ロールの設定」
  - 「AWS SDK for .NET デベロッパーガイド」の「Using IAM roles for Amazon EC2 instances with the SDK for .NET」
  - 「AWS SDK for PHP デベロッパーガイド」の「PHP 用 SDK を使用した Amazon EC2 インスタンスの IAM ロールの設定」
  - 「AWS SDK for Ruby デベロッパーガイド」の「Ruby 用 SDK を使用した Amazon EC2 インスタンスの IAM ロールの設定」

EC2 インスタンスメタデータから一時的な認証情報を取得するには

- 指定された IAM ロールで実行されている EC2 インスタンスから次の URL を呼び出します。これにより、関連する一時的なセキュリティ認証情報 (AccessKeyId、 SecretAccessKey SessionToken、および有効期限) が返されます。次の例では Amazon EMR のデフォルトのインスタンスプロファイル (`EMR_EC2_DefaultRole`) を使用しています。

```
GET http://169.254.169.254/latest/meta-data/iam/security-
credentials/EMR_EC2_DefaultRole
```

IAM ロールを使用するアプリケーションの記述の詳細については、Amazon EC2 インスタンスで実行されるアプリケーションに AWS リソースへのアクセス権を付与する」を参照してください。

一時的なセキュリティ認証情報の詳細については、「一時的セキュリティ認証情報の使用」ガイドの「一時的な認証情報の使用」を参照してください。

ユーザーおよびグループによるロールの作成および変更を許可する

IAM プリンシパル (ユーザーおよびグループ) が、デフォルトロールなど、クラスターのロールを作成、変更、指定するには、次のアクションの実行権限が必要です。各アクションの詳細については、「IAM API リファレンス」の「アクション」を参照してください 。

- `iam:CreateRole`
- `iam:PutRolePolicy`
- `iam:CreateInstanceProfile`
- `iam:AddRoleToInstanceProfile`
- `iam:ListRoles`
- `iam:GetPolicy`
- `iam:GetInstanceProfile`
- `iam:GetPolicyVersion`
- `iam:AttachRolePolicy`
- `iam:PassRole`

`iam:PassRole` アクセス権限では、クラスターを作成できます。残りのアクセス権限では、デフォルトのロールを作成できます。

ユーザーへのアクセス許可の割り当てについては、「IAM ユーザーガイド」の「ユーザーのアクセス許可の変更」を参照してください。

Amazon EMR のアイデンティティベースポリシーの例

デフォルトでは、ユーザーおよびロールには Amazon EMR リソースを作成または変更するアクセス許可はありません。また、、 AWS Management Console AWS CLI、または AWS API を使用してタスクを実行することはできません。IAM 管理者は、ユーザーとロールに必要な、指定されたリソースで特定の API オペレーションを実行する権限をユーザーとロールに付与する IAM ポリシーを作成する必要があります。次に、管理者は、それらのアクセス許可を必要とするユーザーまたはグループにそのポリシーをアタッチする必要があります。

これらの JSON ポリシードキュメント例を使用して IAM のアイデンティティベースのポリシーを作成する方法については、『IAM ユーザーガイド』の「JSON タブでのポリシーの作成」を参照してください。

トピック

- Amazon EMR のポリシーのベストプラクティス
- 自分の権限の表示をユーザーに許可する
- Amazon EMR 管理ポリシー
- クラスターおよび EMR Notebooks に対するタグベースのアクセスの IAM ポリシー
- ModifyInstanceGroup アクションの拒否
- Amazon EMR の ID とアクセスのトラブルシューティング

Amazon EMR のポリシーのベストプラクティス

アイデンティティベースポリシーは非常に強力です。これは、アカウント内で Amazon EMR のリソースを作成、アクセス、または削除できるユーザーを決定します。これらのアクションでは、AWS アカウントのコストが発生する可能性があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

- AWS 管理ポリシーを使用して開始する – Amazon EMR の使用をすばやく開始するには、 AWS 管理ポリシーを使用して、従業員に必要なアクセス許可を付与します。これらのポリシーはアカウントで既に有効になっており、 AWSによって管理および更新されています。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によるアクセス許可の使用を開始する」および「」を参照してくださいAmazon EMR 管理ポリシー。
- 最小特権を付与する - カスタムポリシーを作成するときは、タスクを実行するために必要なアクセス許可のみを付与します。最小限の許可からスタートし、必要に応じて追加の許可を付与します。

この方法は、寛容過ぎる許可から始めて、後から厳しくしようとするよりも安全です。詳細については、IAM ユーザーガイドの「最小特権を認める」を参照してください。

- 機密性の高いオペレーションに MFA を有効にする – 追加セキュリティとして、機密性の高いリソースまたは API オペレーションにアクセスするユーザーに対して、多要素認証 (MFA) の使用を要求します。詳細については、IAM ユーザーガイドの「AWSでの多要素認証 (MFA) の使用」を参照してください。
- 追加セキュリティに対するポリシー条件を使用する - 実行可能な範囲内で、アイデンティティベースのポリシーがリソースにアクセスできる条件を定義します。例えば、あるリクエストの送信が許可される IP アドレスの範囲を指定するための条件を記述できます。指定された日付または時間範囲内でのみリクエストを許可する条件を書くことも、SSL や MFA の使用を要求することもできます。詳細については、「IAM ユーザーガイド」の「IAM JSON ポリシー要素: 条件」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 ID にアタッチされたインラインポリシーおよび管理ポリシーの表示をユーザーに許可するポリシーを作成する方法を示します。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:GetUser",
            "iam:GetUserPolicy",
            "iam:ListAttachedUserPolicies",
            "iam:ListGroupsForUser",
            "iam:ListUserPolicies"
         ],
         "Resource":[
            "arn:aws:iam::*:user/${aws:username}"
         ]
      },
      {
         "Sid":"NavigateInConsole",
```

```
         "Effect":"Allow",
         "Action":[
            "iam:GetGroupPolicy",
            "iam:GetPolicy",
            "iam:GetPolicyVersion",
            "iam:ListAttachedGroupPolicies",
            "iam:ListGroupPolicies",
            "iam:ListGroups",
            "iam:ListPolicies",
            "iam:ListPolicyVersions",
            "iam:ListUsers"
         ],
         "Resource":"*"
      }
   ]
}
```

Amazon EMR 管理ポリシー

必要な Amazon EMR アクションにフルアクセスまたは読み取り専用アクセスを付与する最も簡単な方法は、Amazon EMR の IAM 管理ポリシーを使用することです。管理ポリシーは、アクセス権限の条件が変更された場合に、自動的に更新されるというメリットを提供します。インラインポリシーを使用する場合は、サービスの変更によってアクセス許可エラーが発生する場合があります。

Amazon EMR は、新しい管理ポリシー (v2 ポリシー) を優先して、既存の管理ポリシー (v1 ポリシー) を廃止します。新しい マネージドポリシーは、 AWS ベストプラクティスに合わせてスコープダウンされています。既存の v1 管理ポリシーが廃止されると、これらのポリシーを新しい IAM ロールまたはユーザーにアタッチできなくなります。廃止されたポリシーを使用している既存のロールおよびユーザーは、それらを引き続き使用できます。v2 管理ポリシーは、タグを使用してアクセスを制限します。指定された Amazon EMR アクションのみが許可され、EMR 固有のキーでタグ付けされたクラスターリソースが必要です。新しい v2 ポリシーを使用する前に、ドキュメントを慎重に確認することをお勧めします。

v1 ポリシーには非推奨のマークが付けられ、IAM コンソールの [Policies] (ポリシー) リストの横に警告アイコンとともに表示されます。非推奨ポリシーには、次の特徴があります。

- 現在アタッチされているすべてのユーザー、グループ、およびロールで引き続き機能します。中断される処理はありません。
- 新しいユーザー、グループ、またはロールにアタッチすることはできません。現在のエンティティからポリシーのいずれかをデタッチした場合、再アタッチすることはできません。

- 現在のすべてのエンティティから v1 ポリシーをデタッチすると、ポリシーは表示されなくなり、使用できなくなります。

次の表は、現在のポリシー (v1) ポリシーと v2 ポリシーの変更点をまとめたものです。

Amazon EMR マネージドポリシーの変更

ポリシータイプ	ポリシー名	ポリシーの目的	v2 ポリシーへの変更
デフォルトの EMR サービスロールとアタッチされた管理ポリシー	ロール名: EMR_DefaultRole V1 ポリシー (廃止予定): AmazonElasticMapReduceRole (EMR サービスロール) V2 (範囲制限) ポリシー名: **AmazonEMRServicePolicy_v2**	リソースをプロビジョニングし、 AWS サービスレベルのアクションを実行するときに、Amazon EMR がユーザーに代わって他の サービスを呼び出すことを許可します。このロールは、すべてのクラスターに必須です。	ポリシーは新しいアクセス許可 を追加します`"ec2:DescribeInstanceTypeOfferings"` 。この API オペレーションは、特定のアベイラビリティーゾーンのリストでサポートされているインスタンスタイプのリストを返します。
アタッチされたユーザー、ロール、またはグループによる完全な Amazon EMR アクセスのための IAM 管理ポリシー	V2 (範囲制限) ポリシー名: **AmazonEMRServicePolicy_v2**	EMR アクションに対する完全なアクセス許可をユーザーに許可します。リソースの iam:PassRole permissions が含まれます。	ポリシーは、ユーザーがこのポリシーを使用する前にリソースにユーザータグを追加する必要があるという前提条件を追加します。管理ポリシーを使用するためにリソースにタグを付ける を参照してください。 iam:PassRole action には、iam:PassedToService

ポリシータイプ	ポリシー名	ポリシーの目的	v2 ポリシーへの変更
			condition を指定されたサービスに設定する必要があります。デフォルトでは、Amazon EC2、Amazon S3、およびその他のサービスへのアクセスは許可されていません。「フルアクセス用の IAM 管理ポリシー (v2 管理デフォルトポリシー)」を参照してください。
アタッチされたユーザー、ロール、またはグループによる読み取り専用アクセスのための IAM 管理ポリシー	V1 ポリシー (非推奨化予定): **AmazonElasticMapReduceReadOnlyAccess** V2 (範囲制限) ポリシー名: **AmazonEMRReadOnlyAccessPolicy_v2**	Amazon EMR アクションに対する読み取り専用アクセス許可をユーザーに許可します。	アクセス許可によって、指定された elasticmapreduce の読み取り専用アクションのみが許可されます。Amazon S3 へのアクセスは、デフォルトではアクセスが許可されていません。「読み取り専用アクセス用の IAM 管理ポリシー (v2 管理デフォルトポリシー)」を参照してください。

ポリシータイプ	ポリシー名	ポリシーの目的	v2 ポリシーへの変更
デフォルトの EMR サービスロールとアタッチされた管理ポリシー	ロール名: EMR_DefaultRole V1 ポリシー (廃止予定): AmazonElasticMapReduceRole (EMR サービスロール) V2 (範囲制限) ポリシー名: **AmazonEMRServicePolicy_v2**	リソースをプロビジョニングし、 AWS サービスレベルのアクションを実行するときに、Amazon EMR がユーザーに代わって他の サービスを呼び出すことを許可します。このロールは、すべてのクラスターに必須です。	v2 サービスロールと v2 デフォルトポリシーは、非推奨のロールとポリシーを置き換えます。このポリシーは、ユーザーがこのポリシーを使用する前にリソースにユーザータグを追加する必要があるという前提条件を追加します。「管理ポリシーを使用するためにリソースにタグを付ける」を参照してください「Amazon EMR のサービスロール (EMR ロール)」を参照してください

ポリシータイプ	ポリシー名	ポリシーの目的	v2 ポリシーへの変更
クラスター EC2 インスタンスのサービスロール (EC2 インスタンスプロファイル)	V1 ポリシー (廃止予定): EMR_EC2_DefaultRole (インスタンスプロファイル) 廃止されたポリシー名: AmazonElasticMapReduceforEC2Role	EMR クラスターで実行されているアプリケーションが、Amazon S3 などの他の AWS リソースにアクセスできるようにします。例えば、Amazon S3 からのデータを処理する Apache Spark ジョブを実行する場合、ポリシーでそのようなリソースへのアクセスを許可する必要があります。	デフォルトロールとデフォルトポリシーの両方が非推奨予定です。代替の AWS デフォルトのマネージドロールやポリシーはありません。リソースベースまたは ID ベースのポリシーを提供する必要があります。つまり、デフォルトでは、EMR クラスターで実行されているアプリケーションは、ポリシーに手動で追加しない限り、Amazon S3 やその他のリソースにアクセスできません。デフォルトのロールとデフォルトの管理ポリシー を参照してください。

ポリシータイプ	ポリシー名	ポリシーの目的	v2 ポリシーへの変更
その他の EC2 サービスロールポリシー	現在のポリシー名: AmazonElasticMapReduceforAutoScalingRole、AmazonElasticMapReduceEditorsRole、AmazonEMRCleanupPolicy	自動スケーリング、ノートブック、または EC2 リソースのクリーンアップを使用する場合に、Amazon EMR が他の AWS リソースにアクセスしてアクションを実行するために必要なアクセス許可を提供します。	v2 で変更はありません。

iam の保護：PassRole

Amazon EMR のフルアクセス許可のデフォルトの管理ポリシーには、次を含む `iam:PassRole` セキュリティ設定が組み込まれています。

- 特定のデフォルトの Amazon EMR ロール専用の `iam:PassRole` アクセス許可。
- `iam:PassedToService` や などの指定された AWS サービスでのみポリシーを使用できるようにする `elasticmapreduce.amazonaws.com` 条件`ec2.amazonaws.com`。

AmazonEMRFullAccessPolicy_v2 および AmazonEMRServicePolicy_v2 ポリシーの JSON バージョンは、IAM コンソールで表示できます。v2 管理ポリシーを使用して新しいクラスターを作成することが推奨されます。

カスタムポリシーを作成するには、管理ポリシーから始め、条件にしたがって編集することをお勧めします。

ユーザー (プリンシパル) にポリシーをアタッチする方法については、「IAM ユーザーガイド」の「Working with managed policies using the AWS Management Console」を参照してください。

管理ポリシーを使用するためにリソースにタグを付ける

AmazonEMRServicePolicy_v2 AmazonEMRFullAccessPolicy _v2 は、Amazon EMR がプロビジョニングまたは使用するリソースへのスコープダウンアクセスに依存します。事前に定義されたユーザー

タグが関連付けられているリソースのみにアクセスを制限することで、範囲制限を行います。これらの 2 つのポリシーのいずれかを使用する場合は、クラスターをプロビジョニングする際に、事前定義のユーザータグ `for-use-with-amazon-emr-managed-policies = true` を渡す必要があります。Amazon EMR はそのタグを自動的に伝播します。さらに、次のセクションにリストされているリソースにユーザータグを追加する必要があります。Amazon EMR コンソールを使用してクラスターを起動する場合は、「Amazon EMR コンソールで v2 管理ポリシーを使用してクラスターを起動する際の考慮事項」を参照してください。

管理ポリシーを使用するには、CLI、SDK、またはその他の方法を使用してクラスターをプロビジョニングする際に、ユーザータグ `for-use-with-amazon-emr-managed-policies = true` を渡します。

タグを渡すと、Amazon EMR は、作成したプライベートサブネット ENI、EC2 インスタンス、および EBS ボリュームにタグを伝播します。Amazon EMR は、作成するセキュリティグループにも自動的にタグ付けします。ただし、特定のセキュリティグループで Amazon EMR を起動するには、タグを付ける必要があります。Amazon EMR によって作成されていないリソースの場合は、それらのリソースにタグを追加する必要があります。例えば、Amazon EC2 サブネット、EC2 セキュリティグループ (Amazon EMR によって作成されていない場合)、VPC (Amazon EMR でセキュリティグループを作成する場合) にタグを付ける必要があります。VPC で v2 管理ポリシーを使用してクラスターを起動するには、事前定義のユーザータグでそれらの VPC にタグを付ける必要があります。「Amazon EMR コンソールで v2 管理ポリシーを使用してクラスターを起動する際の考慮事項」を参照してください。

伝播されたユーザー指定のタグ付け

Amazon EMR は、クラスターの作成時に指定した Amazon EMR タグを使用して、作成したリソースにタグを付けます。Amazon EMR は、クラスターの存続期間中に作成したリソースにタグを適用します。

Amazon EMR は、次のリソースのユーザータグを伝播します。

- プライベートサブネット ENI (サービスアクセス Elastic Network Interface)
- EC2 インスタンス
- EBS ボリューム
- EC2 起動テンプレート

自動的にタグ付けされたセキュリティグループ

Amazon EMR は、create cluster コマンドで指定したタグに関係なく、Amazon EMR の v2 管理ポリシーに必要なタグ `for-use-with-amazon-emr-managed-policies` を使用して、作成した EC2 セキュリティグループにタグ付けします。v2 管理ポリシーの導入前に作成されたセキュリティグループの場合、Amazon EMR はセキュリティグループに自動的にタグを付けません。アカウントにすでに存在するデフォルトのセキュリティグループで v2 管理ポリシーを使用する場合は、`for-use-with-amazon-emr-managed-policies = true` を使用して、セキュリティグループに手動でタグを付ける必要があります。

手動でタグ付けされたクラスターリソース

Amazon EMR のデフォルトロールからアクセスできるように、一部のクラスターリソースに手動でタグ付けする必要があります。

- Amazon EMR 管理ポリシータグ `for-use-with-amazon-emr-managed-policies` を使用して、EC2 セキュリティグループと EC2 サブネットに手動でタグ付けする必要があります。
- Amazon EMR でデフォルトのセキュリティグループを作成する場合は、VPC に手動でタグ付けする必要があります。EMR は、デフォルトのセキュリティグループがまだ存在しない場合、特定のタグを使用してセキュリティグループを作成しようとします。

Amazon EMR は、次のリソースに自動的にタグ付けします。

- EMR が作成した EC2 セキュリティグループ

次のリソースに手動でタグ付けを行う必要があります。

- EC2 サブネット
- EC2 セキュリティグループ

必要に応じて、次のリソースに手動でタグ付けできます。

- VPC - Amazon EMR でセキュリティグループを作成する場合のみ

Amazon EMR コンソールで v2 管理ポリシーを使用してクラスターを起動する際の考慮事項

Amazon EMR コンソールで v2 管理ポリシーを使用してクラスターをプロビジョニングできます。コンソールを使用して Amazon EMR クラスターを起動する場合の考慮事項を以下にいくつか示します。

> **Note**
>
> Amazon EMR コンソールを再設計しました。新しいコンソールでは自動タグ機能はまだ使用できません。また、新しいコンソールでは、タグ付けが必要なリソース (VPC/サブネット) も表示されません。古いコンソールと新しいコンソールエクスペリエンスの違いの詳細については、「Amazon EMR コンソール」を参照してください。

- 事前定義のタグを渡す必要はありません。Amazon EMR はタグを自動的に追加し、適切なコンポーネントに伝播します。
- 手動でタグ付けする必要があるコンポーネントの場合、リソースにタグ付けするのに必要なアクセス許可を持っていれば、古い Amazon EMR コンソールがそれらのコンポーネントに自動的にタグ付けしようとします。リソースにタグ付けするためのアクセス許可を持っていない場合、または新しいコンソールを使用する場合は、管理者にリソースのタグ付けを依頼してください。
- すべての前提条件が満たされない限り、v2 管理ポリシーを使用してクラスターを起動することはできません。
- 古い Amazon EMR コンソールには、タグ付けする必要があるリソース (VPC/サブネット) が表示されます。

フルアクセス用の IAM 管理ポリシー (v2 管理デフォルトポリシー)

v2 範囲制限 EMR デフォルト管理ポリシーは、特定のアクセス権限をユーザーに付与します。事前定義された Amazon EMR リソースタグおよび Amazon EMR で使用されるリソース (クラスターの起動に使用する `Subnet` や `SecurityGroup` など) への `iam:PassRole` 条件キーが必要です。

Amazon EMR を対象とした必要なアクションを許可するには、`AmazonEMRFullAccessPolicy_v2` 管理ポリシーをアタッチします。この更新されたデフォルト管理ポリシーは、`AmazonElasticMapReduceFullAccess` 管理ポリシーを置き換えます。

`AmazonEMRFullAccessPolicy_v2` は、Amazon EMR がプロビジョニングまたは使用するリソースへの、範囲が制限されたアクセス権限に依存します。このポリシーを使用する場合は、クラスターのプロビジョニング時にユーザータグ `for-use-with-amazon-emr-managed-policies = true` を渡す必要があります。Amazon EMR はタグを自動的に伝播します。さらに、Amazon EMR によって作成されていない EC2 セキュリティグループなど、特定のタイプのリソースにユーザータグを手動で追加する必要がある場合があります。詳細については、「管理ポリシーを使用するためにリソースにタグを付ける」を参照してください。

AmazonEMRFullAccessPolicy_v2 ポリシーは、以下のようにしてリソースを保護します。

- クラスターの作成と Amazon EMR アクセス用に、事前定義された Amazon EMR 管理ポリシータグ `for-use-with-amazon-emr-managed-policies` を使用してリソースにタグを付ける必要があります。
- `iam:PassRole` アクションを特定のデフォルトロールに制限し、`iam:PassedToService` アクセスを特定のサービスに制限します。
- デフォルトでは、Amazon EC2、Amazon S3、およびその他のサービスへのアクセス権限は提供されなくなっています。

このポリシーの内容は次のとおりです。

> **Note**
>
> コンソールリンク AmazonEMRFullAccessPolicy_v2 を使用してポリシーを表示することもできます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RunJobFlowExplicitlyWithEMRManagedTag",
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:RunJobFlow"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/for-use-with-amazon-emr-managed-policies": "true"
                }
            }
        },
        {
            "Sid": "ElasticMapReduceActions",
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:AddInstanceFleet",
                "elasticmapreduce:AddInstanceGroups",
```

```
                "elasticmapreduce:AddJobFlowSteps",
                "elasticmapreduce:AddTags",
                "elasticmapreduce:CancelSteps",
                "elasticmapreduce:CreateEditor",
                "elasticmapreduce:CreateSecurityConfiguration",
                "elasticmapreduce:DeleteEditor",
                "elasticmapreduce:DeleteSecurityConfiguration",
                "elasticmapreduce:DescribeCluster",
                "elasticmapreduce:DescribeEditor",
                "elasticmapreduce:DescribeJobFlows",
                "elasticmapreduce:DescribeSecurityConfiguration",
                "elasticmapreduce:DescribeStep",
                "elasticmapreduce:DescribeReleaseLabel",
                "elasticmapreduce:GetBlockPublicAccessConfiguration",
                "elasticmapreduce:GetManagedScalingPolicy",
                "elasticmapreduce:GetAutoTerminationPolicy",
                "elasticmapreduce:ListBootstrapActions",
                "elasticmapreduce:ListClusters",
                "elasticmapreduce:ListEditors",
                "elasticmapreduce:ListInstanceFleets",
                "elasticmapreduce:ListInstanceGroups",
                "elasticmapreduce:ListInstances",
                "elasticmapreduce:ListSecurityConfigurations",
                "elasticmapreduce:ListSteps",
                "elasticmapreduce:ListSupportedInstanceTypes",
                "elasticmapreduce:ModifyCluster",
                "elasticmapreduce:ModifyInstanceFleet",
                "elasticmapreduce:ModifyInstanceGroups",
                "elasticmapreduce:OpenEditorInConsole",
                "elasticmapreduce:PutAutoScalingPolicy",
                "elasticmapreduce:PutBlockPublicAccessConfiguration",
                "elasticmapreduce:PutManagedScalingPolicy",
                "elasticmapreduce:RemoveAutoScalingPolicy",
                "elasticmapreduce:RemoveManagedScalingPolicy",
                "elasticmapreduce:RemoveTags",
                "elasticmapreduce:SetTerminationProtection",
                "elasticmapreduce:StartEditor",
                "elasticmapreduce:StopEditor",
                "elasticmapreduce:TerminateJobFlows",
                "elasticmapreduce:ViewEventsFromAllClustersInConsole"
            ],
            "Resource": "*"
        },
        {
```

```
            "Sid": "ViewMetricsInEMRConsole",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricStatistics"
            ],
            "Resource": "*"
        },
        {
            "Sid": "PassRoleForElasticMapReduce",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [
                "arn:aws:iam::*:role/EMR_DefaultRole",
                "arn:aws:iam::*:role/EMR_DefaultRole_V2"
            ],
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "elasticmapreduce.amazonaws.com*"
                }
            }
        },
        {
            "Sid": "PassRoleForEC2",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::*:role/EMR_EC2_DefaultRole",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "ec2.amazonaws.com*"
                }
            }
        },
        {
            "Sid": "PassRoleForAutoScaling",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::*:role/EMR_AutoScaling_DefaultRole",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "application-autoscaling.amazonaws.com*"
                }
            }
        },
        {
```

```
            "Sid": "ElasticMapReduceServiceLinkedRole",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/
elasticmapreduce.amazonaws.com*/AWSServiceRoleForEMRCleanup*",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": [
                        "elasticmapreduce.amazonaws.com",
                        "elasticmapreduce.amazonaws.com.cn"
                    ]
                }
            }
        },
        {
            "Sid": "ConsoleUIActions",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeImages",
                "ec2:DescribeKeyPairs",
                "ec2:DescribeNatGateways",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:DescribeVpcEndpoints",
                "s3:ListAllMyBuckets",
                "iam:ListRoles"
            ],
            "Resource": "*"
        }
    ]
}
```

フルアクセス用の IAM 管理ポリシー (非推奨化予定)

`AmazonElasticMapReduceFullAccess` および `AmazonEMRFullAccessPolicy_v2` AWS Identity and Access Management (IAM) 管理ポリシーは、Amazon EMR およびその他の サービスに必要なすべてのアクションを付与します。

> **Important**
>
> AmazonElasticMapReduceFullAccess 管理ポリシーは廃止予定であり、Amazon EMR での使用は推奨されなくなりました。代わりに AmazonEMRFullAccessPolicy_v2 を使用してください。最終的に IAM サービスで v1 ポリシーが廃止されると、このポリシーはロールにアタッチできなくなります。ただし、既存のロールが廃止されたポリシーを使用している場合でも、そのロールをクラスターにアタッチすることはできます。

Amazon EMR のフルアクセス許可のデフォルトの管理ポリシーには、次を含む iam:PassRole セキュリティ設定が組み込まれています。

- 特定のデフォルトの Amazon EMR ロール専用の iam:PassRole アクセス許可。
- iam:PassedToService や などの指定された AWS サービスでのみポリシーを使用できるようにする elasticmapreduce.amazonaws.com 条件ec2.amazonaws.com。

AmazonEMRFullAccessPolicy AmazonEMRServicePolicy_v2 ポリシーの JSON バージョンは、IAM コンソールで表示できます。v2 管理ポリシーを使用して新しいクラスターを作成することが推奨されます。

非推奨の v1 ポリシーの内容は、 の AWS Management Console で確認できますAmazonElasticMapReduceFullAccess。ポリシー内の ec2:TerminateInstances アクションは、IAM アカウントに関連付けられている Amazon EC2 インスタンスを終了するためのアクセス許可をユーザーまたはロールに付与します。これには、EMR クラスターの一部ではないインスタンスも含まれます。

読み取り専用アクセス用の IAM 管理ポリシー (v2 管理デフォルトポリシー)

Amazon EMR に読み取り専用権限を付与するには、AmazonEMRReadOnlyAccessPolicy管理ポリシーをアタッチします。このデフォルト管理ポリシーは、AmazonElasticMapReduceReadOnlyAccess 管理ポリシーを置き換えます。次のスニペットは、このポリシーステートメントの内容を示しています。AmazonElasticMapReduceReadOnlyAccess ポリシーと比較すると、AmazonEMRReadOnlyAccessPolicy_v2 ポリシーでは、elasticmapreduce 要素にワイルドカード文字を使用しません。代わりに、デフォルトの v2 ポリシーでは、許容される elasticmapreduce アクションの範囲を設定します。

> **Note**
>
> AWS Management Console リンクを使用してポリシー[AmazonEMRReadOnlyAccessPolicy_v2](#)を表示することもできます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ElasticMapReduceActions",
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:DescribeCluster",
                "elasticmapreduce:DescribeEditor",
                "elasticmapreduce:DescribeJobFlows",
                "elasticmapreduce:DescribeSecurityConfiguration",
                "elasticmapreduce:DescribeStep",
                "elasticmapreduce:DescribeReleaseLabel",
                "elasticmapreduce:GetBlockPublicAccessConfiguration",
                "elasticmapreduce:GetManagedScalingPolicy",
                "elasticmapreduce:GetAutoTerminationPolicy",
                "elasticmapreduce:ListBootstrapActions",
                "elasticmapreduce:ListClusters",
                "elasticmapreduce:ListEditors",
                "elasticmapreduce:ListInstanceFleets",
                "elasticmapreduce:ListInstanceGroups",
                "elasticmapreduce:ListInstances",
                "elasticmapreduce:ListSecurityConfigurations",
                "elasticmapreduce:ListSteps",
                "elasticmapreduce:ListSupportedInstanceTypes",
                "elasticmapreduce:ViewEventsFromAllClustersInConsole"
            ],
            "Resource": "*"
        },
        {
            "Sid": "ViewMetricsInEMRConsole",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricStatistics"
            ],
            "Resource": "*"
```

```
        }
    ]
}
```

読み取り専用アクセス用の IAM 管理ポリシー (非推奨化予定)

AmazonElasticMapReduceReadOnlyAccess 管理ポリシーは、非推奨になる予定です。新しいクラスターの起動時に、このポリシーをアタッチすることはできません。AmazonElasticMapReduceReadOnlyAccess は、Amazon EMR デフォルト管理ポリシーとして AmazonEMRReadOnlyAccessPolicy_v2 に置き換えられました。次のスニペットは、このポリシーステートメントの内容を示しています。elasticmapreduce 要素のワイルドカード文字は、指定文字列で始まるアクションのみが許可されるように指定します。このポリシーは明示的にアクションを拒否しないため、別のポリシーステートメントが指定したアクションへのアクセス許可に使用される場合があることに注意してください。

> **Note**
>
> を使用してポリシー AWS Management Console を表示することもできます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:Describe*",
        "elasticmapreduce:List*",
        "elasticmapreduce:ViewEventsFromAllClustersInConsole",
        "s3:GetObject",
        "s3:ListAllMyBuckets",
        "s3:ListBucket",
        "sdb:Select",
        "cloudwatch:GetMetricStatistics"
      ],
      "Resource": "*"
    }
  ]
}
```

AWS マネージドポリシー: EMR DescribeClusterPolicyForEMRWAL

IAM エンティティに EMRDescribeClusterPolicyForEMRWAL をアタッチすることはできません。このポリシーは、Amazon EMR がユーザーに代わってアクションを実行することを許可するサービスにリンクされたロールにアタッチされます。このサービスにリンクされたロールの詳細については、「」を参照してください先行書き込みログ記録にサービスにリンクされたロールを使用する。

このポリシーは、Amazon EMR の WAL サービスがクラスターのステータスを検索して返すための読み取り専用アクセス許可を付与します。Amazon EMR WAL の詳細については、「Amazon EMR のログ先行書き込み (WAL)」を参照してください。

許可の詳細

このポリシーには、以下の許可が含まれています。

- emr — プリンシパルが Amazon EMR からクラスターステータスを記述できるようにします。これは、Amazon EMR がクラスターが終了した日時を確認し、30 日後にクラスターによって残された WAL ログをクリーンアップできるようにするために必要です。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:DescribeCluster"
      ],
      "Resource": "*"
    }
  ]
}
```

AWS Amazon EMR の マネージドポリシー

AWS 管理ポリシーは、 によって作成および管理されるスタンドアロンポリシーです AWS。 AWS 管理ポリシーは、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ように設計されているため、ユーザー、グループ、ロールにアクセス許可の割り当てを開始できます。

AWS 管理ポリシーは、すべての AWS お客様が使用できるため、特定のユースケースに対して最小特権のアクセス許可を付与しない場合があることに注意してください。ユースケース別にカスタマーマネージドポリシーを定義して、マネージドポリシーを絞り込むことをお勧めします。

AWS 管理ポリシーで定義されているアクセス許可は変更できません。が AWS 管理ポリシーで定義されたアクセス許可 AWS を更新すると、ポリシーがアタッチされているすべてのプリンシパル ID (ユーザー、グループ、ロール) が更新されます。 AWS のサービス は、新しい が起動されたとき、または既存のサービスで新しい API AWS オペレーションが使用可能になったときに、 AWS 管理ポリシーを更新する可能性が最も高くなります。

詳細については、「IAM ユーザーガイド」の「AWS 管理ポリシー」を参照してください。

AWS マネージドポリシーに対する Amazon EMR の更新

Amazon EMR の AWS マネージドポリシーの更新に関する詳細を、このサービスがこれらの変更の追跡を開始した以降の分について表示します。

変更	説明	日付
EMRDescribeClusterPolicyForEMRWAL - 新しいポリシー	Amazon EMR がクラスターの終了から 30 日後に WAL クリーンアップのクラスターステータスを判断できるように、新しいポリシーを追加しました。	2023 年 8 月 10 日
AmazonEMRFullAccessPolicy_v2 と **AmazonEMRReadOnlyAccessPolicy_v2** – 既存のポリシーに対する更新	`elasticmapreduce:DescribeReleaseLabel` と `elasticmapreduce:GetAutoTerminationPolicy` が追加されました。	2022 年 4 月 21 日
AmazonEMRFullAccessPolicy_v2 – 既存ポリシーへの更新	「カスタム AMI の使用」に `ec2:DescribeImages` が追加されました。	2022 年 2 月 15 日
Amazon EMR 管理ポリシー	定義済みのユーザータグの使用を明確にするために更新されました。	2021 年 9 月 29 日

変更	説明	日付
	AWS コンソールを使用して v2 管理ポリシーで clsuter を起動する手順に関するセクションを追加しました。	
AmazonEMRFullAccessPolicy_v2 – 既存ポリシーへの更新	PassRoleForAutoScaling および PassRoleForEC2 のアクションが変更され、それぞれ "iam:PassedToService":"application-autoscaling.amazonaws.com*" および "iam:PassedToService":"ec2.amazonaws.com*" に一致するように StringLike 条件演算子が使用されるようになりました。	2021 年 5 月 20 日
AmazonEMRFullAccessPolicy_v2 – 既存ポリシーへの更新	無効なアクション s3:ListBuckets が削除され、s3:ListAllMyBuckets アクションに置き換えられました。 サービスにリンクされたロール (SLR) の作成を、明示的なサービス原則で Amazon EMR が備える唯一の SLR に明示的に範囲制限するように更新しました。作成できる SLR は、この変更前とまったく同じです。	2021 年 3 月 23 日

変更	説明	日付
AmazonEMRFullAccessPolicy_v2 - 新しいポリシー	Amazon EMR で新しいアクセス許可が追加され、リソースへのアクセスの範囲が制限され、ユーザーが Amazon EMR 管理ポリシーを使用する前に事前定義されたユーザータグをリソースに追加する必要があるという前提条件が追加されました。 `iam:PassRole` アクションを実行するには、指定されたサービスに対して `iam:PassedToService` 条件が設定されている必要があります。デフォルトでは、Amazon EC2、Amazon S3、およびその他のサービスへのアクセスは許可されていません。	2021 年 3 月 11 日
AmazonEMRServicePolicy_v2 - 新しいポリシー	このポリシーを使用するには、ユーザーがリソースにユーザータグを追加する必要があるという前提条件を追加します。	2021 年 3 月 11 日
AmazonEMRReadOnlyAccessPolicy_v2 - 新しいポリシー	アクセス許可によって、指定された elasticmapreduce の読み取り専用アクションのみが許可されます。Amazon S3 へのアクセスは、デフォルトではアクセスが許可されていません。	2021 年 3 月 11 日

変更	説明	日付
Amazon EMR が変更の追跡を開始しました。	Amazon EMR が AWS マネージドポリシーの変更の追跡を開始しました。	2021 年 3 月 11 日

クラスターおよび EMR Notebooks に対するタグベースのアクセスの IAM ポリシー

アイデンティティベースのポリシーで条件を使用し、タグに基づいてクラスターおよび EMR ノートブックへのアクセスを制御できます。

EMR クラスターにタグを追加する方法の詳細については、「EMR クラスターにタグを付ける」を参照してください。

次の例では、Amazon EMR 条件キーで条件演算子を使用するさまざまなシナリオと方法について説明しています。これらの IAM ポリシーステートメントは、デモンストレーションのみを目的としており、本稼働環境で使用しないでください。要件に応じて、アクセス権限を付与または拒否するようにポリシーステートメントを組み合わせる複数の方法があります。IAM ポリシーの計画およびテストの詳細については、「IAM ユーザーガイド」を参照してください。

> **Important**
>
> アクションをタグ付けするための権限を明示的に拒否することは重要な考慮事項です。これにより、ユーザーがリソースをタグ付けして意図せずにアクセス許可を付与することを防ぎます。リソースのタグ付けアクションを拒否しない場合、ユーザーはタグを変更して、タグベースのポリシーの意図を回避できます。

クラスターのアイデンティティベースのポリシーステートメントの例

以下の例では、EMR クラスターで許可されるアクションを制御するために使用するアイデンティティベースの許可ポリシーを説明しています。

> **Important**
>
> Amazon EMR の `ModifyInstanceGroup` アクションでは、クラスター ID を指定する必要はありません。そのため、クラスタータグに基づいてこのアクションを拒否するには、さら

に考慮する必要があります。詳細については、「ModifyInstanceGroup アクションの拒否」を参照してください。

トピック

- 特定のタグの値があるクラスター上でのみアクションを許可する
- クラスターの作成時にクラスターのタグ付けを要求する
- タグの値に関わらず、特定のタグがあるクラスター上でアクションを許可する

特定のタグの値があるクラスター上でのみアクションを許可する

次の例では、値が *dev* に設定されたクラスタータグ *department* に基づいてユーザーのアクション実行を許可し、またその同じタグでクラスターにタグ付けすることを許可するポリシーを示します。最後のポリシー例では、同じタグ以外では EMR クラスターにタグ付けする権限を拒否する方法について説明します。

次のポリシーの例では、StringEquals 条件演算子は、*dev* をタグ *department* の値 と一致させるように試みます。タグ *department* がクラスターに追加されていない、または値 *dev* を含んでいない場合は、ポリシーは適用されず、このアクションは、ポリシーによって許可されません。アクションを許可するポリシーステートメントが他にない場合は、ユーザーはこの値を持つこのタグを持っているクラスターとのみ作業できます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt12345678901234",
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:DescribeCluster",
        "elasticmapreduce:ListSteps",
        "elasticmapreduce:TerminateJobFlows",
        "elasticmapreduce:SetTerminationProtection",
        "elasticmapreduce:ListInstances",
        "elasticmapreduce:ListInstanceGroups",
        "elasticmapreduce:ListBootstrapActions",
        "elasticmapreduce:DescribeStep"
      ],
      "Resource": [
```

```
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "elasticmapreduce:ResourceTag/department": "dev"
        }
      }
    }
  ]
}
```

条件付き演算子を使用して複数のタグ値を指定できます。たとえば、*department* タグに値 *dev* または *test* が含まれるクラスターですべてのアクションを許可するには、以下のように、前術の例のような条件ブロックを置き換えることもできます。

```
            "Condition": {
              "StringEquals": {
                "elasticmapreduce:ResourceTag/department":["dev", "test"]
              }
            }
```

クラスターの作成時にクラスターのタグ付けを要求する

上記の例と同様に、次のポリシー例でも、同じ一致タグ (*department* タグの値 *dev*) を検索します。ただし、この例では、RequestTag 条件キーによって、タグの作成時にポリシーが適用されることが指定されています。そのため、指定された値と一致するタグを使用してクラスターを作成する必要があります。

タグ付きのクラスターを作成するには、elasticmapredue:AddTags アクションを実行するためのアクセス許可も必要です。このステートメントでは、elasticmapreduce:ResourceTag 条件キーによって、IAM は *department* タグの値が *dev* であるリソースにタグ付けするためのアクセス権限のみを付与するようになります。Resource 要素は、このアクセス許可をクラスターリソースに制限するために使用されます。

PassRole リソースには、 AWS アカウント ID またはエイリアス、 PassRoleForEMRステートメントのサービスロール名、および PassRoleForEC2ステートメントのインスタンスプロファイル名を指定する必要があります。IAM ARN のフォーマットの詳細については、「IAM ユーザーガイド」の「IAM ARN」を参照してください。

一致するタグとキーの値の詳細については、「IAM ユーザーガイド」の「aws:RequestTag/tag-key」を参照してください。

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Sid": "RunJobFlowExplicitlyWithTag",
        "Effect": "Allow",
        "Action": [
          "elasticmapreduce:RunJobFlow"
        ],
        "Resource": "*",
        "Condition": {
          "StringEquals": {
            "aws:RequestTag/department": "dev"
          }
        }
      },
      {
        "Sid": "AddTagsForDevClusters",
        "Effect": "Allow",
        "Action": "elasticmapreduce:AddTags",
        "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*",
        "Condition": {
          "StringEquals": {
            "elasticmapreduce:ResourceTag/department": "dev"
          }
        }
      },
      {
        "Sid": "PassRoleForEMR",
        "Effect": "Allow",
        "Action": "iam:PassRole",
        "Resource": "arn:aws:iam::AccountId:role/Role-Name-With-Path",
        "Condition": {
          "StringLike": {
            "iam:PassedToService": "elasticmapreduce.amazonaws.com*"
          }
        }
      },
      {
        "Sid": "PassRoleForEC2",
```

```
        "Effect": "Allow",
        "Action": "iam:PassRole",
        "Resource": "arn:aws:iam::AccountId:role/Role-Name-With-Path",
        "Condition": {
          "StringLike": {
            "iam:PassedToService": "ec2.amazonaws.com*"
          }
        }
      }
    ]
}
```

タグの値に関わらず、特定のタグがあるクラスター上でアクションを許可する

タグ値にかかわらず、特定のタグを持つクラスターでのみアクションを許可できます。これを行うには、Null を使用できます。詳細については、「IAM ユーザーガイド」の「条件キーの有無をチェックする条件演算子」を参照してください。たとえば、*department* タグを持つ EMR クラスターでのみアクションを許可するには、それが含んでいる値にかかわらず、次のいずれかを持つ、前述の例のような条件ブロックを置き換えることもできます。Null 演算子は EMR クラスターでタグ *department* の存在を探します。タグがある場合、このポリシーステートメントに指定された条件に合致し、Null ステートメントは false と評価され、適切なアクションが許可されます。

```
"Condition": {
  "Null": {
    "elasticmapreduce:ResourceTag/department":"false"
  }
}
```

次のポリシーステートメントは、クラスターに任意の値を含めることができる *department* タグがある場合のみ、ユーザーに EMR クラスターを作成することを許可します。PassRole リソースには、 AWS アカウント ID またはエイリアス、およびサービスロール名を指定する必要があります。IAM ARN のフォーマットの詳細については、「IAM ユーザーガイド」の「IAM ARN」を参照してください。

ヌル (「false」) 条件演算子の指定の詳細については、「IAM ユーザーガイド」の「条件キーの有無をチェックする条件演算子」を参照してください。

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "CreateClusterTagNullCondition",
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:RunJobFlow"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "Null": {
                    "aws:RequestTag/department": "false"
                }
            }
        },
        {
            "Sid": "AddTagsNullCondition",
            "Effect": "Allow",
            "Action": "elasticmapreduce:AddTags",
            "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*",
            "Condition": {
                "Null": {
                    "elasticmapreduce:ResourceTag/department": "false"
                }
            }
        },
        {
            "Sid": "PassRoleForElasticMapReduce",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::AccountId:role/Role-Name-With-Path",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "elasticmapreduce.amazonaws.com*"
                }
            }
        },
        {
            "Sid": "PassRoleForEC2",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::AccountId:role/Role-Name-With-Path",
            "Condition": {
```

```
                "StringLike": {
                    "iam:PassedToService": "ec2.amazonaws.com*"
                }
            }
        }
    ]
}
```

EMR Notebooks のアイデンティティベースのポリシーステートメントの例

このセクションの IAM ポリシーステートメントの例では、EMR Notebooks で許可されるアクションを制限するためのキーの使い方に関する一般的なシナリオを示します。プリンシパル (ユーザー) に関連付けられたその他のポリシーによってアクションが許可されない限り、示されたとおり条件コンテキストキーによって許可されるアクションが制限されます。

Example – ユーザーがタグ付けに基づいて作成した EMR Notebooks へのアクセスのみ許可する

以下のポリシーステートメントの例では、ロールまたはユーザーにアタッチされている場合、ユーザーは自分で作成したノートブックのみを使用できます。このポリシーステートメントは、ノートブックが作成されたときに適用されるデフォルトのタグを使用します。

この例では、StringEquals 条件演算子は、現在のユーザーの IAM ユーザー ID ({aws:userId}) を表す変数とタグ creatorUserID の値のマッチングを試みます。タグ creatorUserID がノートブックに追加されていない、または現在のユーザー ID の値を含んでいない場合は、ポリシーは適用されず、このアクションは、ポリシーによって許可されません。アクションを許可するポリシーステートメントが他にない場合は、ユーザーはこの値を持つこのタグを持っているノートブックとのみ作業できます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "elasticmapreduce:DescribeEditor",
                "elasticmapreduce:StartEditor",
                "elasticmapreduce:StopEditor",
                "elasticmapreduce:DeleteEditor",
                "elasticmapreduce:OpenEditorInConsole"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
```

```
                "StringEquals": {
                    "elasticmapreduce:ResourceTag/creatorUserId": "${aws:userId}"
                }
            }
        }
    ]
}
```

Example - ノートブックの作成時にノートブックのタグ付けを求める

この例では、RequestTag コンテキストキーが使用されています。CreateEditor アクションは、デフォルトで追加された creatorUserID タグをユーザーが変更または削除しない場合にのみ許可されます。変数 ${aws:userId} は現在アクティブなユーザーのユーザー ID を指定します。これはタグのデフォルト値です。

ポリシーステートメントは、ユーザーが createUserId タグを削除したり、値を変更したりしていないことを確認するのに役立ちます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "elasticmapreduce:CreateEditor"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "elasticmapreduce:RequestTag/creatorUserId": "${aws:userid}"
                }
            }
        }
    ]
}
```

この例では、ユーザーはキー文字 dept および次のいずれかに設定された値を持つタグを使用してクラスターを作成する必要があります。datascience、analytics、operations。

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Action": [
                "elasticmapreduce:CreateEditor"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "elasticmapreduce:RequestTag/dept": [
                        "datascience",
                        "analytics",
                        "operations"
                    ]
                }
            }
        }
    ]
}
```

Example - タグ付けされたクラスターにノートブックの作成を制限して、ノートブックタグを求める

この例では、指定された値のいずれかに設定されたキー文字 owner を持つタグを使用してノートブックが作成された場合にのみ、ノートブックの作成が許可されます。さらに、指定された値のいずれかに設定されたキー文字 department を持つタグがクラスターにある場合のみ、ノートブックを作成できます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "elasticmapreduce:CreateEditor"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "elasticmapreduce:RequestTag/owner": [
                        "owner1",
                        "owner2",
                        "owner3"
                    ],
                    "elasticmapreduce:ResourceTag/department": [
```

```
                        "dep1",
                        "dep3"
                    ]
                }
            }
        }
    ]
}
```

Example - タグに基づいてノートブックを開始する機能を制限する

この例では、ノートブックをスタートする権限を、指定された値のいずれかに設定されたキー文字 owner があるタグを持つノートブックのみに制限します。Resource 要素は editor のみを指定することに使用されるため、条件はクラスターには適用されず、タグ付けされる必要はありません。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "elasticmapreduce:StartEditor"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:elasticmapreduce:*:123456789012:editor/*",
            "Condition": {
                "StringEquals": {
                    "elasticmapreduce:ResourceTag/owner": [
                        "owner1",
                        "owner2"
                    ]
                }
            }
        }
    ]
}
```

この例は、上記の例と似ています。ただし、制限はノートブックではなく、タグ付けされたクラスターにのみ適用されます。

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Action": [
                "elasticmapreduce:StartEditor"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:elasticmapreduce:*:123456789012:cluster/*",
            "Condition": {
                "StringEquals": {
                    "elasticmapreduce:ResourceTag/department": [
                        "dep1",
                        "dep3"
                    ]
                }
            }
        }
    ]
}
```

この例では、さまざまなノートブックおよびクラスタータグが使用されます。これにより、ノートブックは次の場合にのみ起動できます。

- ノートブックに、指定された値のいずれかに設定されたキー文字 owner を持つタグがある場合

  および

- クラスターに、指定された値のいずれかに設定されたキー文字 department を持つタグがある場合

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "elasticmapreduce:StartEditor"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:elasticmapreduce:*:123456789012:editor/*",
            "Condition": {
                "StringEquals": {
                    "elasticmapreduce:ResourceTag/owner": [
                        "user1",
                        "user2"
```

```
                    ]
                }
            }
        },
        {
            "Action": [
                "elasticmapreduce:StartEditor"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:elasticmapreduce:*:123456789012:cluster/*",
            "Condition": {
                "StringEquals": {
                    "elasticmapreduce:ResourceTag/department": [
                        "datascience",
                        "analytics"
                    ]
                }
            }
        }
    ]
}
```

Example - タグに基づいてノートブックエディタを開く機能を制限する

この例では、ノートブックエディタは次の場合にのみ開くことができます。

- ノートブックに、指定された値のいずれかに設定されたキー文字 owner を持つタグがある場合

  および
- クラスターに、指定された値のいずれかに設定されたキー文字 department を持つタグがある場合

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "elasticmapreduce:OpenEditorInConsole"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:elasticmapreduce:*:123456789012:editor/*",
            "Condition": {
```

```
                "StringEquals": {
                    "elasticmapreduce:ResourceTag/owner": [
                        "user1",
                        "user2"
                    ]
                }
            }
        },
        {
            "Action": [
                "elasticmapreduce:OpenEditorInConsole"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:elasticmapreduce:*:123456789012:cluster/*",
            "Condition": {
                "StringEquals": {
                    "elasticmapreduce:ResourceTag/department": [
                        "datascience",
                        "analytics"
                    ]
                }
            }
        }
    ]
}
```

ModifyInstanceGroup アクションの拒否

Amazon EMR の ModifyInstanceGroupsアクションでは、 アクションでクラスター ID を指定する必要はありません。代わりに、インスタンスグループ ID のみを指定できます。このため、クラスター ID またはクラスタータグに基づくこのアクションに対する単純な拒否ポリシーでは、意図した効果が得られない可能性があります。次のポリシー例を考えます。

```
{
 "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "elasticmapreduce:ModifyInstanceGroups"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
```

```
        {
            "Action": [
                "elasticmapreduce:ModifyInstanceGroups"
            ],
            "Effect": "Deny",
            "Resource": "arn:aws:elasticmapreduce:us-east-1:123456789012:cluster/
j-12345ABCDEFG67"
        }
    ]

}
```

このポリシーがアタッチされたユーザーが ModifyInstanceGroup アクションを実行し、インスタンスグループ ID のみを指定した場合、このポリシーは適用されません。アクションは他のすべてのリソースで許可されるため、アクションは成功します。

この問題の解決策は、 NotResource要素を使用してクラスター ID なしで発行されたModifyInstanceGroupアクションを拒否するポリシーステートメントをアイデンティティにアタッチすることです。次のポリシー例では、このような deny ステートメントを追加して、クラスター ID が指定されていない限り、ModifyInstanceGroups リクエストが失敗するようにします。ID はアクションでクラスター ID を指定する必要があるため、クラスター ID に基づく deny ステートメントが有効になります。

```
{
 "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "elasticmapreduce:ModifyInstanceGroups"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "elasticmapreduce:ModifyInstanceGroups"
            ],
            "Effect": "Deny",
            "Resource": "arn:aws:elasticmapreduce:us-east-1:123456789012:cluster/
j-12345ABCDEFG67"
        },
        {
```

```
            "Action": [
                "elasticmapreduce:ModifyInstanceGroups"
            ],
            "Effect": "Deny",
            "NotResource": "arn:*:elasticmapreduce:*:*:cluster/*"
        }
    ]

}
```

クラスタータグに関連付けられた値に基づいて ModifyInstanceGroups アクションを拒否する場合にも同様の問題が存在します。解決策も同様です。タグ値を指定する deny ステートメントに加えて、値に関係なく、指定したタグが存在しない場合に ModifyInstanceGroup アクションを拒否するポリシーステートメントを追加できます。

次の例では、ID にアタッチされたときに、タグ department が dev に設定されているクラスターでその ID に対して ModifyInstanceGroups アクションを拒否するポリシーを示しています。StringNotLike 条件を使用して department タグが存在しない限りアクションを拒否する deny ステートメントのため、このステートメントのみが有効です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "elasticmapreduce:ModifyInstanceGroups"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "elasticmapreduce:ModifyInstanceGroups"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/department": "dev"
                }
            },
            "Effect": "Deny",
            "Resource": "*"
        },
```

```
        {
            "Action": [
                "elasticmapreduce:ModifyInstanceGroups"
            ],
            "Condition": {
                "StringNotLike": {
                    "aws:ResourceTag/department": "?*"
                }
            },
            "Effect": "Deny",
            "Resource": "*"
        }
    ],
}
```

Amazon EMR の ID とアクセスのトラブルシューティング

Amazon EMR と 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や修正には、次の情報が役立ちます。

トピック

- Amazon EMR でアクションを実行する権限がない
- iam を実行する権限がありません。PassRole
- AWS アカウント以外のユーザーに Amazon EMR リソースへのアクセスを許可したい

Amazon EMR でアクションを実行する権限がない

がアクションを実行する権限がないと AWS Management Console 通知した場合は、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 EMR:*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:
 EMR:GetWidget on resource: my-example-widget
```

この場合、Mateo は、EMR:*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。

iam を実行する権限がありません。PassRole

`iam:PassRole` アクションを実行する権限がないというエラーが表示された場合は、ポリシーを更新して Amazon EMR にロールを渡せるようにする必要があります。

一部の AWS のサービス では、新しいサービスロールまたはサービスにリンクされたロールを作成する代わりに、そのサービスに既存のロールを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。

次の例のエラーは、`marymajor` という IAM ユーザーがコンソールを使用して Amazon EMR でアクションを実行しようとした場合に発生します。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。メアリーには、ロールをサービスに渡す許可がありません。

```
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。サインイン認証情報を提供した担当者が管理者です。

AWS アカウント以外のユーザーに Amazon EMR 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:

- Amazon EMR がこれらの機能をサポートしているかどうかを確認するには、「Amazon EMR で IAM が機能する仕組み」を参照してください。
- 所有している 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 AWS アカウント「所有している別の 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。AWS アカウント
- 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でのロールとリソースベースのポリシーの使用の違いについては、IAM ユーザーガイドの「IAM でのクロスアカウントリソースアクセス」を参照してください。

Amazon EMR での Amazon S3 Access Grants の使用

Amazon EMR 用 S3 Access Grants の概要

Amazon EMR リリース 6.15.0 以降では、Amazon S3 Access Grants によるスケーラブルなアクセスコントロールソリューションの提供により、Amazon EMR からの Amazon S3 データへのアクセスが強化されています。S3 データで複雑または大規模なアクセス許可設定が必要となる場合は、Access Grants を使用して、ユーザー、グループ、ロール、アプリケーションの S3 データ権限をスケーリングできます。

S3 Access Grants を使用すると、EMR クラスターへあのアクセス権を持つ ID に関連付けられているランタイム ロールまたは IAM ロールによって付与されるアクセス許可を超えたレベルまで、Amazon S3 データへのアクセスを強化することができます。詳細については、「Amazon S3 ユーザーガイド」の「S3 Access Grants によるアクセス管理」を参照してください。

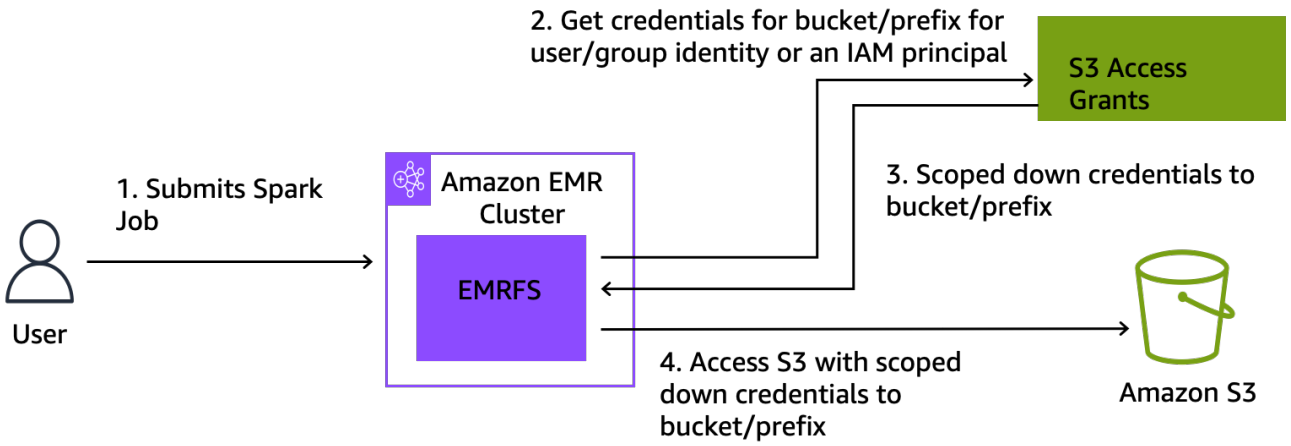
他の Amazon EMR デプロイで S3 Access Grants を使用する手順については、以下のドキュメントを参照してください。

- EKS 上の Amazon EMR での S3 Access Grants の使用
- Amazon EMR Serverless での S3 Access Grants の使用

Amazon EMR が S3 Access Grants と連携する仕組み

Amazon EMR リリース 6.15.0 以降では、S3 Access Grants がネイティブ統合されています。Amazon EMR で S3 ア Access Grants を有効にし、Spark ジョブを実行することができます。Spark ジョブが S3 データをリクエストすると、Amazon S3 は特定のバケット、プレフィックス、またはオブジェクトを対象とする一時的な認証情報を提供します。

以下に、Amazon EMR が S3 Access Grants で保護されたデータにアクセスする方法の概要を示します。



1. ユーザーが、Amazon S3 に保存されているデータを使用する Amazon EMR Spark ジョブを送信します。
2. Amazon EMR はユーザーに代わって、バケット、プレフィックス、またはオブジェクトへのアクセス許可を S3 Access Grants に対してリクエストします。
3. Amazon S3 は、ユーザーの AWS Security Token Service (STS) トークンの形式で一時的な認証情報を返します。トークンのスコープは、S3 バケット、プレフィックス、またはオブジェクトへのアクセスを対象としています。
4. Amazon EMR は STS トークンを使用して S3 からデータを取得します。
5. Amazon EMR は S3 からデータを受け取り、結果をユーザーに返します。

Amazon EMR での S3 Access Grants の使用に関する考慮事項

Amazon EMR で S3 Access Grants を使用するときは、以下の動作と制限に注意してください。

機能のサポート

- S3 Access Grants は、Amazon EMR リリース 6.15.0 以降でサポートされています。
- Amazon EMR で S3 Access Grants を使用する際にサポートされるクエリエンジンは Spark のみです。
- Amazon EMR で S3 Access Grants を使用する際にサポートされるオープンテーブル形式は、Delta Lake と Hudi だけです。
- 以下の Amazon EMR 機能は、S3 Access Grants との使用においてはサポートされません。
  - Apache Iceberg テーブル
  - LDAP ネイティブ認証

  - Apache Ranger ネイティブ認証
  - AWS CLI IAM ロールを使用する Amazon S3 への リクエスト
  - オープンソースの S3A プロトコルによる S3 アクセス
- `fallbackToIAM` オプションは、IAM Identity Center による信頼できる ID 伝達を使用する EMR クラスターではサポートされていません。
- AWS Lake Formationでの S3 Access Grants は、Amazon EC2 で動作する Amazon EMR クラスターでのみサポートされます。

動作に関する注意事項

- Apache Ranger と Amazon EMR のネイティブ統合は、EMRFS S3 Apache Ranger プラグインの一部として S3 Access Grants に対応する機能を備えています。Apache Ranger をきめ細かなアクセスコントロール (FGAC) に使用する場合は、S3 Access Grants の代わりに Apache Ranger プラグインを使用することをお勧めします。
- Amazon EMR では、ユーザーが Spark ジョブ内で同じ認証情報を繰り返しリクエストする必要がないように、EMRFS 内に認証情報キャッシュを持ちます。そのため、Amazon EMR は認証情報をリクエストするときは常にデフォルトレベルの権限を要求します。詳細については、「Amazon S3 ユーザーガイド」の「S3 データへのアクセスをリクエストする」を参照してください。
- Amazon EMR は、S3 Access Grants でサポートしていないアクションをユーザーが実行する場合にジョブ実行用に指定された IAM ロールを使用するように設定されています。詳細については、「IAM ロールにフォールバックする」を参照してください。

S3 Access Grants を使用した Amazon EMR クラスターの起動

このセクションでは、Amazon EC2 で動作する EMR クラスターを起動し、S3 Access Grants を使用して Amazon S3 内のデータへのアクセスを管理する方法について説明します。他の Amazon EMR デプロイで S3 Access Grants を使用する手順については、以下のドキュメントを参照してください。

- EKS 上の Amazon EMR での S3 Access Grants の使用
- EMR Serverless での S3 Access Grants の使用

Amazon EC2 で実行される、S3 Access Grants を使用して Amazon S3 内のデータへのアクセスを管理する EMR クラスターを起動するには、以下の手順に従います。

1. EMR クラスターのジョブ実行ロールを設定します。Spark ジョブの実行に必要な IAM アクセス許可 (s3:GetDataAccess および s3:GetAccessGrantsInstanceForPrefix) を含めてください。

```
{
    "Effect": "Allow",
    "Action": [
    "s3:GetDataAccess",
    "s3:GetAccessGrantsInstanceForPrefix"
    ],
    "Resource": [     //LIST ALL INSTANCE ARNS THAT THE ROLE IS ALLOWED TO QUERY
         "arn:aws_partition:s3:Region:account-id1:access-grants/default",
         "arn:aws_partition:s3:Region:account-id2:access-grants/default"
    ]
}
```

> **Note**
>
> Amazon EMR では、S3 Access Grants は IAM ロールに設定されたアクセス権限を補完します。ジョブ実行用に指定した IAM ロールに S3 に直接アクセスする権限が含まれている場合、ユーザーは S3 Access Grants で定義するデータだけでなく、さらに多くのデータにアクセスできる可能性があります。

2. 次に、 を使用して Amazon EMR 6.15 以降でクラスター AWS CLI を作成し、 emrfs-site分類を使用して S3 Access Grants を有効にします。例は次のとおりです。

```
aws emr create-cluster
  --release-label emr-6.15.0 \
  --instance-count 3 \
  --instance-type m5.xlarge \
  --configurations '[{"Classification":"emrfs-site",
 "Properties":{"fs.s3.s3AccessGrants.enabled":"true",
 "fs.s3.s3AccessGrants.fallbackToIAM":"false"}}]'
```

を使用した S3 Access Grants AWS Lake Formation

AWS Lake Formation 統合で Amazon EMR を使用する場合、Amazon S3 Access Grants を使用すれば、Amazon S3 のデータに直接アクセスすることも、表形式でアクセスすることもできます。

> **Note**
>
> を使用した S3 Access Grants AWS Lake Formation は、Amazon EC2 で実行される Amazon EMR クラスターでのみサポートされます。

直接アクセス

直接アクセスには、Lake Formation が Amazon EMR のメタストアとして使用する AWS Glue サービスの API を呼び出さない S3 データにアクセスするためのすべての呼び出しが含まれます。例えば、 を呼び出す場合ですspark.read。

```
spark.read.csv("s3://...")
```

Amazon EMR AWS Lake Formation で S3 Access Grants を で使用すると、すべての直接アクセスパターンが S3 Access Grants を通過して一時的な S3 認証情報を取得します。

表形式のアクセス

表形式アクセスは、Lake Formation がメタストア API を呼び出して S3 ロケーションにアクセスし、たとえばテーブルデータをクエリするときに発生します。

```
spark.sql("select * from test_tbl")
```

Amazon EMR AWS Lake Formation で S3 Access Grants を で使用すると、すべての表形式のアクセスパターンが Lake Formation を通過します。

IAM ロールにフォールバックする

S3 Access Grants がサポートしていないアクションをユーザーが実行しようとすると、Amazon EMR は、`fallbackToIAM` 設定が `true` の場合、ジョブ実行用に指定された IAM ロールをデフォルトとして使用します。これにより、S3 Access Grants が対象としていないシナリオでも、ユーザーはジョブ実行ロールにフォールバックして S3 アクセス用の認証情報を付与できます。

`fallbackToIAM` を有効にすると、ユーザーは Access Grants で許可されているデータにアクセスできます。ターゲットデータ用の S3 Access Grants トークンがない場合、Amazon EMR はジョブ実行ロールの権限を確認します。

> **Note**
>
> 本番環境のワークロードではこのオプションを無効にする予定がある場合でも、一度 `fallbackToIAM` 設定を有効にしてみてアクセス権限をテストすることをお勧めします。Spark ジョブでは、ユーザーが IAM 認証情報を使用してすべての権限セットにアクセスできる方法が他にもあります。EMR クラスターで S3 からの許可を有効にすると、Spark ジョブは S3 ロケーションにアクセスできます。これらの S3 ロケーションを EMRFS の外部からのアクセスから確実に保護する必要があります。たとえば、ノートブックで使用している S3 クライアントや、Hive や Presto などの S3 Access Grants でサポートされていないアプリケーションからのアクセスから S3 ロケーションを保護する必要があります。

Amazon EMR クラスターノードへのアクセス認証

SSH クライアントは、クラスターインスタンスへのアクセス認証に Amazon EC2 キーペアを使用します。または、Amazon EMR リリース 5.10.0 以降では、Kerberos を設定してユーザーおよびプライマリノードへの SSH 接続を認証することができます。また、Amazon EMR リリース 5.12.0 以降では、LDAP を使用して認証することができます。

トピック

- SSH 認証情報に EC2 キーペアを使用する
- Amazon EMR での認証に Kerberos を使用する
- Amazon EMR での認証に Active Directory または LDAP サーバーを使用する

SSH 認証情報に EC2 キーペアを使用する

Amazon EMR クラスターノードは、Amazon EC2 インスタンスで実行されます。クラスターノードには、Amazon EC2 インスタンスへの接続と同じ方法で接続できます。Amazon EC2 を使用してキーペアを作成するか、またはキーペアをインポートできます。クラスター作成時には、すべてのクラスターインスタンスへの SSH 接続に使用される Amazon EC2 キーペアを指定できます。キーペアを指定せずにクラスターを作成することもできます。この手順は、通常、自動的に起動および作動して段階的処理を実行した後に終了する一時クラスターの場合に実施します。

クラスターへの接続に使用する SSH クライアントには、このキーペアに関連付けられた秘密鍵ファイルが必要です。このファイルは、Linux、Unix および macOS を使用する SSH 接続の場合、.pem ファイルとなります。アクセス権限は、キーの所有者のみが該当ファイルにアクセスできるよう

に設定する必要があります。これは Windows を使用する SSH クライアントでは .ppk ファイルです。.ppk ファイルは、通常は .pem ファイルから作成されます。

- Amazon EC2 キーペアの作成の詳細については、Amazon EC2 ユーザーガイド」の「Amazon EC2 キーペアAmazon EC2」を参照してください。
- PuTTYgen を使用して .pem ファイルから .ppk ファイルを作成する手順については、Amazon EC2 ユーザーガイドのPuTTYgen を使用したプライベートキーの変換」を参照してください。
- .pem ファイルのアクセス許可の設定、および Linux または macOS からの 、Windows sshからの PuTTY、またはサポートされているオペレーティングシステム AWS CLI からの など、さまざまな方法を使用して EMR クラスターのプライマリノードに接続する方法の詳細については、「」を参照してくださいSSH を使用してプライマリノードに接続する。 macOS

Amazon EMR での認証に Kerberos を使用する

Amazon EMR リリース 5.10.0 以降では、Kerberos がサポートされています。Kerberos は、シークレットキーの暗号化を使用して強力な認証を提供するネットワーク認証プロトコルであり、パスワードやその他の認証情報が暗号化されていない形式でネットワーク経由で送信されることはありません。

Kerberos では、認証を必要とするサービスやユーザーはプリンシパルと呼ばれます。プリンシパルは Kerberos 領域内に存在します。キー配布センター (KDC) として知られる Kerberos サーバーは、この領域内で、プリンシパルを認証する手段を提供します。KDC はチケットを発行してこの認証を行います。KDC は、領域内にあるプリンシパルのデータベースに加え、プリンシパルのパスワードや、各プリンシパルに関するその他の管理情報を保持しています。KDC は、他の領域からプリンシパルの認証情報を受け入れることもできます。これは、クロス領域信頼と呼ばれます。また、EMR クラスターはプリンシパルを認証するために外部の KDC を使用できます。

クロス領域信頼を確立するため、あるいは外部の KDC を使用するためによくあるシナリオは、Active Directory ドメインからユーザーを認証することです。これにより、ユーザーは SSH を使用してクラスターに接続する場合やビッグデータアプリケーションで作業する場合に、ドメインアカウントを使用して EMR クラスターにアクセスできます。

Kerberos 認証を使用する場合、Amazon EMR はクラスターにインストールされているアプリケーションやコンポーネント、サブシステムに Kerberos を設定し、相互に認証できるようにします。

> **Important**
>
> Amazon EMR は、クロス領域信頼 AWS Directory Service for Microsoft Active Directory または外部 KDC では をサポートしていません。

Amazon EMR を使用して Kerberos を設定する前に、KDC で実行される Kerberos サービスの概念、および Kerberos サービスの管理ツールに慣れておくことをお勧めします。詳細については、Kerberos Consortium によって発行されている MIT Kerberos ドキュメントを参照してください。

トピック

サポートされているアプリケーション

EMR クラスターで、Kerberos プリンシパルは、すべてのクラスターノードで実行されているビッグデータのアプリケーションサービスとサブシステムです。Amazon EMR は、Kerberos を使用するために、以下に表示されているアプリケーションとコンポーネントを設定します。各アプリケーションには、Kerberos ユーザープリンシパルが関連付けられています。

Amazon EMR は、 AWS Directory Service for Microsoft Active Directoryを使用したクロス領域信頼をサポートしません。

Amazon EMR では、以下に示すアプリケーションやコンポーネント向けにオープンソースの Kerberos 認証機能の設定のみを行います。インストールされているその他のアプリケーションはいずれも Kerberos 認証されていません。そのため、Kerberos 認証済みのコンポーネントと通信できず、アプリケーションエラーが発生することがあります。Kerberos 認証されていないアプリケーションおよびコンポーネントの認証は無効です。サポートされるアプリケーションやコンポーネントは、Amazon EMR リリースによって異なる場合があります。

Livy ユーザーインターフェイスは、Kerberos 認証済みクラスターでホストされている唯一のウェブユーザーインターフェイスです。

- Hadoop MapReduce
- Hbase
- HCatalog
- HDFS
- [Hive]
  - LDAP 認証を使用して Hive を有効にしないでください。これが原因で、Kerberos 認証済みの YARN との通信に問題が生じることがあります。
- Hue
  - Hue ユーザー認証は自動的に設定されません。設定するには、設定 API を使用します。
  - Hue サーバーは Kerberos 認証済みです。Hue フロントエンド (UI) の認証が設定されていません。Hue UI には LDAP 認証を設定できます。
- Livy
  - Kerberos 認証済みクラスターを使用した Livy 偽装は、Amazon EMR リリース 5.22.0 以降でサポートされています。
- Oozie
- フェニックス
- Presto
  - Presto は、Amazon EMR リリース 6.9.0 以降での Kerberos 認証をサポートしています。
  - Presto で Kerberos 認証を使用するには、転送時の暗号化を有効にする必要があります。
- Spark
- Tez
- Trino
  - Trino は、Amazon EMR リリース 6.11.0 以降での Kerberos 認証をサポートしています。
  - Trino で Kerberos 認証を使用するには、転送時の暗号化を有効にする必要があります。
- YARN
- Zeppelin
  - Zeppelin は、必ず Kerberos と Spark インタプリタを一緒に使用することを目的として設定されています。他のインタプリタには設定されません。

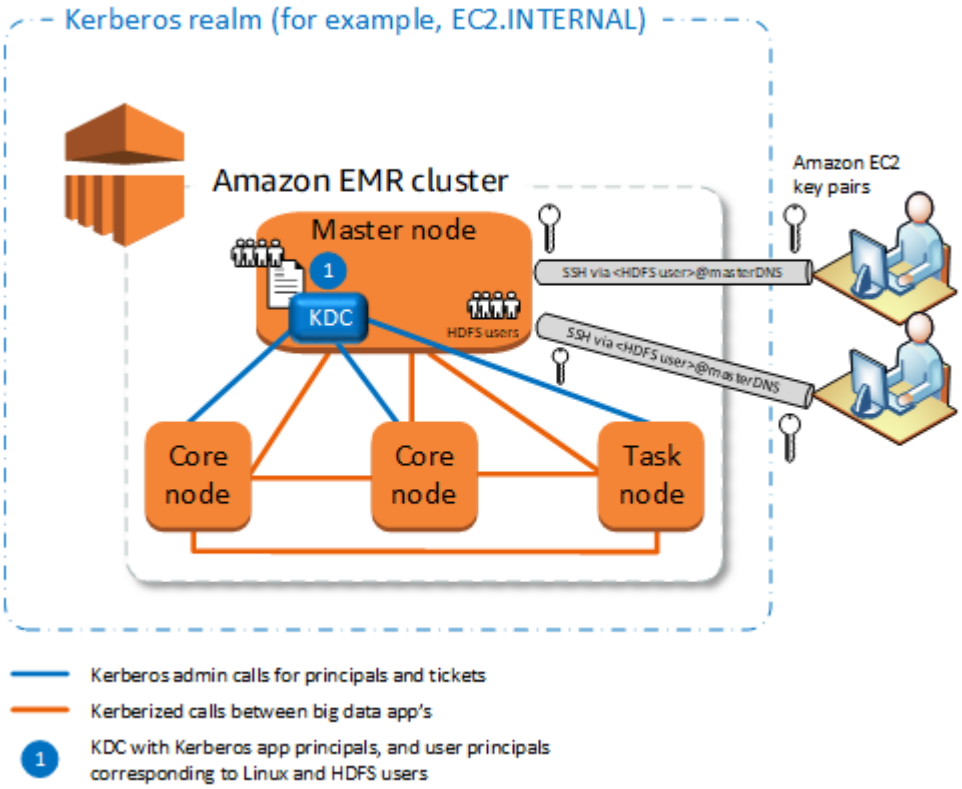
  - Spark 以外の Kerberos 認証済み Zeppelin インタプリタでは、ユーザー偽装はサポートされていません。
- Zookeeper
  - Zookeeper クライアントはサポートされていません。

Kerberos アーキテクチャオプション

Amazon EMR で Kerberos を使用する場合、このセクションに記載されているアーキテクチャから選択できます。選択したアーキテクチャに関係なく、同じ手順を使用して Kerberos を設定します。セキュリティ設定を作成し、セキュリティ設定を指定して、クラスターを作成する場合には互換性のあるクラスターに対応する Kerberos オプションを指定します。また、KDC のユーザープリンシパルに一致するクラスターで Linux 用の HDFS ディレクトリを作成します。設定オプションおよび各アーキテクチャの設定例については、「Amazon EMR での Kerberos の設定」を参照してください。

クラスター専用 KDC (プライマリノードでの KDC)

この設定は、Amazon EMR リリース 5.10.0 以降で使用できます。



利点

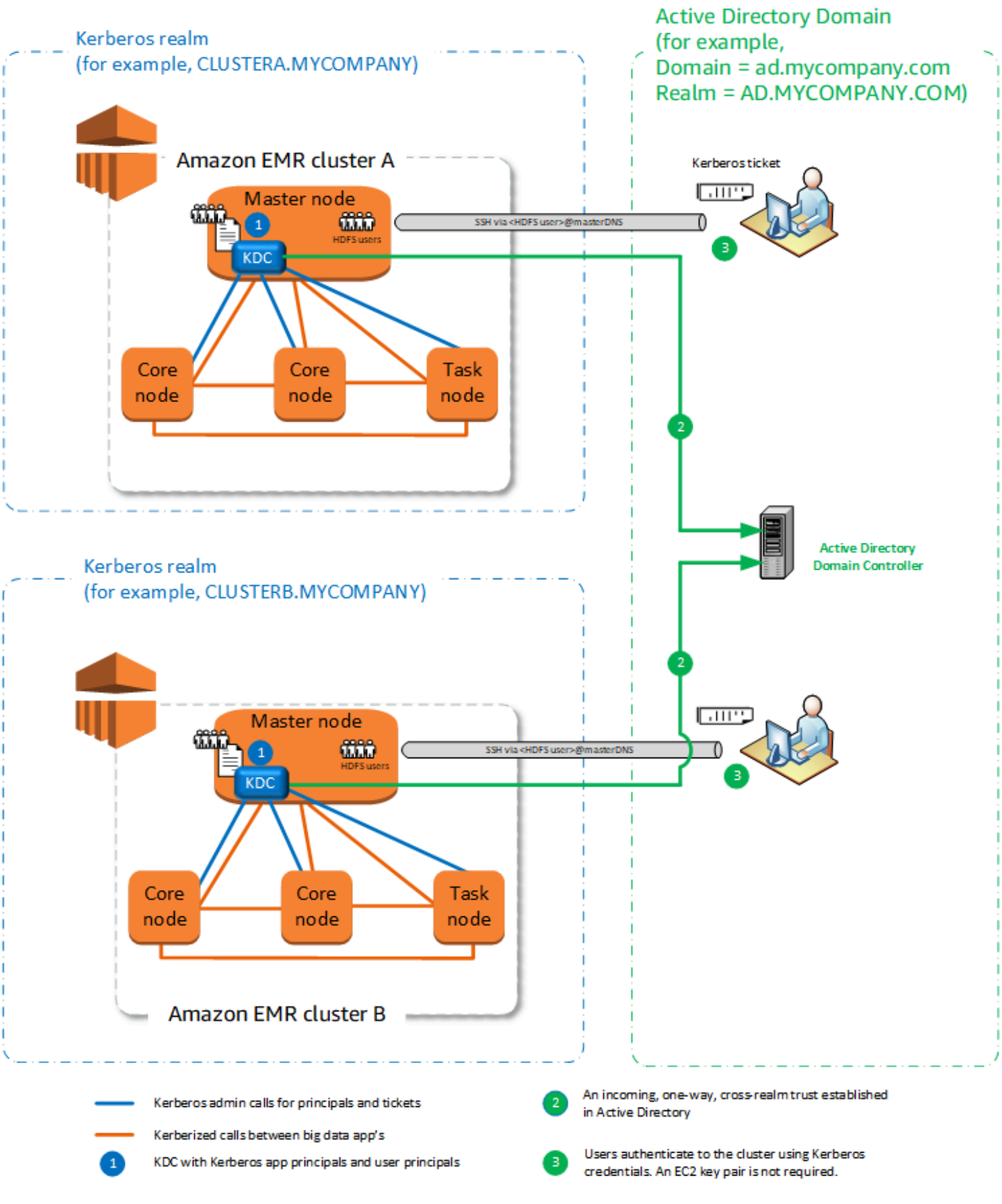
- Amazon EMR には KDC の完全な所有権があります。
- EMR クラスターの KDC は、 AWS Managed Microsoft ADの Microsoft Active Directory などの一元化された KDC 実装からは独立したものです。
- KDC はクラスター内のローカルノードのみの認証を管理するため、パフォーマンスへの影響は最小限に抑えられます。
- 必要に応じて、Kerberos 認証済みの他のクラスターは KDC を外部の KDC として参照できます。詳細については、「外部 KDC – 別のクラスターのプライマリノード」を参照してください。

考慮事項と制約事項

- Kerberos 認証済みクラスターは相互に認証できないため、アプリケーションを相互運用することはできません。クラスターのアプリケーションを相互運用する場合には、クラスター間でクロス領域信頼を確立するか、または 1 つのクラスターを他のクラスターの外部 KDC をして設定する必要があります。クロス領域信頼が確立された場合、KDC には 異なる Kerberos 領域がある必要があります。
- KDC ユーザープリンシパルに対応するプライマリノードの EC2 インスタンス上で Linux ユーザーを作成し、各ユーザー用の HDFS ディレクトリを作成する必要があります。
- ユーザープリンシパルは、SSH を使用してクラスターに接続するために、EC2 プライベートキーファイルおよび `kinit` 認証情報を使用する必要があります。

クロス領域信頼

この設定では、別の Kerberos 領域のプリンシパル (通常はユーザー) が、独自の KDC がある Kerberos 認証済みの EMR クラスター上のアプリケーションコンポーネントを認証します。プライマリノード上の KDC は、両方の KDC に存在するクロス領域のプリンシパルを使用して、別の KDC との信頼関係を確立します。各 KDC ではプリンシパル名とパスワードが正確に一致しています。次の図で示すように、クロス領域信頼は最も一般的な Active Directory の実装です。クロス領域信頼を外部の MIT KDC あるいは別の Amazon EMR クラスター上の KDC と使用することもサポートされています。


Active Directory Domain
(for example,
Domain = ad.mycompany.com
Realm = AD.MYCOMPANY.COM)
Kerberos realm
(for example, CLUSTERA.MYCOMPANY)
Amazon EMR cluster A
Master node
1
KDC
HDFS users
SSH via <HDFS user>@masterDNS
Kerberos ticket
3
Core node
Core node
Task node
2
Active Directory Domain Controller
Kerberos realm
(for example, CLUSTERB.MYCOMPANY)
Master node
1
KDC
HDFS users
SSH via <HDFS user>@masterDNS
2
3
Core node
Core node
Task node
Amazon EMR cluster B
Kerberos admin calls for principals and tickets
Kerberized calls between big data app's
1
KDC with Kerberos app principals and user principals
2
An incoming, one-way, cross-realm trust established in Active Directory
3
Users authenticate to the cluster using Kerberos credentials. An EC2 key pair is not required.

利点

- KDC がインストールされた EMR クラスターには、その KDC の完全な所有権があります。
- Active Directory を使用すると、Amazon EMR は KDC のユーザープリンシパルに対応する Linux ユーザーを自動的に作成します。ただし、各ユーザーに HDFS ディレクトリも作成する必要があります。さらに、Active Directory ドメインのユーザープリンシパルは、EC2 プライベートキーファイルの必要なく、`kinit` 認証情報を使用して Kerberos 認証済みクラスターにアクセスできます。これにより、クラスターユーザー間でプライベートキーを共有する必要がなくなります。
- 各クラスター KDC はクラスター内のノードの認証情報を管理するため、ネットワークのレイテンシーおよびクラスター全体における多数のノード数の処理オーバーヘッドから生じる影響は最低限に抑えられます。

考慮事項と制約事項

- Active Directory 領域を使用して信頼性を確立する場合、クラスター作成時に、ドメインにプリンシパルを結合する許可がある Active Directory のユーザー名とパスワードを提供する必要があります。
- クロス領域信頼を同名の Kerberos 領域間に確立することはできません。
- クロス領域信頼は明示的に確立する必要があります。たとえば、クラスター A とクラスター B の両方で KDC にクロス領域信頼を確立する場合、クラスター間で本質的にお互いを信頼することはできず、また、そのアプリケーションは相互に認証し合うことはできません。
- ユーザープリンシパルの認証情報が正確に一致するように、KDC は個別に管理して整合する必要があります。

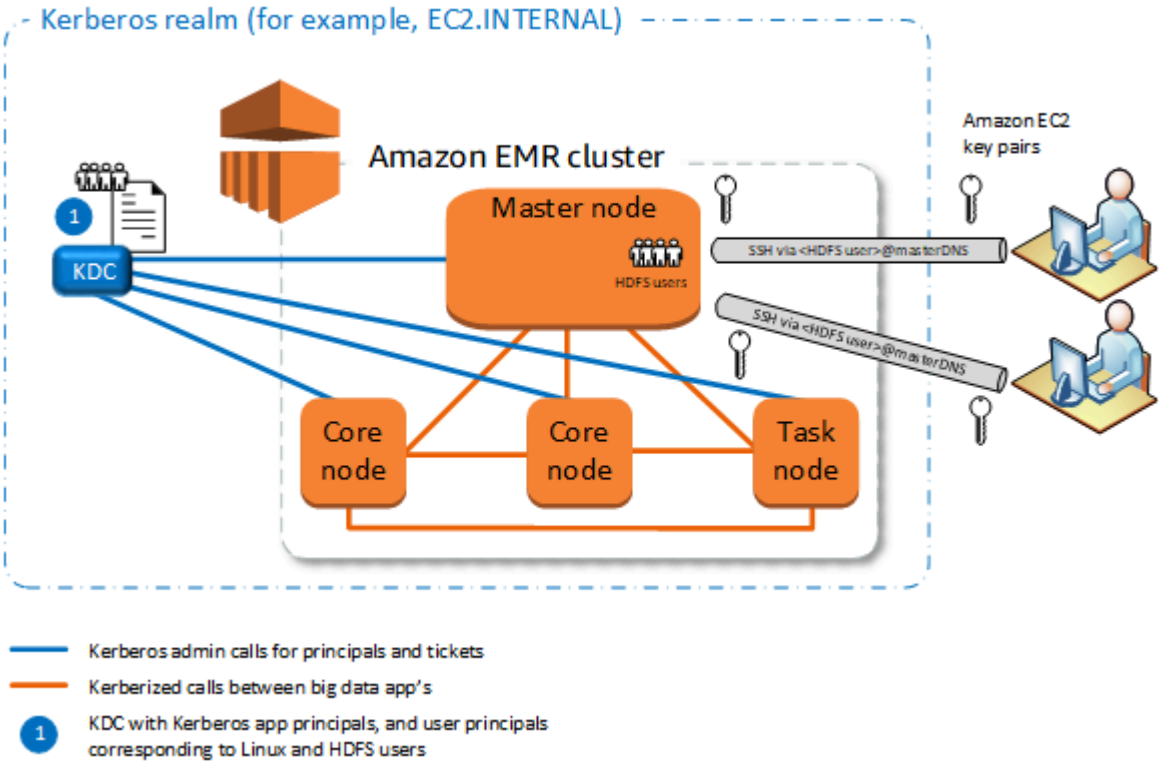
外部 KDC

外部の KDC を使用した設定は、Amazon EMR 5.20.0 以降でサポートされます。

- 外部 KDC - MIT KDC
- 外部 KDC – 別のクラスターのプライマリノード
- 外部 KDC - Active Directory クロス領域信頼を使用した別のクラスター上のクラスター KDC

外部 KDC - MIT KDC

この設定によって、1 つ以上の EMR クラスターが MIT KDC サーバーで定義されて維持されるプリンシパルを使用することが許可されます。



利点

- プリンシパルの管理は単一の KDC に統合されています。
- 複数のクラスターは、同じ Kerberos 領域の同じ KDC を使用できます。詳細については、「同じ KDC で複数のクラスターを使用するための要件」を参照してください。
- Kerberos 認証済みクラスターのプライマリノードには、KDC の維持に関連するパフォーマンスの負担はありません。

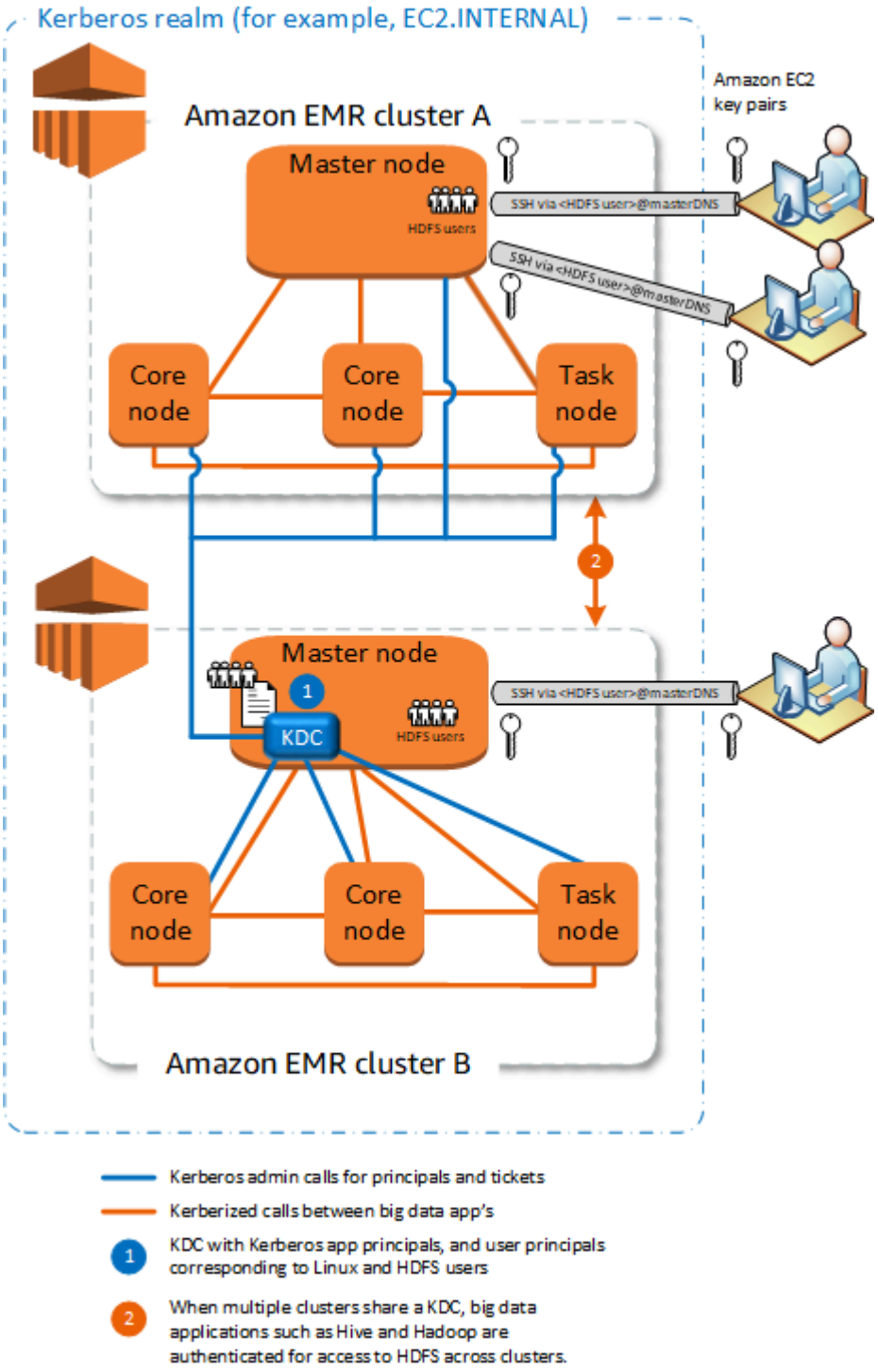
考慮事項と制約事項

- 各 Kerberos 認証済みクラスターのプライマリノード (KDC ユーザープリンシパルに対応するもの) の EC2 インスタンス上で Linux ユーザーを作成し、各ユーザー用の HDFS ディレクトリを作成する必要があります。
- ユーザープリンシパルは、SSH を使用して Kerberos 認証済みのクラスターに接続するために、EC2 プライベートキーファイルおよび `kinit` 認証情報を使用する必要があります。
- Kerberos 認証済みの EMR クラスター内の各ノードには、KDC へのネットワークルートがあることが必要です。

- Kerberos 認証済みのクラスターの各ノードは、KDC の設定がクラスターのパフォーマンスに影響を及ぼすように、外部 KDC に認証情報の負担を設置します。KDC サーバーのハードウェアを設定する際に、同時にサポートする Amazon EMR ノードの最大数を考慮してください。
- クラスターのパフォーマンスは、Kerberos 認証済みのクラスターと KDC のノード間のネットワークにおけるレイテンシーに応じます。
- 相互依存関係のため、トラブルシューティングはより困難になることがあります。

外部 KDC – 別のクラスターのプライマリノード

この設定は、KDC が EMR クラスターのプライマリノード上にあることを除いて、上記の外部 MIT KDC の実装とほぼ同じです。詳細については、「クラスター専用 KDC (プライマリノードでの KDC)」および「チュートリアル: Active Directory ドメインを使用したクロス領域信頼の設定」を参照してください。



利点

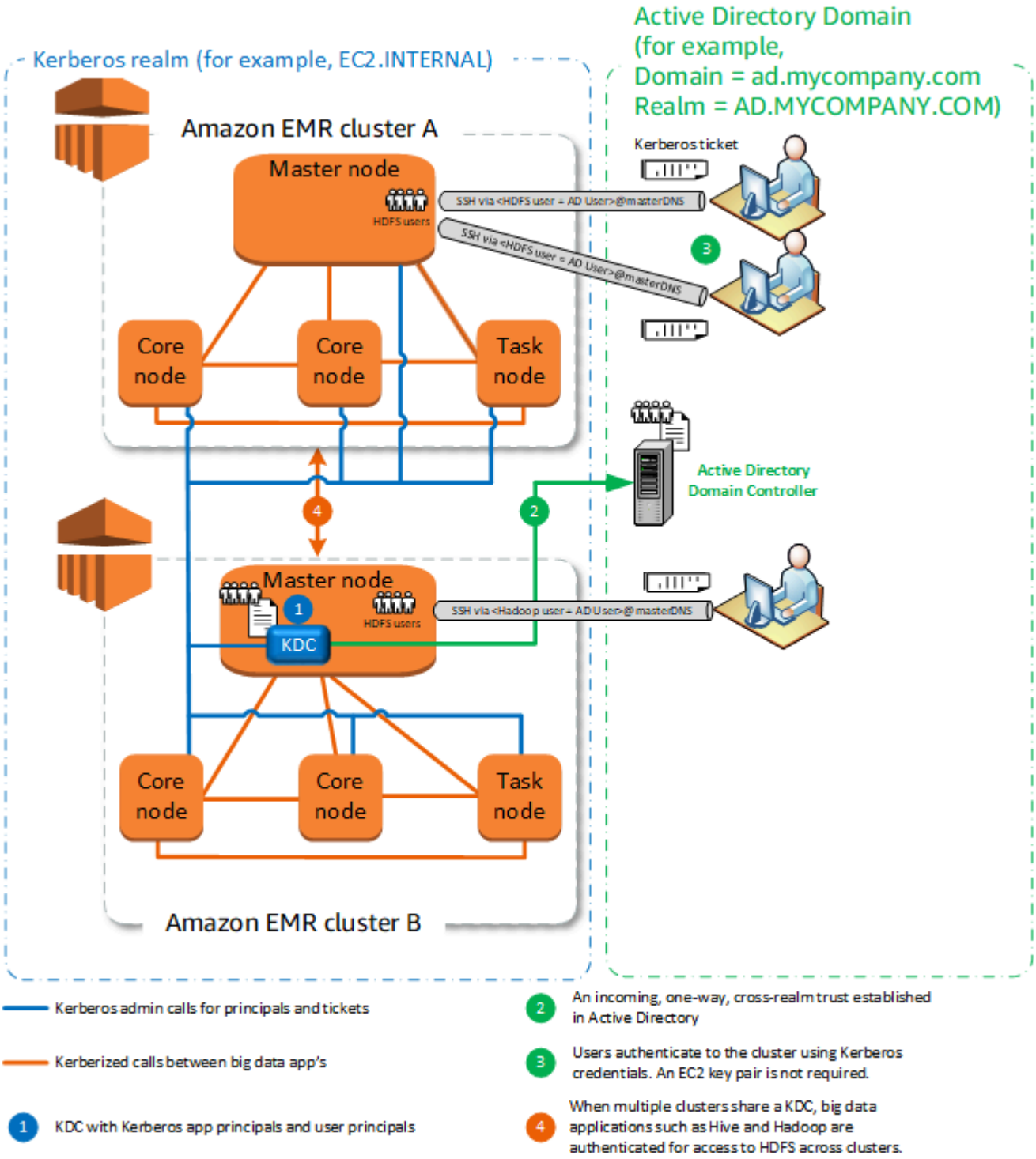
- プリンシパルの管理は単一の KDC に統合されています。
- 複数のクラスターは、同じ Kerberos 領域の同じ KDC を使用できます。詳細については、「同じ KDC で複数のクラスターを使用するための要件」を参照してください。

考慮事項と制約事項

- 各 Kerberos 認証済みクラスターのプライマリノード (KDC ユーザープリンシパルに対応するもの) の EC2 インスタンス上で Linux ユーザーを作成し、各ユーザー用の HDFS ディレクトリを作成する必要があります。
- ユーザープリンシパルは、SSH を使用して Kerberos 認証済みのクラスターに接続するために、EC2 プライベートキーファイルおよび `kinit` 認証情報を使用する必要があります。
- 各 EMR クラスター内の各ノードには、KDC へのネットワークルートがあることが必要です。
- Kerberos 認証済みのクラスターの各 Amazon EMR ノードは、KDC の設定がクラスターのパフォーマンスに影響を及ぼすように、外部 KDC に認証情報の負担を設置します。KDC サーバーのハードウェアを設定する際に、同時にサポートする Amazon EMR ノードの最大数を考慮してください。
- クラスターのパフォーマンスは、クラスターと KDC のノード間のネットワークにおけるレイテンシーに応じます。
- 相互依存関係のため、トラブルシューティングはより困難になることがあります。

外部 KDC - Active Directory クロス領域信頼を使用した別のクラスター上のクラスター KDC

この設定ではまず、Active Directory と一方通行のクロス領域信頼があるクラスター専用の KDC を使用して、クラスターを作成します。詳細なチュートリアルについては、「チュートリアル: Active Directory ドメインを使用したクロス領域信頼の設定」を参照してください。次に、外部 KDC としての信頼があるクラスター KDC を参照する追加のクラスターを起動します。例については、Active Directory クロス領域信頼を使用した外部のクラスター KDCを参照してください。これにより、外部の KDC を使用する各 Amazon EMR クラスターが Microsoft Active Directory ドメインで定義されて維持されるプリンシパルを認証することが許可されます。



利点

- プリンシパルの管理は Active Directory ドメインに統合されています。

- Amazon EMR は Active Directory 領域を結合するため、Active Directory ユーザーに対応する Linux ユーザーを作成する必要がなくなります。ただし、各ユーザーに HDFS ディレクトリも作成する必要があります。
- 複数のクラスターは、同じ Kerberos 領域の同じ KDC を使用できます。詳細については、「同じ KDC で複数のクラスターを使用するための要件」を参照してください。
- Active Directory ドメインのユーザープリンシパルは、EC2 プライベートキーファイルの必要なく、`kinit` 認証情報を使用して Kerberos 認証済みクラスターにアクセスできます。これにより、クラスターユーザー間でプライベートキーを共有する必要がなくなります。
- KDC の維持を負担するのは 1 つの Amazon EMR プライマリノードだけであり、KDC と Active Directory の間のクロス領域信頼を確保するためには、そのクラスターのみを Active Directory 認証情報を使用して作成する必要があります。

考慮事項と制約事項

- 各 EMR クラスターの各ノードには、KDC および Active Directory ドメインコントローラーへのネットワークルートがあることが必要です。
- 各 Amazon EMR ノードは、KDC の設定がクラスターのパフォーマンスに影響を及ぼすように、外部 KDC に認証情報の負担を設置します。KDC サーバーのハードウェアを設定する際に、同時にサポートする Amazon EMR ノードの最大数を考慮してください。
- クラスターのパフォーマンスは、クラスターと KDC サーバーのノード間のネットワークにおけるレイテンシーに応じます。
- 相互依存関係のため、トラブルシューティングはより困難になることがあります。

同じ KDC で複数のクラスターを使用するための要件

複数のクラスターは、同じ Kerberos 領域の同じ KDC を使用できます。ただし、クラスターが同時に実行されると、競合する Kerberos ServicePrincipal 名を使用するとクラスターが失敗する可能性があります。

同じ外部 KDC を持つ複数の同時クラスターがある場合は、それらのクラスターで異なる Kerberos 領域が使用されていることを確認してください。これらのクラスターで同じ Kerberos 領域を使用する必要がある場合は、クラスターが異なるサブネット内にあり、CIDR 範囲が重複していないことを確認してください。

Amazon EMR での Kerberos の設定

このセクションでは、一般的なアーキテクチャを使用して Kerberos を設定する詳細と例を示します。選択するアーキテクチャに関係なく設定の基本は同じであり、3 つのステップで行います。外部の KDC を使用する、あるいはクロス領域信頼を設定する場合には、クラスターのすべてのノードに外部の KDC へのネットワークルートがあること (インバウンドおよびアウトバウンドの Kerberos トラフィックを許可する適用されるセキュリティグループの設定など) を確認する必要があります。

ステップ 1: Kerberos プロパティでセキュリティ設定を作成する

このセキュリティ設定は Kerberos KDC についての詳細を指定し、クラスターを作成するたびにこの Kerberos 設定を再利用できるようにします。セキュリティ設定は、Amazon EMR コンソール、AWS CLI、または EMR API を使用して作成できます。セキュリティ設定には、他にも暗号化などのセキュリティオプションがあります。セキュリティ設定の作成、およびクラスター作成時のセキュリティ設定の指定に関する詳細については、「セキュリティ設定を使用してクラスターセキュリティをセットアップする」を参照してください。セキュリティ設定の Kerberos プロパティに関する詳細は、「セキュリティ設定における Kerberos セキュリティ」を参照してください。

ステップ 2: クラスターを作成し、クラスター固有の Kerberos 属性を指定する

クラスター作成時に、クラスター固有の Kerberos オプションと Kerberos セキュリティ設定を指定します。Amazon EMR コンソールを使用する場合、指定するセキュリティ設定と互換性がある Kerberos オプションのみを使用できます。 AWS CLI または Amazon EMR API を使用する場合は、指定したセキュリティ設定と互換性のある Kerberos オプションを必ず指定してください。たとえば、CLI を使用してクラスターを作成する際にクロス領域信頼にプリンシパルのパスワードを指定するとき、指定したセキュリティ設定がクロス領域信頼で設定されていない場合、エラーが生じます。詳細については、「クラスターの Kerberos 設定」を参照してください。

ステップ 3: クラスターのプライマリノードを設定する

アーキテクチャと実装の要件に応じて、クラスターで追加の設定が必要になることがあります。これは、クラスターの作成後、あるいは作成プロセス中にステップまたはブートストラップを使用して行うことができます。

SSH を使用してクラスターに接続した Kerberos 認証済みの各ユーザーについて、Kerberos ユーザーに対応する Linux アカウントが作成されていることを確認する必要があります。ユーザープリンシパルが外部 KDC として、あるいはクロス領域信頼を介して Active Directory ドメインコントローラーによって提供されている場合、Amazon EMR は Linux アカウントを自動的に作成します。Active Directory が使用されていない場合は、各ユーザーに Linux ユーザーに対応するプリンシ

パルを作成する必要があります。詳細については、「Kerberos 認証済み HDFS ユーザーと SSH 接続向けクラスターの設定」を参照してください。

また、各ユーザーには所有する HDFS ユーザーディレクトリ (要作成) がある必要があります。さらに、Kerberos 認証済みのユーザーからの接続を許可するように、SSH は GSSAPI を有効にして設定する必要があります。GSSAPI がプライマリノードで有効にされている必要があり、クライアントの SSH アプリケーションが GSSAPI を使用するように設定されている必要があります。詳細については、「Kerberos 認証済み HDFS ユーザーと SSH 接続向けクラスターの設定」を参照してください。

Amazon EMR 上の Kerberos のセキュリティ設定およびクラスター設定

Kerberos 認証済みのクラスターを作成する場合、セキュリティ設定と合わせて、クラスター固有の Kerberos 属性を指定します。必ず、他の属性と一緒に指定します。それ以外の場合はエラーが発生します。

このトピックでは、セキュリティ設定およびクラスターを作成するときに、Kerberos で利用できる設定パラメータの概要を示します。また、互換性のあるセキュリティ設定およびクラスター作成用の CLI 例は、一般的なアーキテクチャ向けに提供されています。

セキュリティ設定における Kerberos セキュリティ

Amazon EMR コンソール、、 AWS CLIまたは EMR API を使用して、Kerberos 属性を指定するセキュリティ設定を作成できます。セキュリティ設定には、他にも暗号化などのセキュリティオプションがあります。詳細については、「セキュリティ設定を作成する」を参照してください。

次のリファレンスを使用して、選択する Kerberos アーキテクチャで利用できるセキュリティ構成の設定を理解します。Amazon EMR コンソールの設定が表示されます。対応する CLI オプションについては、「を使用した Kerberos 設定の指定 AWS CLI」または「設定例」を参照してください。

パラメータ		説明
Kerberos		このセキュリティ設定を使用するクラスターで Kerberos を有効にすることを指定します。クラスターがこのセキュリティ設定を使用する場合、クラスターで Kerberos 設定も指定する必要があります。そうしないと、エラーが発生します。
プロバイダー	クラスター専用 KDC	Amazon EMR が、このセキュリティ設定を使用するクラスターのプライマリノードに KDC を作成するこ

パラメータ		説明
		とを指定します。領域名と KDC 管理者パスワードは、クラスターの作成時に指定します。 必要に応じて、この KDC を他のクラスターから参照できます。異なるセキュリティ設定を使用してこれらのクラスターを作成し、外部 KDC を指定し、クラスター専用 KDC に指定した領域名と KDC 管理者パスワードを使用します。
	外部 KDC	Amazon EMR 5.20.0 以降でのみ利用できます。このセキュリティ設定を使用するクラスターが、クラスター外の KDC サーバーを使用して Kerberos プリンシパルを認証するように指定します。KDC はクラスター上に作成されません。クラスターの作成時に、外部 KDC の領域名と KDC 管理者パスワードを指定します。
チケットのライフタイム		オプション。このセキュリティ設定を使用するクラスターで KDC によって発行された Kerberos チケットが有効である期間を指定します。 チケットの有効期間は、セキュリティ上の理由により制限されます。クラスターアプリケーションとサービスでは、期限が切れるとチケットを自動更新します。Kerberos 認証情報を使用して SSH 経由でクラスターに接続する場合は、チケットの有効期限が切れたら、プライマリノードのコマンドラインから `kinit` を実行して更新する必要があります。

パラメータ		説明
クロス領域信頼		このセキュリティ設定を使用するクラスター上のクラスター専用 KDC と、異なる Kerberos 領域内の KDC との間のクロス領域信頼を指定します。 別の領域のプリンシパル (通常はユーザー) は、この設定を使用するクラスターに対して認証されます。他の Kerberos 領域での追加設定が必要です。詳細については、「チュートリアル: Active Directory ドメインを使用したクロス領域信頼の設定」を参照してください。
クロス領域信頼プロパティ	領域	信頼関係の他の領域の Kerberos 領域名を指定します。慣例により、Kerberos 領域名はドメイン名と同じにします。ただし、すべて大文字にします。
	[ドメイン]	信頼関係の他の領域のドメイン名を指定します。
	[Admin server] (管理者サーバー)	信頼関係の他の領域の管理サーバーの完全修飾ドメイン名 (FQDN) または IP アドレスを指定します。通常、管理サーバーと KDC サーバーは同じ FQDN を持つ同じマシン上で実行されますが、通信には別のポートを使用します。 ポートを指定しない場合、Kerberos のデフォルトであるポート 749 が使用されます。オプションで、ポート (`domain.example.com :749` など) を指定できます。

パラメータ		説明
	[KDC server] (KDC サーバー)	信頼関係の他の領域の KDC サーバーの完全修飾ドメイン名 (FQDN) または IP アドレスを指定します。通常、KDC サーバーと管理サーバーは同じ FQDN を持つ同じマシン上で実行されますが、別のポートを使用します。 ポートを指定しない場合、Kerberos のデフォルトであるポート 88 が使用されます。オプションで、ポート (`domain.example.com :88` など) を指定できます。
外部 KDC		クラスター外部 KDC がクラスターで使用されることを指定します。
外部 KDC プロパティ	[Admin server] (管理者サーバー)	外部管理サーバーの完全修飾ドメイン名 (FQDN) または IP アドレスを指定します。通常、管理サーバーと KDC サーバーは同じ FQDN を持つ同じマシン上で実行されますが、通信には別のポートを使用します。 ポートを指定しない場合、Kerberos のデフォルトであるポート 749 が使用されます。オプションで、ポート (`domain.example.com :749` など) を指定できます。
	[KDC server] (KDC サーバー)	外部 KDC サーバーの完全修飾ドメイン名 (FQDN) を指定します。通常、KDC サーバーと管理サーバーは同じ FQDN を持つ同じマシン上で実行されますが、別のポートを使用します。 ポートを指定しない場合、Kerberos のデフォルトであるポート 88 が使用されます。オプションで、ポート (`domain.example.com :88` など) を指定できます。

パラメータ			説明
	[Active Directory Integration] (Active Directory の統合)		Kerberos プリンシパル認証が Microsoft Active Directory ドメインに統合されることを指定します。
	Active Directory 統合プロパティ	[Active Directory realm] (Active Directory 領域)	Active Directory ドメインの Kerberos 領域名を指定します。慣例により、Kerberos 領域名は通常、ドメイン名と同じにします。ただし、すべて大文字にします。
		[Active Directory domain] (Active Directory ドメイン)	Active Directory ドメイン名を指定します。
		[Active Directory server] (Active Directory サーバー)	Microsoft Active Directory ドメインコントローラーの完全修飾ドメイン名 (FQDN) を指定します。

クラスターの Kerberos 設定

Amazon EMR コンソール、、 AWS CLIまたは EMR API を使用してクラスターを作成するときに、Kerberos 設定を指定できます。

次のリファレンスを使用して、選択する Kerberos アーキテクチャで利用できるクラスター構成の設定を理解します。Amazon EMR コンソールの設定が表示されます。対応する CLI オプションについては、「設定例」を参照してください。

パラメータ	説明

パラメータ	説明
領域	クラスターの Kerberos 領域名。Kerberos の規則では、ドメイン名と同じ名前に設定する必要がありますが、大文字にします。たとえば、ドメイン (`ec2.internal`) の場合は、領域名に `EC2.INTERNAL` を使用します。
KDC 管理者パスワード	`kadmin` または `kadmin.local` 向けにクラスター内で使用されるパスワード。以下は、Kerberos V5 管理システムのコマンドラインインターフェイスです。Kerberos プリンシパル、パスワードポリシー、クラスターのキータブを管理できます。
クロス領域信頼プリンシパルのパスワード (オプション)	クロス領域信頼の確立時に必要。クロス領域プリンシパルのパスワード。領域内では同一である必要があります。強力なパスワードを選択します。
Active Directory ドメイン結合ユーザー (オプション)	クロス領域信頼で Active Directory を使用するときに必要です。これは、コンピュータをドメインに結合するアクセス許可がある Active Directory アカウントのユーザーログオン名です。Amazon EMR は、この識別子を使用して、クラスターをドメインに結合します。詳細については、「the section called “ステップ 3: EMR クラスターのドメインにアカウントを追加する”」を参照してください。
Active Directory ドメイン結合パスワード (オプション)	Active Directory ドメイン結合ユーザーのパスワード。詳細については、「the section called “ステップ 3: EMR クラスターのドメインにアカウントを追加する”」を参照してください。

設定例

以下の例は、一般的な scenarios. AWS CLI commands のセキュリティ設定とクラスター設定を簡略化するために示しています。

ローカル KDC

次のコマンドは、プライマリノード上で実行されているクラスター専用 KDC を備えたクラスターを作成します。クラスターでの追加設定が必要になります。詳細については、「Kerberos 認証済み HDFS ユーザーと SSH 接続向けクラスターの設定」を参照してください。

セキュリティ設定の作成

```
aws emr create-security-configuration --name LocalKDCSecurityConfig \
--security-configuration '{"AuthenticationConfiguration": \
{"KerberosConfiguration": {"Provider": "ClusterDedicatedKdc",\
"ClusterDedicatedKdcConfiguration": {"TicketLifetimeInHours": 24 }}}}'
```

クラスターの作成

```
aws emr create-cluster --release-label emr-7.1.0 \
--instance-count 3 --instance-type m5.xlarge \
--applications Name=Hadoop Name=Hive --ec2-attributes
 InstanceProfile=EMR_EC2_DefaultRole,KeyName=MyEC2Key \
--service-role EMR_DefaultRole \
--security-configuration LocalKDCSecurityConfig \
--kerberos-attributes Realm=EC2.INTERNAL,KdcAdminPassword=MyPassword
```

Active Directory クロス領域信頼を使用したクラスター専用 KDC

次のコマンドは、Active Directory ドメインへのクロス領域信頼を持つ、プライマリノード上で実行されているクラスター専用 KDC を備えたクラスターを作成します。クラスターと Active Directory における追加設定が必要になります。詳細については、「チュートリアル: Active Directory ドメインを使用したクロス領域信頼の設定」を参照してください。

セキュリティ設定の作成

```
aws emr create-security-configuration --name LocalKDCWithADTrustSecurityConfig \
--security-configuration '{"AuthenticationConfiguration": \
{"KerberosConfiguration": {"Provider": "ClusterDedicatedKdc", \
"ClusterDedicatedKdcConfiguration": {"TicketLifetimeInHours": 24, \
"CrossRealmTrustConfiguration": {"Realm":"AD.DOMAIN.COM", \
```

```
"Domain":"ad.domain.com", "AdminServer":"ad.domain.com", \
"KdcServer":"ad.domain.com"}}}}}'
```

クラスターの作成

```
aws emr create-cluster --release-label emr-7.1.0 \
--instance-count 3 --instance-type m5.xlarge --applications Name=Hadoop Name=Hive \
--ec2-attributes InstanceProfile=EMR_EC2_DefaultRole,KeyName=MyEC2Key \
--service-role EMR_DefaultRole --security-configuration KDCWithADTrustSecurityConfig \
--kerberos-attributes Realm=EC2.INTERNAL,KdcAdminPassword=MyClusterKDCAdminPassword,\
ADDomainJoinUser=ADUserLogonName,ADDomainJoinPassword=ADUserPassword,\
CrossRealmTrustPrincipalPassword=MatchADTrustPassword
```

別のクラスター上の外部 KDC

次のコマンドは、プリンシパルを認証するために、別のクラスターのプライマリノード上のクラスター専用 KDC を参照するクラスターを作成します。クラスターでの追加設定が必要になります。詳細については、「Kerberos 認証済み HDFS ユーザーと SSH 接続向けクラスターの設定」を参照してください。

セキュリティ設定の作成

```
aws emr create-security-configuration --name ExtKDCOnDifferentCluster \
--security-configuration '{"AuthenticationConfiguration": \
{"KerberosConfiguration": {"Provider": "ExternalKdc", \
"ExternalKdcConfiguration": {"KdcServerType": "Single", \
"AdminServer": "MasterDNSOfKDCMaster:749", \
"KdcServer": "MasterDNSOfKDCMaster:88"}}}}'
```

クラスターの作成

```
aws emr create-cluster --release-label emr-7.1.0 \
--instance-count 3 --instance-type m5.xlarge \
--applications Name=Hadoop Name=Hive \
--ec2-attributes InstanceProfile=EMR_EC2_DefaultRole,KeyName=MyEC2Key \
--service-role EMR_DefaultRole --security-configuration ExtKDCOnDifferentCluster \
--kerberos-attributes Realm=EC2.INTERNAL,KdcAdminPassword=KDCOnMasterPassword
```

Active Directory クロス領域信頼を使用した外部のクラスター KDC

次のコマンドは KDC がないクラスターを作成します。このクラスターは、プリンシパルを認証するために、別のクラスターのプライマリノード上で実行されているクラスター専用 KDC を参照し

ます。この KDC には Active Directory ドメインコントローラーを使用したクロス領域信頼があります。KDC を備えたプライマリノードでの追加設定が必要です。詳細については、「チュートリアル: Active Directory ドメインを使用したクロス領域信頼の設定」を参照してください。

セキュリティ設定の作成

```
aws emr create-security-configuration --name ExtKDCWithADIntegration \
--security-configuration '{"AuthenticationConfiguration": \
{"KerberosConfiguration": {"Provider": "ExternalKdc", \
"ExternalKdcConfiguration": {"KdcServerType": "Single", \
"AdminServer": "MasterDNSofClusterKDC:749", \
"KdcServer": "MasterDNSofClusterKDC.com:88", \
"AdIntegrationConfiguration": {"AdRealm":"AD.DOMAIN.COM", \
"AdDomain":"ad.domain.com", \
"AdServer":"ad.domain.com"}}}}}'
```

クラスターの作成

```
aws emr create-cluster --release-label emr-7.1.0 \
--instance-count 3 --instance-type m5.xlarge --applications Name=Hadoop Name=Hive \
--ec2-attributes InstanceProfile=EMR_EC2_DefaultRole,KeyName=MyEC2Key \
--service-role EMR_DefaultRole --security-configuration ExtKDCWithADIntegration \
--kerberos-attributes Realm=EC2.INTERNAL,KdcAdminPassword=KDCOnMasterPassword,\
ADDomainJoinUser=MyPrivilegedADUserName,ADDomainJoinPassword=PasswordForADDomainJoinUser
```

Kerberos 認証済み HDFS ユーザーと SSH 接続向けクラスターの設定

Amazon EMR は、クラスターで実行されるアプリケーションに対して、Kerberos 認証済みユーザークライアント (例: hadoop ユーザー、spark ユーザーなど) を作成します。Kerberos を使用して、クラスタープロセスに認証されているユーザーを追加することもできます。これで、認証されたユーザーは、Kerberos 認証情報を使用してクラスターに接続し、アプリケーションを操作することができます。クラスターに対してユーザーを認証するには、次の設定が必要です。

- KDC で Kerberos プリンシパルと一致する Linux アカウントがクラスターにある必要があります。Amazon EMR は、Active Directory と統合しているアーキテクチャでこれを自動的に行いません。
- 各ユーザーに対してプライマリノード上で HDFS ユーザーディレクトリを作成し、このディレクトリへのアクセス許可をユーザーに付与する必要があります。
- GSSAPI がプライマリノードで有効化されるように SSH サービスを設定する必要があります。また、ユーザーには GSSAPI が有効化された SSH クライアントがある必要があります。

プライマリノードに Linux ユーザーと Kerberos プリンシパルを追加する

Active Directory を使用しない場合、クラスターのプライマリノードで Linux アカウントを作成し、これらの Linux ユーザーに対するプリンシパルを KDC に追加する必要があります。これには、プライマリノードに対する KDC のプリンシパルが含まれます。ユーザープリンシパルに加えて、プライマリノード上で実行されている KDC にはローカルホストに対するプリンシパルが必要です。

使用するアーキテクチャに Active Directory が統合されている場合、ローカル KDC の Linux ユーザーおよびプリンシパルは、該当する場合に自動的に作成されます。このため、このステップはスキップできます。詳細については、「クロス領域信頼」および「外部 KDC - Active Directory クロス領域信頼を使用した別のクラスター上のクラスター KDC」を参照してください。

> **Important**
>
> プライマリノードがエフェメラルストレージを使用するため、プライマリノードが終了すると、KDC はプリンシパルのデータベースと共に失われます。SSH 接続用のユーザーを作成する場合は、高可用性用に構成された外部 KDC を使用してクロス領域信頼を確立することをお勧めします。または、Linux アカウントを使用して SSH 接続用のユーザーを作成する場合は、ブートストラップアクションとスクリプトを使用してアカウント作成プロセスを自動化して、新しいクラスターを作成するときに繰り返すことができるようにします。

ユーザーおよび KDC プリンシパルを追加するには、クラスター作成後あるいは作成時にクラスターへステップを送信することが最も簡単な方法です。または、デフォルトの hadoop ユーザーとして EC2 キーペアを使用してプライマリノードに接続し、コマンドを実行することもできます。詳細については、「SSH を使用してプライマリノードに接続する」を参照してください。

次の例では、既存のクラスターに Bash スクリプト (configureCluster.sh) を送信し、クラスター ID を参照します。このスクリプトは Amazon S3 に保存されます。

```
aws emr add-steps --cluster-id <j-2AL4XXXXXX5T9> \
--steps Type=CUSTOM_JAR,Name=CustomJAR,ActionOnFailure=CONTINUE,\
Jar=s3://region.elasticmapreduce/libs/script-runner/script-runner.jar,\
Args=["s3://DOC-EXAMPLE-BUCKET/configureCluster.sh"]
```

次の例は、configureCluster.sh スクリプトの内容を示します。このスクリプトでは、HDFS ユーザーディレクトリの処理と SSH 向けの GSSAPI の有効化も処理します。これについては、次のセクションで説明します。

```
#!/bin/bash
#Add a principal to the KDC for the primary node, using the primary node's returned
 host name
sudo kadmin.local -q "ktadd -k /etc/krb5.keytab host/`hostname -f`"
#Declare an associative array of user names and passwords to add
declare -A arr
arr=([lijuan]=pwd1 [marymajor]=pwd2 [richardroe]=pwd3)
for i in ${!arr[@]}; do
    #Assign plain language variables for clarity
     name=${i}
     password=${arr[${i}]}

     # Create a principal for each user in the primary node and require a new password
 on first logon
     sudo kadmin.local -q "addprinc -pw $password +needchange $name"

     #Add hdfs directory for each user
     hdfs dfs -mkdir /user/$name

     #Change owner of each user's hdfs directory to that user
     hdfs dfs -chown $name:$name /user/$name
done

# Enable GSSAPI authentication for SSH and restart SSH service
sudo sed -i 's/^.*GSSAPIAuthentication.*$/GSSAPIAuthentication yes/' /etc/ssh/
sshd_config
sudo sed -i 's/^.*GSSAPICleanupCredentials.*$/GSSAPICleanupCredentials yes/' /etc/ssh/
sshd_config
sudo systemctl restart sshd
```

ユーザー HDFS ディレクトリの追加

ユーザーがクラスターにログインして Hadoop ジョブを実行できるようにするには、Linux アカウント向けに HDFS ユーザーディレクトリを追加し、そのディレクトリの所有権を各ユーザーに付与します。

HDFS ディレクトリを作成するには、クラスター作成後あるいは作成時にクラスターへステップを送信することが最も簡単な方法です。または、デフォルトの hadoop ユーザーとして EC2 キーペアを使用してプライマリノードに接続し、コマンドを実行することもできます。詳細については、「SSH を使用してプライマリノードに接続する」を参照してください。

次の例では、既存のクラスターに Bash スクリプト (AddHDFSUsers.sh) を送信し、クラスター ID を参照します。このスクリプトは Amazon S3 に保存されます。

```
aws emr add-steps --cluster-id <j-2AL4XXXXXX5T9> \
--steps Type=CUSTOM_JAR,Name=CustomJAR,ActionOnFailure=CONTINUE,\
Jar=s3://region.elasticmapreduce/libs/script-runner/script-runner.jar,Args=["s3://DOC-EXAMPLE-BUCKET/AddHDFSUsers.sh"]
```

次の例は、AddHDFSUsers.sh スクリプトの内容を示します。

```
#!/bin/bash
# AddHDFSUsers.sh script

# Initialize an array of user names from AD, or Linux users created manually on the
 cluster
ADUSERS=("lijuan" "marymajor" "richardroe" "myusername")

# For each user listed, create an HDFS user directory
# and change ownership to the user

for username in ${ADUSERS[@]}; do
     hdfs dfs -mkdir /user/$username
     hdfs dfs -chown $username:$username /user/$username
done
```

SSH で GSSAPI を有効化する

Kerberos 認証済みユーザーが SSH を使用してプライマリノードに接続するには、SSH サービスで GSSAPI 認証が有効にされている必要があります。GSSAPI を有効にするには、プライマリノードのコマンドラインから次のコマンドを実行するか、それをスクリプトとして実行するステップを使用します。SSH を再設定したら、サービスを再起動する必要があります。

```
sudo sed -i 's/^.*GSSAPIAuthentication.*$/GSSAPIAuthentication yes/' /etc/ssh/
sshd_config
sudo sed -i 's/^.*GSSAPICleanupCredentials.*$/GSSAPICleanupCredentials yes/' /etc/ssh/
sshd_config
sudo systemctl restart sshd
```

SSH を使用して Kerberos 認証済みのクラスターに接続する

このセクションでは、Kerberos 認証済みユーザーが EMR クラスターのプライマリノードに接続するためのステップを示しています。

SSH 接続のために使用する各コンピューターには、SSH クライアントおよび Kerberos クライアントアプリケーションがインストールされている必要があります。ほとんどの Linux コンピューターにはデフォルトでこれがあります。たとえば、ほとんどの Linux、Unix、および Mac OS オペレーティングシステムには OpenSSH がインストールされています。SSH クライアントがあるかどうかを確認するには、コマンドラインで ssh と入力します。ご使用のコンピュータでこのコマンドが認識されない場合、プライマリノードに接続するために SSH クライアントをインストールします。OpenSSH プロジェクトが、SSH ツールの完全なスイートの無料実装を提供しています。詳細については、OpenSSH のウェブサイトを参照してください。Windows ユーザーは、PuTTY などのアプリケーションを SSH クライアントとして使用できます。

SSH 接続の詳細については、「クラスターに接続する」を参照してください。

SSH は GSSAPI を使用して Kerberos クライアントを認証するため、クラスターのプライマリノードで SSH サービスに対して GSSAPI 認証を有効にする必要があります。詳細については、「SSH で GSSAPI を有効化する」を参照してください。また、SSH クライアントは GSSAPI も使用する必要があります。

次の例では、*MasterPublicDNS* には、クラスターの詳細ペインの 概要タブに表示されるマスターパブリック DNS の値を使用します。例えば、*ec2-11-222-33-44.compute-1.amazonaws.com* です。

krb5.conf (非 Active Directory) の前提条件

Active Directory 統合なしの設定を使用する場合、SSH クライアントと Kerberos クライアントのアプリケーションに加えて、各クライアントコンピュータには、クラスターのプライマリノード上の `/etc/krb5.conf` ファイルに一致する `/etc/krb5.conf` ファイルのコピーがあることが必要です。

krb5.conf ファイルをコピーするには

1. SSH を使用してプライマリノードに接続するには、EC2 キーペアおよびデフォルト hadoop ユーザー (例: hadoop@*MasterPublicDNS*) を使用します。詳細な手順については、「クラスターに接続する」を参照してください。
2. プライマリノードから `/etc/krb5.conf` ファイルの内容をコピーします。詳細については、「クラスターに接続する」を参照してください。

3. クラスターに接続する各クライアントコンピュータで、前のステップで作成したコピーに基づいて同一の `/etc/krb5.conf` ファイルを作成します。

kinit および SSH を使用する

ユーザーが Kerberos 認証情報を使用してクライアントコンピューターから接続するたびに、ユーザーはまずクライアントコンピューター上でユーザーに対する Kerberos チケットを更新する必要があります。また、SSH クライアントは GSSAPI 認証を使用するように設定されている必要があります。

SSH を使用して Kerberos 認証済みの EMR クラスターに接続するには

1. 次の例に示すように、`kinit` を使用して Kerberos チケットを更新する

```
kinit user1
```

2. `ssh` クライアントをクラスター専用 KDC で作成したプリンシパルあるいは Active Directory ユーザー名と共に使用します。次の例に示すように、GSSAPI 認証が有効になっていることを確認します。

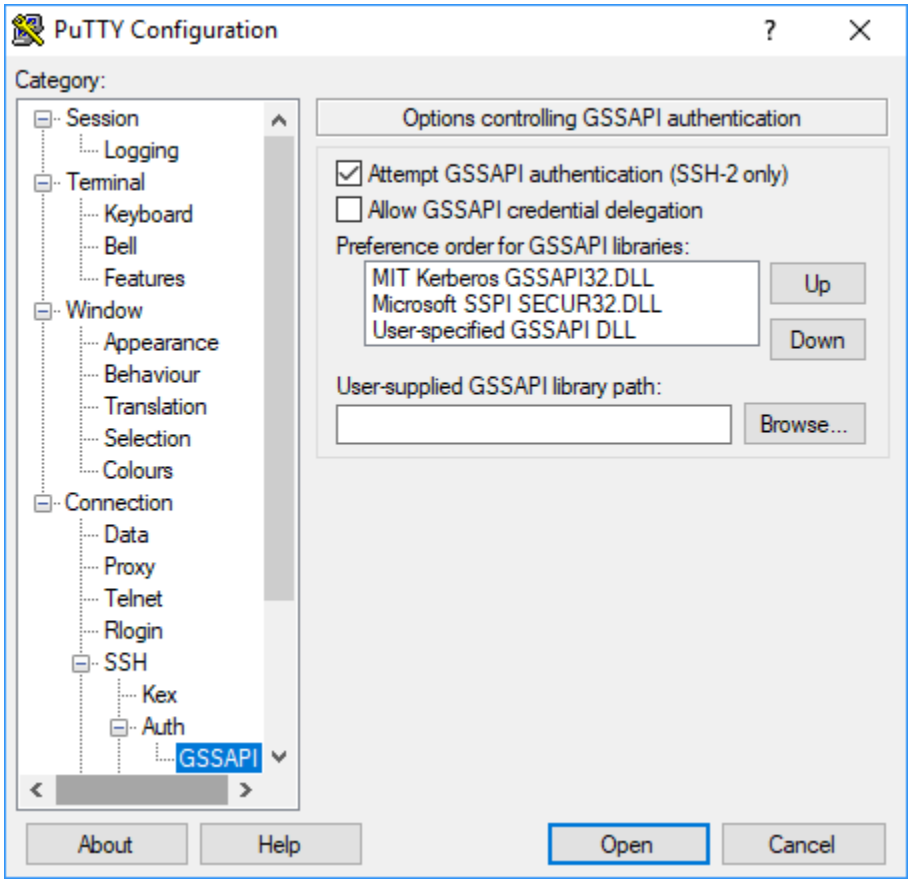
   例: Linux ユーザー

   `-K` オプションは GSSAPI 認証を指定します。

```
ssh -K user1@MasterPublicDNS
```

   例: Windows ユーザー (PuTTY)

   次に示すように、セッションの GSSAPI 認証が有効化されていることを確認します。



チュートリアル: クラスター専用の KDC を設定する

このトピックでは、クラスター専用 KDC (キー配布センター) を使用したクラスターの作成、すべてのクラスターノードへの Linux アカウントの手動での追加、プライマリノードでの KDC への Kerberos プリンシパルの追加、クライアントコンピュータに Kerberos クライアントがインストールされていることの確認を行う説明を示します。

Kerberos および KDC に対する Amazon EMR サポート、および MIT Kerberos ドキュメントへのリンクの詳細については、「Amazon EMR での認証に Kerberos を使用する」を参照してください。

ステップ 1: Kerberos 認証済みクラスターを作成する

1. Kerberos を有効にするセキュリティ設定を作成します。次の例は、セキュリティ設定 AWS CLI をインライン JSON 構造として指定する を使用するcreate-security-configurationコマンドを示しています。ローカルに保存されたファイルを参照することもできます。

```
aws emr create-security-configuration --name MyKerberosConfig \
--security-configuration '{"AuthenticationConfiguration": {"KerberosConfiguration":
```

```
{"Provider": "ClusterDedicatedKdc", "ClusterDedicatedKdcConfiguration":
 {"TicketLifetimeInHours": 24}}}}'
```

2. セキュリティ設定を参照して、クラスターの Kerberos 属性を確立し、ブートストラップアクションを使用して Linux アカウントを追加するクラスターを作成します。次の例は、 AWS CLIで create-cluster コマンドを使用する方法を示しています。このコマンドを使用すると、上記で作成したセキュリティ設定 (MyKerberosConfig) が参照されます。また、ブートストラップアクションとして、クラスターを作成する前に作成し、Amazon S3 にアップロードしたシンプルなスクリプト (createlinuxusers.sh) も参照されます。

```
aws emr create-cluster --name "MyKerberosCluster" \
--release-label emr-7.1.0 \
--instance-type m5.xlarge \
--instance-count 3 \
--ec2-attributes InstanceProfile=EMR_EC2_DefaultRole,KeyName=MyEC2KeyPair \
--service-role EMR_DefaultRole \
--security-configuration MyKerberosConfig \
--applications Name=Hadoop Name=Hive Name=Oozie Name=Hue Name=HCatalog Name=Spark \
--kerberos-attributes Realm=EC2.INTERNAL,\
KdcAdminPassword=MyClusterKDCAdminPwd \
--bootstrap-actions Path=s3://DOC-EXAMPLE-BUCKET/createlinuxusers.sh
```

次のコードでは、createlinuxusers.sh スクリプトの内容を示します。このスクリプトでは、user1、user2、user3 がクラスター内の各ノードに追加されます。次のステップでは、これらのユーザーを KDC プリンシパルとして追加します。

```
#!/bin/bash
sudo adduser user1
sudo adduser user2
sudo adduser user3
```

ステップ 2: KDC にプリンシパルを追加する、HDFS ユーザーディレクトリを作成する、SSH を設定する

プライマリノードで実行されている KDC には、ローカルホストと、クラスターで作成した各ユーザーに対するプリンシパルを追加する必要があります。また、クラスターに接続し、Hadoop ジョブを実行する必要がある場合は、各ユーザー向けに HDFS ディレクトリを作成します。同様に、SSH サービスを設定し、GSSAPI 認証を有効にします。Kerberos で必要になります。GSSAPI を有効にしたら、SSH サービスを再起動します。

最も簡単にこれらのタスクを実行するには、クラスターにステップを送信します。次の例では、以前作成したクラスターに Bash スクリプト (configurekdc.sh) を送信し、クラスター ID を参照します。このスクリプトは Amazon S3 に保存されます。または、EC2 キーペアを使用してプライマリノードに接続し、コマンドを実行したり、クラスター作成時にステップを送信したりすることもできます。

```
aws emr add-steps --cluster-id <j-2AL4XXXXXX5T9> --steps
 Type=CUSTOM_JAR,Name=CustomJAR,ActionOnFailure=CONTINUE,Jar=s3://
myregion.elasticmapreduce/libs/script-runner/script-runner.jar,Args=["s3://DOC-EXAMPLE-
BUCKET/configurekdc.sh"]
```

次のコードでは、configurekdc.sh スクリプトの内容を示します。

```
#!/bin/bash
#Add a principal to the KDC for the primary node, using the primary node's returned
 host name
sudo kadmin.local -q "ktadd -k /etc/krb5.keytab host/`hostname -f`"
#Declare an associative array of user names and passwords to add
declare -A arr
arr=([user1]=pwd1 [user2]=pwd2 [user3]=pwd3)
for i in ${!arr[@]}; do
    #Assign plain language variables for clarity
     name=${i}
     password=${arr[${i}]}

     # Create principal for sshuser in the primary node and require a new password on
 first logon
     sudo kadmin.local -q "addprinc -pw $password +needchange $name"

     #Add user hdfs directory
     hdfs dfs -mkdir /user/$name

     #Change owner of user's hdfs directory to user
     hdfs dfs -chown $name:$name /user/$name
done

# Enable GSSAPI authentication for SSH and restart SSH service
sudo sed -i 's/^.*GSSAPIAuthentication.*$/GSSAPIAuthentication yes/' /etc/ssh/
sshd_config
sudo sed -i 's/^.*GSSAPICleanupCredentials.*$/GSSAPICleanupCredentials yes/' /etc/ssh/
sshd_config
sudo systemctl restart sshd
```

追加したユーザーはこれで SSH を使用してクラスターに接続できるようになります。詳細については、「SSH を使用して Kerberos 認証済みのクラスターに接続する」を参照してください。

チュートリアル: Active Directory ドメインを使用したクロス領域信頼の設定

クロス領域信頼をセットアップする際、別の Kerberos 領域のプリンシパル (通常はユーザー) が、EMR クラスターのアプリケーションコンポーネントを認証できるようにします。クラスター専用 KDC (キー配布センター) は、両方の KDC に存在するクロス領域のプリンシパルを使用して、他の KDC で信頼関係を確立します。プリンシパル名とパスワードが正確に一致しています。

クロス領域信頼では、KDC がネットワーク経由で相互に到達し、相互のドメイン名を解決できる必要があります。以下に、必要な接続とドメイン名の解決を行うネットワークステップの例と合わせて、EC2 インスタンスとして実行されている Microsoft AD ドメインコントローラーを使用してクロス領域の信頼関係を確立するステップを示します。KDC 間の必要なネットワークトラフィックを許可するすべてのネットワークのセットアップは許容範囲内です。

必要に応じて、1 つのクラスター上で KDC を使用して Active Directory でクロス領域信頼を確立したら、異なるセキュリティ設定を使用して別のクラスターを作成し、最初のクラスターの KDC を外部 KDC として参照することができます。セキュリティグループ設定およびクラスターのセットアップの例については、「Active Directory クロス領域信頼を使用した外部のクラスター KDC」を参照してください。

Kerberos および KDC に対する Amazon EMR サポート、および MIT Kerberos ドキュメントへのリンクの詳細については、「Amazon EMR での認証に Kerberos を使用する」を参照してください。

> **Important**
>
> Amazon EMR は、 とのクロス領域信頼をサポートしていません AWS Directory Service for Microsoft Active Directory。

ステップ 1: VPC とサブネットをセットアップする

ステップ 2: Active Directory ドメインコントローラーの起動とインストール

ステップ 3: EMR クラスターのドメインにアカウントを追加する

ステップ 4: Active Directory ドメインコントローラーで受信の信頼を設定する

ステップ 5: DHCP オプションセットを使用して VPC DNS サーバーとして Active Directory ドメインコントローラーを指定する

ステップ 6: Kerberos 認証済みの EMR クラスターを起動する

ステップ 7: HDFS ユーザーを作成して、Active Directory アカウントのクラスターに対するアクセス許可を設定する

ステップ 1: VPC とサブネットをセットアップする

クラスター専用 KDC が Active Directory ドメインコントローラーに到達し、ドメイン名を解決できるように、VPC およびサブネットの作成手順を以下に示します。これらのステップでは、DHCP オプションセットのドメイン名サーバーとして Active Directory ドメインコントローラーを参照し、ドメイン名を解決します。詳細については、「ステップ 5: DHCP オプションセットを使用して VPC DNS サーバーとして Active Directory ドメインコントローラーを指定する」を参照してください。

KDC と Active Directory ドメインコントローラーは、相互のドメイン名を解決できる必要があります。これにより、Amazon EMR はコンピュータをドメインに参加させ、クラスターインスタンス上で対応する Linux アカウントおよび SSH パラメータを自動的に設定することができます。

Amazon EMR でドメイン名を解決できない場合は、Active Directory ドメインコントローラーの IP アドレスを使用して信頼を参照できます。ただし、Linux アカウントの追加、クラスター専用 KDC への対応するプリンシパルの追加、SSH の設定は手動で行う必要があります。

VPC とサブネットをセットアップするには

1. 1 つのパブリックサブネットを持つ Amazon VPC を作成する 詳細については、「Amazon VPC 入門ガイド」の「ステップ 1: VPC の作成」を参照してください。

   > **Important**
   >
   > Microsoft Active Directory ドメインコントローラーを使用している場合は、すべての IPv4 アドレスが 9 文字未満 (例: 10.0.0.0/16) になるように EMR クラスターの CIDR ブロックを選択します。これは、クラスターコンピュータの DNS 名は、コンピュータが Active Directory ディレクトリに参加するときに使用されます。 はIPv4DNS 名が 15 文字を超える可能性があるように、IPv4 アドレスに基づいて DNS ホスト名を AWS 割り当てます。Active Directory には、結合されるコンピュータ名の登録に 15 文字の制限があるため、長い場合は切り捨てられます。これが原因で、予測できないエラーが発生することがあります。

2. VPC に割り当てられているデフォルトの DHCP オプションセットを削除します。詳細については、「DHCP オプションを使用しないように VPC を変更する」を参照してください。Active

Directory ドメインコントローラーを DNS サーバーとして指定する新しい設定を後で追加します。

3. VPC に対して DNS サポートが有効になっていること、つまり、DNS ホスト名および DNS 解決がいずれも有効であることを確認します。デフォルトでは有効化されています。詳細については、「VPC の DNS サポートを更新する」を参照してください。
4. VPC にインターネットゲートウェイがアタッチされていることを確認します。この状態がデフォルトです。詳細については、「インターネットゲートウェイの作成とアタッチ」を参照してください。

> **Note**
>
> VPC 向けに新しいドメインコントローラーを確立しているため、この例ではインターネットゲートウェイが使用されます。お客様のアプリケーションでは、インターネットゲートウェイが必要ではない場合があります。唯一の要件は、クラスター専用 KDC が Active Directory ドメインコントローラーにアクセスできることです。

5. カスタムルートテーブルを作成して、インターネットゲートウェイをターゲットにしたルートを追加し、サブネットにアタッチします。詳細については、「カスタムルートテーブルを作成する」を参照してください。
6. ドメインコントローラーで EC2 インスタンスを起動する場合は、静的なパブリック IPv4 アドレスを使用して RDP で接続する必要があります。これを行う最も簡単な方法は、サブネットを自動割り当てパブリック IPv4 アドレスに設定することです。サブネットを作成する際、これはデフォルトで設定されていません。詳細については、「サブネットのパブリック IPv4 アドレス属性を変更する」を参照してください。インスタンスを起動する際、オプションでそのアドレスを割り当てることができます。詳細については、「インスタンス起動時のパブリック IPv4 アドレスの割り当て」を参照してください。
7. 終了したら、VPC とサブネット ID を書きとめておきます。後に Active Directory ドメインコントローラーおよびクラスターを起動する際に使用します。

ステップ 2: Active Directory ドメインコントローラーの起動とインストール

1. Microsoft Windows Server 2016 Base の AMI に基づき、EC2 インスタンスを起動します。m4.xlarge またはそれ以上のインスタンスタイプの使用をお勧めします。詳細については、Amazon EC2 ユーザーガイド」の AWS Marketplace 「インスタンスの起動」を参照してください。

2. EC2 インスタンスに関連付けられているセキュリティグループのグループ ID をメモします。この情報は「ステップ 6: Kerberos 認証済みの EMR クラスターを起動する」に必要です。*sg-012xrlmdomain345* を使用します。または、EMR クラスターとこのインスタンス用に、それらの間のトラフィックを許可する異なるセキュリティグループを指定できます。詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EC2 security groups for Linux instances」(Linux インスタンス用の Amazon EC2 セキュリティグループ) を参照してください。
3. RDP を使用して EC2 インスタンスに接続します。詳細については、Amazon EC2 ユーザーガイド」の「Windows インスタンスへの接続」を参照してください。
4. サーバーで Active Directory ドメインサービスの役割をインストールして設定するには、[Server Manager] (サーバー マネージャ) を起動します。サーバーをドメインコントローラーに昇格させ、ドメイン名 (ここでは *ad.domain.com*) を割り当てます。後に EMR セキュリティ設定およびクラスターを作成する際に必要になるため、ドメイン名を書きとめておきます。Active Directory を初めてセットアップする場合は、「How to set up Active Directory (AD) in Windows Server 2016」の手順に従います。

   完了したらインスタンスが再起動されます。

ステップ 3: EMR クラスターのドメインにアカウントを追加する

Active Directory ドメインコントローラーに RDP 接続し、各クラスターユーザー向けに Active Directory ユーザーおよびコンピュータにユーザーアカウントを作成します。詳細については、Microsoft Learn サイトの「Create a User Account in Active Directory Users and Computers」を参照してください。各ユーザーの [User logon name (ユーザーのログオン名)] を書きとめておきます。これらは、後にクラスターを構成する際に必要になります。

さらに、コンピュータをドメインに参加させるのに十分な権限を持つユーザーアカウントを作成します。クラスターを作成する際、このアカウントを指定します。Amazon EMR は、このアカウントを使用して、クラスターインスタンスをドメインに結合します。このアカウントとパスワードは「ステップ 6: Kerberos 認証済みの EMR クラスターを起動する」で指定します。コンピュータ参加権限をアカウントに委任するには、参加権限を持つグループを作成し、ユーザーをそのグループに割り当てることが推奨されます。手順については、「AWS Directory Service 管理ガイド」の「ディレクトリ結合権限の委任」を参照してください。

ステップ 4: Active Directory ドメインコントローラーで受信の信頼を設定する

以下のコマンド例では、Active Directory に信頼を作成します。一方向、着信、非推移に加え、クラスター専用 KDC を使用した領域信頼があります。クラスターの領域では例として *EC2.INTERNAL* を使用します。*KDC-FQDN* は、KDC をホストしている Amazon EMR プライマリノードでリス

トされている [パブリック DNS] の名前に置き換えてください。passwordt パラメータでは、[cross-realm principal password (クロス領域プリンシパルのパスワード)] を指定します。このパスワードは、クラスター作成時に、クラスターの [realm (領域)] と合わせて指定したものです。この領域名は、クラスターの us-east-1 のデフォルトのドメイン名から取得されます。Domain は Active Directory ドメインで信頼を作成する場合、規則により小文字で表記されます。この例では *ad.domain.com* を使用します

管理者権限で Windows コマンドプロンプトを開き、以下のコマンドを入力して、Active Directory ドメインコントローラーで信頼関係を作成します。

```
C:\Users\Administrator> ksetup /addkdc EC2.INTERNAL KDC-FQDN
C:\Users\Administrator> netdom trust EC2.INTERNAL /Domain:ad.domain.com /add /realm /
passwordt:MyVeryStrongPassword
C:\Users\Administrator> ksetup /SetEncTypeAttr EC2.INTERNAL AES256-CTS-HMAC-SHA1-96
```

ステップ 5: DHCP オプションセットを使用して VPC DNS サーバーとして Active Directory ドメインコントローラーを指定する

これで、Active Directory ドメインコントローラーが設定されたため、VPC 内で名前解決できるように、VPC を設定してドメイン名サーバーとして使用する必要があります。これを行うには、DHCP オプションセットをアタッチします。[Domain name] (ドメイン名) をクラスターのドメイン名に指定します (例: クラスターが us-east-1 にある場合は ec2.internal、他のリージョンにある場合は *region*.compute.internal)。ドメインネームサーバー の場合、最初のエントリとして Active Directory ドメインコントローラーの IP アドレス (クラスターから到達可能である必要があります) を指定し、その後に AmazonProvidedDNS (***xx.xx.xx.xx*** 、AmazonProvidedDNS など) を指定する必要があります。詳細については、「DHCP オプションセットを変更する」を参照してください。

ステップ 6: Kerberos 認証済みの EMR クラスターを起動する

1. Amazon EMR で、前のステップで作成した Active Directory ドメインコントローラーを指定するセキュリティ設定を作成します。次にコマンドの例を示します。ドメイン (*ad.domain.com*) を、「ステップ 2: Active Directory ドメインコントローラーの起動とインストール」で指定したドメイン名に置き換えます。

```
aws emr create-security-configuration --name MyKerberosConfig \
--security-configuration '{
  "AuthenticationConfiguration": {
    "KerberosConfiguration": {
      "Provider": "ClusterDedicatedKdc",
      "ClusterDedicatedKdcConfiguration": {
```

```
        "TicketLifetimeInHours": 24,
        "CrossRealmTrustConfiguration": {
          "Realm": "AD.DOMAIN.COM",
          "Domain": "ad.domain.com",
          "AdminServer": "ad.domain.com",
          "KdcServer": "ad.domain.com"
        }
      }
    }
  }
}'
```

2. 以下の属性を使用してクラスターを作成します。

- `--security-configuration` オプションを使用して、作成したセキュリティ設定を指定します。この例では を使用します*MyKerberosConfig*。
- `--ec2-attributes option` の SubnetId プロパティを使用して、「ステップ 1: VPC とサブネットをセットアップする」で作成したサブネットを指定します。この例では *step1-subnet* を使用します。
- `--ec2-attributes` オプションの AdditionalMasterSecurityGroups と AdditionalSlaveSecurityGroups を使用して、「ステップ 2: Active Directory ドメインコントローラーの起動とインストール」の AD ドメインコントローラーに関連付けられているセキュリティグループが、クラスターのプライマリノード、コアノード、タスクノードにも関連付けられるように指定します。この例では *sg-012xrlmdomain345* を使用します。

`--kerberos-attributes` を使用して、以下のクラスター固有の Kerberos 属性を指定します。

- Active Directory ドメインコントローラーの設定時に指定したクラスターの領域。
- 「ステップ 4: Active Directory ドメインコントローラーで受信の信頼を設定する」で `passwordt` として指定したクロス領域信頼プリンシパルのパスワード。
- `KdcAdminPassword` は、クラスター専用 KDC を管理するために使用します。
- 「ステップ 3: EMR クラスターのドメインにアカウントを追加する」で作成したコンピュータ結合権限を持つ Active Directory アカウントのユーザーログオン名およびパスワード。

次の例では、Kerberos 認証済みクラスターを起動します。

```
aws emr create-cluster --name "MyKerberosCluster" \
--release-label emr-5.10.0 \
--instance-type m5.xlarge \
--instance-count 3 \
--ec2-attributes InstanceProfile=EMR_EC2_DefaultRole,KeyName=MyEC2KeyPair,\
```

```
SubnetId=step1-subnet, AdditionalMasterSecurityGroups=sg-012xrlmdomain345,
AdditionalSlaveSecurityGroups=sg-012xrlmdomain345\
--service-role EMR_DefaultRole \
--security-configuration MyKerberosConfig \
--applications Name=Hadoop Name=Hive Name=Oozie Name=Hue Name=HCatalog Name=Spark \
--kerberos-attributes Realm=EC2.INTERNAL,\
KdcAdminPassword=MyClusterKDCAdminPwd,\
ADDomainJoinUser=ADUserLogonName,ADDomainJoinPassword=ADUserPassword,\
CrossRealmTrustPrincipalPassword=MatchADTrustPwd
```

ステップ 7: HDFS ユーザーを作成して、Active Directory アカウントのクラスターに対するアクセス許可を設定する

Active Directory で信頼関係を設定すると、Amazon EMR は各 Active Directory アカウントのクラスターで Linux ユーザーを作成します。例えば、Active Directory のユーザーログオン名が `LiJuan` の場合、Linux アカウントは `lijuan` になります。Active Directory のユーザー名には、大文字を含めることができますが、Linux では Active Directory のこの区別は採用されません。

ユーザーがクラスターにログインして Hadoop ジョブを実行できるようにするには、Linux アカウント向けに HDFS ユーザーディレクトリを追加し、そのディレクトリの所有権を各ユーザーに付与します。そのためには、Amazon S3 に保存されているスクリプトをクラスターのステップとして実行することをお勧めします。あるいは、以下のスクリプトのコマンドをプライマリノードのコマンドラインから実行することもできます。Hadoop ユーザーとして SSH 経由でプライマリノードに接続するには、クラスターの作成時に指定した EC2 キーペアを使用します。詳細については、「SSH 認証情報に EC2 キーペアを使用する」を参照してください。

スクリプト (*AddHDFSUsers.sh*) を実行するクラスターにステップを追加するには、以下のコマンドを実行します。

```
aws emr add-steps --cluster-id <j-2AL4XXXXXX5T9> \
--steps Type=CUSTOM_JAR,Name=CustomJAR,ActionOnFailure=CONTINUE,\
Jar=s3://region.elasticmapreduce/libs/script-runner/script-runner.jar,Args=["s3://DOC-EXAMPLE-BUCKET/AddHDFSUsers.sh"]
```

ファイル (*AddHDFSUsers.sh*) の内容は次のとおりです。

```
#!/bin/bash
# AddHDFSUsers.sh script
```

```
# Initialize an array of user names from AD or Linux users and KDC principals created
 manually on the cluster
ADUSERS=("lijuan" "marymajor" "richardroe" "myusername")

# For each user listed, create an HDFS user directory
# and change ownership to the user

for username in ${ADUSERS[@]}; do
     hdfs dfs -mkdir /user/$username
     hdfs dfs -chown $username:$username /user/$username
done
```

Hadoop グループにマッピングされた Active Directory グループ

Amazon EMR では、System Security Services Daemon (SSD) を使用して、Active Directory グループを Hadoop グループにマッピングします。グループマッピングを確認するには、「SSH を使用して Kerberos 認証済みのクラスターに接続する」の説明に従ってプライマリノードにログインした後、`hdfs groups` コマンドを使用して、Active Directory アカウントが属する Active Directory グループが、クラスター上の対応する Hadoop ユーザーの Hadoop グループにマッピングされていることを確認します。また、他のユーザーのグループマッピングを確認するには、同コマンドを使用してユーザー名を 1 つ以上指定します (例: `hdfs groups lijuan`)。詳細については、「Apache HDFS コマンドガイド」の「グループ」を参照してください。

Amazon EMR での認証に Active Directory または LDAP サーバーを使用する

Amazon EMR リリース 6.12.0 以降では、LDAP over SSL (LDAPS) プロトコルを使用して、企業の ID サーバーとネイティブに統合されるクラスターを起動できます。LDAP (Lightweight Directory Access Protocol) は、データにアクセスして管理する、ベンダーに依存しないオープンなアプリケーションプロトコルです。LDAP は一般的に、Active Directory (AD) や OpenLDAP などのアプリケーションでホストされている企業 ID サーバーに対するユーザー認証に使用されます。このネイティブ統合により、LDAP サーバーを使用して Amazon EMR のユーザーを認証できます。

Amazon EMR LDAP 統合の主な特徴は次のとおりです。

- ユーザーの代わりに Amazon EMR が、LDAP 認証を使用して認証するようにサポート対象のアプリケーションを設定します。
- Amazon EMR は、Kerberos プロトコルを使用して、サポート対象のアプリケーションのセキュリティを設定および管理します。ユーザーがコマンドやスクリプトを入力する必要はありません。

- Hive メタストアデータベースとテーブルの Apache Ranger 認証により、きめ細かなアクセスコントロール (FGAC) が可能になります。詳細については、「Amazon EMR と Apache Ranger を統合する」を参照してください。
- クラスターにアクセスするために LDAP 認証情報が必要な場合は、SSH 経由で EMR クラスターにアクセスできるユーザーについて、きめ細かなアクセスコントロール (FGAC) が可能です。

以下のページでは、Amazon EMR LDAP 統合で EMR クラスターを起動するための概念、前提条件、ステップについて説明します。

トピック

Amazon EMR での LDAP の概要

Lightweight Directory Access Protocol (LDAP) は、ネットワーク管理者が企業ネットワーク内のユーザーを認証することでデータへのアクセスを管理および制御するために使用するソフトウェアプロトコルです。LDAP プロトコルは、情報を階層化されたツリーディレクトリ構造で格納します。詳細については、LDAP.com の「Basic LDAP Concepts」を参照してください。

企業ネットワーク内では、多くのアプリケーションが LDAP プロトコルを使用してユーザーを認証することがあります。Amazon EMR LDAP 統合により、EMR クラスターは、セキュリティ設定を追加することで、同じ LDAP プロトコルをネイティブに使用できます。

Amazon EMR がサポートする LDAP プロトコルには、Active Directory と OpenLDAP という 2 つの主要な実装があります。他の実装も可能ですが、ほとんどが Active Directory または OpenLDAP と同じ認証プロトコルに適合しています。

Active Directory (AD)

Active Directory (AD) は、Windows ドメインネットワーク向けの Microsoft のディレクトリサービスです。AD はほとんどの Windows Server オペレーティングシステムに組み込まれており、LDAP プロトコルと LDAPS プロトコルを介してクライアントと通信できます。認証のために、Amazon EMR

は識別名およびパスワードとしてユーザープリンシパル名 (UPN) を使用し、AD インスタンスとのユーザーバインドを試みます。UPN で使用される標準形式は `username@domain_name` です。

OpenLDAP

OpenLDAP は、無料で利用できるオープンソースの LDAP プロトコル実装です。認証のために、Amazon EMR は識別名およびパスワードとして完全修飾ドメイン名 (FQDN) を使用し、OpenLDAP インスタンスとのユーザーバインドを試みます。FQDN で使用される標準形式は `username_attribute=username,LDAP_user_search_base` です。通常、`username_attribute` 値は `uid` で、`LDAP_user_search_base` 値にはユーザーにつながるツリーの属性が含まれます。例えば `ou=People,dc=example,dc=com` です。

LDAP プロトコルの他の無料のオープンソース実装では、通常、ユーザーの識別名は OpenLDAP と同様の FQDN に従います。

Amazon EMR 用の LDAP コンポーネント

以下のコンポーネントを通じて、Amazon EMR およびユーザーが EMR クラスターで直接使用する任意のアプリケーションでの認証に LDAP サーバーを使用することができます。

シークレットエージェント

シークレットエージェントは、すべてのユーザーリクエストを認証するクラスター上のプロセスです。シークレットエージェントは、EMR クラスター上のサポート対象のアプリケーションに代わって、LDAP サーバーへのユーザーバインドを作成します。シークレットエージェントは `emrsecretagent` ユーザーとして実行され、`/emr/secretagent/log` ディレクトリにログを書き込みます。これらのログには、各ユーザーの認証リクエストの状態や、ユーザー認証中に発生する可能性のあるエラーに関する詳細が記録されます。

System Security Services Daemon (SSSD)

SSSD は LDAP 対応 EMR クラスターの各ノードで実行されるデーモンです。SSSD は UNIX ユーザーを作成して管理し、リモートの企業 ID を各ノードに同期させます。Hive や Spark などの YARN ベースのアプリケーションでは、ユーザーのクエリを実行するすべてのノードにローカル UNIX ユーザーが存在する必要があります。

Amazon EMR での LDAP に関するアプリケーションサポートと考慮事項

Amazon EMR での LDAP でサポートされているアプリケーション

> **Important**
>
> Amazon EMR が LDAP でサポートしているアプリケーションは、このページにリストされているアプリケーションだけです。クラスターのセキュリティを確保するために、LDAP を有効にした EMR クラスターを作成する場合は、LDAP 互換アプリケーションのみを含めることができます。サポートされていない他のアプリケーションをインストールしようとすると、Amazon EMR は新しいクラスターに対するリクエストを拒否します。

Amazon EMR リリース 6.12 以降は、以下のアプリケーションとの LDAP 統合をサポートしています。

- Apache Livy
- Apache Hive から HiveServer2 (HS2)
- Trino
- Presto
- Hue

次のアプリケーションを EMR クラスターにインストールし、セキュリティニーズを満たすように設定することができます。

- Apache Spark
- Apache Hadoop

Amazon EMR での LDAP でサポートされている機能

LDAP 統合では、次の Amazon EMR 機能を使用できます。

> **Note**
>
> LDAP 認証情報の安全を確保するには、転送時の暗号化を使用して、クラスターに出入りするデータフローを保護する必要があります。転送時の暗号化の詳細については、「保管中と転送中のデータの暗号化」を参照してください。

- 転送時 (必須) と保管中の暗号化
- インスタンスグループ、インスタンスフリート、スポットインスタンス
- 実行中のクラスター上のアプリケーションの再設定
- EMRFS サーバー側の暗号化 (SSE)

サポートされていない 機能

Amazon EMR の LDAP 統合を使用する場合は、次の制限事項を考慮してください。

- Amazon EMR は、LDAP が有効になっているクラスターのステップを無効にします。
- Amazon EMR は、LDAP が有効になっているクラスターのランタイムロールと AWS Lake Formation 統合をサポートしていません。
- Amazon EMR は、StartTLS を使用した LDAP をサポートしていません。
- Amazon EMR は、LDAP が有効になっているクラスターでの高可用性モード (複数のプライマリノードを持つクラスター) をサポートしていません。
- LDAP が有効になっているクラスターのバインド認証情報や証明書をローテーションすることはできません。これらのフィールドのいずれかがローテーションされた場合は、更新されたバインド認証情報または証明書を使用して新しいクラスターを起動することが推奨されます。
- LDAP では正確な検索ベースを使用する必要があります。LDAP ユーザーとグループの検索ベースは、LDAP 検索フィルターをサポートしていません。

LDAP を備えた EMR クラスターを設定して起動する

このセクションでは、LDAP 認証で使用するために Amazon EMR を設定する方法について説明します。

トピック

Amazon EMR インスタンスロールにアクセス AWS Secrets Manager 許可を追加する

Amazon EMR は IAM サービスロールを使用して、ユーザーの代わりにクラスターのプロビジョニングと管理を行うためのアクションを実行します。クラスター EC2 インスタンスのサービスロール

(Amazon EMR の EC2 インスタンスプロファイルとも呼ばれます) は、Amazon EMR が起動時にクラスター内のすべての EC2 インスタンスに割り当てる特殊なタイプのサービスロールです。

EMR クラスターが Amazon S3 データやその他の AWS のサービスと対話するためのアクセス許可を定義するには、クラスターの起動時に、EMR_EC2_DefaultRole ではなくカスタム Amazon EC2 インスタンスプロファイルを定義します。詳細については、「クラスター EC2 インスタンスのサービスロール (EC2 インスタンスプロファイル)」および「IAM ロールのカスタマイズ」を参照してください。

次のステートメントをデフォルトの EC2 インスタンスプロファイルに追加して、Amazon EMR AWS Secrets Manager がセッションにタグを付け、LDAP 証明書を保存する にアクセスできるようにします。

```
    {
      "Sid": "AllowAssumeOfRolesAndTagging",
      "Effect": "Allow",
      "Action": ["sts:TagSession", "sts:AssumeRole"],
      "Resource": [
        "arn:aws:iam::111122223333:role/LDAP_DATA_ACCESS_ROLE_NAME",
        "arn:aws:iam::111122223333:role/LDAP_USER_ACCESS_ROLE_NAME"
      ]
    },
    {
        "Sid": "AllowSecretsRetrieval",
        "Effect": "Allow",
        "Action": "secretsmanager:GetSecretValue",
        "Resource": [
            "arn:aws:secretsmanager:us-east-1:111122223333:secret:LDAP_SECRET_NAME*",
            "arn:aws:secretsmanager:us-
east-1:111122223333:secret:ADMIN_LDAP_SECRET_NAME*"
        ]
    }
```

> **Note**
>
> Secrets Manager のアクセス許可を設定するときに、シークレット名の末尾にワイルドカード文字 * を付け忘れると、クラスターリクエストは失敗します。ワイルドカードは、シークレットバージョンを表します。
> AWS Secrets Manager ポリシーの範囲は、クラスターがインスタンスをプロビジョニングするために必要な証明書のみに制限する必要があります。

LDAP 統合用の Amazon EMR セキュリティ設定を作成する

LDAP 統合で EMR クラスターを起動する前に、「セキュリティ設定を作成する」のステップを使用してクラスターの Amazon EMR セキュリティ設定を作成します。`AuthenticationConfiguration` の下の `LDAPConfiguration` ブロック、または Amazon EMR コンソールの [セキュリティ設定] セクションの対応するフィールドで、以下の設定を完了します。

`EnableLDAPAuthentication`

コンソールオプション: 認証プロトコル: LDAP

LDAP 統合を使用するには、このオプションを `true` に設定するか、コンソールでクラスターを作成するときに認証プロトコルとして選択します。Amazon EMR コンソールでセキュリティ設定を作成する場合、デフォルトでは `EnableLDAPAuthentication` は `true` です。

`LDAPServerURL`

コンソールオプション: LDAP サーバーの場所

LDAP サーバーの場所 (プレフィックスを含む): `ldaps://`*`location_of_server`*

`BindCertificateARN`

コンソールオプション: LDAP SSL 証明書

LDAP サーバーが使用する SSL 証明書に署名するための証明書を含む AWS Secrets Manager ARN。LDAP サーバーがパブリック認証局 (CA) によって署名されている場合は、ARN AWS Secrets Manager に空白のファイルを提供できます。Secrets Manager に証明書を保存する方法の詳細については、「AWS Secrets Managerに TLS 証明書を保存する」を参照してください。

`BindCredentialsARN`

コンソールオプション: LDAP サーバーバインド認証情報

LDAP 管理者ユーザーのバインド認証情報を含む AWS Secrets Manager ARN。認証情報は JSON オブジェクトとして保存されます。このシークレットにはキーと値のペアが 1 つしかありません。ペアのキーはユーザー名で、値はパスワードです。例えば `{"uid=admin,cn=People,dc=example,dc=com": "AdminPassword1"}` です。EMR クラスターの SSH ログインを有効にしない限り、このフィールドはオプションです。多くの設定で、Active Directory インスタンスには、SSSD がユーザーを同期できるようにするためのバインド認証情報が必要です。

LDAPAccessFilter

コンソールオプション: LDAP アクセスフィルター

認証可能な LDAP サーバー内のオブジェクトのサブセットを指定します。例えば、LDAP サーバー内の posixAccount オブジェクトクラスを持つすべてのユーザーにアクセスを許可する場合、アクセスフィルターを (objectClass=posixAccount) として定義します。

LDAPUserSearchBase

コンソールオプション: LDAP ユーザー検索ベース

LDAP サーバー内のユーザーが属する検索ベース。例えば cn=People,dc=example,dc=com です。

LDAPGroupSearchBase

コンソールオプション: LDAP グループ検索ベース

LDAP サーバー内のグループが属する検索ベース。例えば cn=Groups,dc=example,dc=com です。

EnableSSHLogin

コンソールオプション: SSH ログイン

LDAP 認証情報を使用したパスワード認証を許可するかどうかを指定します。このオプションを有効にすることは推奨されません。キーペアは EMR クラスターへのアクセスを可能にする、より安全なルートです。このフィールドはオプションであり、デフォルトは false です。

LDAPServerType

コンソールオプション: LDAP サーバータイプ

Amazon EMR が接続する LDAP サーバーのタイプを指定します。サポートされているオプションは、Active Directory と OpenLDAP です。その他の LDAP サーバータイプでも機能する可能性はありますが、Amazon EMR では他のサーバータイプを公式にはサポートしていません。詳細については、「Amazon EMR 用の LDAP コンポーネント」を参照してください。

ActiveDirectoryConfigurations

Active Directory サーバータイプを使用するセキュリティ設定に必要なサブブロック。

ADDomain

コンソールオプション: Active Directory ドメイン

Active Directory サーバータイプを使用するセキュリティ設定でのユーザー認証用のユーザープリンシパル名 (UPN) の作成に使用されるドメイン名。

LDAP と Amazon EMR を使用するセキュリティ設定に関する考慮事項

- Amazon EMR LDAP 統合を使用するセキュリティ設定を作成するには、転送時の暗号化を使用する必要があります。転送時の暗号化については、「保管中と転送中のデータの暗号化」を参照してください。
- Kerberos 設定を同じセキュリティ設定で定義することはできません。Amazon EMR は、専用の KDC を自動的にプロビジョニングし、この KDC の管理者パスワードを管理します。ユーザーはこの管理者パスワードにアクセスできません。
- IAM ランタイムロール および を同じセキュリティ設定 AWS Lake Formation で定義することはできません。
- `LDAPServerURL` の値には `ldaps://` プロトコルが含まれている必要があります。
- `LDAPAccessFilter` を空にすることはできません。

Amazon EMR の Apache Ranger 統合で LDAP を使用する

Amazon EMR の LDAP 統合により、Apache Ranger との統合をさらに進めることができます。LDAP ユーザーを Ranger に取り込むと、それらのユーザーを Apache Ranger ポリシーサーバーに関連付けて Amazon EMR やその他のアプリケーションと統合できるようになります。そのためには、LDAP クラスターで使用するセキュリティ設定の `AuthorizationConfiguration` 内にある `RangerConfiguration` フィールドを定義します。セキュリティ設定の設定方法の詳細については、「EMR セキュリティ設定を作成する」を参照してください。

Amazon EMR で LDAP を使用する場合、Apache Ranger の Amazon EMR 統合で `KerberosConfiguration` を指定する必要はありません。

LDAP で認証する EMR クラスターを起動する

次のステップを使用して、LDAP または Active Directory を使用する EMR クラスターを起動します。

1. 環境をセットアップします。
   - EMR クラスターのノードが Amazon S3 および と通信できることを確認します AWS Secrets Manager。これらのサービスと通信できるように EC2 インスタンスプロファイルロールを変

更する方法の詳細については、「Amazon EMR インスタンスロールにアクセス AWS Secrets Manager 許可を追加する」を参照してください。

- プライベートサブネットで EMR クラスターを実行する場合は、 AWS PrivateLink と Amazon VPC エンドポイントを使用するか、ネットワークアドレス変換 (NAT) を使用して S3 および Secrets Manager と通信するように VPC を設定する必要があります。詳細については、「Amazon VPC 入門ガイド」の「AWS PrivateLink および VPC エンドポイント」と「NAT インスタンス」を参照してください。
- EMR クラスターと LDAP サーバーの間にネットワーク接続があることを確認します。EMR クラスターは、ネットワーク経由で LDAP サーバーにアクセスする必要があります。クラスターのプライマリノード、コアノード、およびタスクノードは、LDAP サーバーと通信してユーザーデータを同期します。LDAP サーバーが Amazon EC2 で実行されている場合は、EMR クラスターからのトラフィックを受け入れるように EC2 セキュリティグループを更新します。詳細については、「Amazon EMR インスタンスロールにアクセス AWS Secrets Manager 許可を追加する」を参照してください。

2. LDAP 統合用の Amazon EMR セキュリティ設定を作成します。詳細については、「LDAP 統合用の Amazon EMR セキュリティ設定を作成する」を参照してください。
3. 設定が完了したら、「Amazon EMR クラスターを起動する」のステップを使用して、以下の設定でクラスターを起動します。

   - Amazon EMR リリース 6.12 以降を選択します。最新の Amazon EMR リリースを使用することが推奨されます。
   - クラスターのアプリケーションには、LDAP をサポートするものだけを指定または選択してください。Amazon EMR で LDAP をサポートするアプリケーションのリストについては、「Amazon EMR での LDAP に関するアプリケーションサポートと考慮事項」を参照してください。
   - 前のステップで作成したセキュリティ設定を適用します。

Amazon EMR で LDAP を使用する例

LDAP 統合を使用する EMR クラスターをプロビジョニングすると、組み込みのユーザー名とパスワードによる認証メカニズムを通じて、任意のサポート対象のアプリケーションに LDAP 認証情報を提供できます。このページでは、いくつかの例を示します。

Apache Hive での LDAP 認証の使用

Example - Apache Hive

次のコマンド例では、HiveServer2 と Beeline を使用して Apache Hive セッションを開始します。

```
beeline -u "jdbc:hive2://$HOSTNAME:10000/default;ssl=true;sslTrustStore=
$TRUSTSTORE_PATH;trustStorePassword=$TRUSTSTORE_PASS"  -n LDAP_USERNAME -
p LDAP_PASSWORD
```

Apache Livy での LDAP 認証の使用

Example - Apache Livy

次のコマンド例は、cURL を使用して Livy セッションを開始します。*ENCODED-KEYPAIR* は、`username:password` の Base64 エンコード文字列に置き換えてください。

```
curl -X POST --data '{"proxyUser":"LDAP_USERNAME","kind": "pyspark"}' -H "Content-Type:
 application/json" -H "Authorization: Basic ENCODED-KEYPAIR" DNS_OF_PRIMARY_NODE:8998/
sessions
```

Presto での LDAP 認証の使用

Example - Presto

次のコマンド例は、Presto CLI を使用して Presto セッションを開始します。

```
presto-cli --user "LDAP_USERNAME" --password --catalog hive
```

このコマンドを実行した後、プロンプトに LDAP パスワードを入力します。

Trino での LDAP 認証の使用

Example - Trino

次のコマンド例は、Trino CLI を使用して Trino セッションを開始します。

```
trino-cli --user "LDAP_USERNAME" --password --catalog hive
```

このコマンドを実行した後、プロンプトに LDAP パスワードを入力します。

Hue での LDAP 認証の使用

クラスター上に作成した SSH トンネル経由で Hue UI にアクセスすることも、Hue への接続を公開ブロードキャストするようにプロキシサーバーを設定することもできます。Hue はデフォルトでは HTTPS モードで実行されないため、クライアントと Hue UI の間の通信が HTTPS で暗号化されるように、追加の暗号化レイヤーを使用することが推奨されます。これにより、ユーザーの認証情報が誤ってプレーンテキストで公開される可能性が低くなります。

Hue UI を使用するには、ブラウザで Hue UI を開き、LDAP ユーザー名とパスワードを入力してログインします。認証情報が正しければ、Hue はユーザーをログインさせ、サポート対象のすべてのアプリケーションで ID を使用してユーザーを認証します。

パスワード認証には SSH を使用し、他のアプリケーションには Kerberos チケットを使用する

> **Important**
>
> EMR クラスターへの SSH 接続でパスワード認証を使用することは推奨されません。

LDAP 認証情報を使用して EMR クラスターに SSH 接続できます。これを行うには、クラスターの起動に使用する Amazon EMR セキュリティ設定で `EnableSSHLogin` 設定を `true` に設定します。次に、起動したら次のコマンドを使用して、クラスターに SSH 接続します。

```
ssh username@EMR_PRIMARY_DNS_NAME
```

このコマンドを実行した後、プロンプトに LDAP パスワードを入力します。

Amazon EMR には、LDAP 認証情報を直接受け付けないサポート対象のアプリケーションで使用する Kerberos キータブファイルとチケットをユーザーが生成できる、クラスター上で使用するスクリプトが含まれています。これらのアプリケーションには`spark-submit`、、Spark SQL、 などがあります PySpark。

「`ldap-kinit`」と入力し、プロンプトに従います。認証が成功すると、Kerberos キータブファイルと有効な Kerberos チケットがホームディレクトリに表示されます。Kerberos チケットを使用すると、他の Kerberos 環境と同様にアプリケーションを実行できます。

Amazon EMR を と統合する AWS IAM Identity Center

Amazon EMR リリース 6.15.0 以降では、 の ID を使用して Amazon EMR クラスターで AWS IAM Identity Center 認証できます。以下のセクションでは、Identity Center と統合された EMR クラスター起動のコンセプト、前提条件、操作手順について説明します。

トピック

概要

IAM Identity Center を介した信頼できる ID の伝播は、ワークフォース ID を安全に作成または接続し、 AWS アカウントとアプリケーション間のアクセスを一元管理するのに役立ちます。この機能を使用すると、ユーザーは信頼できる ID 伝達を使用するアプリケーションにサインインでき、そのアプリケーションは、信頼できる ID 伝達も使用する AWS サービスのデータにアクセスするために行うリクエストでユーザーの ID を渡すことができます。アクセスはユーザーの ID に基づいて管理されるため、ユーザーはデータにアクセスするためにデータベースのローカルユーザー認証情報を使用したり、IAM ロールを引き受けたりする必要はありません。

Identity Center は、あらゆる規模とタイプの組織に対して、 でのワークフォース認証と認可 AWS に推奨されるアプローチです。Identity Center を使用すると、 でユーザー ID を作成および管理したり AWS、Microsoft Active Directory、Okta、Ping Identity、 JumpCloudGoogle Workspace、Microsoft Entra ID (以前の Azure AD) などの既存の ID ソースに接続したりできます。

詳細については、「 ユーザーガイド」の「 AWS IAM Identity Centerとは」および「アプリケーション間での信頼できる ID の伝播」を参照してください。 https://docs.aws.amazon.com/singlesignon/latest/userguide/trustedidentitypropagation.html AWS IAM Identity Center

特徴と利点

Amazon EMR と IAM Identity Center の統合には、以下の利点があります。

- Amazon EMR は、お使いの Identity Center ID を EMR クラスターに引き継ぐするための認証情報を提供します。

- Amazon EMR は、サポートされているすべてのアプリケーションを、クラスター認証情報を使用して認証するように設定します。
- Amazon EMR が Kerberos プロトコルを使用してサポート対象のアプリケーションのセキュリティを設定および管理するので、ユーザー側でコマンドやスクリプトを提供する必要はありません。
- S3 Access Grants が管理する S3 プレフィックスの Identity Center ID を使用して Amazon S3 プレフィックスレベルの認証を執行できます。
- AWS Lake Formation マネージド AWS Glue テーブルで Identity Center ID を使用してテーブルレベルの認証を強制する機能。

Amazon EMR AWS IAM Identity Center の統合の開始方法

このセクションでは、 と統合するように Amazon EMR を設定する方法について説明します AWS IAM Identity Center。

トピック

- Identity Center インスタンスを作成する
- Identity Center 用の IAM ロールを作成します。
- Identity Center 対応のセキュリティ設定を作成します。
- Identify Center 対応クラスターを作成して起動します。
- IAM Identity Center 対応 EMR クラスター用の Lake Formation を設定する
- IAM Identity Center 対応 EMR クラスターでの S3 Access Grants の使用

Identity Center インスタンスを作成する

Identity Center インスタンスをまだお持ちでない場合は、EMR クラスターを起動する AWS リージョン に Identity Center インスタンスを作成します。各 Identity Center インスタンスは、 AWS アカウントの 1 つのリージョンにのみ存在できます。

次の AWS CLI コマンドを使用して、 という名前の新しいインスタンスを作成します*MyInstance*。

```
aws sso-admin create-instance --name MyInstance
```

Identity Center 用の IAM ロールを作成します。

Amazon EMR を と統合するには AWS IAM Identity Center、EMR クラスターから Identity Center で認証する IAM ロールを作成します。内部的には、Amazon EMR は SigV4 認証情報を使用して、Identity Center ID を AWS Lake Formationなどの下流のサービスに引き継ぎます。また、お使いのロールには下流のサービスを呼び出すための所定の権限が必要です。

ロールを作成するときは、以下のアクセス許可ポリシーを使用してください。

```
{
  "Statement": [
    {
      "Sid": "IdCPermissions",
      "Effect": "Allow",
      "Action": [
        "sso-oauth:*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "GlueandLakePermissions",
      "Effect": "Allow",
      "Action": [
        "glue:*",
        "lakeformation:GetDataAccess"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AccessGrantsPermissions",
      "Effect": "Allow",
      "Action": [
        "s3:GetDataAccess",
        "s3:GetAccessGrantsInstanceForPrefix"
      ],
      "Resource": "*"
    }
  ]
}
```

このロールの信頼ポリシーにより、InstanceProfile ロールにロールを引き継がせることができます。

```
{
```

```
   "Sid": "AssumeRole",
   "Effect": "Allow",
   "Principal": {
        "AWS": "arn:aws:iam::12345678912:role/EMR_EC2_DefaultRole"
    },
    "Action": [
               "sts:AssumeRole",
               "sts:SetContext"
    ]
 }
```

Identity Center 対応のセキュリティ設定を作成します。

IAM Identity Center 統合で EMR クラスターを起動するには、以下のサンプルコマンドを使用して、Identity Center を有効にする Amazon EMR セキュリティ設定を作成します。各設定について以下で説明します。

```
aws emr create-security-configuration --name "IdentityCenterConfiguration-with-lf-
accessgrants" --region "us-west-2" --security-configuration '{
 "AuthenticationConfiguration":{
  "IdentityCenterConfiguration":{
   "EnableIdentityCenter":true,
   "IdentityCenterApplicationAssigmentRequired":false,
   "IdentityCenterInstanceARN": "arn:aws:sso:::instance/ssoins-123xxxxxxxxx789",
   "IAMRoleForEMRIdentityCenterApplicationARN": "arn:aws:iam::123456789012:role/tip-
role"
     }
 },
 "AuthorizationConfiguration": {
  "LakeFormationConfiguration": {
   "EnableLakeFormation": true
  }
 },
 "EncryptionConfiguration": {
  "EnableInTransitEncryption": true,
  "EnableAtRestEncryption": false,
  "InTransitEncryptionConfiguration": {
   "TLSCertificateConfiguration": {
    "CertificateProviderType": "PEM",
    "S3Object": "s3://my-bucket/cert/my-certs.zip"
   }
  }
 }
```

```
}'
```

- **`EnableIdentityCenter`** - (必須) Identity Center 統合を有効にします。
- **`IdentityCenterApplicationARN`** - (必須) Identify Center インスタンスの ARN。
- **`IAMRoleForEMRIdentityCenterApplicationARN`** - (必須) クラスターから Identity Center トークンを取得する IAM ロール。
- **`IdentityCenterApplicationAssignmentRequired`** - (ブーリアン型) Identity Center アプリケーションを使用するために割り当てが必要かどうかを制御します。デフォルト値は、`true`です。
- **`AuthorizationConfiguration`** / **`LakeFormationConfiguration`** – オプションで、認証を設定します。
  - **`EnableLakeFormation`** - クラスタ上で Lake Formation 認証を有効にします。

Identity Center と Amazon EMR の統合を有効にするには、`EncryptionConfiguration` と `IntransitEncryptionConfiguration` を指定する必要があります。

Identify Center 対応クラスターを作成して起動します。

ここまでで、Identity Center で認証する IAM ロールを設定し、Identity Center を有効にする Amazon EMR セキュリティ設定を作成できたので、Identify Center 対応クラスターを作成して起動できます。必要なセキュリティ設定を使用してクラスターを起動するステップについては、「クラスターのセキュリティ設定を指定する」を参照してください。

必要に応じて、Identity Center 対応クラスターを Amazon EMR がサポートする他のセキュリティオプションとともに使用する場合は、以下のセクションを参照してください。

- IAM Identity Center 対応 EMR クラスターでの S3 Access Grants の使用
- IAM Identity Center 対応 EMR クラスター用の Lake Formation を設定する

IAM Identity Center 対応 EMR クラスター用の Lake Formation を設定する

AWS IAM Identity Center を有効にした EMR クラスターAWS Lake Formationと統合できます。

まず、クラスターと同じリージョンに Identity Center インスタンスが設定されていることを確認します。詳細については、「Identity Center インスタンスを作成する」を参照してください。IAM Identity Center コンソールでインスタンスの詳細を確認するか、または以下のコマンドを使用して CLI からすべてのインスタンスの詳細を確認することにより、インスタンス ARN を見つけます。

```
aws sso-admin list-instances
```

次に、次のコマンドで ARN と AWS アカウント ID を使用して、Lake Formation を IAM Identity Center と互換性があるように設定します。

```
aws lakeformation create-lake-formation-identity-center-configuration --cli-input-json
 file://create-lake-fromation-idc-config.json
json input:
{
    "CatalogId": "account-id/org-account-id",
    "InstanceArn": "identity-center-instance-arn"
}
```

次に、`put-data-lake-settings` を呼び出して、Lake Formation に対して `AllowFullTableExternalDataAccess` を有効にします。

```
aws lakeformation put-data-lake-settings --cli-input-json file://put-data-lake-
settings.json
json input:
{
    "DataLakeSettings": {
        "DataLakeAdmins": [
            {
                "DataLakePrincipalIdentifier": "admin-ARN"
            }
        ],
        "CreateDatabaseDefaultPermissions": [...],
        "CreateTableDefaultPermissions": [...],
        "AllowExternalDataFiltering": true,
        "AllowFullTableExternalDataAccess": true
    }
}
```

最後に、EMR クラスターにアクセスするユーザーの ID ARN に、フルテーブルアクセス許可を付与します。ARN には Identity Center から取得したユーザー ID が含まれます。コンソールで Identity Center に移動し、[ユーザー]を選択し、該当するユーザーを選択して[一般情報]設定を表示します。

ユーザー ID をコピーして、以下の *user-id* 用 ARN に貼り付けます。

```
arn:aws:identitystore:::user/user-id
```

> **Note**
>
> EMR クラスターへのクエリは、その IAM Identity Center ID が、Lake Formation で保護されたテーブルに対するフルテーブルアクセス許可を持っている場合にのみ機能します。ID がフルテーブルアクセス許可を持っていない場合、クエリは失敗します。

以下のコマンドを使用して、ユーザーにフルテーブルアクセス許可を付与します。

```
aws lakeformation grant-permissions --cli-input-json file://grantpermissions.json
json input:
{
    "Principal": {
        "DataLakePrincipalIdentifier": "arn:aws:identitystore:::user/user-id"
    },
    "Resource": {
        "Table": {
            "DatabaseName": "tip_db",
            "Name": "tip_table"
        }
    },
    "Permissions": [
        "ALL"
    ],
    "PermissionsWithGrantOption": [
        "ALL"
    ]
}
```

IAM Identity Center 対応 EMR クラスターでの S3 Access Grants の使用

S3 Access Grants は、 AWS IAM Identity Center 有効な EMR クラスターと統合できます。

S3 Access Grants を使用して、Identity Center を使用するクラスターからのデータセットへのアクセスを許可します。許可を作成して、IAM ユーザー、グループ、ロール、または社内ディレクトリ用に設定した権限を補強します。詳細については、「Amazon EMR での S3 Access Grants の使用」を参照してください。

トピック

- S3 Access Grants インスタンスおよびロケーションを作成する
- Identity Center の ID 用許可を作成します。

S3 Access Grants インスタンスおよび口ケーションを作成する

まだ S3 Access Grants インスタンスを作成していない場合は、EMR クラスターを起動する AWS リージョン に S3 Access Grants インスタンスを作成します。

次の AWS CLI コマンドを使用して、 という名前の新しいインスタンスを作成します*MyInstance*。

```
aws s3control-access-grants create-access-grants-instance \
--account-id 12345678912 \
--identity-center-arn "identity-center-instance-arn" \
```

次に S3 Access Grants 口ケーションを作成し、赤色の値を自分の使用する値に置き換えます。

```
aws s3control-access-grants create-access-grants-location \
--account-id 12345678912 \
--location-scope s3:// \
--iam-role-arn "access-grant-role-arn" \
--region aa-example-1
```

> **Note**
>
> iam-role-arn パラメータを accessGrantRole ARN として定義します。

Identity Center の ID 用許可を作成します。

最後に、クラスターにアクセスできる ID の権限を作成します。

```
aws s3control-access-grants create-access-grant \
--account-id 12345678912 \
--access-grants-location-id "default" \
--access-grants-location-configuration S3SubPrefix="s3-bucket-prefix"
--permission READ \
--grantee GranteeType=DIRECTORY_USER,GranteeIdentifier="your-identity-center-user-id"
```

出力例:

```
{
"CreatedAt": "2023-09-21T23:47:24.870000+00:00",
"AccessGrantId": "1234-12345-1234-1234567",
```

```
"AccessGrantArn": "arn:aws:s3:aa-example-1-1:123456789012:access-grants/default/grant/
xxxx1234-1234-5678-1234-1234567890",
"Grantee": {
"GranteeType": "DIRECTORY_USER",
"GranteeIdentifier": "5678-56789-5678-567890"
},
"AccessGrantsLocationId": "default",
"AccessGrantsLocationConfiguration": {
"S3SubPrefix": "myprefix/*"
},
"Permission": "READ",
"GrantScope": "s3://myprefix/*"
}
```

Amazon EMR と Identity Center の統合に関する考慮事項と制限

Amazon EMR で IAM Identity Center を使用するときは、以下の点に考慮してください。

- アイデンティティセンターが提供する信頼できる ID 伝達は、Amazon EMR 6.15.0 以降でサポートされており、Apache Spark でのみ動作します。
- 信頼できる ID の伝播で EMR クラスターを有効にするには、 を使用して、信頼できる ID の伝播が有効になっているセキュリティ設定 AWS CLI を作成し、クラスターの起動時にそのセキュリティ設定を使用する必要があります。詳細については、「Identity Center 対応のセキュリティ設定を作成します。」を参照してください。
- 信頼できる ID 伝達を使用する EMR クラスターは、同じく信頼できる ID 伝達を使用するサービスのみを呼び出すことができます。
- 信頼 AWS Lake Formation された ID 伝達を使用する EMR クラスターでは、 に基づくテーブルレベルのアクセスコントロールのみを使用できます。
- 信頼できる ID 伝達を使用する EMR クラスターでの、Apache Spark と Lake Formation に基づくアクセス制御をサポートする操作には、`SELECT`、`ALTER TABLE`、`DROP TABLE` が含まれます
- 信頼できる ID 伝達を使用する EMR クラスターでの、Apache Spark でサポートしていない Lake Formation ベースのアクセス制御には `INSERT` ステートメントが含まれます。
- Amazon EMR による信頼できる ID の伝播は、次の でサポートされています AWS リージョン。
  - `ap-east-1` – アジアパシフィック (香港)
  - `ap-northeast-1` – アジアパシフィック (東京)
  - `ap-northeast-2` – アジアパシフィック (ソウル)
  - `ap-south-1` – アジアパシフィック (ムンバイ)

- ap-southeast-1 – アジアパシフィック (シンガポール)
- ap-southeast-2 – アジアパシフィック (シドニー)
- ca-central-1 — カナダ (中部)
- eu-central-1 – 欧州 (フランクフルト)
- eu-north-1 – 欧州 (ストックホルム)
- eu-west-1 – 欧州 (アイルランド)
- eu-west-2 – 欧州 (ロンドン)
- eu-west-3 – 欧州 (パリ)
- me-south-1 – 中東 (バーレーン)
- sa-east-1 – 南米 (サンパウロ)
- us-east-1 — 米国東部 (バージニア北部)
- us-east-2 — 米国東部 (オハイオ)
- us-west-1 — 米国西部 (北カリフォルニア)
- us-west-2 — 米国西部 (オレゴン)

Amazon EMR を と統合する AWS Lake Formation

AWS Lake Formation は、Amazon Simple Storage Service (S3) データレイク内のデータを検出、カタログ化、クレンジング、保護するのに役立つマネージドサービスです。Lake Formation は、 AWS Glue データカタログ内のデータベースとテーブルへのきめ細かな列レベルのアクセスを提供します。詳細については、「What is AWS Lake Formation?」を参照してください。

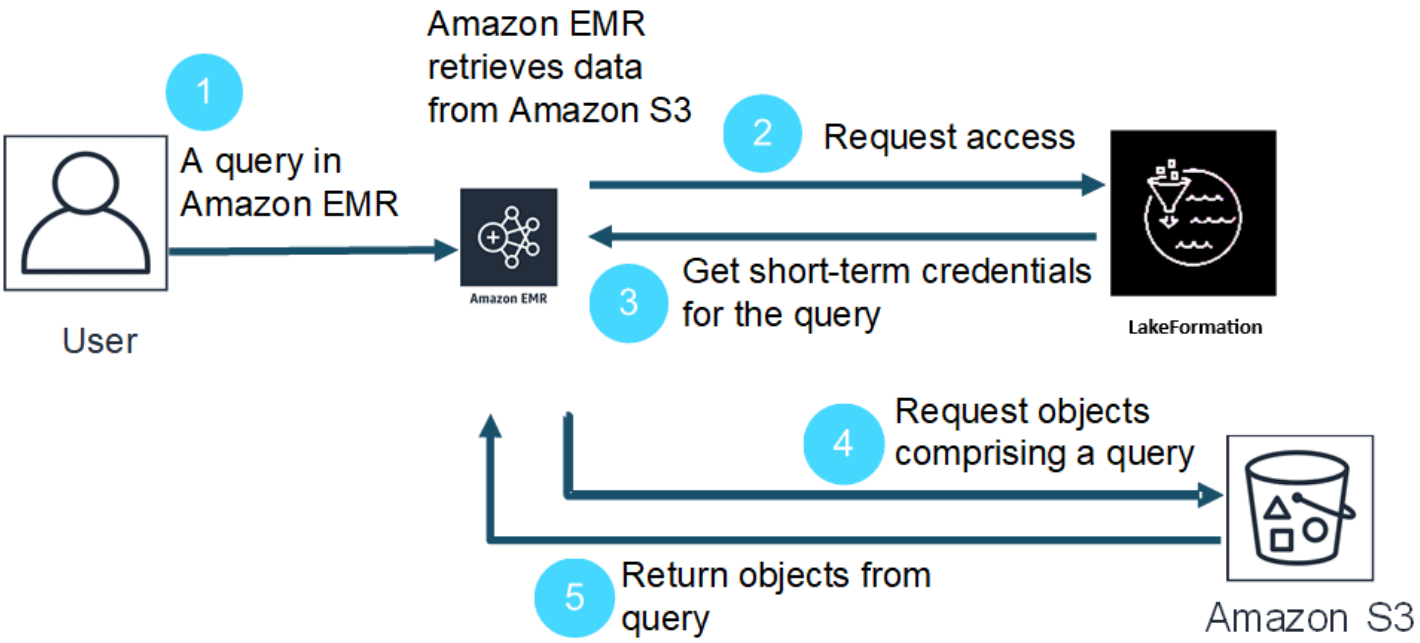
Amazon EMR リリース 6.7.0 以降では、Amazon EMR クラスターに送信する Spark、Hive、Presto のジョブに Lake Formation ベースのアクセスコントロールを適用できます。Lake Formation と統合するには、ランタイムロールを使用する EMR クラスターを作成する必要があります。ランタイムロールは、Amazon EMR のジョブまたはクエリに関連付ける AWS Identity and Access Management (IAM) ロールです。その後、Amazon EMR はこのロールを使用して AWS リソースにアクセスします。詳細については、「Amazon EMR ステップのランタイムロール」を参照してください。

Amazon EMR と Lake Formation の連携の仕組み

Amazon EMR を Lake Formation と統合したら、 Step API または SageMaker Studio を使用して Amazon EMR クラスターへのクエリを実行できます。その後、Lake Formation は Amazon EMR 用

の一時的な認証情報を使用してデータへのアクセスを提供します。このプロセスは、認証情報の供給と呼ばれます。詳細については、「What is AWS Lake Formation?」を参照してください。

以下は、Amazon EMR が Lake Formation セキュリティポリシーで保護されたデータにアクセスする方法の概要を示します。



1. ユーザーが Lake Formation 内のデータに対して Amazon EMR クエリを送信します。
2. Amazon EMR は、ユーザーにデータアクセス権を付与するために、Lake Formation に一時的な認証情報をリクエストします。
3. Lake Formation が一時的な認証情報を返します。
4. Amazon EMR は、Amazon S3 からデータを取り出すためのクエリリクエストを送信します。
5. Amazon EMR は Amazon S3 からデータを受信し、ユーザーが Lake Formation で定義したユーザーアクセス許可に基づいてフィルタリングし、結果を返します。

Lake Formation ポリシーへのユーザーおよびグループの追加については、「Granting Data Catalog permissions」を参照してください。

前提条件

Amazon EMR と Lake Formation を統合する前に、次の要件を満たす必要があります。

- Amazon EMR クラスターでランタイムロール認証を有効にします。

- AWS Glue データカタログをメタデータストアとして使用します。
- AWS Glue Data Catalog のデータベース、テーブル、列にアクセスするためのアクセス許可を Lake Formation で定義および管理します。詳細については、「What is AWS Lake Formation?」を参照してください。

トピック

Amazon EMR での Lake Formation の有効化

Amazon EMR 6.15.0 以降では、 AWS Glue Data Catalog のデータにアクセスする EC2 クラスターで Amazon EMR で Spark ジョブを実行すると、 AWS Lake Formation を使用して Hudi、Iceberg、または Delta Lake ベースのテーブルにテーブル、行、列、およびセルレベルのアクセス許可を適用できます。

このセクションでは、セキュリティ設定を作成し、Amazon EMR と連携するように Lake Formation を設定する方法について説明します。Lake Formation 用に作成したセキュリティ設定を使用してクラスターを起動する方法についても説明します。

ステップ 1: EMR クラスターのランタイムロールを設定する

EMR クラスターにランタイムロールを使用するには、セキュリティ設定を作成する必要があります。セキュリティ設定により、クラスター全体で一貫したセキュリティ、承認、認証のオプションを適用できます。

1. 次のセキュリティ設定で `lf-runtime-roles-sec-cfg.json` というファイルを作成します。

```
{
    "AuthorizationConfiguration": {
        "IAMConfiguration": {
            "EnableApplicationScopedIAMRole": true,
            "ApplicationScopedIAMRoleConfiguration": {
                "PropagateSourceIdentity": true
```

```
            }
        },
        "LakeFormationConfiguration": {
            "AuthorizedSessionTagValue": "Amazon EMR"
        }
    },
    "EncryptionConfiguration": {
            "EnableInTransitEncryption": true,
            "InTransitEncryptionConfiguration": {
            "TLSCertificateConfiguration": {<certificate-configuration>}
        }
    }
}
```

2. 次に、セッションタグが Lake Formation を承認できるようにするために、`LakeFormationConfiguration/AuthorizedSessionTagValue` プロパティを `Amazon EMR` に設定します。
3. 次のコマンドを使用して、Amazon EMR セキュリティ設定を作成します。

```
aws emr create-security-configuration \
--name 'iamconfig-with-iam-lf' \
--security-configuration file://lf-runtime-roles-sec-cfg.json
```

または、Amazon EMR コンソールを使用して、カスタム設定でセキュリティ設定を作成することもできます。

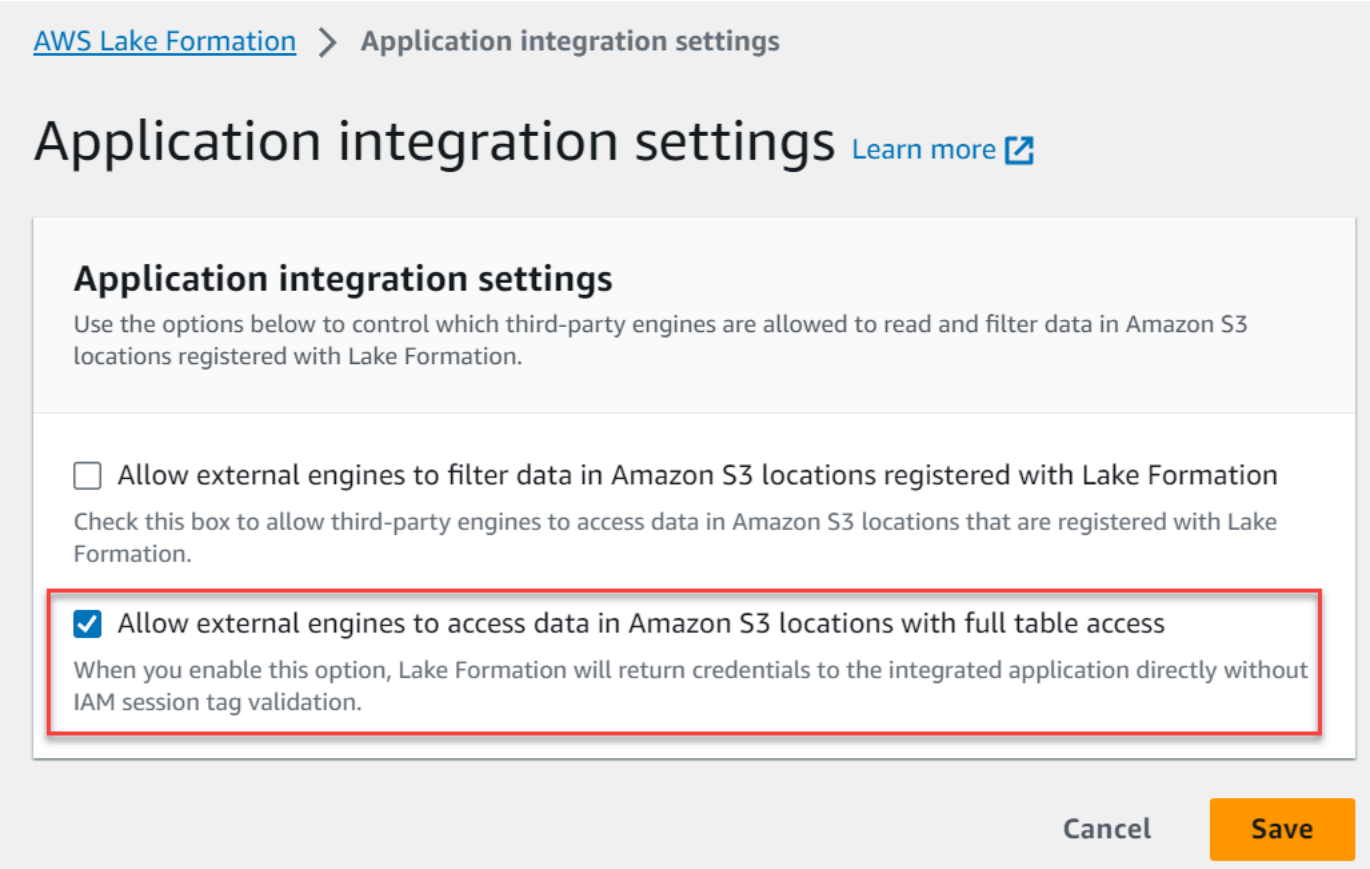
ステップ 2: Amazon EMR クラスターを起動します

これで、前のステップで作成したセキュリティ設定を使用して EMR クラスターを起動する準備ができました。セキュリティ設定の詳細については、「セキュリティ設定を使用してクラスターセキュリティをセットアップする」および「Amazon EMR ステップのランタイムロール」を参照してください。

ステップ 3a: Amazon EMR ランタイムロールを使用して、Lake Formation ベースのテーブルレベルのアクセス許可を設定する

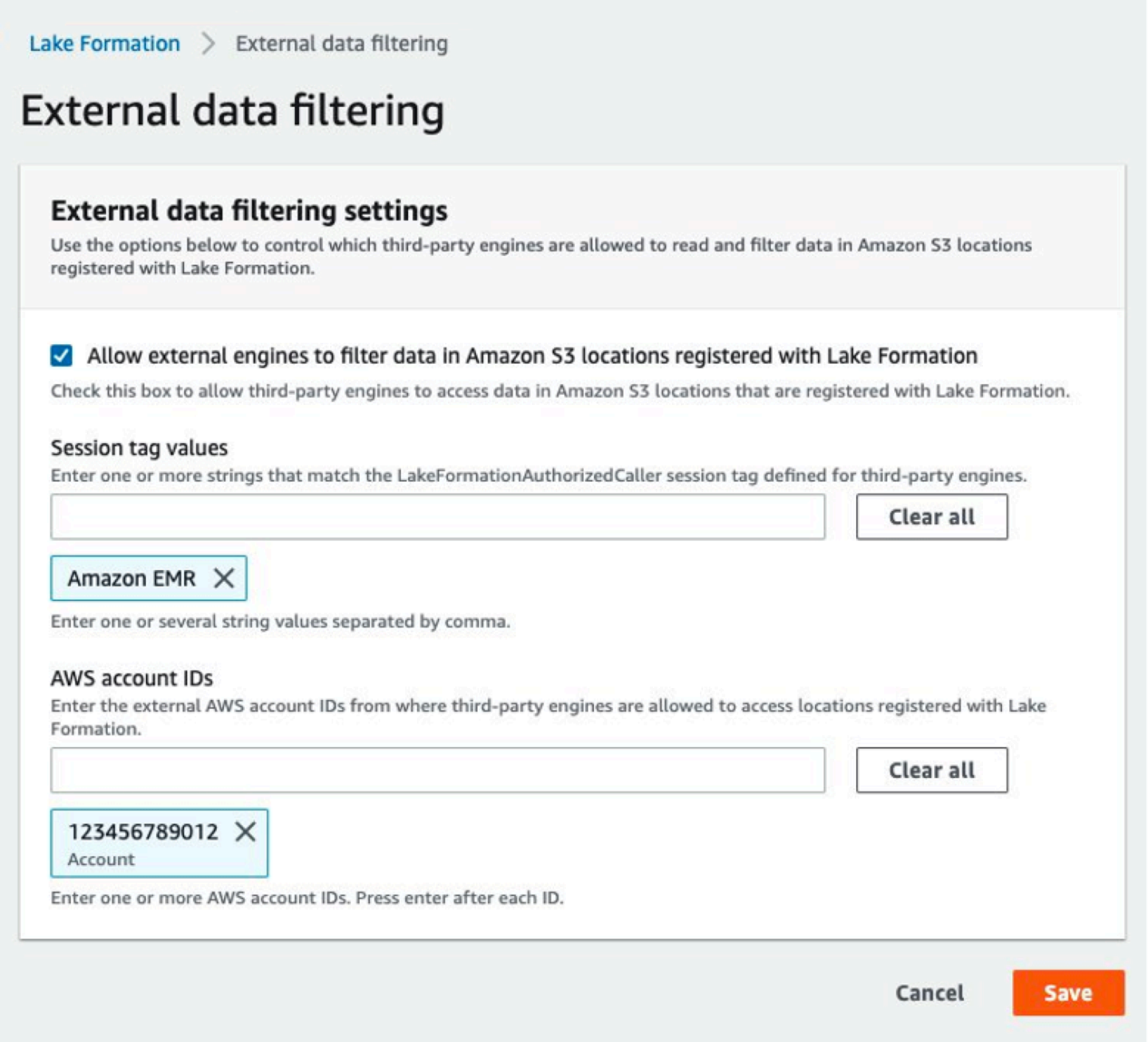
列、行、またはセルレベルでの細粒度なアクセス制御が必要ない場合は、Glue Data Catalog を使用してテーブルレベルの権限を設定できます。テーブルレベルのアクセスを有効にするには、 AWS Lake Formation コンソールに移動し、サイドバーの管理セクションからアプリケーション統合設定オプションを選択します。次に、以下のオプションを有効にして [保存] を選択します。

[部エンジンに、Amazon S3 ロケーション内のデータに対するフルアクセス許可を付与する]



ステップ 3b: Amazon EMR ランタイムロールを使用して Lake Formation ベースの列、行、セルレベルのアクセス許可を設定する

Lake Formation でテーブルレベルおよび列レベルのアクセス許可を適用するには、データレイク管理者がセッションタグ設定 `AuthorizedSessionTagValue` の値として `Amazon EMR` を設定する必要があります。Lake Formation は、このセッションタグを使用して発信者を承認し、データレイクへのアクセス権限を付与します。このセッションタグは、Lake Formation コンソールの [外部データフィルタリング] セクションで設定できます。*123456789012* は、自分の AWS アカウント ID に置き換えてください。



ステップ 4: Amazon EMR ランタイムロールの AWS Glue および Lake Formation 許可を設定する

Amazon EMR ランタイムロールを使用した Lake Formation ベースのアクセスコントロールの設定を続行するには、Amazon EMR ランタイムロールの AWS Glue および Lake Formation 許可を設定する必要があります。IAM ランタイムロールが Lake Formation と対話できるようにするには、`lakeformation:GetDataAccess` および `glue:Get*` を使用してアクセス権限を付与します。

Lake Formation のアクセス許可は、 AWS Glue Data Catalog リソース、Amazon S3 ロケーション、およびそれらのロケーションの基盤となるデータへのアクセスを制御します。IAM アクセス許可は、Lake Formation および AWS Glue API とリソースへのアクセスを制御します。データカタログ内のテーブルにアクセスするための Lake Formation アクセス許可 (SELECT) を持っていても、`glue:Get*` API に対する IAM アクセス許可がない場合、操作は失敗します。Lake Formation のアクセスコントロールの詳細については、「Lake Formation アクセスコントロールの概要」を参照してください。

1. 次の内容で `emr-runtime-roles-lake-formation-policy.json` ファイルを作成します。

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Sid": "LakeFormationManagedAccess",
        "Effect": "Allow",
        "Action": [
            "lakeformation:GetDataAccess",
            "glue:Get*",
            "glue:Create*",
            "glue:Update*"
        ],
        "Resource": "*"
    }
}
```

2. 関連する IAM ポリシーを作成します。

```
aws iam create-policy \
--policy-name emr-runtime-roles-lake-formation-policy \
--policy-document file://emr-runtime-roles-lake-formation-policy.json
```

3. このポリシーを IAM ランタイムロールに割り当てるには、「AWS Lake Formation 許可の管理」のステップを実行します。

ランタイムロールと Lake Formation を使用して、テーブルレベルと列レベルのアクセス許可を適用できるようになりました。ソース ID を使用して、 でアクションを制御し、オペレーションをモニタリングすることもできます AWS CloudTrail。詳細な end-to-end 例については、「Amazon EMR ステップのランタイムロールの紹介」を参照してください。

Apache Hudi と Lake Formation

Amazon EMR リリース 6.15.0 以降には、Spark SQL AWS Lake Formation でデータを読み書きするときに、Apache Hudi を使用した に基づくきめ細かなアクセスコントロールのサポートが含まれています。Amazon EMR は、Apache Hudi を使用した、テーブル、行、列、セルレベルのアクセスコントロールをサポートしています。この機能を使用すると、 copy-on-write テーブルに対してスナップショットクエリを実行して、特定のコミットまたは圧縮の瞬間にテーブルの最新のスナップショットをクエリできます。

現在、Lake Formation 対応 Amazon EMR クラスターは、増分クエリとタイムトラベルクエリを実行するために Hudi のコミット時間列を取得する必要があります。Spark の`timestamp as of`構文と `Spark.read()`関数はサポートされていません。正しい構文は ですselect * from table where _hoodie_commit_time <= point_in_time。詳細については、「Hudi テーブル でのポイントインタイム- クエリ」を参照してください。

以下のサポートマトリックスには、Apache Hudi と Lake Formation の連携の主要機能の一部がリストされています。

	Copy on Write	読み取り時マージ
スナップショットクエリ - Spark SQL	✓	✓
読み取り最適化クエリ - Spark SQL	✓	✓
増分クエリ	✓	✓
タイムトラベルクエリ	✓	✓
メタデータテーブル	✓	✓
DML **`INSERT`** コマンド	✓	✓
DDL コマンド		
Spark データソースクエリ		
Spark データソース書き込み		

Hudi テーブルのクエリ

このセクションでは、Lake Formation 対応クラスターで前述のサポート対象クエリを実行する方法を示します。テーブルは、登録済みのカタログテーブルである必要があります。

1. Spark シェルを起動するには、次のコマンドを使用します。

```
spark-sql
--jars /usr/lib/hudi/hudi-spark-bundle.jar \
--conf spark.serializer=org.apache.spark.serializer.KryoSerializer \
--conf
spark.sql.catalog.spark_catalog=org.apache.spark.sql.hudi.catalog.HoodieCatalog \
--conf
spark.sql.extensions=org.apache.spark.sql.hudi.HoodieSparkSessionExtension,com.amazonaws.em
 \
--conf spark.sql.catalog.spark_catalog.lf.managed=true
```

   Lake Formation でレコードサーバーを使用して Spark カタログを管理する場合は、`spark.sql.catalog.<managed_catalog_name>.lf.managed`を true に設定します。

2. copy-on-write テーブルの最新のスナップショットをクエリするには、次のコマンドを使用します。

```
SELECT * FROM my_hudi_cow_table
```

```
spark.read.table("my_hudi_cow_table")
```

3. MOR テーブルの最新の圧縮データをクエリするには、読み取り最適化テーブル (サフィックス `_ro` 付き) に対するクエリを実行します。

```
SELECT * FROM my_hudi_mor_table_ro
```

```
spark.read.table("my_hudi_mor_table_ro")
```

> **Note**
>
> Lake Formation クラスターでの読み取りのパフォーマンスは、最適化がサポートされていないために低くなる可能性があります。これらの機能には、Hudi メタデータに基づくファイ

ルリストやデータスキップなどがあります。アプリケーションのパフォーマンスをテストして、要件を満たしているか確認することをお勧めします。

Apache Iceberg と Lake Formation

Amazon EMR リリース 6.15.0 以降には、Spark SQL AWS Lake Formation でデータを読み書きするときに、Apache Iceberg による に基づくきめ細かなアクセスコントロールのサポートが含まれています。Amazon EMR は、Apache Iceberg を使用した、テーブル、行、列、セルレベルのアクセスコントロールをサポートしています。この機能を使用すると、 copy-on-write テーブルに対してスナップショットクエリを実行して、特定のコミットまたは圧縮の瞬間にテーブルの最新のスナップショットをクエリできます。

Iceberg 形式を使用する場合は、以下の設定を設定します。*DB_LOCATION* を Iceberg テーブルが置かれている Amazon S3 パスに置き換え、リージョンとアカウント ID のプレースホルダーを自分の使用する値に置き換えます。

```
spark-sql \
--conf
 spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions,com.ama

--conf spark.sql.catalog.iceberg_catalog=org.apache.iceberg.spark.SparkCatalog
--conf spark.sql.catalog.iceberg_catalog.warehouse=s3://DB_LOCATION
--conf spark.sql.catalog.iceberg_catalog.catalog-
impl=org.apache.iceberg.aws.glue.GlueCatalog
--conf spark.sql.catalog.iceberg_catalog.io-impl=org.apache.iceberg.aws.s3.S3FileIO
--conf spark.sql.catalog.iceberg_catalog.glue.account-id=ACCOUNT_ID
--conf spark.sql.catalog.iceberg_catalog.glue.id=ACCOUNT_ID
--conf spark.sql.catalog.iceberg_catalog.client.assume-role.region=AWS_REGION
--conf spark.sql.secureCatalog=iceberg_catalog
```

Lake Formation でレコードサーバーを使用して Spark カタログを管理する場合は、`spark.sql.catalog.<managed_catalog_name>.lf.managed`を true に設定します。

また、以下のロール継承設定を渡さないように注意する必要があります。

```
--conf spark.sql.catalog.my_catalog.client.assume-role.region
--conf spark.sql.catalog.my_catalog.client.assume-role.arn
```

```
--conf spark.sql.catalog.my_catalog.client.assume-
role.tags.LakeFormationAuthorizedCaller
```

以下のサポートマトリックスには、Apache Iceberg と Lake Formation のコア機能の一部がリストされています。

	Copy on Write	読み取り時マージ
スナップショットクエリ - Spark SQL	✓	✓
読み取り最適化クエリ - Spark SQL	✓	✓
増分クエリ	✓	✓
タイムトラベルクエリ	✓	✓
メタデータテーブル	✓	✓
DML **`INSERT`** コマンド	✓	✓
DDL コマンド		
Spark データソースクエリ		
Spark データソース書き込み		

Delta Lake と Lake Formation

Amazon EMR リリース 6.15.0 以降には、Spark SQL AWS Lake Formation でデータを読み書きする場合に、Delta Lake による に基づくきめ細かなアクセスコントロールのサポートが含まれています。Amazon EMR は、Delta Lake を使用した、テーブル、行、列、セルレベルのアクセスコントロールをサポートしています。この機能を使用すると、 copy-on-write テーブルに対してスナップショットクエリを実行して、特定のコミットまたは圧縮の瞬間にテーブルの最新のスナップショットをクエリできます。

Lake Formation で Delta Lake を使用するには、次のコマンドを実行します。

```
spark-sql \
```

```
--conf
 spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension,com.amazonaws.emr.recordserver.co
 \
--conf spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog
 \
--conf spark.sql.catalog.spark_catalog.lf.managed=true
```

Lake Formation でレコードサーバーを使用して Spark カタログを管理する場合は、`spark.sql.catalog.<managed_catalog_name>.lf.managed`を true に設定します。

次のサポートマトリックスは、Lake Formation を使用した Delta Lake のいくつかのコア機能を示しています。

	Copy on Write	読み取り時マージ
スナップショットクエリ - Spark SQL	✓	✓
読み取り最適化クエリ - Spark SQL	✓	✓
増分クエリ	サポートされません	サポートされません
タイムトラベルクエリ	サポートされません	サポートされません
メタデータテーブル	✓	✓
DML **`INSERT`** コマンド	✓	✓
DDL コマンド		
Spark データソースクエリ		
Spark データソース書き込み		

AWS Glue Data Catalog での Delta Lake テーブルの作成

Amazon EMR with Lake Formation は DDL コマンドと Delta テーブルの作成をサポートしていません。 AWS Glue データカタログにテーブルを作成するには、次の手順に従います。

1. 次の例を使用して Delta テーブルを作成します。S3 の場所が存在することを確認します。

   ```
   spark-sql \
   ```

```
--conf "spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension" \
--conf
 "spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog"

> CREATE DATABASE if not exists <DATABASE_NAME> LOCATION 's3://<S3_LOCATION>/
transactionaldata/native-delta/<DATABASE_NAME>/';
> CREATE TABLE <TABLE_NAME> (x INT, y STRING, z STRING) USING delta;
> INSERT INTO <TABLE_NAME> VALUES (1, 'a1', 'b1');
```

2. テーブルの詳細を確認するには、https://console.aws.amazon.com/glue/ にアクセスしてください。
3. 左側のナビゲーションで、データカタログ を展開し、テーブル を選択し、作成したテーブルを選択します。スキーマ では、Spark で作成した Delta テーブルに、 AWS Glue のデータ型のすべての列が格納されていることがわかりarray<string>ます。
4. Lake Formation で列とセルレベルのフィルターを定義するには、スキーマからcol1列を削除し、テーブルスキーマにある列を追加します。この例では、列 x、、 yを追加しますz。

Amazon EMR での Lake Formation の使用に関する考慮事項

で Amazon EMR を使用する場合は、次の点を考慮してください AWS Lake Formation。

- テーブルレベルのアクセスコントロールは、Amazon EMR リリース 6.13 以降のクラスターで使用できます。
- Amazon EMR リリース 6.15 以降のクラスターでは、行、列、セルレベルでのきめ細かなアクセス制御が可能です。
- テーブルにアクセスできるユーザーは、そのテーブルのすべてのプロパティにアクセスできます。Lake Formation ベースのアクセスコントロールをテーブルで使用している場合は、テーブルを確認して、プロパティに機密データや情報が含まれていないことを確認してください。
- Lake Formation を使用する Amazon EMR クラスターでは、Spark がテーブル統計を収集する場合の Spark の HDFS へのフォールバックをサポートしていません。これは通常、クエリのパフォーマンスを最適化するのに役立ちます。
- Lake Formation に基づくアクセスコントロールをサポートする、非管理対象の Apache Spark テーブルによる操作には、INSERT INTO や INSERT OVERWRITE などがあります。
- Lake Formation に基づくアクセスコントロールをサポートする、Apache Spark と Apache Hive での操作には、SELECT、DESCRIBE、SHOW DATABASE、SHOW TABLE、SHOW COLUMN、SHOW PARTITION などがあります。

- Amazon EMR は、以下の Lake Formation ベースの操作用アクセスコントロールをサポートしていません。
  - 管理対象テーブルへの書き込み
  - Amazon EMR は `CREATE TABLE` をサポートしていません。Amazon EMR 6.10.0 以降は、`ALTER TABLE`をサポートしています。
  - `INSERT` コマンド以外の DML ステートメント。
- Lake Formation ベースのアクセスコントロールを使用する場合と使用しない場合では、同じクエリでパフォーマンスに差が生じます。

Amazon EMR と Apache Ranger を統合する

Amazon EMR 5.32.0 以降では、Apache Ranger とネイティブに統合されるクラスターを起動できます。Apache Ranger は、Hadoop プラットフォーム全体で包括的なデータセキュリティを有効化、監視、管理するためのオープンソースフレームワークです。詳細については、「Apache Ranger」を参照してください。ネイティブ統合により、独自の Apache Ranger を導入して、Amazon EMR できめ細かなデータアクセス制御を実施できます。

このセクションでは、Amazon EMR と Apache Ranger の統合の概念的な概要を示します。また、Apache Ranger と統合された Amazon EMR クラスターを起動するために必要な前提条件と手順も示します。

Amazon EMR と Apache Ranger のネイティブ統合には、次の主な利点があります。

- Hive Metastore データベースおよびテーブルへのきめ細かなアクセスコントロール。これにより、Apache Spark および Apache Hive アプリケーションにおいてデータベース、テーブル、列のレベルでデータフィルタリングポリシーを定義できます。行レベルのフィルタリングとデータマスキングは、Hive アプリケーションでサポートされています。
- Hive アプリケーション用に Amazon EMR で既存の Hive ポリシーを直接使用する機能。
- プレフィックスおよびオブジェクトレベルでの Amazon S3 データへのアクセスコントロール。これにより、EMR ファイルシステムを使用して S3 データにアクセスするためのデータフィルタリングポリシーを定義できます。
- CloudWatch ログを使用して一元的な監査を行う機能。
- Amazon EMR はユーザーに代わって Apache Ranger プラグインをインストールおよび管理します。

Apache Ranger

Apache Ranger は、Hadoop プラットフォーム全体で包括的なデータセキュリティを有効化、監視、管理するためのフレームワークです。

Apache Ranger には以下の機能があります。

- 中央の UI または REST API を使用して、セキュリティ関連のすべてのタスクを管理するための一元化されたセキュリティ管理。
- 中央管理ツールで管理される、Hadoop コンポーネントまたはツールで特定のアクションまたはオペレーションを行うためのきめ細かな認可。
- すべての Hadoop コンポーネントで標準化された認可方法。
- さまざまな認可方法のサポートの強化。
- Hadoop のすべてのコンポーネント内のユーザーアクセスと管理アクション (セキュリティ関連) の一元的な監査。

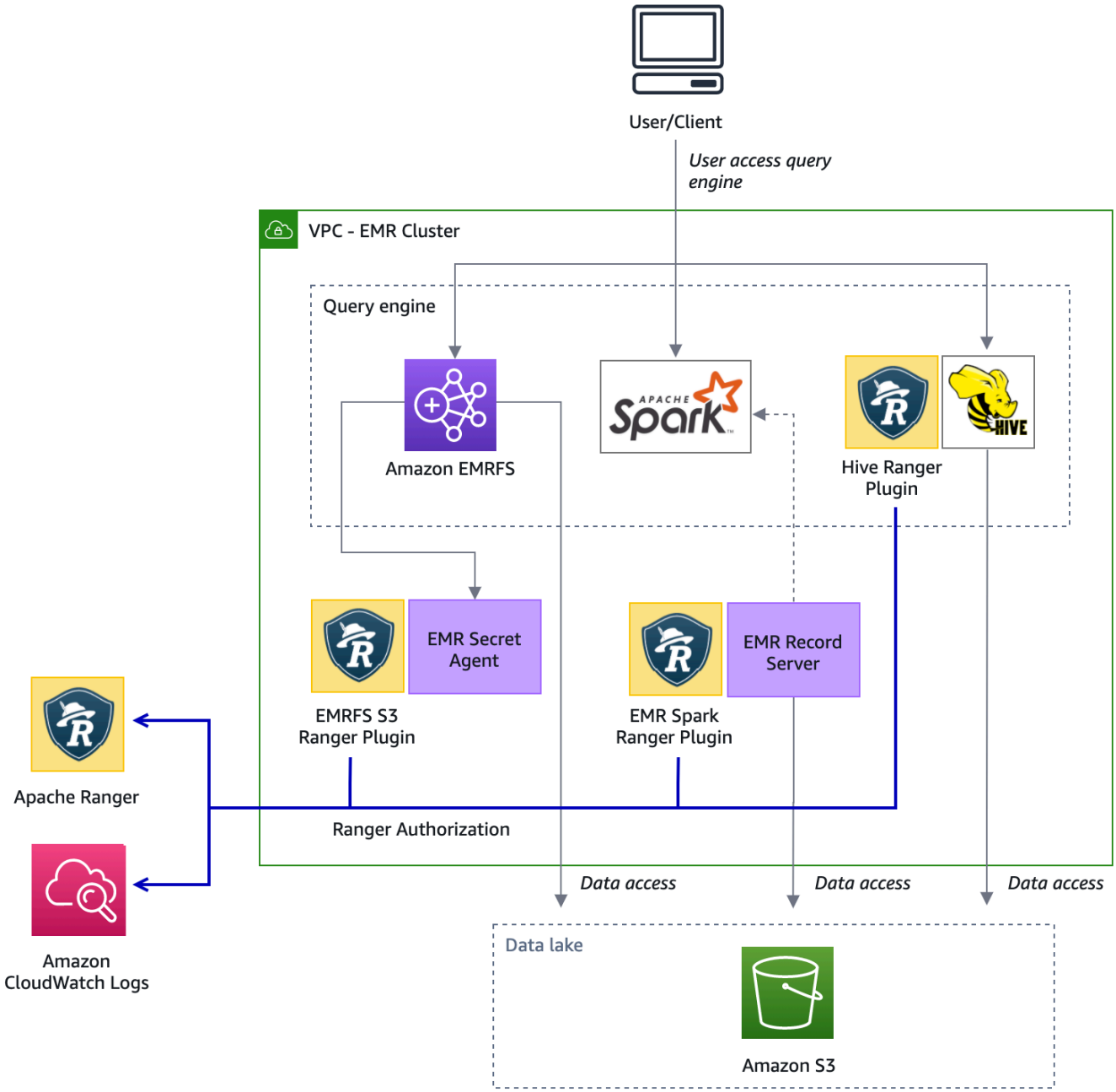
Apache Ranger は、認可に次の 2 つの主要コンポーネントを使用します。

- Apache Ranger ポリシー管理サーバー - このサーバーにより、Hadoop アプリケーションの認可ポリシーを定義できます。Amazon EMR と統合する場合、Hive メタストアにアクセスし、Amazon S3 データの EMR ファイルシステム (EMRFS) にアクセスするための Apache Spark と Hive 用のポリシーを定義および適用できます。新しい Apache Ranger ポリシー管理サーバーを設定するか、既存の Apache Ranger ポリシー管理サーバーを使用して Amazon EMR と統合できます。
- Apache Ranger プラグイン - このプラグインは、Apache Ranger ポリシー管理サーバーで定義された認可ポリシーに対してユーザーのアクセスを検証します。Amazon EMR は、Apache Ranger 設定で選択された各 Hadoop アプリケーションに対して Apache Ranger プラグインを自動的にインストールして設定します。

トピック

- Amazon EMR と Apache Ranger 統合のアーキテクチャ
- Amazon EMR コンポーネント

Amazon EMR と Apache Ranger 統合のアーキテクチャ



Amazon EMR コンポーネント

Amazon EMR は、以下のコンポーネントを使用して Apache Ranger を使用したきめ細かなアクセスコントロールを有効にします。これらの Amazon EMR コンポーネントと Apache Ranger プラグインの視覚的な表現については、「アーキテクチャ図」を参照してください。

シークレットエージェント - シークレットエージェントはシークレットを安全に保存し、他の Amazon EMR コンポーネントまたはアプリケーションにシークレットを分散します。シークレットには、一時的なユーザー認証情報、暗号化キー、または Kerberos チケットを含めることができます。シークレットエージェントは、クラスター内のすべてのノードで実行され、インスタンスメタデータサービスへの呼び出しをインターセプトします。インスタンスプロファイルロールの認証情報へのリクエストの場合、シークレットエージェントは、EMRFS S3 Ranger プラグインでリクエストを認可した後、リクエストしたユーザーおよびリクエストされたリソースに応じて認証情報を送信します。シークレットエージェントは *emrsecretagent* ユーザーとして実行され、/emr/secretagent/log ディレクトリにログを書き込みます。このプロセスは、機能の特定の `iptables` ルールのセットに依存しています。`iptables` が無効になっていることを確認することが重要です。`iptables` 設定をカスタマイズする場合、NAT テーブルルールを保持し、変更しないようにする必要があります。

EMR レコードサーバー - レコードサーバーは、Spark からのデータへのアクセスリクエストを受け取ります。その後、リクエストされたリソースを Amazon EMR の Spark Ranger プラグインに転送して、リクエストを認可します。レコードサーバーは Amazon S3 からデータを読み取り、Ranger ポリシーに基づいてユーザーがアクセスを許可されている、フィルタリングされたデータを返します。レコードサーバーは emr_record_server ユーザーとしてクラスター内のすべてのノードで実行され、/var/log/emr-record-server ディレクトリにログを書き込みます。

アプリケーションのサポートと制限

サポートされているアプリケーション

EMR が Ranger プラグインをインストールする Amazon EMR と Apache Ranger との統合は、現在、以下のアプリケーションをサポートしています。

- Apache Spark (EMR 5.32 以上および EMR 6.3 以上で利用可能)
- Apache Hive (EMR 5.32 以上および EMR 6.3 以上で利用可能)
- EMRFS 経由の S3 アクセス (EMR 5.32 以上および EMR 6.3 以上で利用可能)

次のアプリケーションを EMR クラスターにインストールでき、セキュリティニーズを満たすように設定する必要がある場合があります。

- Apache Hadoop (YARN と HDFS を含む EMR 5.32 以上および EMR 6.3 以上で利用可能)
- Apache Livy (EMR 5.32 以上および EMR 6.3 以上で利用可能)
- Apache Zeppelin (EMR 5.32 以上および EMR 6.3 以上で利用可能)

- Apache Hue (EMR 5.32 以上および EMR 6.3 以上で利用可能)
- Ganglia (EMR 5.32 以上および EMR 6.3 以上で利用可能)
- HCatalog (EMR 5.32 以上および EMR 6.3 以上で利用可能)
- Mahout (EMR 5.32 以上および EMR 6.3 以上で利用可能)
- MXNet (EMR 5.32 以上および EMR 6.3 以上で利用可能)
- TensorFlow (EMR 5.32 以降および EMR 6.3 以降で使用可能)
- Tez (EMR 5.32 以上および EMR 6.3 以上で利用可能)
- Trino (EMR 6.7 以降で使用可能)
- ZooKeeper (EMR 5.32 以降および EMR 6.3 以降で使用可能)

> **Important**
>
> 上記のアプリケーションのみが、現在サポートされているアプリケーションです。クラスターのセキュリティを確保するために、Apache Ranger が有効になっている場合、上記のリスト内のアプリケーションのみが含まれた EMR クラスターを作成できます。
> 他のアプリケーションは現在サポートされていません。クラスターのセキュリティを確保するために、他のアプリケーションをインストールしようとすると、クラスターが拒否されます。

サポートされている機能

Amazon EMR と Apache Ranger では、次の Amazon EMR の機能を使用できます。

- 保管中と転送中の暗号化
- Kerberos 認証 (必須)
- インスタンスグループ、インスタンスフリート、スポットインスタンス
- 実行中のクラスター上のアプリケーションの再設定
- EMRFS サーバー側の暗号化 (SSE)

> **Note**
>
> Amazon EMR 暗号化設定は SSE を管理します。詳細については「暗号化オプション」を参照してください。

アプリケーションの制限事項

Amazon EMR と Apache Ranger を統合する際に留意すべきいくつかの制限があります。

- 現在、 コンソールを使用して、 で AWS Ranger 統合オプションを指定するセキュリティ設定を作成することはできません AWS GovCloud (US) Region。セキュリティ設定は CLI を使用して実行できます。
- クラスターに Kerberos をインストールする必要があります。
- YARN Resource Manager UIsHDFS UI、Livy NameNode UI などのアプリケーション UI (ユーザーインターフェイス) は、デフォルトでは認証を使用して設定されません。
- HDFS のデフォルトのアクセス許可 umask は、作成されたオブジェクトがデフォルトで `world wide readable` に設定されるように構成されています。
- Amazon EMR は、Apache Ranger での高可用性 (複数のプライマリ) モードをサポートしていません。
- その他の制限については、各アプリケーションの制限を参照してください。

> **Note**
>
> Amazon EMR 暗号化設定は SSE を管理します。詳細については「暗号化オプション」を参照してください。

プラグインの制限事項

各プラグインには特定の制限があります。Apache Hive プラグインの制限については、「Apache Hive プラグインの制限事項」を参照してください。Apache Spark プラグインの制限については、「Apache Spark プラグインの制限事項」を参照してください。EMRFS S3 プラグインの制限については、「EMRFS S3 プラグインの制限事項」を参照してください。

Apache Ranger 用に Amazon EMR を設定する

Apache Ranger をインストールする前に、このセクションの情報を参照して、Amazon EMR が正しく設定されていることを確認します。

トピック

- Ranger 管理サーバーを設定する
- Apache Ranger とのネイティブ統合のための IAM ロール

- EMR セキュリティ設定を作成する
- AWS Secrets Managerに TLS 証明書を保存する
- EMR クラスターを開始する
- Apache Ranger 対応の Amazon EMR クラスター用に Zeppelin を設定する
- 既知の問題

Ranger 管理サーバーを設定する

Amazon EMR 統合の場合、Apache Ranger アプリケーションプラグインは TLS/SSL を使用して管理サーバーと通信する必要があります。

前提条件: Ranger 管理サーバーの SSL 有効化

Amazon EMR での Apache Ranger では、プラグインと Ranger 管理サーバー間の双方向の SSL 通信が必要です。プラグインが SSL 経由で Apache Ranger サーバーと通信するようにするには、Ranger 管理サーバーの ranger-admin-site.xml 内で次の属性を有効にします。

```
<property>
    <name>ranger.service.https.attrib.ssl.enabled</name>
    <value>true</value>
</property>
```

さらに、以下の設定が必要です。

```
<property>
    <name>ranger.https.attrib.keystore.file</name>
    <value>_<PATH_TO_KEYSTORE>_</value>
</property>

<property>
    <name>ranger.service.https.attrib.keystore.file</name>
    <value>_<PATH_TO_KEYSTORE>_</value>
</property>

<property>
    <name>ranger.service.https.attrib.keystore.pass</name>
    <value>_<KEYSTORE_PASSWORD>_</value>
</property>

<property>
```

```
    <name>ranger.service.https.attrib.keystore.keyalias</name>
    <value><PRIVATE_CERTIFICATE_KEY_ALIAS></value>
</property>

<property>
    <name>ranger.service.https.attrib.clientAuth</name>
    <value>want</value>
</property>

<property>
    <name>ranger.service.https.port</name>
    <value>6182</value>
</property>
```

TLS 証明書

Amazon EMR と Apache Ranger を統合するには、Amazon EMR ノードから Ranger 管理サーバーへのトラフィックが TLS を使用して暗号化され、Ranger プラグインが双方向相互 TLS 認証を使用して Apache Ranger サーバーに対して認証する必要があります。Amazon EMR サービスには、Ranger 管理サーバーのパブリック証明書 (前の例で指定) とプライベート証明書が必要です。

Apache Ranger プラグイン証明書

Apache Ranger プラグインパブリック TLS 証明書は、プラグインの接続時に検証するために Apache Ranger 管理サーバーからアクセスできる必要があります。これを行うには、3 つの方法があります。

方法 1: Apache Ranger 管理サーバーでトラストストアを設定する

信頼ストアを設定するには、 ranger-admin-site.xml に次の設定を入力します。

```
<property>
    <name>ranger.truststore.file</name>
    <value><LOCATION TO TRUSTSTORE></value>
</property>

<property>
    <name>ranger.truststore.password</name>
    <value><PASSWORD FOR TRUSTSTORE></value>
</property>
```

方法 2: Java cacerts トラストストアに証明書をロードする

Ranger 管理サーバーが JVM オプションでトラストストアを指定していない場合は、プラグインのパブリック証明書をデフォルトの cacerts ストアに配置できます。

方法 3: トラストストアを作成し、JVM オプションの一部として指定する

{RANGER_HOME_DIRECTORY}/ews/ranger-admin-services.sh 内で JAVA_OPTS を変更して "-Djavax.net.ssl.trustStore=*<TRUSTSTORE_LOCATION>*" および "-Djavax.net.ssl.trustStorePassword=*<TRUSTSTORE_PASSWORD>*" を含めます。例えば、既存の JAVA_OPTS の後に次の行を追加します。

```
JAVA_OPTS=" ${JAVA_OPTS} -Djavax.net.ssl.trustStore=${RANGER_HOME}/truststore/
truststore.jck -Djavax.net.ssl.trustStorePassword=changeit"
```

> **Note**
>
> この指定では、ユーザーが Apache Ranger 管理サーバーにログインし、ps コマンドの使用など、実行中のプロセスを確認できる場合、トラストストアのパスワードが公開されることがあります。

自己署名証明書の使用

自己署名証明書は、証明書としては推奨されません。自己署名証明書は失効できず、自己署名証明書は内部セキュリティ要件に準拠していない可能性があります。

サービス定義のインストール

サービス定義は、アプリケーションのポリシーの属性を記述するために Ranger 管理サーバーによって使用されます。ポリシーは、クライアントがダウンロードできるようにポリシーリポジトリに保存されます。

サービス定義を設定するには、Ranger 管理サーバーに対して REST 呼び出しを行う必要があります。次のセクションで必要な API について、「Apache Ranger PublicAPIsv2」を参照してください。

Apache Spark のサービス定義のインストール

Apache Spark のサービス定義をインストールするには、「Apache Spark プラグイン」を参照してください。

EMRFS サービス定義のインストール

Amazon EMR の S3 サービス定義をインストールするには、「EMRFS S3 プラグイン」を参照してください。

Hive サービス定義を使用する

Apache Hive は Apache Ranger 2.0 以降に付属している既存の Ranger サービス定義を使用できます。詳細については、「Apache Hive プラグイン」を参照してください。

ネットワークトラフィックルール

Apache Ranger が EMR クラスターに統合されている場合、クラスターは追加のサーバーおよびAWSと通信する必要があります。

コアノードとタスクノードを含むすべての Amazon EMR ノードは、ポリシーをダウンロードするために Apache Ranger 管理サーバーと通信できる必要があります。Apache Ranger 管理が Amazon EC2 で実行されている場合、EMR クラスターからのトラフィックを受信できるようにセキュリティグループを更新する必要があります。

Ranger 管理サーバーとの通信に加えて、すべてのノードが次の AWS サービスと通信できる必要があります。

- Amazon S3
- AWS KMS (EMRFS SSE-KMS を使用している場合)
- Amazon CloudWatch
- AWS STS

プライベートサブネット内で EMR クラスターを実行する予定の場合は、「Amazon VPC ユーザーガイド」の「AWS PrivateLink と VPC のエンドポイント」を使用するか、「Amazon VPC ユーザーガイド」の「NAT インスタンス」を使用して、これらのサービスと通信できるように VPC を設定します。

Apache Ranger とのネイティブ統合のための IAM ロール

Amazon EMR と Apache Ranger の統合は、クラスターを起動する前に作成する必要がある次の 3 つの主要なロールに依存します。

- Amazon EMR のカスタム Amazon EC2 インスタンスプロファイル
- Apache Ranger エンジンの IAM ロール
- 他の AWS サービスの IAM ロール

このセクションでは、これらのロールの概要と、各 IAM ロールに含める必要があるポリシーについて説明します。これらのロールの作成については、「Ranger 管理サーバーを設定する」を参照してください。

EC2 インスタンスプロファイル

Amazon EMR は IAM サービスロールを使用して、ユーザーの代わりにクラスターのプロビジョニングと管理を行うためのアクションを実行します。EC2 インスタンスのサービスロール (Amazon EMR の EC2 インスタンスプロファイルとも呼ばれます) は、起動時にクラスター内のすべての EC2 インスタンスに割り当てられる特殊なサービスロールです。

EMR クラスターと Amazon S3 データおよび Apache Ranger およびその他の AWS サービスで保護された Hive メタストアとのインタラクションのアクセス許可を定義するには、クラスターの起動`EMR_EC2_DefaultRole`時に の代わりに使用するカスタム EC2 インスタンスプロファイルを定義します。

詳細については、「クラスター EC2 インスタンスのサービスロール (EC2 インスタンスプロファイル)」および「IAM ロールのカスタマイズ」を参照してください。

Amazon EMR がセッションにタグ付けし、TLS 証明書 AWS Secrets Manager を保存する にアクセスするには、デフォルトの EC2 インスタンスプロファイルに次のステートメントを追加する必要があります。

```
    {
      "Sid": "AllowAssumeOfRolesAndTagging",
      "Effect": "Allow",
      "Action": ["sts:TagSession", "sts:AssumeRole"],
      "Resource": [
        "arn:aws:iam::<AWS_ACCOUNT_ID>:role/<RANGER_ENGINE-
PLUGIN_DATA_ACCESS_ROLE_NAME>",
        "arn:aws:iam::<AWS_ACCOUNT_ID>:role/<RANGER_USER_ACCESS_ROLE_NAME>"
      ]
    },
    {
        "Sid": "AllowSecretsRetrieval",
        "Effect": "Allow",
        "Action": "secretsmanager:GetSecretValue",
        "Resource": [

 "arn:aws:secretsmanager:<REGION>:<AWS_ACCOUNT_ID>:secret:<PLUGIN_TLS_SECRET_NAME>*",

 "arn:aws:secretsmanager:<REGION>:<AWS_ACCOUNT_ID>:secret:<ADMIN_RANGER_SERVER_TLS_SECRET_NAME>
```

```
        ]
    }
```

> **Note**
>
> Secrets Manager アクセス許可については、シークレット名の末尾にあるワイルドカード (「*」) を忘れないでください。そうしないと、リクエストは失敗します。ワイルドカードは、シークレットバージョン用です。

> **Note**
>
> AWS Secrets Manager ポリシーの範囲を、プロビジョニングに必要な証明書のみに制限します。

Apache Ranger の IAM ロール

このロールは、Apache Hive や Amazon EMR Record Server などの信頼できる実行エンジンが Amazon S3 データにアクセスするための認証情報を提供します。S3 SSE-KMS を使用している場合は、このロールだけを使用して、KMS キーを含む Amazon S3 データにアクセスします。

このロールは、次の例に示す最小ポリシーで作成する必要があります。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CloudwatchLogsPermissions",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Effect": "Allow",
      "Resource": [

 "arn:aws:logs:<REGION>:<AWS_ACCOUNT_ID>:<CLOUDWATCH_LOG_GROUP_NAME_IN_SECURITY_CONFIGURATION>:
      ]
    },
```

```
    {
      "Sid": "BucketPermissionsInS3Buckets",
      "Action": [
        "s3:CreateBucket",
        "s3:DeleteBucket",
        "s3:ListAllMyBuckets",
        "s3:ListBucket"
      ],
      "Effect": "Allow",
      "Resource": [
 *"arn:aws:s3:::bucket1",
        "arn:aws:s3:::bucket2"*
        ]
    },
    {
      "Sid": "ObjectPermissionsInS3Objects",
      "Action": [
        "s3:GetObject",
        "s3:DeleteObject",
        "s3:PutObject"
      ],
      "Effect": "Allow",
      "Resource": [
 *"arn:aws:s3:::bucket1/*",
          "arn:aws:s3:::bucket2/*"
*        ]
    }
  ]
}
```

Important

CloudWatch ログストリームに書き込むアクセス許可を付与するには、ログリソースの末尾にアスタリスク「*」を含める必要があります。

Note

EMRFS 整合性ビューまたは S3-SSE 暗号化を使用している場合は、実行エンジンがこれらのエンジンと対話できるように、DynamoDB テーブルおよび KMS キーへのアクセス許可を追加します。

Apache Ranger の IAM ロールは EC2 インスタンスプロファイルロールによって引き受けられます。次の例を使用して、Apache Ranger の IAM ロールが EC2 インスタンスプロファイルロールによって引き受けられるようにする信頼ポリシーを作成します。

```
   {
     "Sid": "",
     "Effect": "Allow",
     "Principal": {
       "AWS": "arn:aws:iam::<AWS_ACCOUNT_ID>:role/<EC2 INSTANCE PROFILE ROLE NAME eg.
EMR_EC2_DefaultRole>"
     },
     "Action": ["sts:AssumeRole", "sts:TagSession"]
   }
```

他の AWS のサービスの IAM ロール

このロールは、信頼された実行エンジンではないユーザーに、必要に応じて AWS サービスとやり取りするための認証情報を提供します。すべてのユーザーからアクセスできるようにするデータでない限り、この IAM ロールを使用して Amazon S3 データへのアクセスを許可しないでください。

このロールは EC2 インスタンスプロファイルロールによって引き受けられます。次の例を使用して、Apache Ranger の IAM ロールが EC2 インスタンスプロファイルロールによって引き受けられるようにする信頼ポリシーを作成します。

```
   {
     "Sid": "",
     "Effect": "Allow",
     "Principal": {
       "AWS": "arn:aws:iam::<AWS_ACCOUNT_ID>:role/<EC2 INSTANCE PROFILE ROLE NAME eg.
EMR_EC2_DefaultRole>"
     },
     "Action": ["sts:AssumeRole", "sts:TagSession"]
   }
```

アクセス許可を検証する

アクセス許可の検証の手順については、「Apache Ranger のトラブルシューティング」を参照してください。

EMR セキュリティ設定を作成する

Apache Ranger 用の Amazon EMR セキュリティ設定を作成する

Apache Ranger と統合された Amazon EMR クラスターを起動する前に、セキュリティ設定を作成します。

Console

AWS Ranger 統合オプションを指定するセキュリティ設定を作成するには

1. Amazon EMR コンソールで [Security configurations] (セキュリティ設定) > [Create] (作成) を選択します。
2. セキュリティ設定の [Name (名前)] を入力します。この名前を使用して、クラスターの作成時に使用するセキュリティ設定を指定します。
3. [AWS Ranger 統合] で [Apache Ranger が管理するきめ細かいアクセスコントロールを有効にする] を選択します。
4. 適用する [IAM role for Apache Ranger] (Apache Ranger 用の IAM ロール) を選択します。詳細については、「Apache Ranger とのネイティブ統合のための IAM ロール」を参照してください。
5. 適用する [IAM role for other services] (他の AWS のサービス用 IAM ロール) を選択します。
6. 管理サーバーのシークレットマネージャー ARN とアドレスを入力して、Ranger 管理サーバーに接続するようにプラグインを設定します。
7. Ranger プラグインを設定するアプリケーションを選択します。プラグインのプライベート TLS 証明書を含むシークレットマネージャー ARN を入力します。

   Apache Spark または Apache Hive が設定されていないが、クラスターのアプリケーションとして選択されている場合、リクエストは失敗します。
8. 必要に応じて他のセキュリティ設定オプションを設定し、[Create (作成)] を選択します。クラスター専用 KDC または外部 KDC を使用して Kerberos 認証を有効にする必要があります。

> **Note**
>
> 現在、 コンソールを使用して、 で AWS Ranger 統合オプションを指定するセキュリティ設定を作成することはできません AWS GovCloud (US) Region。セキュリティ設定は CLI を使用して実行できます。

CLI

Apache Ranger 統合のセキュリティ設定を作成するには

1. を AWS アカウント ID *<ACCOUNT ID>*に置き換えます。
2. *<REGION>* を、リソースが置かれているリージョンに置き換えます。
3. TicketLifetimeInHours の値を指定して、KDC によって発行された Kerberos チケットが有効である期間を決定します。
4. AdminServerURL に Ranger 管理サーバーのアドレスを指定します。

```
{
    "AuthenticationConfiguration": {
        "KerberosConfiguration": {
            "Provider": "ClusterDedicatedKdc",
            "ClusterDedicatedKdcConfiguration": {
                "TicketLifetimeInHours": 24
            }
        }
    },
    "AuthorizationConfiguration":{
      "RangerConfiguration":{
         "AdminServerURL":"https://_<RANGER ADMIN SERVER IP>_:6182",
         "RoleForRangerPluginsARN":"arn:aws:iam::_<ACCOUNT ID>_:role/_<RANGER PLUGIN
 DATA ACCESS ROLE NAME>_",
         "RoleForOtherAWSServicesARN":"arn:aws:iam::_<ACCOUNT ID>_:role/_<USER
 ACCESS ROLE NAME>_",
         "AdminServerSecretARN":"arn:aws:secretsmanager:_<REGION>_:_<ACCOUNT
 ID>_:secret:_<SECRET NAME THAT PROVIDES ADMIN SERVERS PUBLIC TLS CERTIFICATE
 WITHOUT VERSION>_",
         "RangerPluginConfigurations":[
            {
               "App":"Spark",
               "ClientSecretARN":"arn:aws:secretsmanager:_<REGION>_:_<ACCOUNT
 ID>_:secret:_<SECRET NAME THAT PROVIDES SPARK PLUGIN PRIVATE TLS CERTIFICATE
 WITHOUT VERSION>_",
               "PolicyRepositoryName":"<SPARK SERVICE NAME eg. amazon-emr-spark>"
            },
            {
               "App":"Hive",
```

```
                "ClientSecretARN":"arn:aws:secretsmanager:_<REGION>_:_<ACCOUNT
 ID>_:secret:_<SECRET NAME THAT PROVIDES Hive PLUGIN PRIVATE TLS CERTIFICATE WITHOUT
 VERSION>_",
                "PolicyRepositoryName":"<HIVE SERVICE NAME eg. Hivedev>"
            },
            {
                "App":"EMRFS-S3",
                "ClientSecretARN":"arn:aws:secretsmanager:_<REGION>_:_<ACCOUNT
 ID>_:secret:_<SECRET NAME THAT PROVIDES EMRFS S3 PLUGIN PRIVATE TLS CERTIFICATE
 WITHOUT VERSION>_",
                "PolicyRepositoryName":"<EMRFS S3 SERVICE NAME eg amazon-emr-emrfs>"
            },
       {
                "App":"Trino",
                "ClientSecretARN":"arn:aws:secretsmanager:_<REGION>_:_<ACCOUNT
 ID>_:secret:_<SECRET NAME THAT PROVIDES TRINO PLUGIN PRIVATE TLS CERTIFICATE
 WITHOUT VERSION>_",
                "PolicyRepositoryName":"<TRINO SERVICE NAME eg amazon-emr-trino>"
            }
         ],
         "AuditConfiguration":{
            "Destinations":{
               "AmazonCloudWatchLogs":{
                  "CloudWatchLogGroup":"arn:aws:logs:<REGION>:_<ACCOUNT ID>_:log-
group:_<LOG GROUP NAME FOR AUDIT EVENTS>_"
               }
            }
         }
      }
   }
}
```

PolicyRespositoryNames は、Apache Ranger 管理者で指定されているサービス名です。

次のコマンドを使用して、Amazon EMR セキュリティ設定を作成します。security-configuration は、任意の名前に置き換えます。クラスターの作成時に、この設定を名前で選択します。

```
aws emr create-security-configuration \
--security-configuration file://./security-configuration.json \
--name security-configuration
```

追加のセキュリティ機能を設定する

Amazon EMR を Apache Ranger に安全に統合するには、次の EMR セキュリティ機能も設定する必要があります。

- クラスター専用 KDC または外部 KDC を使用して Kerberos 認証を有効にします。手順については、「Amazon EMR での認証に Kerberos を使用する」を参照してください。
- (オプション) 転送中または保管中の暗号化を有効にします。詳細については、「暗号化オプション」を参照してください。

詳細については、「Amazon EMR でのセキュリティ」を参照してください。

AWS Secrets Managerに TLS 証明書を保存する

Amazon EMR クラスターと Ranger 管理サーバーにインストールされた Ranger プラグインは、送信されたポリシーデータやその他の情報が傍受された場合に読み取れないように、TLS を介して通信する必要があります。また、EMR では、プラグインが独自の TLS 証明書を提供して Ranger 管理サーバーに対して認証し、双方向 TLS 認証を実行することを義務付けています。この設定では、4 つの証明書 (プライベート TLS 証明書とパブリック TLS 証明書の 2 つのペア) を作成する必要がありました。Ranger 管理サーバーに証明書をインストールする手順については、「Ranger 管理サーバーを設定する」を参照してください。設定を完了するには、EMR クラスターにインストールされた Ranger プラグインに、管理者サーバーのパブリック TLS 証明書と、プラグインが Ranger 管理サーバーに対する認証に使用するプライベート証明書という 2 つの証明書が必要です。これらの TLS 証明書を提供するには、 にあり AWS Secrets Manager 、EMR セキュリティ設定で提供されている必要があります。

> **Note**
>
> プラグイン証明書のいずれかが侵害された場合の影響を制限するために、アプリケーションごとに証明書ペアを作成することを強く推奨しますが、必須ではありません。

> **Note**
>
> 証明書を追跡し、有効期限が切れる前にローテーションする必要があります。

証明書の形式

証明書の へのインポート AWS Secrets Manager は、プライベートプラグイン証明書かパブリック Ranger 管理者証明書かに関係なく同じです。TLS 証明書をインポートする前に、証明書は 509x PEM 形式である必要があります。

パブリック証明書の例は、次の形式です。

```
-----BEGIN CERTIFICATE-----
...Certificate Body...
-----END CERTIFICATE-----
```

プライベート証明書の例は、次の形式です。

```
-----BEGIN PRIVATE KEY-----
...Private Certificate Body...
-----END PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
...Trust Certificate Body...
-----END CERTIFICATE-----
```

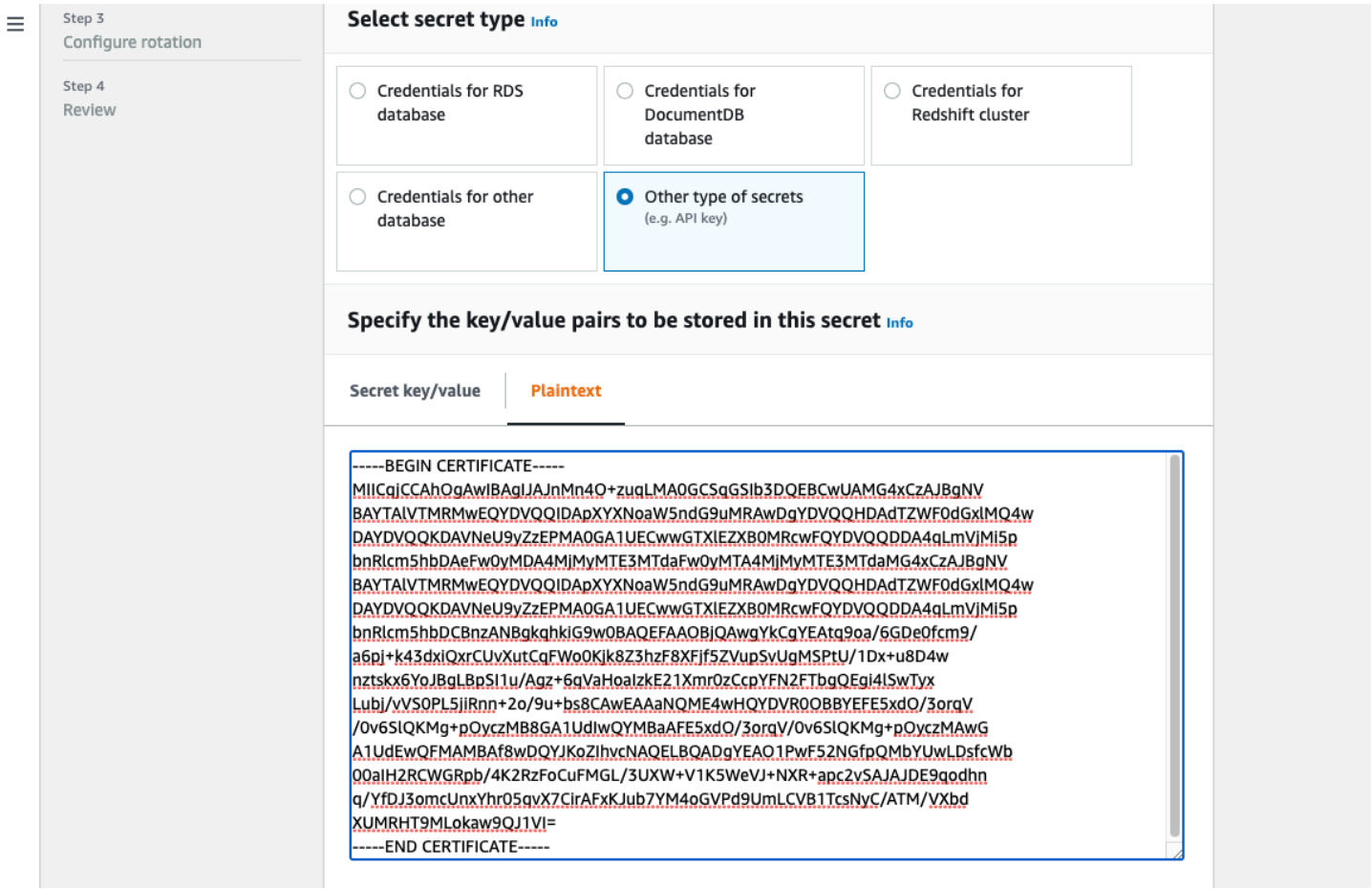
プライベート証明書には、信頼証明書も含まれている必要があります。

次のコマンドを実行して、証明書が正しい形式であることを検証できます。

```
openssl x509 -in <PEM FILE> -text
```

AWS Secrets Managerへの証明書のインポート

シークレットマネージャーでシークレットを作成する場合は、[シークレットのタイプ] で [その他のシークレット] を選択し、[プレーンテキスト] フィールドに PEM エンコードの証明書を貼り付けます。



EMR クラスターを開始する

Apache Ranger を使用する Amazon EMR クラスターを起動する前に、各コンポーネントが次の最小バージョン要件を満たしていることを確認してください。

- Amazon EMR 5.32.0 以降、または 6.3.0 以降 最新の Amazon EMR リリースバージョンを使用することが推奨されます。
- Apache Ranger 管理サーバー 2.x

以下の手順を実行します。

- Apache Ranger をまだインストールしていない場合は、インストールします。詳細については、「Apache Ranger 0.5.0 installation」を参照してください。
- Amazon EMR クラスターと Apache Ranger 管理サーバーの間にネットワーク接続があることを確認します。「Ranger 管理サーバーを設定する」を参照してください。

- 必要な IAM ロールを作成します。Apache Ranger とのネイティブ統合のための IAM ロール を参照してください。
- Apache Ranger インストール環境用の EMR セキュリティ設定を作成します。詳細については、「EMR セキュリティ設定を作成する」を参照してください。

Apache Ranger 対応の Amazon EMR クラスター用に Zeppelin を設定する

このトピックでは、Apache Ranger 対応の Amazon EMR クラスター用に Apache Zeppelin を設定して、インタラクティブなデータ探索用のノートブックとして Zeppelin を使用できるようにする方法について説明します。Zeppelin は、Amazon EMR リリースバージョン 5.0.0 以降に含まれています。以前のリリースバージョンには、サンドボックスアプリケーションとして Zeppelin が含まれています。詳細については、「Amazon EMR リリースガイド」の「Amazon EMR 4.x リリースバージョン」を参照してください。

デフォルトでは、Zeppelin はデフォルトのログインとパスワードで設定されており、これはマルチテナント環境では安全ではありません。

Zeppelin を設定するには、以下のステップを実行します。

1. 認証メカニズムを変更する

   `shiro.ini` ファイルを変更して、優先認証メカニズムを実装します。Zeppelin は Active Directory、LDAP、PAM、Knox SSO をサポートしています。詳細については、「Apache Shiro authentication for Apache Zeppelin」を参照してください。
2. エンドユーザーを偽装するように Zeppelin を設定する

   Zeppelin がエンドユーザーを偽装することを許可すると、Zeppelin から送信されたジョブを、そのエンドユーザーとして実行できます。以下の設定を `core-site.xml` に追加します。

```
[
  {
    "Classification": "core-site",
    "Properties": {
      "hadoop.proxyuser.zeppelin.hosts": "*",
      "hadoop.proxyuser.zeppelin.groups": "*"
    },
    "Configurations": [
    ]
  }
```

```
]
```

次に、/etc/hadoop/conf にある hadoop-kms-site.xml に次の設定を追加します。

```
[
  {
    "Classification": "hadoop-kms-site",
    "Properties": {
      "hadoop.kms.proxyuser.zeppelin.hosts": "*",
      "hadoop.kms.proxyuser.zeppelin.groups": "*"
    },
    "Configurations": [
    ]
  }
]
```

「コンソールでのインスタンスグループの再設定」の手順に従って、コンソールを使用して Amazon EMR クラスターにこれらの設定を追加することもできます。

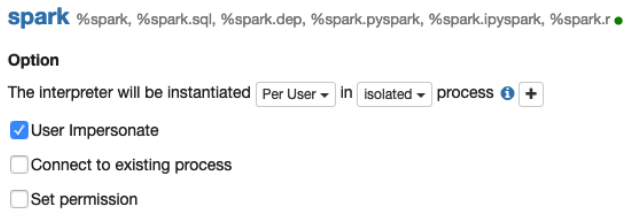
3. Zeppelin にエンドユーザーとして sudo を許可する

   以下が含まれているファイル /etc/sudoers.d/90-zeppelin-user を作成します。

```
zeppelin ALL=(ALL) NOPASSWD:ALL
```

4. インタプリタの設定を変更して、自分のプロセスでユーザージョブを実行する。

   すべてのインタープリタについて、「分離された」プロセスでインタープリタを「ユーザーごとに」インスタンス化するように設定します。



5. **zeppelin-env.sh を変更する**

   以下を zeppelin-env.sh に追加し、Zeppelin がエンドユーザーとして起動インタプリタを開始するようにします。

```
ZEPPELIN_IMPERSONATE_USER=`echo ${ZEPPELIN_IMPERSONATE_USER} | cut -d @ -f1`
```

```
export ZEPPELIN_IMPERSONATE_CMD='sudo -H -u ${ZEPPELIN_IMPERSONATE_USER} bash -c'
```

以下を zeppelin-env.sh に追加し、デフォルトのノートブックアクセス許可を作成者のみの読み取り専用に変更します。

```
export ZEPPELIN_NOTEBOOK_PUBLIC="false"
```

最後に、以下を に追加zeppelin-env.shして、最初のCLASSPATHステートメントの後に EMR RecordServer クラスパスを含めます。

```
export CLASSPATH="$CLASSPATH:/usr/share/aws/emr/record-server/lib/aws-emr-record-server-connector-common.jar:/usr/share/aws/emr/record-server/lib/aws-emr-record-server-spark-connector.jar:/usr/share/aws/emr/record-server/lib/aws-emr-record-server-client.jar:/usr/share/aws/emr/record-server/lib/aws-emr-record-server-common.jar:/usr/share/aws/emr/record-server/lib/jars/secret-agent-interface.jar"
```

6. Zeppelin を再起動する

   以下のコマンドを実行して、Zeppelin を再起動します。

   ```
   sudo systemctl restart zeppelin
   ```

既知の問題

既知の問題

Amazon EMR リリース 5.32 には、hive-site.xml のアクセス許可が変更され、特権ユーザーのみがそれを読み取ることができる (認証情報が保存されている可能性があるため) という既知の問題があります。これにより、Hue が hive-site.xml を読み取ることができず、ウェブページが継続的にリロードされることがあります。この問題が発生した場合は、以下の設定を追加して、問題を解決します。

```
[
  {
    "Classification": "hue-ini",
    "Properties": {},
    "Configurations": [
      {
        "Classification": "desktop",
        "Properties": {
```

```
          "server_group":"hive_site_reader"
         },
        "Configurations":[
        ]
      }
    ]
  }
]
```

Apache Ranger 用の EMRFS S3 プラグインは、現在 Apache Ranger のセキュリティゾーン機能をサポートしていないという既知の問題があります。セキュリティゾーン機能を使用して定義されたアクセスコントロールの制限は、Amazon EMR クラスターには適用されません。

アプリケーション UI

デフォルトでは、アプリケーション UI は認証を実行しません。これには、ResourceManager UI、NodeManager UI、Livy UI などが含まれます。さらに、UI にアクセスできるユーザーは、他のすべてのユーザーのジョブに関する情報を表示できます。

この動作が望ましくない場合は、セキュリティグループを使用してユーザーによるアプリケーション UI へのアクセスを制限する必要があります。

HDFS のデフォルトのアクセス許可

デフォルトでは、ユーザーが HDFS で作成したオブジェクトには、誰でも読み取り可能なアクセス許可が与えられます。このため、アクセスが許可されるべきではないユーザーがデータを読み取ることができる可能性があります。デフォルトのファイルアクセス許可がジョブの作成者のみによる読み書きに設定されるようにこの動作を変更するには、次の手順を実行します。

EMR クラスターを作成するときに、次の設定を指定します。

```
[
  {
    "Classification": "hdfs-site",
    "Properties": {
      "dfs.namenode.acls.enabled": "true",
      "fs.permissions.umask-mode": "077",
      "dfs.permissions.superusergroup": "hdfsadmingroup"
    }
  }
]
```

さらに、次のブートストラップアクションを実行します。

```
--bootstrap-actions Name='HDFS UMask Setup',Path=s3://elasticmapreduce/hdfs/umask/
umask-main.sh
```

Apache Ranger プラグイン

Apache Ranger プラグインは、Apache Ranger ポリシー管理サーバーで定義された認可ポリシーに対してユーザーのアクセスを検証します。

トピック

- Apache Hive プラグイン
- Apache Spark プラグイン
- EMRFS S3 プラグイン
- Trino プラグイン

Apache Hive プラグイン

Apache Hive は Hadoop エコシステム内の普及している実行エンジンです。Amazon EMR は、Hive のきめ細かなアクセスコントロールを提供できる Apache Ranger プラグインを提供しています。このプラグインはオープンソースの Apache Ranger 管理サーバーバージョン 2.0 以降と互換性があります。

トピック

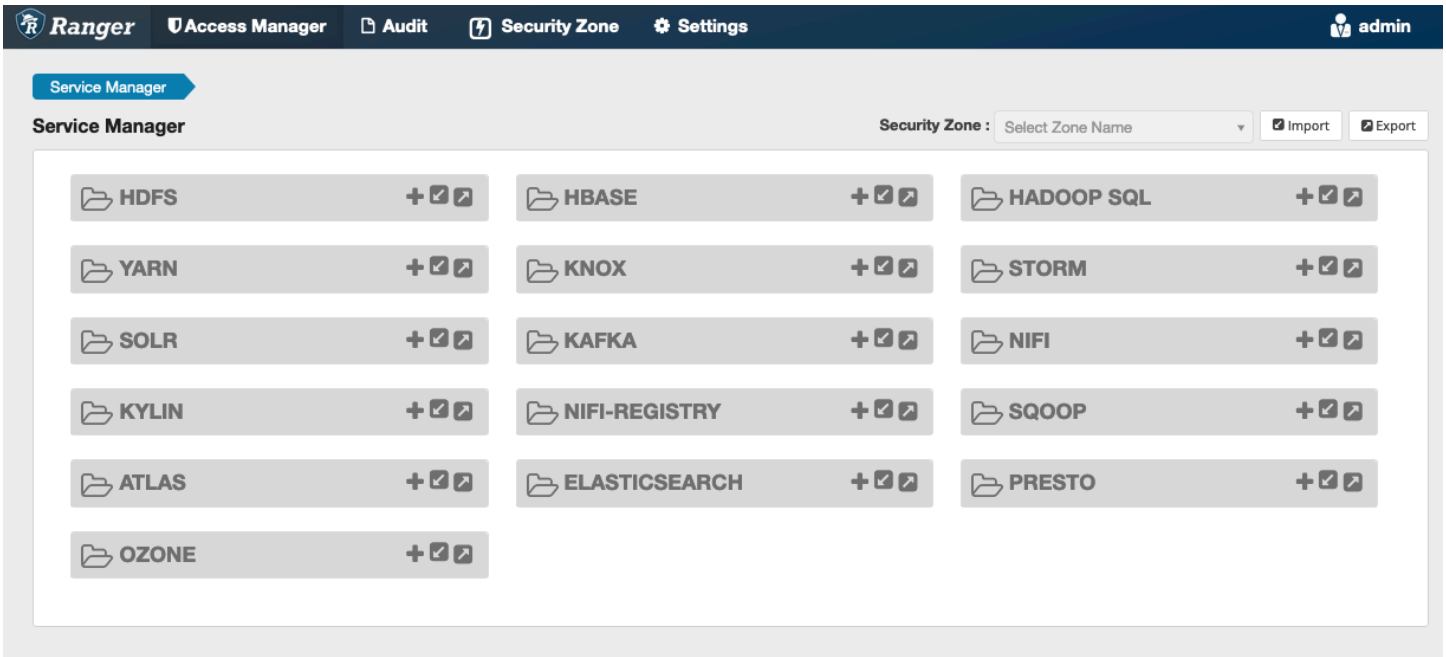
- サポートされている機能
- サービス設定のインストール
- 考慮事項
- 制限事項

サポートされている機能

EMR 上の Hive 用の Apache Ranger プラグインは、データベース、テーブル、列レベルのアクセスコントロール、行フィルタリング、データマスキングなど、オープンソースプラグインのすべての機能をサポートしています。Hive コマンドおよび関連する Ranger アクセス許可の表については、「Hive commands to Ranger permission mapping」を参照してください。

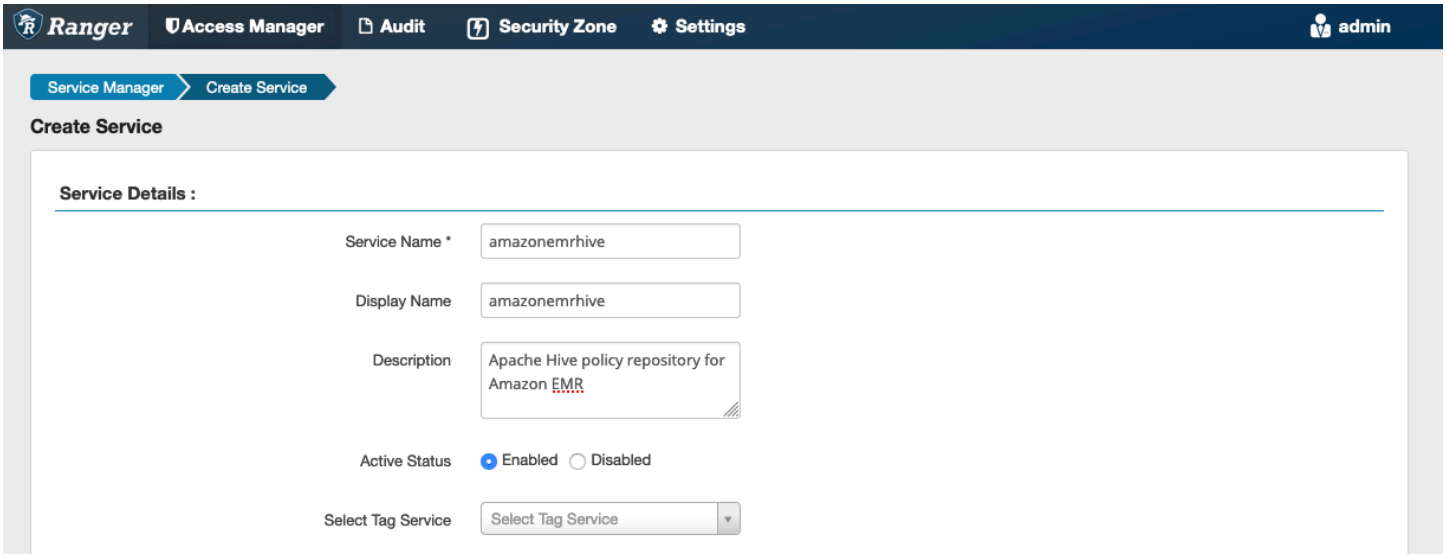
サービス設定のインストール

Apache Hive プラグインは、Apache Hive Hadoop SQL 内の既存の Hive サービス定義と互換性があります。



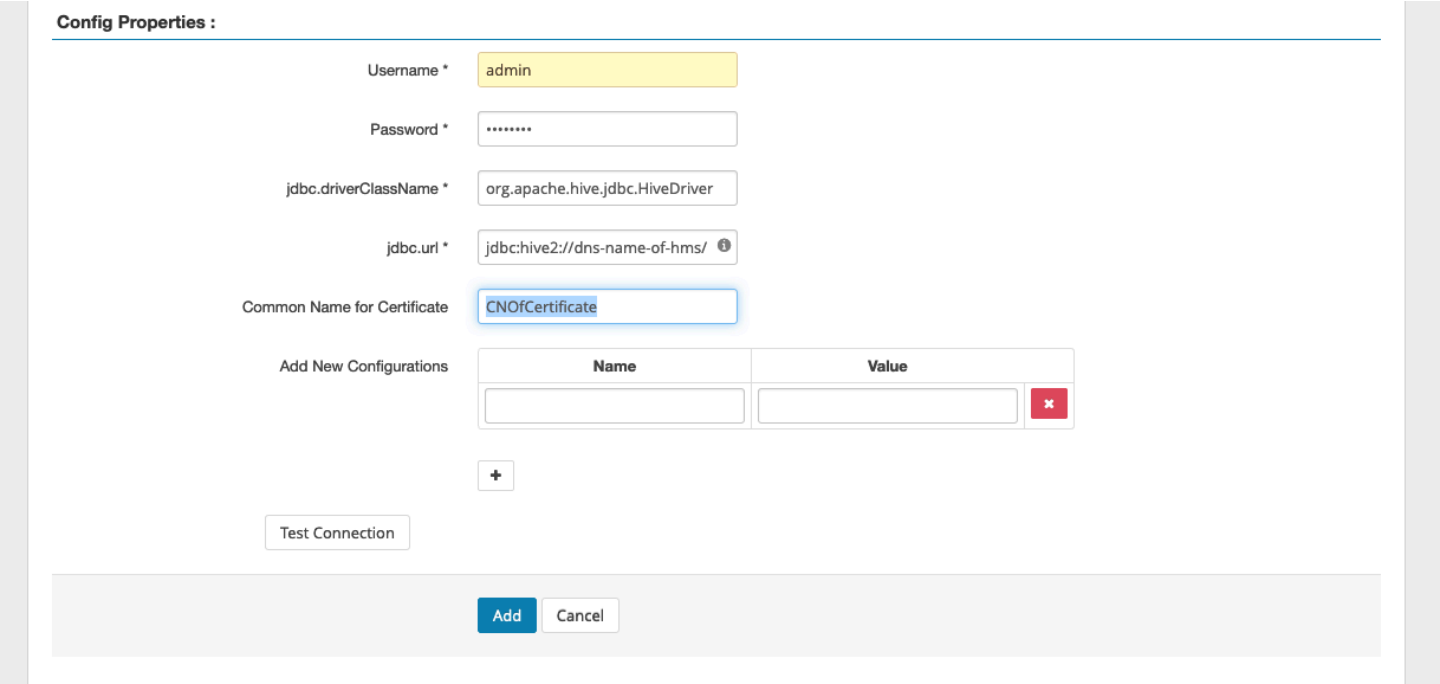
上記のように Hadoop SQL の下にサービスのインスタンスがない場合には、インスタンスを作成できます。Hadoop SQL の横の [+] をクリックします。

1. [Service Name] (サービス名): これが表示されている場合は、サービス名を入力します。推奨値は **`amazonemrhive`** です。このサービス名をメモしておいてください。EMR セキュリティ設定を作成するときに必要です。
2. [Display Name] (表示名): サービスに対して表示する名前を入力します。推奨値は **`amazonemrhive`** です。

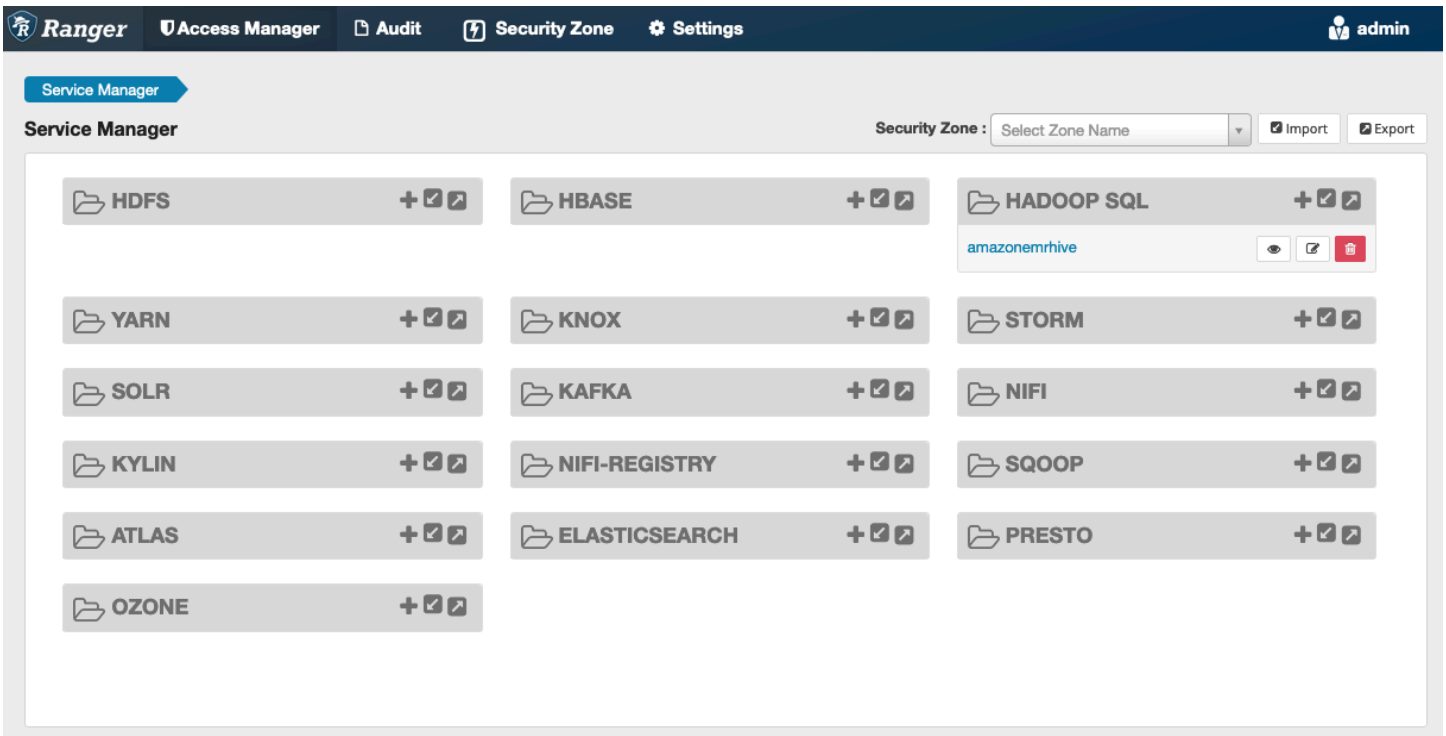


Apache Hive Config プロパティは、ポリシーの作成時に自動完了を実装する HiveServer2 を使用して Apache Ranger 管理サーバーへの接続を確立するために使用されます。永続的な HiveServer2 プロセスがなく、情報を入力できる場合、以下のプロパティは正確である必要はありません。

- ユーザー名 : HiveServer2 つのインスタンスのインスタンスへの JDBC 接続のユーザー名を入力します。
- [Password] (パスワード): 上記のユーザー名のパスワードを入力します。
- jdbc.driver.ClassName: Apache Hive 接続用の JDBC クラスのクラス名を入力します。デフォルト値を使用できます。
- jdbc.url : HiveServer2 に接続するときに使用する JDBC 接続文字列を入力します。
- [Common Name for Certificate] (証明書の共通名): クライアントプラグインから管理サーバーに接続するために使用する証明書内の CN フィールド。この値は、プラグイン用に作成された TLS 証明書の CN フィールドと一致する必要があります。



Test Connection ボタンは、上記の値を使用して HiveServer2 つのインスタンスに正常に接続できるかどうかをテストします。サービスが正常に作成されると、サービスマネージャは以下のようになります。



考慮事項

Hive メタデータサーバー

Hive メタデータサーバーには、不正アクセスから保護するために、信頼されたエンジン (具体的には Hive および `emr_record_server`) からしかアクセスできません。また、Hive メタデータサーバーには、クラスター上のすべてのノードからもアクセスできます。ポート 9083 は必須で、メインノードへのすべてのノードアクセスを提供します。

認証

デフォルトでは、Apache Hive は EMR セキュリティ設定で設定された Kerberos を使用して認証するように設定されています。 HiveServer2 は LDAP を使用してユーザーを認証するように設定することもできます。「Implementing LDAP authentication for Hive on a multi-tenant Amazon EMR cluster」を参照してください。

制限事項

Amazon EMR 5.x での Apache Hive プラグインの現在の制限事項は次のとおりです。

- Hive ロールは、現在サポートされていません。Grant、Revoke ステートメントはサポートされていません。
- Hive CLI はサポートされていません。JDBC/Beeline だけが Hive を接続するために認可されている方法です。
- `hive.server2.builtin.udf.blacklist` 設定には、安全ではないと判断される UDF を入力する必要があります。

Apache Spark プラグイン

Amazon EMR は EMR を統合し RecordServer 、SparkSQL のきめ細かなアクセスコントロールを提供しています。EMR の RecordServer は、Apache Ranger 対応クラスターのすべてのノードで実行される特権プロセスです。Spark ドライバーまたはエグゼキューターが SparkSQL ステートメントを実行すると、すべてのメタデータおよびデータリクエストは を経由します RecordServer。EMR の詳細については RecordServer、Amazon EMR コンポーネント「」ページを参照してください。

トピック

- サポートされている機能
- INSERT、ALTER、または DDL ステートメントを使用するように、サービス定義を再デプロイする

- サービス定義のインストール
- SparkSQL ポリシーの作成
- 考慮事項
- 制限事項

サポートされている機能

SQL ステートメント/Ranger アクション	STATUS	サポートされている EMR リリース
SELECT	サポート	5.32 以降
SHOW DATABASES	サポート	5.32 以降
SHOW COLUMNS	サポート	5.32 以降
SHOW TABLES	サポート	5.32 以降
SHOW TABLE PROPERTIES	サポート	5.32 以降
DESCRIBE TABLE	サポート	5.32 以降
INSERT OVERWRITE	サポート	5.34 および 6.4 以降
INSERT INTO	サポート	5.34 および 6.4 以降
ALTER TABLE	サポート	6.4 以降
CREATE TABLE	サポート	5.35 および 6.7 以降
CREATE DATABASE	サポート	5.35 および 6.7 以降

SQL ステートメント/Ranger アクション	STATUS	サポートされている EMR リリース
DROP TABLE	サポート	5.35 および 6.7 以降
DROP DATABASE	サポート	5.35 および 6.7 以降
DROP VIEW	サポート	5.35 および 6.7 以降
CREATE VIEW	サポート外	

SparkSQL を使用する場合、次の機能がサポートされています。

- Hive Metastore 内のテーブルに対するきめ細かいアクセスコントロール、およびポリシーを、データベース、テーブル、および列レベルで作成できます。
- Apache Ranger ポリシーには、ユーザーおよびグループへの付与ポリシーと拒否ポリシーを含めることができます。
- 監査イベントは CloudWatch ログに送信されます。

INSERT、ALTER、または DDL ステートメントを使用するように、サービス定義を再デプロイする

> **Note**
>
> Amazon EMR 6.4 以降、INSERT INTO、INSERT OVERWRITE、ALTER TABLE の各ステートメントを使用して Spark SQL を使用できます。Amazon EMR 6.7 以降では、Spark SQL を使用してデータベースとテーブルを作成したり削除したりできます。Apache Spark サービス定義がデプロイされた Apache Ranger サーバー上の既存のインストール環境がある場合は、次のコードを使用してサービス定義を再デプロイします。

```
# Get existing Spark service definition id calling Ranger REST API and JSON
 processor
curl --silent -f -u <admin_user_login>:<password_for_ranger_admin_user> \
-H "Accept: application/json" \
-H "Content-Type: application/json" \
```

```
-k 'https://*<RANGER SERVER ADDRESS>*:6182/service/public/v2/api/servicedef/
name/amazon-emr-spark' | jq .id

# Download the latest Service definition
wget https://s3.amazonaws.com/elasticmapreduce/ranger/service-definitions/
version-2.0/ranger-servicedef-amazon-emr-spark.json

# Update the service definition using the Ranger REST API
curl -u <admin_user_login>:<password_for_ranger_admin_user> -X PUT -d @ranger-
servicedef-amazon-emr-spark.json \
-H "Accept: application/json" \
-H "Content-Type: application/json" \
-k 'https://*<RANGER SERVER ADDRESS>*:6182/service/public/v2/api/
servicedef/<Spark service definition id from step 1>'
```

サービス定義のインストール

EMR の Apache Spark サービス定義をインストールするには、レンジャー管理サーバーを設定する必要があります。Ranger 管理サーバーを設定する を参照してください。

Apache Spark サービス定義をインストールするには、次の手順に従います。

ステップ 1: Apache Ranger 管理サーバーに SSH で接続する

例:

```
ssh ec2-user@ip-xxx-xxx-xxx-xxx.ec2.internal
```

ステップ 2: サービス定義および Apache Ranger 管理サーバープラグインをダウンロードする

一時ディレクトリに、サービス定義をダウンロードします。このサービス定義は Ranger 2.x バージョンでサポートされています。

```
mkdir /tmp/emr-spark-plugin/
cd /tmp/emr-spark-plugin/

wget https://s3.amazonaws.com/elasticmapreduce/ranger/service-definitions/version-2.0/
ranger-spark-plugin-2.x.jar
wget https://s3.amazonaws.com/elasticmapreduce/ranger/service-definitions/version-2.0/
ranger-servicedef-amazon-emr-spark.json
```

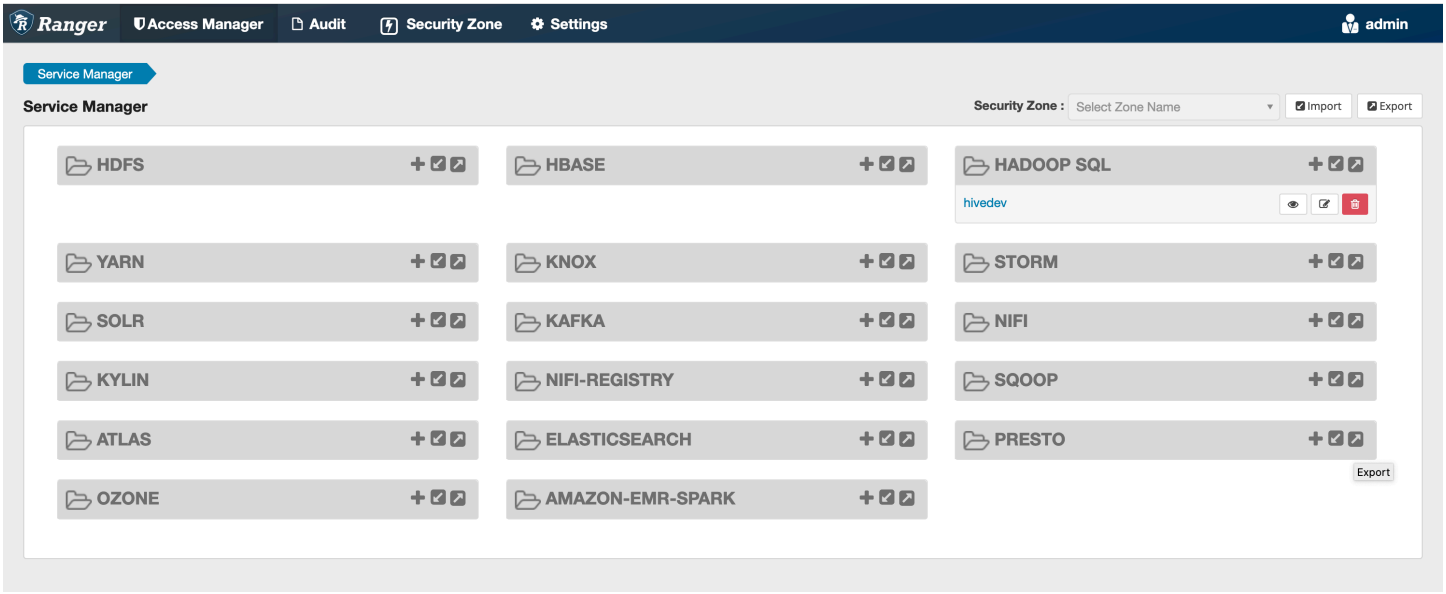
ステップ 3: Amazon EMR 用の Apache Spark プラグインをインストールする

```
export RANGER_HOME=.. # Replace this Ranger Admin's home directory eg /usr/lib/ranger/
ranger-2.0.0-admin
mkdir $RANGER_HOME/ews/webapp/WEB-INF/classes/ranger-plugins/amazon-emr-spark
mv ranger-spark-plugin-2.x.jar $RANGER_HOME/ews/webapp/WEB-INF/classes/ranger-plugins/
amazon-emr-spark
```

ステップ 4: Amazon EMR の Apache Spark サービス定義を登録する

```
curl -u *<admin users login>*:*_<_**_password_ **_for_** _ranger admin user_**_>_* -X
 POST -d @ranger-servicedef-amazon-emr-spark.json \
-H "Accept: application/json" \
-H "Content-Type: application/json" \
-k 'https://*<RANGER SERVER ADDRESS>*:6182/service/public/v2/api/servicedef'
```

このコマンドが正常に実行されると、次の図に示すように、Ranger 管理 UI に「AMAZON-EMR-SPARK」という新しいサービスが表示されます (Ranger バージョン 2.0 が示されています)。

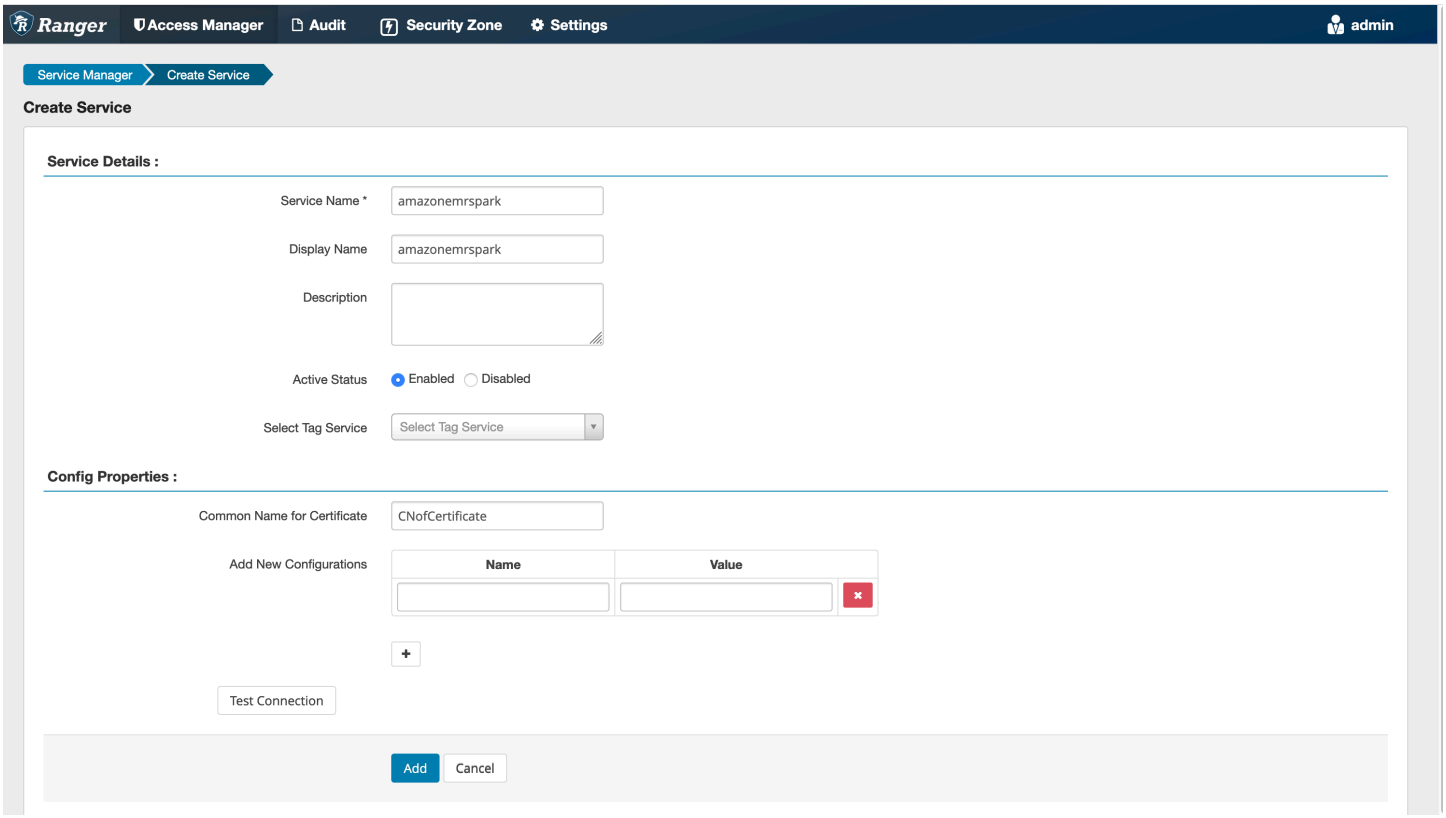


ステップ 5: AMAZON-EMR-SPARK アプリケーションのインスタンスを作成する

[Service Name] (サービス名): 使用されるサービス名 (表示されている場合)。推奨値は **amazonemrspark** です。このサービス名をメモしておいてください。EMR セキュリティ設定を作成するときに必要になります。

[Display Name] (表示名): このインスタンスに表示される名前。推奨値は **amazonemrspark** です。

[Common Name For Certificate] (証明書の共通名): クライアントプラグインから管理サーバーに接続するために使用する証明書内の CN フィールド。この値は、プラグイン用に作成された TLS 証明書の CN フィールドと一致する必要があります。



> **Note**
>
> このプラグインの TLS 証明書は、Ranger 管理サーバーのトラストストアに登録されている必要があります。詳細については、「TLS 証明書」を参照してください。

SparkSQL ポリシーの作成

新しいポリシーを作成する場合、入力するフィールドは次のとおりです。

[Policy Name] (ポリシー名): このポリシーの名前。

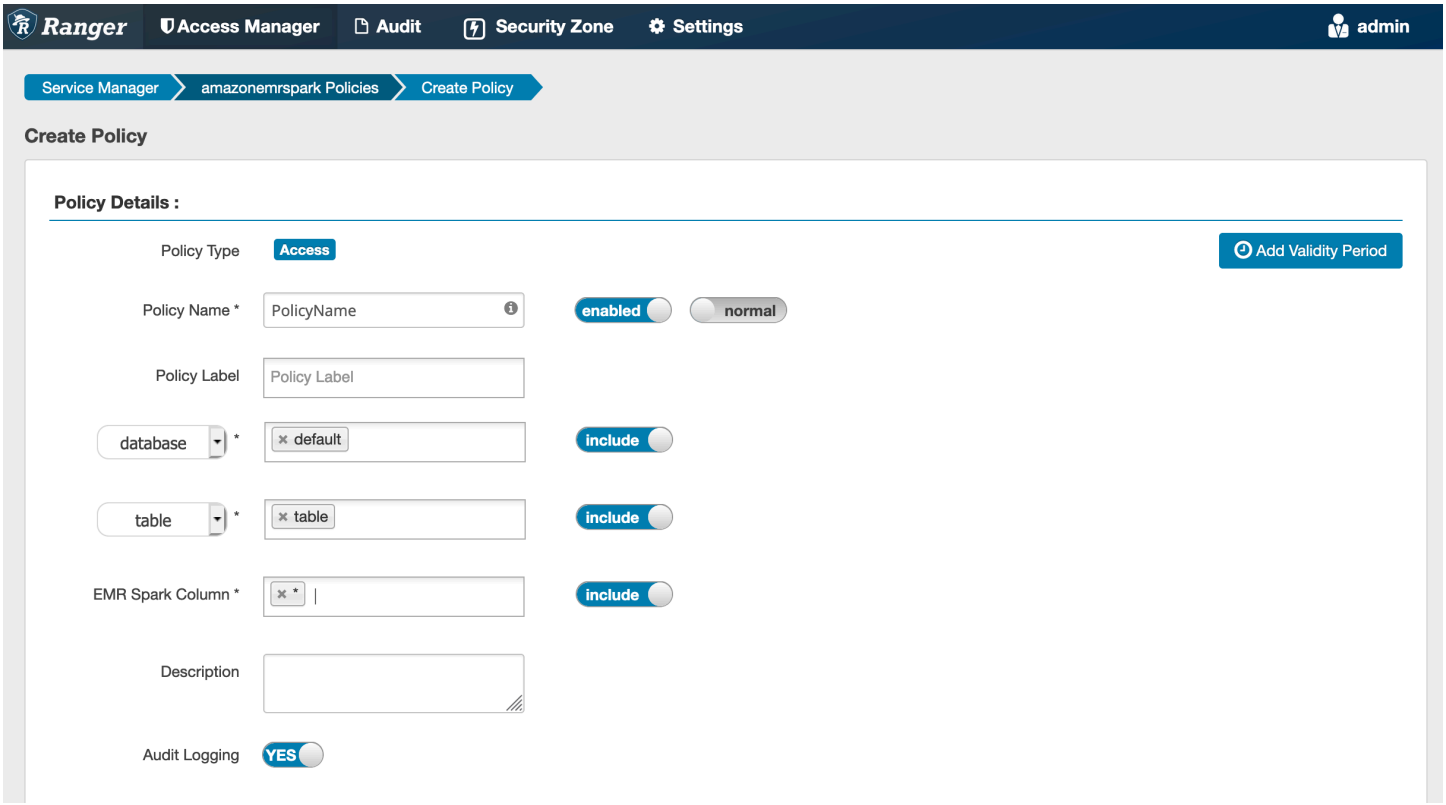
[Policy Label] (ポリシーラベル): このポリシーにつけることができるラベル。

[Database] (データベース): このポリシーが適用されるデータベース。ワイルドカード「*」はすべてのデータベースを表します。

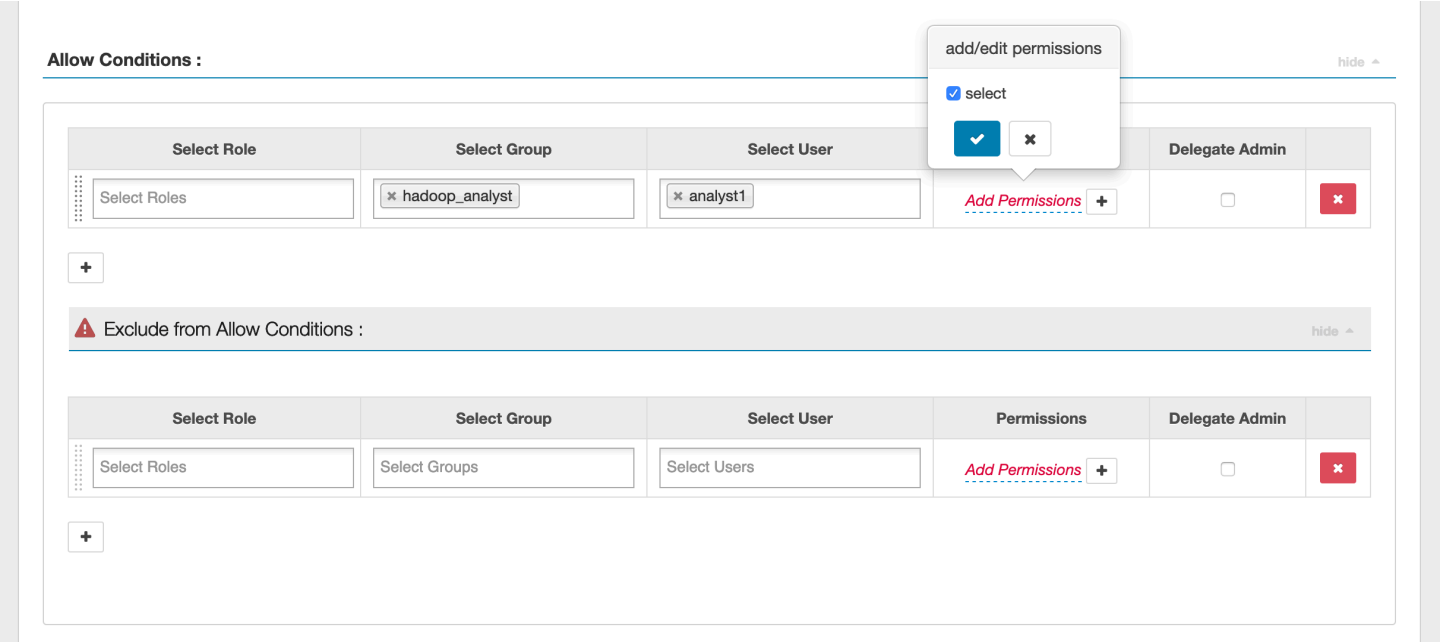
[Table] (テーブル): このポリシーが適用されるテーブル。ワイルドカード「*」はすべてのテーブルを表します。

[EMR Spark Column] (EMR Spark 列): このポリシーが適用される列。ワイルドカード「*」はすべての列を表します。

[Description] (説明): このポリシーの説明。



ユーザーとグループを指定するには、以下のユーザーおよびグループを入力してアクセス許可を付与します。[許可] 条件および [拒否] 条件の除外を指定することもできます。



許可条件と拒否条件を指定したら、[Save] (保存) をクリックします。

考慮事項

EMR クラスター内の各ノードは、ポート 9083 でメインノードに接続できる必要があります。

制限事項

Apache Spark プラグインの現在の制限事項は次のとおりです。

- レコードサーバーは常に Amazon EMR クラスターで実行されている HMS に接続します。必要に応じて、リモートモードに接続するように HMS を設定します。Apache Spark Hive-site.xml 設定ファイル内に設定値を入れないでください。
- CSV または Avro で Spark データソースを使用して作成されたテーブルは、EMR を使用して読み取れません RecordServer。Hive を使用してデータの作成および書き込みを行い、Record を使用して読み取りを行います。
- Delta Lake および Hudi テーブルはサポートされていません。
- ユーザーは、デフォルトのデータベースにアクセスできる必要があります。これは Apache Spark の要件です。
- Ranger 管理サーバーはオートコンプリートをサポートしていません。
- Amazon EMR 用 SparkSQL プラグインは、行フィルターやデータマスキングをサポートしていません。

- Spark SQL で ALTER TABLE を使用する場合、パーティションの場所はテーブルの場所の子ディレクトリである必要があります。パーティションの場所がテーブルの場所と異なるパーティションへのデータの挿入はサポートされていません。

EMRFS S3 プラグイン

マルチテナントクラスター上の S3 内のオブジェクトに対するアクセスコントロールを容易に提供できるようにするために、EMRFS S3 プラグインは、EMRFS 経由で S3 内のデータにアクセスするときに S3 内のデータへのアクセスコントロールを提供します。S3 リソースへのアクセスは、ユーザーおよびグループレベルで許可できます。

これを実現するために、アプリケーションが S3 内のデータにアクセスしようとすると、EMRFS は認証情報のリクエストをシークレットエージェントプロセスに送信し、そこでリクエストが Apache Ranger プラグインに対して認証され、認可されます。リクエストが認可されると、シークレットエージェントは制限されたポリシーを使用して Apache Ranger エンジンの IAM ロールを引き受けて、アクセスを許可した Ranger ポリシーにのみアクセスできる認証情報を生成します。その後、その認証情報が EMRFS に戻され、S3 にアクセスできるようになります。

トピック

サポートされている機能

EMRFS S3 プラグインは、ストレージレベルの認可を提供します。ポリシーを作成して、S3 バケットおよびプレフィックスに対すアクセス権をユーザーおよびグループに提供できます。認可は EMRFS に対してのみ行われます。

サービス設定のインストール

EMRFS サービス定義をインストールするには、Ranger 管理サーバーを設定する必要があります。サーバーをセットアップするには、「」を参照してくださいRanger 管理サーバーを設定する。

EMRFS サービス定義をインストールするには、次の手順に従います。

ステップ 1: Apache Ranger 管理サーバーに SSH で接続する。

例:

```
ssh ec2-user@ip-xxx-xxx-xxx-xxx.ec2.internal
```

ステップ 2: EMRFS サービス定義 をダウンロードします。

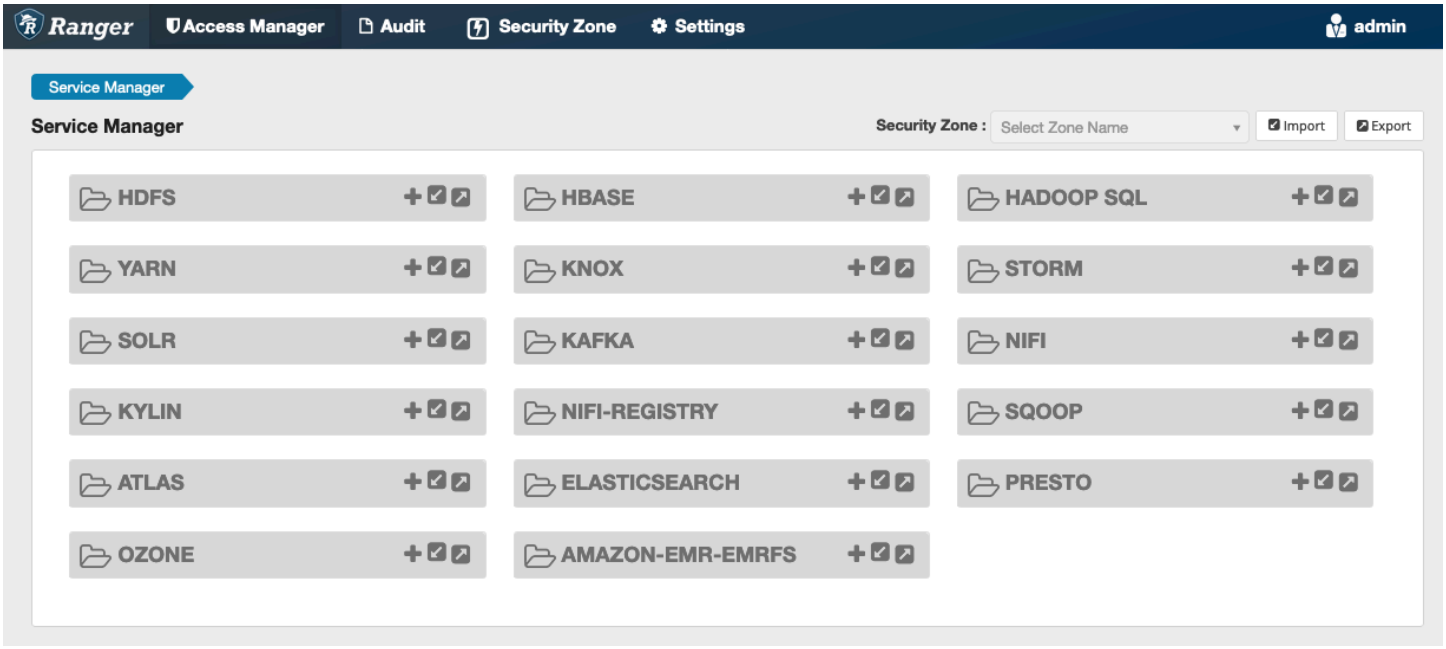
一時ディレクトリに、Amazon EMR サービス定義をダウンロードします。このサービス定義は Ranger 2.x バージョンでサポートされています。

```
wget https://s3.amazonaws.com/elasticmapreduce/ranger/service-definitions/version-2.0/
ranger-servicedef-amazon-emr-emrfs.json
```

ステップ 3: EMRFS S3 サービス定義 を登録する。

```
curl -u *<admin users login>*:*_<_**_password_ **_for_** _ranger admin user_**_>_* -X
 POST -d @ranger-servicedef-amazon-emr-emrfs.json \
-H "Accept: application/json" \
-H "Content-Type: application/json" \
-k 'https://*<RANGER SERVER ADDRESS>*:6182/service/public/v2/api/servicedef'
```

このコマンドが正常に実行されると、次の図に示すように、Ranger 管理 UI に「AMAZON-EMR-S3」という新しいサービスが表示されます (Ranger バージョン 2.0 が示されています)。



ステップ 4: AMAZON-EMR-EMRFS アプリケーションのインスタンスを作成します。

サービス定義のインスタンスを作成します。

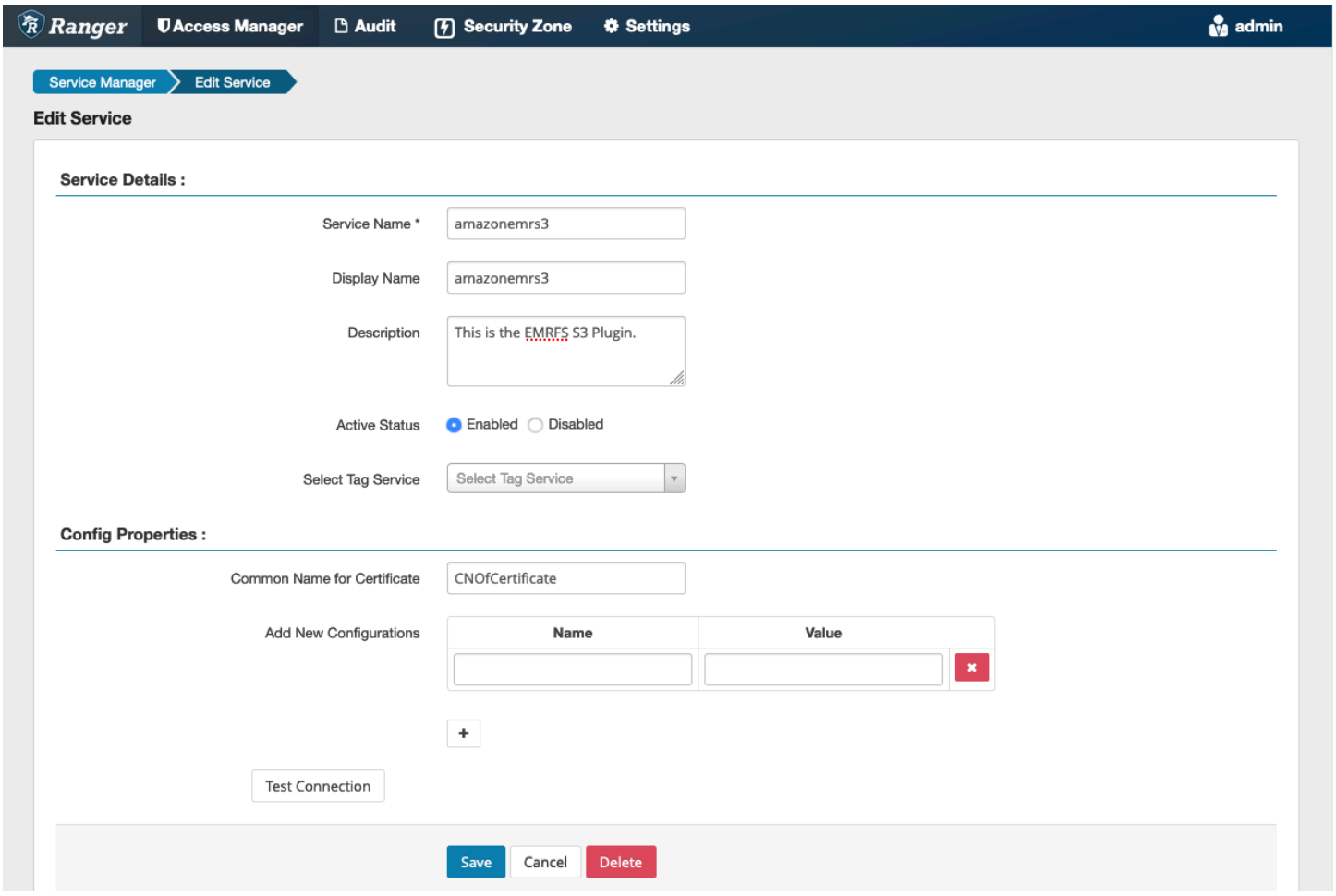
- AMAZON-EMR-EMRFS の横の [+] をクリックします。

以下のフィールドに入力します。

[Service Name] (サービス名) (表示されている場合): 推奨値は **amazonemrspark** です。このサービス名をメモしておいてください。EMR セキュリティ設定を作成するときに必要になります。

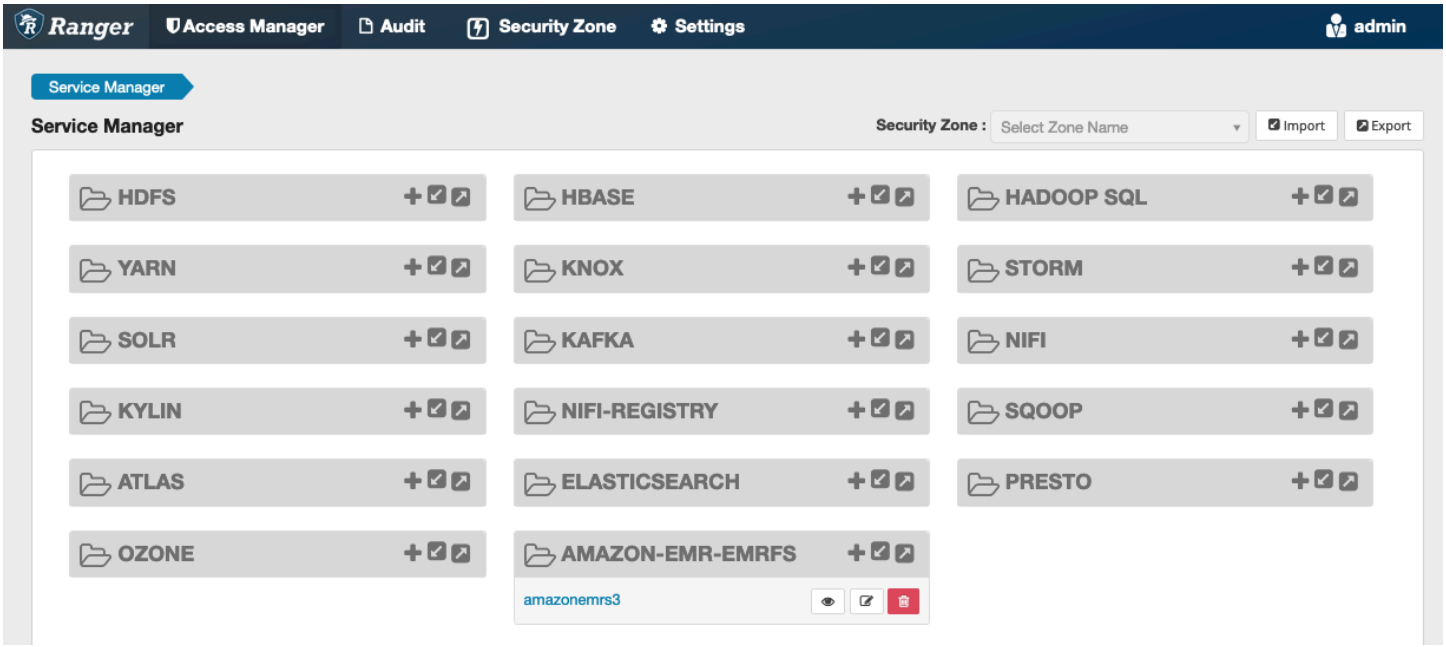
[Display Name] (表示名): このサービスに対して表示する名前。推奨値は **amazonemrspark** です。

[Common Name For Certificate] (証明書の共通名): クライアントプラグインから管理サーバーに接続するために使用する証明書内の CN フィールド。この値は、プラグイン用に作成された TLS 証明書の CN フィールドと一致する必要があります。



> **Note**
>
> このプラグインの TLS 証明書は、Ranger 管理サーバーのトラストストアに登録されている必要があります。詳細については、「TLS 証明書」を参照してください。

サービスが作成されると、次の図に示すように、サービスマネージャーに「AMAZON-EMR-EMRFS」が表示されます。



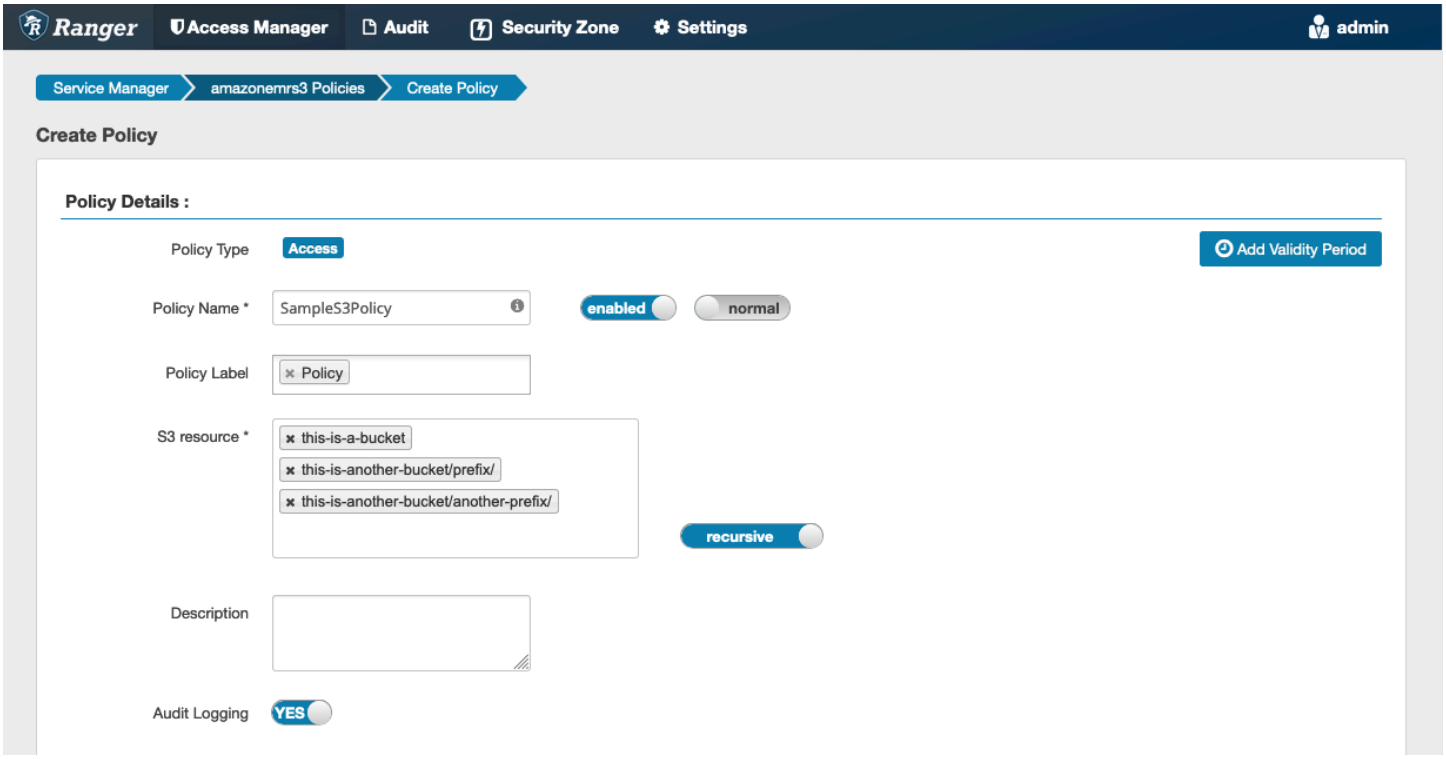
EMRFS S3 ポリシーの作成

新しいポリシーを作成するには、Service Manager の [ポリシーの作成] ページで、次のフィールドに入力します。

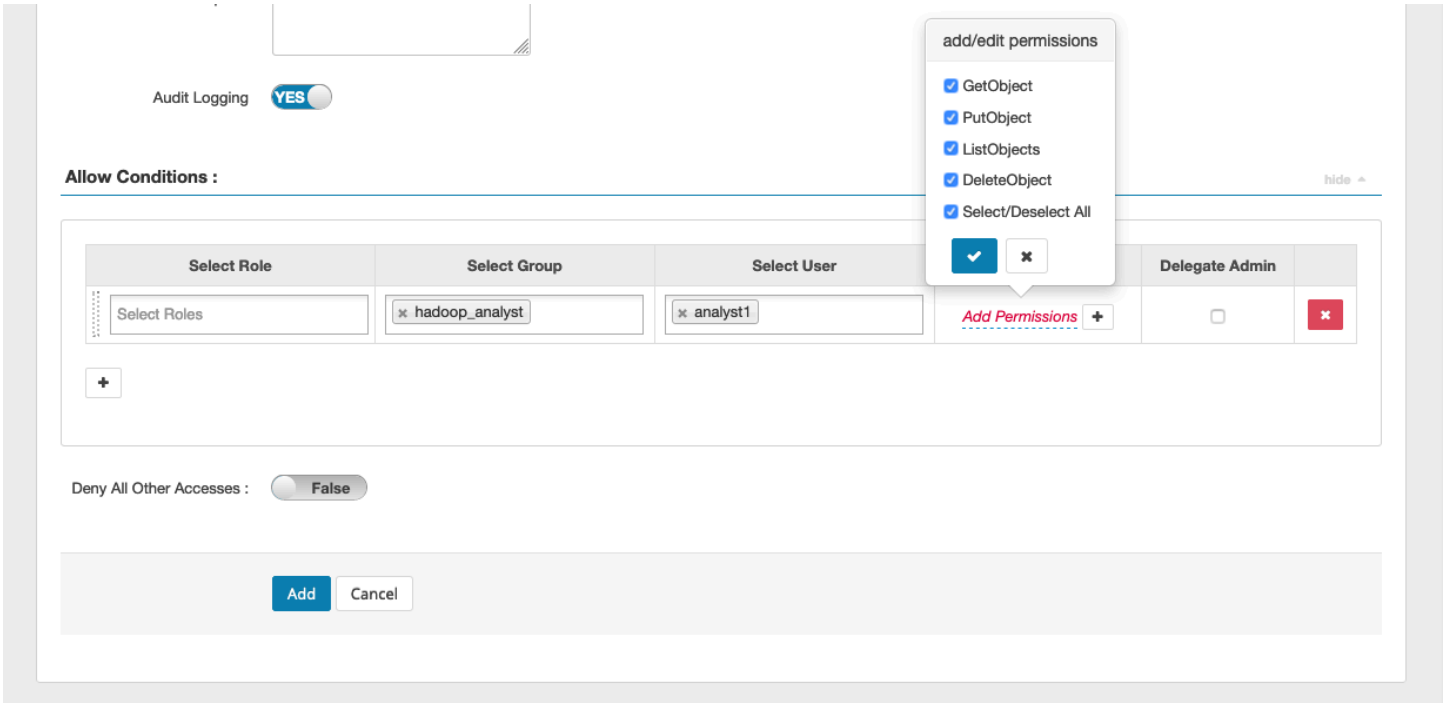
[Policy Name] (ポリシー名): このポリシーの名前。

[Policy Label] (ポリシーラベル): このポリシーにつけることができるラベル。

[S3 Resource] (S3 リソース): バケットとオプションのプレフィックスで始まるリソース。ベストプラクティスについては、「EMRFS S3 ポリシーの使用に関する注意事項」を参照してください。Ranger 管理サーバーのリソースには、**`s3://`**、**`s3a://`**、**`s3n://`** が含まれていてはいけません。



アクセス許可を付与するユーザーおよびグループを指定できます。[許可] 条件および [拒否] 条件の除外を指定することもできます。



> **Note**
>
> 各ポリシーには最大 3 つのリソースを使用できます。3 つを超えるリソースを追加した場合、このポリシーを EMR クラスターで使用すると、エラーが発生することがあります。3 つを超えるポリシーを追加すると、ポリシーの制限に関するリマインダーが表示されます。

EMRFS S3 ポリシーの使用に関する注意事項

Apache Ranger で S3 ポリシーを作成する際には、いくつかの使用上の考慮事項があります。

複数の S3 オブジェクトへのアクセス許可

再帰ポリシーとワイルドカード式を使用して、共通のプレフィックスを持つ複数の S3 オブジェクトに対するアクセス許可を付与できます。再帰ポリシーは、共通のプレフィックスを持つすべてのオブジェクトに対するアクセス許可を付与します。ワイルドカード式では、複数のプレフィクスが選択されます。これらを組み合わせて、次の例に示されているように、複数の共通のプレフィックスを持つすべてのオブジェクトに対するアクセス許可を付与します。

Example 再帰ポリシーを使用する

以下のように整理された S3 バケット内のすべての parquet ファイルをリストするためのアクセス許可が必要であるものとします。

```
s3://sales-reports/americas/
    +- year=2000
    |      +- data-q1.parquet
    |      +- data-q2.parquet
    +- year=2019
    |      +- data-q1.json
    |      +- data-q2.json
    |      +- data-q3.json
    |      +- data-q4.json
    |
    +- year=2020
    |      +- data-q1.parquet
    |      +- data-q2.parquet
    |      +- data-q3.parquet
    |      +- data-q4.parquet
    |      +- annual-summary.parquet
    +- year=2021
```

まず、プレフィックス s3://sales-reports/americas/year=2000 の付いた parquet ファイルについて考えます。これらすべての にアクセスGetObject 許可を付与するには、次の 2 つの方法があります。

非再帰ポリシーを使用する: 1 つのオプションとして、2 つの別々の非再帰ポリシー (1 つはディレクトリ用、もう 1 つはファイル用) を使用できます。

最初のポリシーでは、プレフィックス s3://sales-reports/americas/year=2020 (末尾に / はありません) に対するアクセス許可を付与します。

```
- S3 resource = "sales-reports/americas/year=2000"
- permission = "GetObject"
- user = "analyst"
```

2 番目のポリシーでは、ワイルドカード式を使用して、プレフィックス sales-reports/americas/year=2020/ (末尾の / に注意) を持つすべてのファイルに対するアクセス許可を付与します。

```
- S3 resource = "sales-reports/americas/year=2020/*"
- permission = "GetObject"
- user = "analyst"
```

再帰ポリシーを使用する: より便利な代替方法として、単一の再帰ポリシーを使用し、プレフィックスに再帰的なアクセス許可を付与できます。

```
 - S3 resource = "sales-reports/americas/year=2020"
 - permission = "GetObject"
 - user = "analyst"
 - is recursive = "True"
```

これまでのところ、プレフィックス s3://sales-reports/americas/year=2000 の付いた parquet ファイルのみが含まれています。ここで、次のようにワイルドカード式を導入することで、別のプレフィックス s3://sales-reports/americas/year=2020 が付いた parquet ファイルも同じ再帰ポリシーに含めることができます。

```
 - S3 resource = "sales-reports/americas/year=20?0"
 - permission = "GetObject"
 - user = "analyst"
 - is recursive = "True"
```

PutObject および アクセス DeleteObject 許可のポリシー

EMRFS 上のファイルに対する PutObjectおよび アクセスDeleteObject許可の記述には特別な注意が必要です。アクセス許可とは異なり GetObject、プレフィックスに付与された追加の再帰的なアクセス許可が必要なためです。

Example PutObject および アクセス DeleteObject 許可のポリシー

例えば、ファイルを削除するには、実際のファイルに対する DeleteObject アクセス許可annual-summary.parquetだけでなく、

```
- S3 resource = "sales-reports/americas/year=2020/annual-summary.parquet"
- permission = "DeleteObject"
- user = "analyst"
```

プレフィックスに対する再帰的な GetObject および PutObject アクセス許可を付与するポリシーも必要になります。

同様に、ファイル annual-summary.parquet を修正するために必要となるのも、実際のファイルに対する PutObject アクセス許可だけではありません。

```
- S3 resource = "sales-reports/americas/year=2020/annual-summary.parquet"
- permission = "PutObject"
- user = "analyst"
```

プレフィックスに対する再帰的な GetObject アクセス許可を付与するポリシーも必要になります。

```
- S3 resource = "sales-reports/americas/year=2020"
- permission = "GetObject"
- user = "analyst"
- is recursive = "True"
```

ポリシー内のワイルドカード

ワイルドカードを指定できる領域は 2 つあります。S3 リソースを指定するときには、「*」と「?」を使用できます。「*」は S3 パスに対するマッチングを提供し、プレフィクスの後のすべてのものに一致します。ポリシーの例を次に示します。

```
S3 resource = "sales-reports/americas/*"
```

これは次の S3 パスと一致します。

```
sales-reports/americas/year=2020/
sales-reports/americas/year=2019/
sales-reports/americas/year=2019/month=12/day=1/afile.parquet
sales-reports/americas/year=2018/month=6/day=1/afile.parquet
sales-reports/americas/year=2017/afile.parquet
```

ワイルドカード「?」は、1 文字にのみ一致します。ポリシーの例を次に示します。

```
S3 resource = "sales-reports/americas/year=201?/"
```

これは次の S3 パスと一致します。

```
sales-reports/americas/year=2019/
sales-reports/americas/year=2018/
sales-reports/americas/year=2017/
```

ユーザー内のワイルドカード

ユーザーにアクセス権を提供するためにユーザーを割り当てるときには、2 つの組み込みのワイルドカードを使用できます。1 つ目は、すべてのユーザーにアクセス権を提供する「{USER}」ワイルドカードです。2 番目のワイルドカードは「{OWNER}」です。これは、特定のオブジェクトまたはディレクトリの所有者にアクセス権を提供します。ただし、「{USER}」ワイルドカードは現在サポートされていません。

制限事項

EMRFS S3 プラグインの現在の制限事項は次のとおりです。

- Apache Ranger ポリシーには、最大 3 つのポリシーを設定できます。
- S3 へのアクセスは EMRFS 経由で行う必要があり、Hadoop 関連のアプリケーションで使用できます。以下はサポートされていません。

  - Boto3 ライブラリ

  - AWS SDK および AWK CLI

  - S3A オープンソースコネクタ
- Apache Ranger 拒否ポリシーはサポートされていません。
- CSE-KMS 暗号化を持つキーを使用した S3 でのオペレーションは、現在サポートされていません。

- クロスリージョンサポートはサポートされていません。
- Apache Ranger のセキュリティゾーン機能はサポートされていません。セキュリティゾーン機能を使用して定義されたアクセスコントロールの制限は、Amazon EMR クラスターには適用されません。
- Hadoop は常に EC2 インスタンスプロファイルにアクセスするため、Hadoop ユーザーは監査イベントを生成しません。
- Amazon EMR 整合性ビューを無効にすることをお勧めします。S3 は強固な整合性を備えているため、これは不要です。詳細については、「Amazon S3 strong consistency」を参照してください。
- EMRFS S3 プラグインは多数の STS 呼び出しを行います。開発アカウントで負荷テストを行い、STS 呼び出しボリュームをモニタリングすることをお勧めします。また、STS リクエストを実行してサービス制限を引き上げる AssumeRoleこともお勧めします。
- Ranger 管理サーバーはオートコンプリートをサポートしていません。

Trino プラグイン

Trino (旧称 PrestoSQL) は、HDFS、オブジェクトストレージ、リレーショナルデータベース、NoSQL データベースなどのデータソースでクエリを実行するために使用できる SQL クエリエンジンです。これにより、データを一元的な場所に移行する必要がなくなり、データがどこに置かれていてもクエリすることができます。Amazon EMR には、Trino のきめ細かなアクセスコントロールを提供できる Apache Ranger プラグインが備わっています。このプラグインはオープンソースの Apache Ranger 管理サーバーバージョン 2.0 以降と互換性があります。

トピック

- サポートされている機能
- サービス設定のインストール
- Trino ポリシーの作成
- 考慮事項
- 制限事項

サポートされている機能

Amazon EMR 上の Trino 用の Apache Ranger プラグインは、きめ細かいアクセスコントロールによって保護されている Trino クエリエンジンのすべての機能をサポートしています。これには、データベース、テーブル、列レベルのアクセスコントロール、行フィルタリング、データマスキングが

含まれます。Apache Ranger ポリシーには、ユーザーおよびグループへの付与ポリシーと拒否ポリシーを含めることができます。監査イベントもログに送信されます CloudWatch。

サービス設定のインストール

Trino サービス定義をインストールするには、Ranger 管理サーバーを設定する必要があります。Ranger 管理サーバーを設定するには、「Ranger 管理サーバーを設定する」を参照してください。

Trino サービス定義をインストールするには、次のステップを実行します。

1. Apache Ranger 管理サーバーに SSH 接続します。

```
ssh ec2-user@ip-xxx-xxx-xxx-xxx.ec2.internal
```

2. Presto サーバープラグインをアンインストールします (存在する場合)。以下のコマンドを実行します。[サービスが見つかりません] エラーが表示される場合は、Presto サーバープラグインがサーバーにインストールされていないことを意味します。次のステップに進みます。

```
curl -f -u *<admin users login>*:*_<_**_password_ **_for_** _ranger admin
 user_**_>_* -X DELETE -k 'https://*<RANGER SERVER ADDRESS>*:6182/service/public/
v2/api/servicedef/name/presto'
```

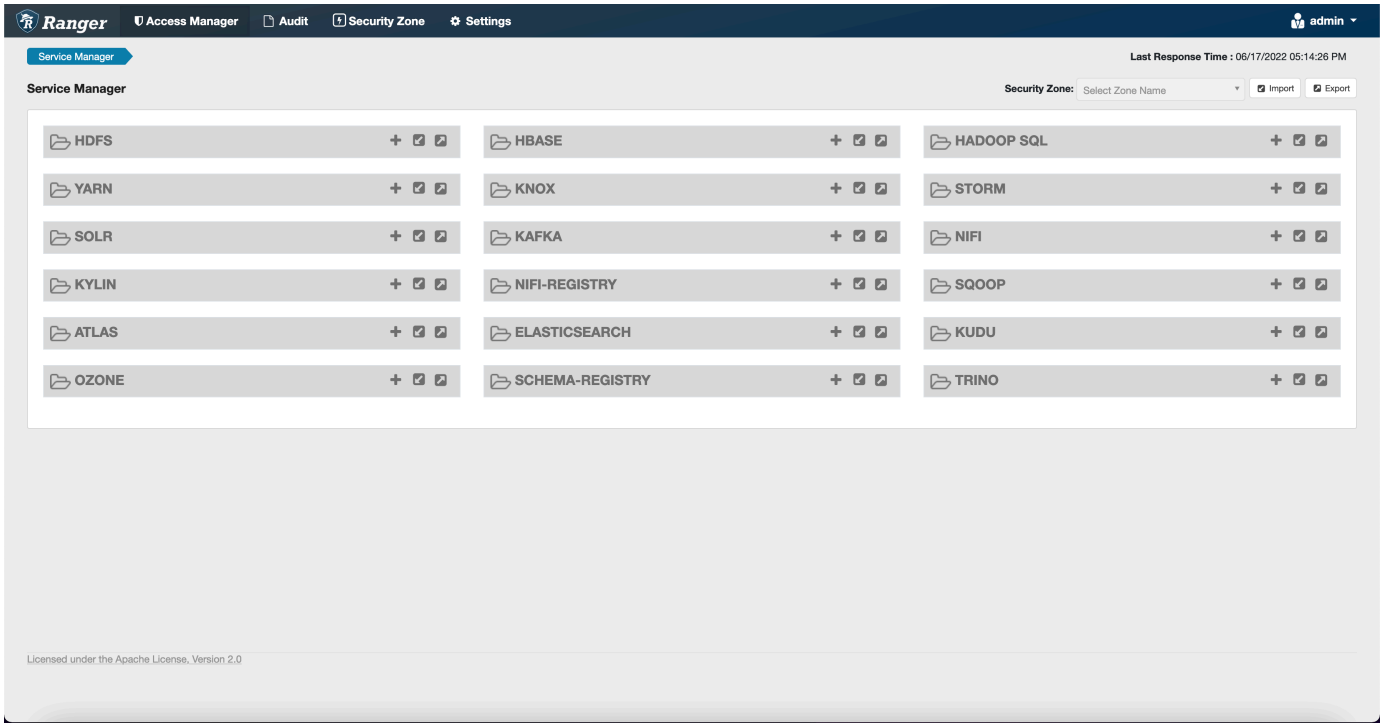
3. サービス定義および Apache Ranger 管理サーバープラグインをダウンロードします。一時ディレクトリに、サービス定義をダウンロードします。このサービス定義は Ranger 2.x バージョンでサポートされています。

```
wget https://s3.amazonaws.com/elasticmapreduce/ranger/service-definitions/
version-2.0/ranger-servicedef-amazon-emr-trino.json
```

4. Amazon EMR の Apache Trino サービス定義を登録します。

```
curl -u *<admin users login>*:*_<_**_password_ **_for_** _ranger admin user_**_>_*
 -X POST -d @ranger-servicedef-amazon-emr-trino.json \
-H "Accept: application/json" \
-H "Content-Type: application/json" \
-k 'https://*<RANGER SERVER ADDRESS>*:6182/service/public/v2/api/servicedef'
```

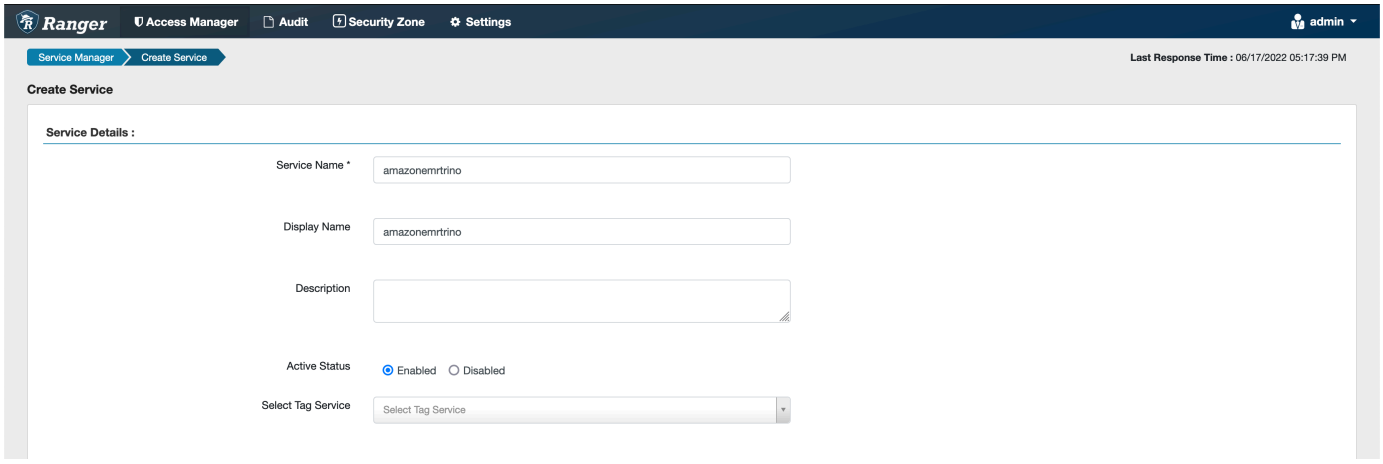
このコマンドが正常に実行されると、次の図に示すように、Ranger 管理 UI に TRINO という新しいサービスが表示されます。



5. 次の情報を入力して、TRINO アプリケーションのインスタンスを作成します。

   サービス名: 使用するサービス名。推奨値は amazonemrtrino です。このサービス名をメモしておいてください。Amazon EMR セキュリティ設定を作成するときに必要になります。

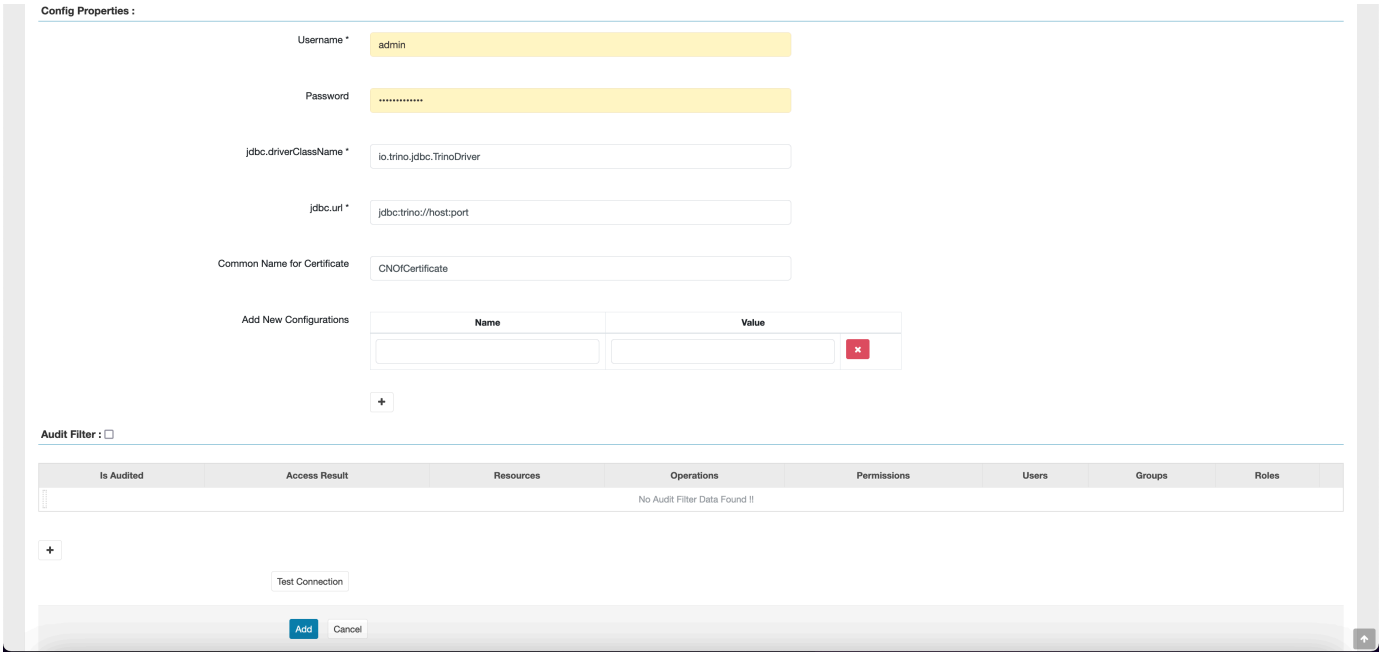
   [Display Name] (表示名): このインスタンスに表示される名前。推奨値は amazonemrtrino です。



   jdbc.driver.ClassName: Trino 接続用の JDBC クラスのクラス名。デフォルト値を使用できます。

   jdbc.url: Trino コーディネーターへの接続時に使用する JDBC 接続文字列。

[Common Name For Certificate] (証明書の共通名): クライアントプラグインから管理サーバーに接続するために使用する証明書内の CN フィールド。この値は、プラグイン用に作成された TLS 証明書の CN フィールドと一致する必要があります。



このプラグインの TLS 証明書は、Ranger 管理サーバーのトラストストアに登録されている必要があるので注意してください。詳細については、「TLS 証明書」を参照してください。

Trino ポリシーの作成

新しいポリシーを作成する際は、次のフィールドに入力します。

[Policy Name] (ポリシー名): このポリシーの名前。

[Policy Label] (ポリシーラベル): このポリシーにつけることができるラベル。

カタログ: このポリシーが適用されるカタログ。ワイルドカード「*」はすべてのカタログを表します。

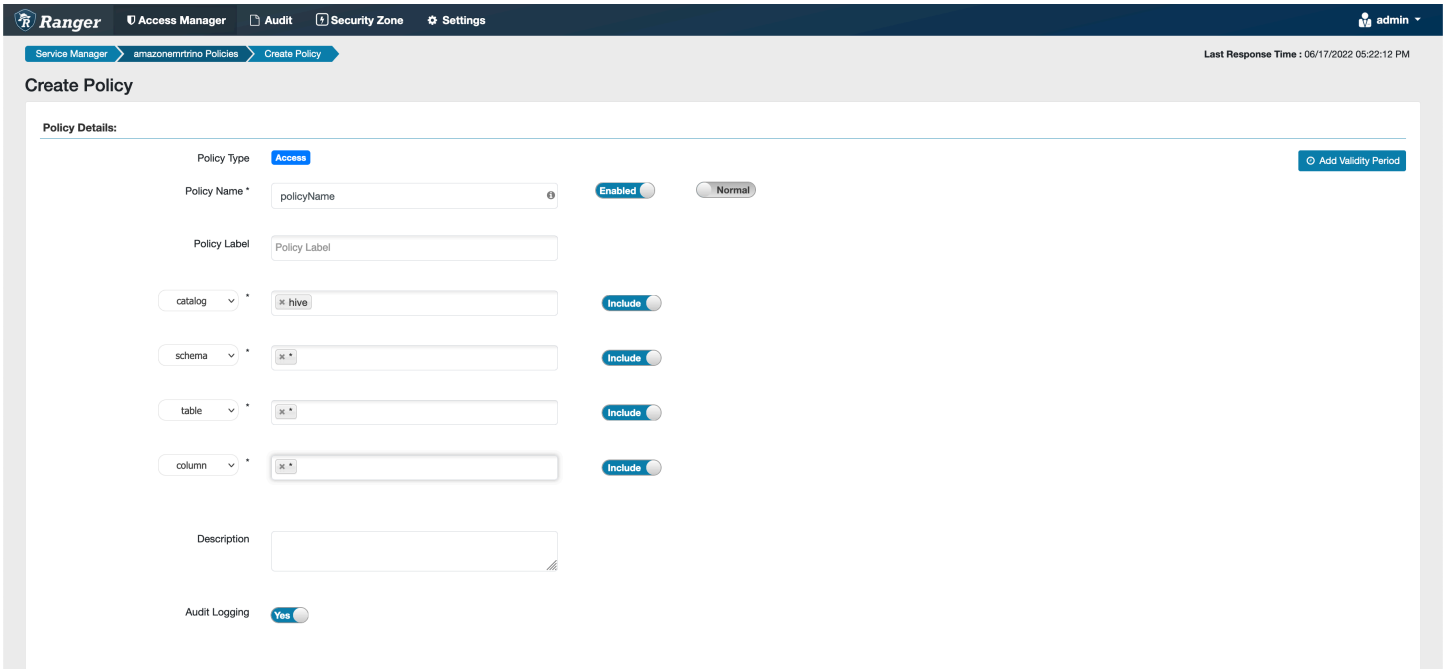
スキーマ: このポリシーが適用されるスキーマ。ワイルドカード「*」はすべてのスキーマを表します。

[Table] (テーブル): このポリシーが適用されるテーブル。ワイルドカード「*」はすべてのテーブルを表します。

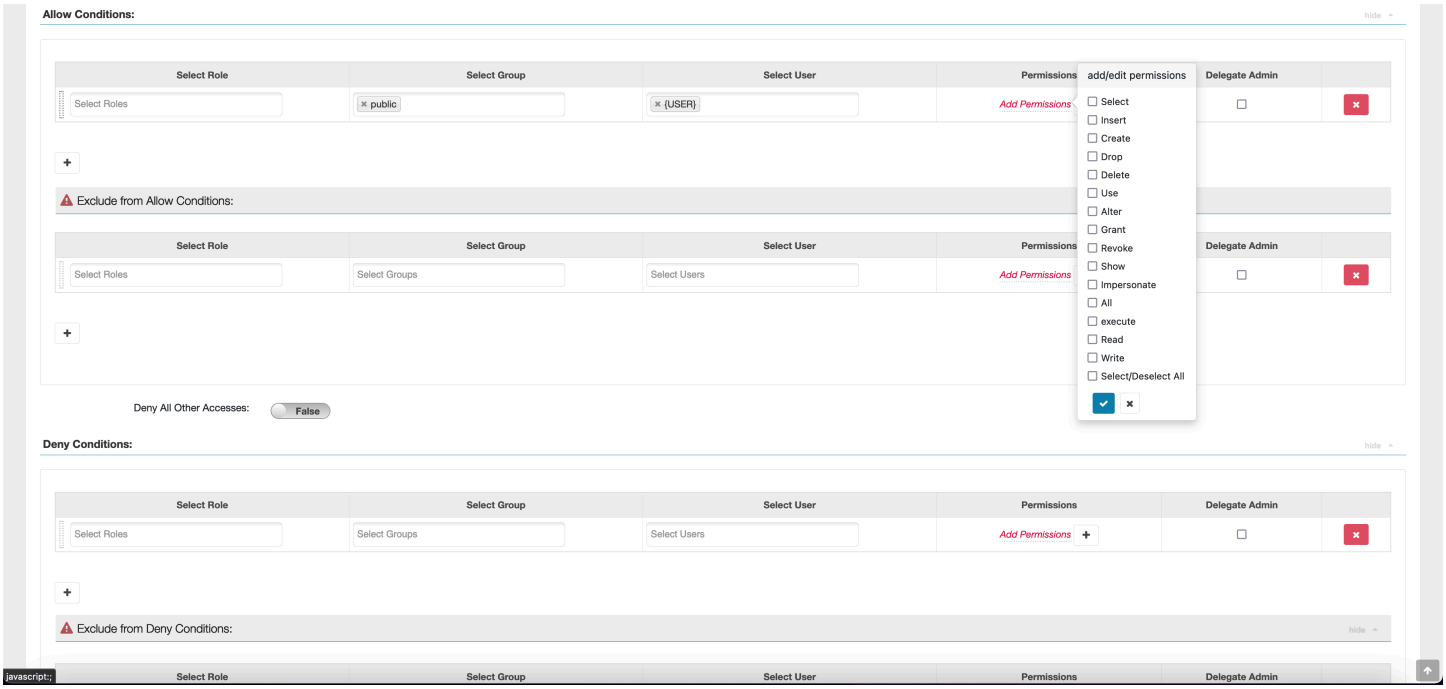
列: このポリシーが適用される列。ワイルドカード「*」はすべての列を表します。

[Description] (説明): このポリシーの説明。

Trino ユーザー (ユーザー偽装アクセス用)、Trino システム/セッションプロパティ (エンジンシステムまたはセッションプロパティの変更用)、関数/プロシージャ (関数またはプロシージャの呼び出しの許可用)、URL (データロケーション上のエンジンへの読み取り/書き込みアクセスの許可用) には、他のタイプのポリシーが存在します。



特定のユーザーとグループにアクセス許可を付与するには、ユーザーおよびグループを入力します。[許可] 条件および [拒否] 条件の除外を指定することもできます。



許可条件と拒否条件を指定したら、[保存] を選択します。

考慮事項

Apache Ranger で Trino ポリシーを作成する際には、注意すべきいくつかの使用上の考慮事項があります。

Hive メタデータサーバー

Hive メタデータサーバーには、不正アクセスから保護するために、信頼されたエンジン (具体的には Trino エンジン) からしかアクセスできません。また、Hive メタデータサーバーには、クラスター上のすべてのノードからもアクセスできます。ポート 9083 は必須で、メインノードへのすべてのノードアクセスを提供します。

認証

デフォルトでは、Trino は Amazon EMR セキュリティ設定で設定された Kerberos を使用して認証するように設定されています。

転送時の暗号化が必要

Trino プラグインを使用するには、Amazon EMR セキュリティ設定で転送時の暗号化が有効になっている必要があります。暗号化を有効にするには、「転送中の暗号化」を参照してください。

制限事項

現在、Trino プラグインには以下の制限事項があります。

- Ranger 管理サーバーはオートコンプリートをサポートしていません。

Apache Ranger のトラブルシューティング

Apache Ranger の使用に関して一般的に診断される問題をいくつか紹介します。

レコメンデーション

- 単一のメインノードクラスターを使用してテストする: 単一ノードマスタークラスターは、マルチノードクラスターよりも迅速にプロビジョニングでき、各テスト反復の時間を短縮できます。
- クラスターで開発モードを設定する: EMR クラスターを開始するときは、`--additional-info"` パラメータを以下に設定します。

  ```
  '{"clusterType":"development"}'
  ```

  このパラメータは AWS CLI または AWS SDK を介してのみ設定でき、Amazon EMR コンソールからは利用できません。このフラグが設定されている場合にマスターのプロビジョニングに失敗すると、Amazon EMR サービスはクラスターを廃止する前にしばらくの間、クラスターの稼働状態を維持します。この時間は、クラスターが終了する前にさまざまなログファイルをプローブするのに非常に便利です。

EMR クラスターのプロビジョニングが失敗した

Amazon EMR クラスターの開始に失敗する理由はいくつかあります。問題を診断するいくつかの方法があります。

EMR プロビジョニングログを確認する

Amazon EMR は、Puppet を使用してクラスターにアプリケーションをインストールおよび設定します。ログを見ると、クラスターのプロビジョニングフェーズ中にエラーが発生したかどうかについての詳細が示されます。クラスターまたは S3 (ログが S3 にプッシュされるように設定されている場合) でログにアクセスできます。

ログは、ディスク上の /var/log/provision-node/apps-phase/0/{UUID}/puppet.log および s3://<LOG LOCATION>/<CLUSTER ID>/node/<EC2 INSTANCE ID>/provision-node/apps-phase/0/{UUID}/puppet.log.gz. に保存されます。

一般的なエラーメッセージ

エラーメッセージ	原因
Puppet (err): emr-record-server failed! journalctl log for emr-record-server：	EMR レコードサーバーの開始に失敗しました。下記の「EMR レコードサーバーログ」を参照してください。
Puppet (err): emrsecretagent の emr-record-server failed! journalctl ログのシステム開始：	EMR シークレットエージェントの開始に失敗しました。下記の「SecretAgent ログの確認」を参照してください。
/Stage[main]/Ranger_plugins::Ranger_hive_plugin/Ranger_plugins::Prepare_two_way_tls[configure 2-way TLS in Hive plugin]/Exec[create keystore and truststore for Ranger Hive plugin]/returns (notice): 140408606197664:error:0906D06C:PEM routines:PEM_read_bio:no start line:pem_lib.c:707:Expecting: ANY PRIVATE KEY	Apache Ranger プラグイン証明書のシークレットマネージャーのプライベート TLS 証明書が正しい形式でないか、プライベート証明書ではありません。証明書の形式について、「TLS 証明書」を参照してください。
/Stage[main]/Ranger_plugins::Ranger_s3_plugin/Ranger_plugins::Prepare_two_way_tls[Configure 2-way TLS in Ranger s3 plugin]/Exec[create keystore and truststore for Ranger amazon-emr-s3 plugin]/returns (notice): GetSecretValue オペレーションを呼び出すときにエラー (AccessDeniedException) が発生しました: User: arn:aws:sts::XXXXXXXXXXXXXXXX:assumed-role/EMR_EC2_DefaultRole/i-XXXXXXXXXXXXXXXX: secretsmanager:Get	EC2 インスタンスプロファイルロールに、シークレットエージェントから TLS 証明書を取得するための適切なアクセス許可がありません。

エラーメッセージ	原因
SecretValue on resource: arn:aws:secrets:Secretsmanager:us-east-1:XXXXXXX:secret:AdminServer-XXXXXX	

SecretAgent ログの確認

シークレットエージェントのログは、EMR ノード上の `/emr/secretagent/log/` または S3 の `s3://<LOG LOCATION>/<CLUSTER ID>/node/<EC2 INSTANCE ID>/daemons/secretagent/` ディレクトリにあります。

一般的なエラーメッセージ

エラーメッセージ	原因
スレッド「main」com.amazonaws.services.securitytoken.model.AWSSecurityTokenServiceException: User: arn:aws:sts::XXXXXXXXXXXX:assumed-role/EMR_EC2_DefaultRole/i-XXXXXXXXXXXXXXXX の例外は、sts:AssumeRole on resource: arn:aws:iam:::XXXXXXXXXXXXXXXX:role/RangerPluginDataAccessRole* (サービス： AWSSecurityTokenService; ステータスコード: 403; エラーコード： AccessDenied; リクエスト ID: XXXXXXXXXXXX-XXXX-XXXX-XXXX XXXXXXXX; Proxy: null) を実行する権限がありません	上記の例外は、EMR EC2 インスタンスプロファイルロール にロール を引き受けるアクセス許可がないことを意味しますRangerPluginDataAccessRole。Apache Ranger とのネイティブ統合のための IAM ロール を参照してください。
ERROR qtp54617902-149: Web App Exception Occurred javax.ws.rs.NotAllowedException: HTTP 405 メソッドは許可されていません	これらのエラーは無視して問題ありません。

レコードサーバーログの確認 (SparkSQL の場合)

EMR Record Server ログは、EMR ノードの /var/log/emr-record-server/ で入手できます。または、S3 の s3://<LOG LOCATION>/<CLUSTER ID>/node/<EC2 INSTANCE ID>/daemons/emr-record-server/ ディレクトリにあります。

一般的なエラーメッセージ

エラーメッセージ	原因
InstanceMetadataServiceResourceFetcher:105 - [] トークン com.amazonaws の取得に失敗しました。SdkClientException: サービスエンドポイントへの接続に失敗しました	EMR の起動に SecretAgent 失敗したか、問題が発生しています。 SecretAgent ログにエラーがないか、puppet スクリプトを調べて、プロビジョニングエラーがあったかどうかを確認します。

クエリが予期せず失敗している

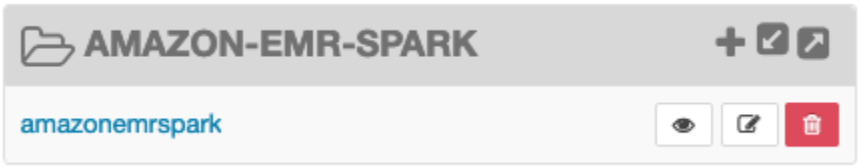
Apache Ranger プラグインログを確認する (Apache Hive、EMR RecordServer、EMR SecretAgent など、ログ)

このセクションは、Apache Hive、EMR Record Server、EMR など、Ranger プラグインと統合するすべてのアプリケーションで共通です SecretAgent。

一般的なエラーメッセージ

エラーメッセージ	原因
エラー PolicyRefresher:272 - [] PolicyRefresher(serviceName =policy-repository): サービスが見つかりませんでした。Will clean up local cache of policies (-1)	このエラーメッセージは、EMR セキュリティ設定で指定したサービス名が Ranger 管理サーバーのサービスポリシーリポジトリと一致していないことを示しています。

Ranger 管理サーバー内で AMAZON-EMR-SPARK サービスが次のようになっている場合は、サービス名として **`amazonemrspark`** を入力する必要があります。



AWS Glue データカタログビューの操作 (プレビュー)

> **Note**
>
> AWS Amazon EMR の Glue Data Catalog ビューはプレビューリリースであり、変更される可能性があります。この機能は、AWS サービス条件で定義されているプレビューサービスとして提供されます。

AWS Glue データカタログで単一の共通ビューを作成および管理できます。1 つの共通ビューは複数の SQL クエリエンジンをサポートしているため、Amazon EMR Amazon Athena や Amazon Redshift など AWS のサービス、さまざまな で同じビューにアクセスできます。

Data Catalog でビューを作成することで、 でリソース許可とタグベースのアクセスコントロールを使用して AWS Lake Formation 、Data Catalog ビューへのアクセスを許可できます。このアクセスコントロール方法を使用すると、ビューの作成時に参照したテーブルへの追加のアクセスを設定する必要はありません。このアクセス許可を付与する方法は、定義セマンティクスと呼ばれ、これらのビューは定義ビューと呼ばれます。Lake Formation でのアクセスコントロールの詳細については、「 AWS Lake Formation デベロッパーガイド」の「Data Catalog リソースに対するアクセス許可の付与と取り消し」を参照してください。

データカタログビューは、以下のユースケースに役立ちます。

- きめ細かなアクセスコントロール – ユーザーが必要とするアクセス許可に基づいてデータアクセスを制限するビューを作成します。例えば、データカタログのビューを使用して、人事 (HR) 部門に属さない従業員が個人を特定できる情報 (PII) を表示できないようにすることができます。
- 完全なビュー定義 — データカタログのビューに特定のフィルターを適用することで、データカタログのビュー内のデータレコードが常に完全であることを確認します。
- セキュリティの強化 — ビューの作成に使用されるクエリ定義は完全である必要があります。この利点は、データカタログ内のビューが悪意のあるプレイヤーからの SQL コマンドの影響を受けにくいことを意味します。

- データの簡単な共有 — データを移動 AWS アカウント せずに他の とデータを共有します。詳細については、「Lake Formation でのクロスアカウントデータ共有」を参照してください。

データカタログビューの作成

> **Important**
>
> このプレビューリリース中、Amazon EMR はビューの作成時に使用する Spark-SQL を検証しません。リスクを軽減するために、ビュー作成アクセス許可を付与するユーザーを制限することをお勧めします。

Data Catalog ビューを作成するには、ビューの作成時に参照するすべてのテーブルGrantableのオプションを含む完全なSELECTアクセス許可を持つ IAM ロールを使用する必要があります。このロールは定義ロールと呼ばれます。データカタログビューの作成に必要なアクセス許可と前提条件の完全なリストについては、「 AWS Lake Formation デベロッパーガイド」の「ビューの使用」を参照してください。IAM ロールを設定するには AWS CLI 、 を使用する必要があります。詳細については、「」の「IAM ロールを使用する AWS CLI」を参照してください。

データカタログビューを作成するには、次の手順に従います。

> **Note**
>
> Amazon EMR の Apache Spark からデータカタログビューにアクセスするには、ダイアレクトを に設定SPARKし、 DialectVersionを に設定する必要があります3.4.1-amzn-2。

1. まずプレビューモデルをダウンロードします。

```
aws s3 cp s3://emr-data-access-control-us-east-1/beta/glue-views/model/
service-2.json
```

2. プレビューモデルを使用する AWS CLI ように を設定します。

```
aws configure add-model --service-model file:///<path-to-preview-model>/
service-2.json --service-name glue-views
```

3. ビューを作成します。

```
aws glue-views create-table --cli-input-json '{
  "DatabaseName": "<database>",
  "TableInput": {
    "Name": "<view>",
    "StorageDescriptor": {
      "Columns": [
        {
          "Name": "<col1>",
          "Type": "<data-type>"
        },
        ...
        {
          "Name": "<colN>",
          "Type": "<data-type>"
        }
      ]
    },
    "ViewDefinition": {
      "SubObjects": [
        "arn:aws:glue:<aws-region;>:<aws-account-id>:table/<database>/<referenced-
table1>",
        ...
        "arn:aws:glue:<aws-region>:<aws-account-id>:table/<database>/<referenced-
tableN>",
       ],
       "IsProtected": true,
      "Representations": [
        {
          "Dialect": "SPARK",
          "DialectVersion": "3.4.1-amzn-2",
          "ViewOriginalText": "<Spark-SQL>",
          "ViewExpandedText": "<Spark-SQL>"
        }
      ]
    }
  }
}'
```

データカタログビューへのアクセスの有効化

> **Important**
>
> Data Catalog ビューへのアクセスは、本番環境ではなく、テスト環境で EMR クラスターでのみ有効にすることをお勧めします。

Amazon EMR の Apache Spark からデータカタログビューにアクセスするには、まず Lake Formation のサポートを有効にし、以下のスクリプトを使用して Amazon EMR の Spark でビューのサポートを有効にする必要があります。サポートの有効化の詳細については、「Amazon EMR で Lake Formation を有効にする」および「カスタムブートストラップアクションを使用する」を参照してください。

```
# Download the script and upload it to Amazon S3
wget https://emr-data-access-control-us-east-1.s3.amazonaws.com/beta/glue-views/ba/
enable-mdv.sh /Users/$USER/enable-mdv.sh
aws s3 cp /Users/$USER/enable-views.sh s3://<bucket>/<prefix>/enable-views.sh

# EMR Security Configuration
cat <<EOT > /Users/$USER/lakeformation-protection.json
{
    "AuthorizationConfiguration":{
        "IAMConfiguration":{
            "EnableApplicationScopedIAMRole":true
        },
        "LakeFormationConfiguration":{
            "AuthorizedSessionTagValue":"Amazon EMR"
        }
    },
     "EncryptionConfiguration": {
            "EnableInTransitEncryption": true,
            "InTransitEncryptionConfiguration": {
            "TLSCertificateConfiguration": {
                 "CertificateProviderType": "PEM",
                 "S3Object": "s3://<BUCKET>/<PREFIX>/certificates.zip"
            }
        }
    }
}
EOT
```

```
SECURITY_CONFIG="RuntimeRolesWithAWSLakeFormation"

aws emr create-security-configuration \
--name $SECURITY_CONFIG \
--security-configuration file:///Users/$USER/lakeformation-protection.json

# EMR Cluster version
RELEASE_LABEL="emr-6.15.0"
```

次に、ブートストラップアクションを使用する次の AWS CLI コマンドを使用して、データカタログビューをサポートする EMR クラスターを作成します。

```
aws emr create-cluster \
...
--release-label $RELEASE_LABEL \
--security-configuration $SECURITY_CONFIG \
--bootstrap-actions \
 Name='Enable Views',Path="s3://<bucket>/<prefix>/enable-views.sh"
```

データカタログビューをクエリする

> **Important**
>
> このプレビューリリースでは、信頼できるソースからのみビューにアクセスすることをお勧めします。プレビューでは、Amazon EMR クラスターを保護する検証の量には制限があります。

データカタログビューを作成した後、IAM ロールを使用してビューをクエリできるようになりました。IAM ロールには、データカタログビューに対する アクセスSELECT許可が必要です。ビューで参照される基盤となるテーブルへのアクセスを許可する必要はありません。この IAM ロールをランタイムロールとして使用する必要があります。Amazon EMR ステップ、EMR Studio、および SageMaker Studio のランタイムロールを使用して、EMR クラスターからビューにアクセスできます。ランタイムロールの詳細については、「Amazon EMR ステップのランタイムロール」を参照してください。

すべてをセットアップしたら、ビューをクエリできます。例えば、EMR Studio の Workspace に EMR クラスターをアタッチした後、次のクエリを実行してビューにアクセスできます。

```
SELECT * from <database>.<glue-data-catalog-view> LIMIT 10
```

制限事項

Data Catalog ビューを使用する場合は、次の制限事項を考慮してください。

- Amazon EMR 6.15.0 でのみデータカタログビューを作成できます。
- ビュー定義では、最大 10 個のテーブルしか参照できません。
- PROTECTED データカタログビューのみを作成できます。 UNPROTECTEDビューはサポートされていません。
- Data Catalog ビューで別の AWS アカウント のテーブルを参照することはできません。
- ユーザー定義関数 (UDFsはサポートされていません。
- データカタログビューで Apache Hudi や Apache Iceberg などのオープンテーブル形式を参照することはできません。
- データカタログビューで他のビューを参照することはできません。

セキュリティグループを使用してネットワークトラフィックを制御する

セキュリティグループは、クラスター内の EC2 インスタンスの仮想ファイアウォールとして機能し、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をコントロールします。各セキュリティグループには、インバウンドトラフィックをコントロールする一連のルールと、アウトバウンドトラフィックをコントロールする別個の一連のルールがあります。詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EC2 security groups for Linux instances」(Linux インスタンス用の Amazon EC2 セキュリティグループ) を参照してください。

Amazon EMR では 2 つのクラスのセキュリティグループ (Amazon EMR マネージドセキュリティグループと追加のセキュリティグループ) を使用します。

すべてのクラスターにはマネージドセキュリティグループが関連付けられています。Amazon EMR が作成するデフォルトのマネージドセキュリティグループを使用することも、カスタムのマネージドセキュリティグループを指定することもできます。いずれにしても、Amazon EMR は、クラスターがクラスターインスタンスと AWS サービス間で通信するために必要なマネージドセキュリティグループにルールを自動的に追加します。

追加セキュリティグループはオプションです。マネージドセキュリティグループに加えて、それらを指定して、クラスターインスタンスへのアクセスを調整できます。追加のセキュリティグループには、定義したルールのみが含まれます。Amazon EMR はそれらを変更しません。

Amazon EMR がマネージドセキュリティグループで作成するルールでは、クラスターは内部コンポーネント間で通信できます。ユーザーおよびアプリケーションがクラスター外からクラスターにアクセスできるようにするには、マネージドセキュリティグループでルールを編集するか、追加のルールを持つ追加のセキュリティグループを作成するか、または両方を実行できます。

> **Important**
>
> マネージドセキュリティグループでルールを編集すると、意図しない結果になることがあります。クラスターが正しく動作するために必要なトラフィックを意図せずにブロックする可能性があり、ノードに到達できないため エラーが発生します。実装の前にセキュリティグループ設定を注意深く計画してテストしてください。

クラスターを作成するときは、セキュリティグループのみを指定できます。クラスターの実行中は、クラスターまたはクラスターインスタンスに追加できませんが、既存のセキュリティグループからルールを編集、追加および削除できます。ルールを保存するとすぐに有効になります。

デフォルトでは、セキュリティグループは制限されています。トラフィックを許可するルールが追加されていない限り、トラフィックは拒否されます。同じトラフィックと同じソースに適用されるルールが複数ある場合、最も許容度の大きいルールが適用されます。たとえば、IP アドレス 192.0.2.12/32 からの SSH を許可するルールと、IP アドレス範囲 192.0.2.0/24 からのすべての TCP トラフィックへのアクセスを許可する別のルールがある場合、192.0.2.12 を含む範囲からのすべての TCP トラフィックを許可するルールが優先されます。この場合、192.0.2.12 のクライアントには、意図した以上のアクセスがある場合があります。

> **Important**
>
> セキュリティグループのルールを編集してポートを開くときには注意が必要です。ワークロードの実行に必要なプロトコルとポートには、必ず信頼できる認証済みのクライアントからのトラフィックのみを許可するルールを追加してください。

例外のリストに追加していない任意のポートでのパブリックアクセスをルールで許可する場合は、クラスターの作成を防止するために使用する Amazon EMR のブロックパブリックアクセスを各リー

ジョンで設定できます。2019 年 7 月以降に作成された AWS アカウントの場合、Amazon EMR のパブリックアクセスブロックはデフォルトでオンになっています。2019 年 7 月より前にクラスターを作成した AWS アカウントの場合、Amazon EMR のパブリックアクセスブロックはデフォルトでオフになっています。詳細については、「Amazon EMR のパブリックアクセスブロックの使用」を参照してください。

トピック

> **Note**
>
> Amazon EMR では、「マスター」や「スレーブ」などの不快感を与える可能性のある、あるいは包括的でない業界用語の代わりに、包括的な用語を使用することを目指しています。より包括的なエクスペリエンスを促進し、サービスコンポーネントを理解しやすくするために、新しい用語に移行しました。
> ここでは、「ノード」をインスタンスとして説明し、Amazon EMR インスタンスタイプをプライマリインスタンス、コアインスタンス、タスクインスタンスとして記述します。移行中は、Amazon EMR のセキュリティグループに関連する用語など、従来の古い用語への参照が残っている場合があります。

Amazon EMR マネージドセキュリティグループでの作業

> **Note**
>
> Amazon EMR では、「マスター」や「スレーブ」などの不快感を与える可能性のある、あるいは包括的でない業界用語の代わりに、包括的な用語を使用することを目指しています。より包括的なエクスペリエンスを促進し、サービスコンポーネントを理解しやすくするために、新しい用語に移行しました。
> ここでは、「ノード」をインスタンスとして説明し、Amazon EMR インスタンスタイプをプライマリインスタンス、コアインスタンス、タスクインスタンスとして記述します。移行中

は、Amazon EMR のセキュリティグループに関連する用語など、従来の古い用語への参照が残っている場合があります。

プライマリインスタンスと、クラスター内のコアインスタンスおよびタスクインスタンスには、異なるマネージドセキュリティグループが関連付けられています。サービスへのアクセスの追加マネージドセキュリティグループは、プライベートサブネットにクラスターを作成する場合に必要です。ネットワーク設定に関するマネージドセキュリティグループの役割の詳細については、「Amazon VPC オプション」を参照してください。

クラスターのマネージドセキュリティグループを指定する場合は、すべてのマネージドセキュリティグループに対して、デフォルトまたはカスタムの同じタイプのセキュリティグループを使用する必要があります。例えば、プライマリインスタンスにカスタムセキュリティグループを指定して、コアインスタンスとタスクインスタンスにカスタムセキュリティグループを指定しないということはできません。

デフォルトのマネージドセキュリティグループを使用する場合、クラスターの作成時にそれらを指定する必要はありません。Amazon EMR では、デフォルトが自動的に使用されます。さらに、クラスターの VPC にデフォルトがまだ存在しない場合、Amazon EMR によって作成されます。Amazon EMR は、明示的に指定した場合やまだ存在しない場合にもそれらを作成します。

クラスターの作成後にマネージドセキュリティグループでルールを編集できます。新しいクラスターを作成するときは、Amazon EMR によって指定したマネージドセキュリティグループのルールがチェックされ、以前追加された可能性があるルールに加えて、新しいクラスターに必要な不足しているインバウンドルールが作成されます。特に明記しない限り、デフォルトの Amazon EMR マネージドセキュリティグループ用の各ルールも、指定したカスタム Amazon EMR マネージドセキュリティグループに追加されます。

デフォルトのマネージドセキュリティグループは次のとおりです。

- ElasticMapReduce-プライマリ

  このセキュリティグループのルールについては、「プライマリインスタンスの Amazon EMR マネージドセキュリティグループ (パブリックサブネット)」を参照してください。
- ElasticMapReduceコア

  このセキュリティグループのルールについては、「コアインスタンスとタスクインスタンスの Amazon EMR マネージドセキュリティグループ (パブリックサブネット)」を参照してください。
- ElasticMapReduce-プライマリプライベート

このセキュリティグループのルールについては、「プライマリインスタンスの Amazon EMR マネージドセキュリティグループ (プライベートサブネット)」を参照してください。

- ElasticMapReduce-コアプライベート

  このセキュリティグループのルールについては、「コアインスタンスとタスクインスタンスの Amazon EMR マネージドセキュリティグループ (プライベートサブネット)」を参照してください。

- ElasticMapReduce-ServiceAccess

  このセキュリティグループのルールについては、「サービスアクセスの Amazon EMR マネージドセキュリティグループ (プライベートサブネット)」を参照してください。

プライマリインスタンスの Amazon EMR マネージドセキュリティグループ (パブリックサブネット)

パブリックサブネットのプライマリインスタンスのデフォルトのマネージドセキュリティグループのグループ名は ElasticMapReduce-プライマリ です。これには、以下のルールが含まれています。カスタムマネージドセキュリティグループを指定すると、Amazon EMR によってすべての同じルールがカスタムセキュリティグループに追加されます。

タイプ	プロトコル	ポート範囲	ソース	詳細
インバウンドルール				
すべての ICMP-IPv4	すべて	該当なし	プライマリインスタンスのマネージドセキュリティグループのグループ ID。言い換えると、ルールが表示される同じセキュリティグループです。	反射的なルールにより、指定されたセキュリティグループに関連付けられた任意のインスタンスからのインバウンドトラフィックが許可されます。複数のクラスターに対してデフォルトの `ElasticMapReduce-primary` を使用することにより、クラスターのコアノードとタスクノードは、どの ICMP、TCP、UDP ポートでも相互に通信できます。カスタムのマネージドセキュリティ
すべての TCP	TCP	すべて		
すべての UDP	UDP	すべて		

タイプ	プロトコル	ポート範囲	ソース	詳細
				グループを指定して、クラスター間のアクセスを制限します。
すべての ICMP-IPV4	すべて	該当なし	コアノードとタスクノードに指定されたマネージドセキュリティグループのグループ ID。	これらのルールでは、インスタンスが異なるクラスターにある場合でも、すべてのインバウンド ICMP トラフィックと、指定されたセキュリティグループに関連付けられたすべてのコアインスタンスとタスクインスタンスからの TCP、UDP ポート経由のトラフィックが許可されます。
すべての TCP	TCP	すべて		
すべての UDP	UDP	すべて		
カスタム	TCP	8443	さまざまな Amazon IP アドレス範囲	これらのルールにより、クラスターマネージャーはプライマリノードと通信できます。

コンソールを使用して、信頼できるソースにプライマリセキュリティグループへの SSH アクセスを許可するには

セキュリティグループを編集するには、クラスターが存在する VPC のセキュリティグループを管理する権限が必要です。詳細については、「IAM ユーザーガイド」の「IAM ユーザーのアクセス許可の変更」と、EC2 セキュリティグループを管理できるようにする「ポリシーの例」を参照してください。

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. [Clusters] を選択します。変更するクラスターの ID を選択します。
3. ネットワークとセキュリティペインで、EC2 セキュリティグループ (ファイアウォール) ドロップダウンを展開します。
4. プライマリノード で、セキュリティグループを選択します。
5. [Edit inbound rules] (インバウンドルールの編集) を選択します。

6. 次の設定で、パブリックアクセスを許可するインバウンドルールを確認します。存在する場合は、[Delete] (削除) をクリックして削除します。

   - タイプ

     SSH

   - [ポート]

     22

   - ソース

     Custom 0.0.0.0/0

   > **Warning**
   >
   > 2020 年 12 月以前は、ポート 22 ですべてのソースからのインバウンドトラフィックを許可する事前設定済みのルールがありました。このルールは、プライマリノードへの最初の SSH 接続を簡素化するために作成されたものです。このインバウンドルールを削除して、信頼できる送信元のみにトラフィックを制限することを強くお勧めします。

7. ルールのリストの下部までスクロールし、[Add Rule] (ルールの追加) を選択します。
8. [Type (タイプ)] で、[SSH] を選択します。

   SSH を選択すると、[Protocol] (プロトコル) に [TCP] が、[Port Range] (ポート範囲) に [22] が自動的に入力されます。
9. [source] (送信元) には、[My IP] (マイ IP) を選択して、IP アドレスを送信元アドレスとして自動的に追加します。[Custom] (カスタム) で信頼済みクライアントの IP アドレスの範囲を追加することも、他のクライアントに追加のルールを作成することもできます。多くのネットワーク環境では IP アドレスを動的に割り当てるため、将来的に信頼済みクライアントの IP アドレスを更新することが必要になる場合があります。
10. [保存] を選択します。
11. 必要に応じて、ネットワークとセキュリティペインの Core ノードとタスクノードで他のセキュリティグループを選択し、上記のステップを繰り返して、SSH クライアントがコアノードとタスクノードにアクセスできるようにします。

コアインスタンスとタスクインスタンスの Amazon EMR マネージドセキュリティグループ (パブリックサブネット)

パブリックサブネットのコアインスタンスとタスクインスタンスのデフォルトのマネージドセキュリティグループのグループ名は ElasticMapReduce-core です。デフォルトのマネージドセキュリティグループには次のルールがあり、カスタムマネージドセキュリティグループを指定すると、Amazon EMR により同じルールが追加されます。

タイプ	プロトコル	ポート範囲	ソース	詳細
インバウンドルール				
すべての ICMP-IPV4	すべて	該当なし	コアインスタンスとタスクインスタンスのマネージドセキュリティグループのグループ ID。言い換えると、ルールが表示される同じセキュリティグループです。	反射的なルールにより、指定されたセキュリティグループに関連付けられた任意のインスタンスからのインバウンドトラフィックが許可されます。複数のクラスターに対してデフォルトの ElasticMapReduce-core を使用することにより、クラスターのコアインスタンスとタスクインスタンスは、どの ICMP、TCP、UDP ポートでも相互に通信できます。カスタムのマネージドセキュリティグループを指定して、クラスター間のアクセスを制限します。
すべての TCP	TCP	すべて		
すべての UDP	UDP	すべて		
すべての ICMP-IPV4	すべて	該当なし	プライマリインスタンスのマネージドセキュリティグループのグループ ID。	これらのルールでは、インスタンスが異なるクラスターにある場合でも、すべてのインバウンド ICMP トラフィックと、指定されたセキュリティグループに関連付けられたすべてのプライマリインスタンスからの TCP または UDP ポート経由のトラフィックが許可されます。
すべての TCP	TCP	すべて		
すべての UDP	UDP	すべて		

プライマリインスタンスの Amazon EMR マネージドセキュリティグループ (プライベートサブネット)

プライベートサブネット内のプライマリインスタンスのデフォルトのマネージドセキュリティグループには、グループ名 -ElasticMapReducePrimary-Private があります。デフォルトのマネージドセキュリティグループには次のルールがあり、カスタムマネージドセキュリティグループを指定すると、Amazon EMR により同じルールが追加されます。

タイプ	プロトコル	ポート範囲	ソース	詳細
インバウンドルール				
すべての ICMP-IPv4	すべて	該当なし	プライマリインスタンスのマネージドセキュリティグループのグループ ID。言い換えると、ルールが表示される同じセキュリティグループです。	反射的なルールにより、指定されたセキュリティグループに関連付けられた任意のインスタンスからのインバウンドトラフィックが許可され、プライベートサブネット内からアクセス可能です。複数のクラスターに対してデフォルトの ElasticMapReduce-Primary-Private を使用することにより、クラスターのコアノードとタスクノードは、どの ICMP、TCP、UDP ポートでも相互に通信できます。カスタムのマネージドセキュリティグループを指定して、クラスター間のアクセスを制限します。
すべての TCP	TCP	すべて		
すべての UDP	UDP	すべて		
すべての ICMP-IPV4	すべて	該当なし	コアノードとタスクノードのマネージドセキュリティグループのグループ ID。	これらのルールでは、インスタンスが異なるクラスターにある場合でも、すべてのインバウンド ICMP トラフィックと、指定されたセキュリティグループに関連付けられたすべてのコアインスタンスとタスクインスタンスからの TCP、UDP ポート経由のトラフィックが許可され、プライベートサブネット内からアクセス可能です。
すべての TCP	TCP	すべて		
すべての UDP	UDP	すべて		

タイプ	プロトコル	ポート範囲	ソース	詳細
HTTPS (8443)	TCP	8443	プライベートサブネット内のサービスアクセスのマネージドセキュリティグループのグループ ID。	このルールにより、クラスターマネージャーはプライマリノードと通信できます。
アウトバウンドルール				
すべてのトラフィック	すべて	すべて	0.0.0.0/0	インターネットへのアウトバウンドアクセスを提供する。
カスタム TCP	TCP	9443	プライベートサブネット内のサービスアクセスのマネージドセキュリティグループのグループ ID。	上記の「すべてのトラフィック」のデフォルトのアウトバウンドルールを削除した場合、このルールが Amazon EMR 5.30.0 以降の最小要件になります。 **Note** カスタムマネージドセキュリティグループを使用する場合、Amazon EMR はこのルールを追加しません。

タイプ	プロトコル	ポート範囲	ソース	詳細
カスタム TCP	TCP	80 (http) または 443 (https)	プライベートサブネット内のサービスアクセスのマネージドセキュリティグループのグループ ID。	上記の「すべてのトラフィック」のデフォルトのアウトバウンドルールを削除した場合、このルールが Amazon EMR 5.30.0 以降で HTTPS 経由で Amazon S3 に接続するための最小要件になります。 **Note** カスタムマネージドセキュリティグループを使用する場合、Amazon EMR はこのルールを追加しません。

コアインスタンスとタスクインスタンスの Amazon EMR マネージドセキュリティグループ (プライベートサブネット)

プライベートサブネットのコアインスタンスとタスクインスタンスのデフォルトのマネージドセキュリティグループのグループ名は ElasticMapReduce-Core-Private です。デフォルトのマネージドセキュリティグループには次のルールがあり、カスタムマネージドセキュリティグループを指定すると、Amazon EMR により同じルールが追加されます。

タイプ	プロトコル	ポート範囲	ソース	詳細
インバウンドルール				
すべての ICMP-IPV4	すべて	該当なし	コアインスタンスとタスクインスタンスのマネージドセキュリティグループのグループ ID。言い換える	反射的なルールにより、指定されたセキュリティグループに関連付けられた任意のインスタンスからのインバウンドトラフィックが許可されます。複数のクラスターに対してデフォルトの `ElasticMapReduce-core` を使用することにより、クラスターのコアイ
すべての TCP	TCP	すべて		

タイプ	プロトコル	ポート範囲	ソース	詳細
すべての UDP	UDP	すべて	と、ルールが表示される同じセキュリティグループです。	ンスタンスとタスクインスタンスは、どの ICMP、TCP、UDP ポートでも相互に通信できます。カスタムのマネージドセキュリティグループを指定して、クラスター間のアクセスを制限します。
すべての ICMP-IPV4	すべて	該当なし	プライマリインスタンスのマネージドセキュリティグループのグループ ID。	これらのルールでは、インスタンスが異なるクラスターにある場合でも、すべてのインバウンド ICMP トラフィックと、指定されたセキュリティグループに関連付けられたすべてのプライマリインスタンスからの TCP または UDP ポート経由のトラフィックが許可されます。
すべての TCP	TCP	すべて		
すべての UDP	UDP	すべて		
HTTPS (8443)	TCP	8443	プライベートサブネット内のサービスアクセスのマネージドセキュリティグループのグループ ID。	このルールにより、クラスターマネージャはコアノードおよびタスクノードと通信できます。
アウトバウンドルール				
すべてのトラフィック	すべて	すべて	0.0.0.0/0	以下の「アウトバウンドルールの編集」を参照してください。

タイプ	プロトコル	ポート範囲	ソース	詳細
カスタム TCP	TCP	80 (http) または 443 (https)	プライベートサブネット内のサービスアクセスのマネージドセキュリティグループのグループ ID。	上記の「すべてのトラフィック」のデフォルトのアウトバウンドルールを削除した場合、このルールが Amazon EMR 5.30.0 以降で HTTPS 経由で Amazon S3 に接続するための最小要件になります。 **Note** カスタムマネージドセキュリティグループを使用する場合、Amazon EMR はこのルールを追加しません。

アウトバウンドルールの編集

デフォルトでは、Amazon EMR は、すべてのプロトコルとポートですべてのアウトバウンドトラフィックを許可するアウトバウンドルールを使用して、このセキュリティグループを作成します。Amazon EMR クラスターで実行できるさまざまな Amazon EMR およびカスタマーアプリケーションには、異なる送信ルールが必要になる場合があるため、すべてのアウトバウンドトラフィックの許可が選択されています。Amazon EMR は、デフォルトのセキュリティグループを作成するときに、これらの特定の設定を予測できません。セキュリティグループの送信の範囲を制限して、ユースケースとセキュリティポリシーに合ったルールだけを含めることができます。少なくとも、このセキュリティグループには次のアウトバウンドルールが必要ですが、一部のアプリケーションでは追加の送信ルールが必要になる場合があります。

タイプ	プロトコル	ポート範囲	デスティネーション	詳細
すべての TCP	TCP	すべて	pl-*xxxxxxxx*	マネージド Amazon S3 プレフィクスリスト `com.amazonaws.` *`MyRegion`*`.s3`。

タイプ	プロトコル	ポート範囲	デスティネーション	詳細
すべてのトラフィック	すべて	すべて	sg-*xxxxxxxxx xxxxxxxx*	ElasticMapReduce-Core-Private セキュリティグループの ID。
すべてのトラフィック	すべて	すべて	sg-*xxxxxxxxx xxxxxxxx*	ElasticMapReduce-Primary-Private セキュリティグループの ID。
カスタム TCP	TCP	9443	sg-*xxxxxxxxx xxxxxxxx*	ElasticMapReduce-ServiceAccess セキュリティグループの ID。

サービスアクセスの Amazon EMR マネージドセキュリティグループ (プライベートサブネット)

プライベートサブネットのサービスアクセスのデフォルトのマネージドセキュリティグループのグループ名は ElasticMapReduce-ServiceAccess です。インバウンドルールと、プライベートサブネット内の他のマネージドセキュリティグループへの HTTPS (ポート 8443、ポート 9443) 経由のトラフィックを許可するアウトバウンドルールがあります。これらのルールにより、クラスターマネージャーはプライマリノードとも、コアノードおよびタスクノードとも通信できます。カスタムセキュリティグループを使用している場合は、同じルールが必要です。

タイプ	プロトコル	ポート範囲	ソース	詳細
インバウンドルール (Amazon EMR リリース 5.30.0 以降の Amazon EMR クラスターに必要)				
カスタム TCP	TCP	9443	プライマリインスタンスのマネージドセキュリティグループのグループ ID。	このルールにより、プライマリインスタンスのセキュリティグループとサービスアクセスセキュリティグループの間で通信が可能になります。

タイプ	プロトコル	ポート範囲	ソース	詳細
アウトバウンドルール (すべての Amazon EMR クラスターに必要)				
カスタム TCP	TCP	8443	プライマリインスタンスのマネージドセキュリティグループのグループ ID。	これらのルールにより、クラスターマネージャーはプライマリノードとも、コアノードおよびタスクノードとも通信できます。
カスタム TCP	TCP	8443	コアインスタンスとタスクインスタンスのマネージドセキュリティグループのグループ ID。	これらのルールにより、クラスターマネージャーはプライマリノードとも、コアノードおよびタスクノードとも通信できます。

追加のセキュリティグループを使用する

デフォルトのマネージドセキュリティグループを使用する場合でも、カスタムのマネージドセキュリティグループを指定する場合でも、追加のセキュリティグループを使用できます。追加セキュリティグループでは、異なるクラスター間のアクセスや外部クライアント、リソース、アプリケーションからのアクセスを柔軟に調整できます。

たとえば、次のシナリオを考えてみます。相互に通信する必要がある複数のクラスターがありますが、クラスターの特定のサブセットのみに対してプライマリインスタンスへのインバウンド SSH アクセスを許可する必要があります。これを行うには、クラスターに対して同じマネージドセキュリティグループのセットを使用します。次に、信頼済みクライアントからのインバウンド SSH アクセスを許可する追加セキュリティグループを作成し、サブセット内で各クラスターにプライマリインスタンスの追加セキュリティグループを指定します。

プライマリインスタンスには最大 15 個の追加セキュリティグループを適用でき、コアインスタンスとタスクインスタンスには 15 個、サービスアクセスには 15 個 (プライベートサブネット内) を追加できます。必要な場合は、プライマリインスタンス、コアインスタンスとタスクインスタンス、サービスアクセスに対して同じ追加セキュリティグループを指定することができます。アカウント内の

セキュリティグループとルールの最大数はアカウント制限に従います。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制限」を参照してください。

Amazon EMR マネージドセキュリティグループと追加セキュリティグループを指定する

セキュリティグループは、 AWS Management Console、、 AWS CLIまたは Amazon EMR API を使用して指定できます。セキュリティグループを指定しない場合、Amazon EMR によってデフォルトのセキュリティグループが作成されます。追加セキュリティグループの指定はオプションです。プライマリインスタンス、コアインスタンスとタスクインスタンス、サービスアクセス (プライベートサブネットのみ) に対して追加セキュリティグループを割り当てることができます。

New console

> **Note**
>
> Amazon EMR コンソールは、再設計され、使いやすくなりました。古いコンソールと新しいコンソールエクスペリエンスの違いについては、「Amazon EMR コンソール」を参照してください。

新しいコンソールを使用してセキュリティグループを指定する方法

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で、[クラスター] を選択し、[クラスターの作成] を選択します
3. [ネットワーク] の下で、[EC2 セキュリティグループ (ファイアウォール)] の横にある矢印を選択してこのセクションを展開します。[プライマリノード] と [コアノードとタスクノード] の下で、デフォルトの Amazon EMR マネージドセキュリティグループがデフォルトで選択されます。プライベートサブネットを使用する場合は、サービスアクセス用のセキュリティグループを選択するオプションもあります。
4. Amazon EMR マネージドセキュリティグループを変更するには、[セキュリティグループの選択] ドロップダウンメニューを使用して、[Amazon EMR マネージドセキュリティグループ] オプションリストから別のオプションを選択します。[プライマリノード] と [コアノードとタスクノード] の両方に 1 つの Amazon EMR マネージドセキュリティグループがあります。

5. カスタムセキュリティグループを追加するには、同じ [セキュリティグループの選択] ドロップダウンメニューを使用して、[カスタムセキュリティグループ] オプションリストから最大 4 つのカスタムセキュリティグループを選択します。[プライマリノード] と [コアノードとタスクノード] の両方に最大 4 つのカスタムセキュリティグループを設定できます。
6. クラスターに適用するその他のオプションを選択します。
7. クラスターを起動するには、[クラスターの作成] を選択します。

Old console

古いコンソールを使用してセキュリティグループを指定する方法

1. 新しい Amazon EMR コンソールに移動し、サイドナビゲーションから [古いコンソールに切り替え] を選択します。古いコンソールに切り替えたときの動作の詳細については、「Using the old console」を参照してください。
2. [Create cluster (クラスターの作成)]、[Go to advanced options (詳細オプションに移動する)] の順に選択します。
3. [Step 4:Security (ステップ 4: セキュリティ)] が表示されるまでクラスターのオプションを選択します。
4. [EC2 Security Groups (EC2 セキュリティグループ)] を選択してセクションを展開します。

   [EMR managed security groups (EMR マネージドセキュリティグループ)] では、デフォルトのマネージドセキュリティグループがデフォルトで選択されています。VPC で [Master] (マスター)、[Core & Task] (コアおよびタスク)、[Service Access] (サービスアクセス) (プライベートサブネットのみ) にデフォルトが存在しない場合、関連するセキュリティグループ名の前に [Create] (作成) が表示されます。
5. カスタムのマネージドセキュリティグループを使用する場合、[EMR managed security groups (EMR マネージドセキュリティグループ)] リストから選択します。

   カスタムのマネージドセキュリティグループを選択する場合、他のインスタンスのカスタムセキュリティグループを選択することを通知するメッセージが表示されます。クラスターにはカスタムのみ、またはデフォルトのみのマネージドセキュリティグループを使用できます。
6. 必要に応じて、[Additional security groups (追加セキュリティグループ)] で鉛筆アイコンを選択し、リストから最大 4 つまでのセキュリティグループを選択して、[Assign security groups (セキュリティグループの割り当て)] を選択します。必要に応じて、[Master (マスター)]、

[Core & Task (コアおよびタスク)]、[Service Access (サービスアクセス)] それぞれで繰り返します。

7. [クラスターの作成] を選択します。

AWS CLIを使用してセキュリティグループを指定する

を使用してセキュリティグループを指定するには AWS CLI 、 `--ec2-attributes`オプションの以下のパラメータを指定して `create-cluster` コマンドを使用します。

パラメータ	説明
`EmrManagedPrimarySecurityGroup`	このパラメータを使用して、プライマリインスタンスのカスタムマネージドセキュリティグループを指定します。このパラメータを指定する場合は、`EmrManagedCoreSecurityGroup` も指定する必要があります。プライベートサブネットのクラスターの場合は、`ServiceAccessSecurityGroup` も指定する必要があります。
`EmrManagedCoreSecurityGroup`	このパラメータを使用して、コアインスタンスとタスクインスタンスのカスタムマネージドセキュリティグループを指定します。このパラメータを指定する場合は、`EmrManagedPrimarySecurityGroup` も指定する必要があります。プライベートサブネットのクラスターの場合は、`ServiceAccessSecurityGroup` も指定する必要があります。
`ServiceAccessSecurityGroup`	このパラメータを使用して、サービスアクセスのカスタムマネージドセキュリティグループを指定します。これは、プライベートサブネットのクラスターにのみ適用されます。`ServiceAccessSecurityGroup` として指定するセキュリティグループは、他の目的

パラメータ	説明
	に使用してはならず、また、Amazon EMR 用に予約する必要があります。このパラメータを指定する場合は、EmrManagedPrimarySecurityGroup も指定する必要があります。
AdditionalPrimarySecurityGroups	このパラメータを使用して、プライマリインスタンスに最大 4 つの追加セキュリティグループを指定します。
AdditionalCoreSecurityGroups	このパラメータを使用して、コアインスタンスとタスクインスタンスに最大 4 つの追加セキュリティグループを指定します。

Example - カスタム Amazon EMR マネージドセキュリティグループと追加セキュリティグループを指定する

次の例では、プライベートサブネット内のクラスターに対してカスタム Amazon EMR マネージドセキュリティグループを指定し、プライマリインスタンスに対して複数の追加セキュリティグループを指定し、コアインスタンスとタスクインスタンスに対して 1 つの追加セキュリティグループを指定します。

> **Note**
>
> 読みやすくするために、Linux 行連続文字 (\) が含まれています。Linux コマンドでは、これらは削除することも、使用することもできます。Windows の場合、削除するか、キャレット (^) に置き換えてください。

```
aws emr create-cluster --name "ClusterCustomManagedAndAdditionalSGs" \
--release-label emr-emr-7.1.0 --applications Name=Hue Name=Hive \
Name=Pig --use-default-roles --ec2-attributes \
SubnetIds=subnet-xxxxxxxxxxxx,KeyName=myKey,\
ServiceAccessSecurityGroup=sg-xxxxxxxxxxxx,\
EmrManagedPrimarySecurityGroup=sg-xxxxxxxxxxxx,\
```

```
EmrManagedCoreSecurityGroup=sg-xxxxxxxxxxx,\
AdditionalPrimarySecurityGroups=['sg-xxxxxxxxxxx',\
'sg-xxxxxxxxxxx','sg-xxxxxxxxxx'],\
AdditionalCoreSecurityGroups=sg-xxxxxxxxxxx \
--instance-type m5.xlarge
```

詳細については、「AWS CLI コマンドリファレンス」の「create-cluster」を参照してください。

EMR Notebooks の EC2 セキュリティグループの指定

EMR ノートブックを作成する場合、ノートブック エディタの使用時には EMR ノートブックと Amazon EMR クラスターの間のネットワークトラフィックを制御するために 2 つのセキュリティグループが使用されます。デフォルトのセキュリティグループには、ノートブックがアタッチされる、EMR Notebooks サービスとクラスター間のネットワークトラフィックのみを許可する最小限のルールがあります。

EMR ノートブックは、TCP ポート 18888 を使用してプロキシ経由でクラスターと通信するために Apache Livy を使用します。環境に合わせて調整されたルールを使用してカスタムセキュリティグループを作成すると、ノートブックのサブセットのみが特定のクラスター上のノートブック エディタ内でコードを実行できるように、ネットワークトラフィックを制限できます。クラスターは、クラスターのデフォルトのセキュリティグループに加えて、カスタムセキュリティを使用します。詳細については、「Amazon EMR 管理ガイド」の「セキュリティグループを使用してネットワークトラフィックを制御する」および「EMR Notebooks の EC2 セキュリティグループの指定」を参照してください。

プライマリインスタンスのデフォルトの EC2 セキュリティグループ

プライマリインスタンスのデフォルトの EC2 セキュリティグループは、プライマリインスタンスのクラスターのセキュリティグループに加えて、プライマリインスタンスに関連付けられます。

グループ名: ElasticMapReduceEditors-Livy

ルール

- インバウンド

  EMR Notebooks のデフォルトの EC2 セキュリティグループのすべてのリソースから TCP ポート 18888 を許可する
- アウトバウンド

なし

EMR Notebooks のデフォルトの EC2 セキュリティグループ

EMR ノートブックのデフォルトの EC2 セキュリティグループは、割り当てられているすべての EMR ノートブックのノートブック工ディタに関連付けられます。

グループ名: ElasticMapReduceEditors-Editor

ルール

- インバウンド

  なし
- アウトバウンド

  EMR Notebooks のデフォルトの EC2 セキュリティグループのすべてのリソースへの TCP ポート 18888 を許可する

ノートブックを Git リポジトリに関連付ける場合の EMR Notebooks のカスタム EC2 セキュリティグループ

Git リポジトリをノートブックにリンクするには、EMR ノートブックのセキュリティグループに、ノートブックがインターネットにトラフィックをルーティングできるようにするアウトバウンドルールが含まれている必要があります。このためには、新しいセキュリティグループを作成することをお勧めします。デフォルトの ElasticMapReduceEditors-Editor セキュリティグループを更新すると、このセキュリティグループにアタッチされている他のノートブックにも同じアウトバウンドルールが付与される場合があります。

ルール

- インバウンド

  なし
- アウトバウンド

  以下の例に示すように、クラスター経由でインターネットにトラフィックをルーティングすることをノートブックに許可します。値 0.0.0.0/0 は例の目的に使用されています。このルールを変更して、Git ベースのリポジトリの IP アドレスを指定できます。

タイプ	プロトコル	ポート範囲	デスティネーション
カスタム TCP ルール	TCP	18888	SG-
HTTPS	TCP	443	0.0.0.0/0

Amazon EMR のパブリックアクセスブロックの使用

Amazon EMR のブロックパブリックアクセス (BPA) は、クラスターのセキュリティ設定でポートのパブリック IP アドレスからのインバウンドトラフィックが許可されている場合に、ユーザーがパブリックサブネットでクラスターを起動するのを防止します。

> **Important**
>
> ブロックパブリックアクセスはデフォルトで有効になっています。アカウントの保護を強化するには、これを有効のままにしておくことが推奨されます。

ブロックパブリックアクセスについて

ブロックパブリックアクセスのアカウントレベル設定を使用して、Amazon EMR クラスターへのパブリックネットワークアクセスを一元管理することができます。

のユーザーがクラスター AWS アカウント を起動すると、Amazon EMR はクラスターのセキュリティグループ内のポートルールをチェックし、インバウンドトラフィックルールと比較します。セキュリティグループに、パブリック IP アドレス IPv4 0.0.0.0/0 または IPv6 ::/0 に対してポートを開くインバウンドルールがあり、それらのポートがアカウントで適切に指定されていない場合、Amazon EMR はユーザーにクラスターの作成を許可しません。

ユーザーがパブリックサブネットで実行中のクラスターのセキュリティグループルールを変更して、アカウントの BPA 設定に違反するパブリックアクセスルールを設定した場合、Amazon EMR は新しいルールを取り消します (それを行うアクセス許可がある場合)。Amazon EMR にルールを取り消すアクセス許可がない場合は、 AWS Health ダッシュボードで違反を説明するイベントを作成します。Amazon EMR にルール取り消しのアクセス許可を付与するには、「セキュリティグループのルールを取り消すために Amazon EMR を設定する」を参照してください。

ご使用の AWS アカウントのすべての AWS リージョン にあるすべてのクラスターに対して、ブロックパブリックアクセスはデフォルトで有効になっています。BPA はクラスターのライフサイクル全体に適用されますが、プライベートサブネットで作成したクラスターには適用されません。BPA ルールには例外を設定できます。ポート 22 はデフォルトでは例外です。例外の設定の詳細については、「パブリックアクセスブロックの設定」を参照してください。

パブリックアクセスブロックの設定

アカウントのセキュリティグループとパブリックアクセスブロック設定は、いつでも更新できます。

、 ()、および Amazon EMR API を使用して AWS Management Console、パブリックアクセスブロック AWS Command Line Interface (BPA AWS CLI) 設定のオンとオフを切り替えることができます。設定はリージョンごとにアカウント全体に適用されます。クラスターセキュリティを維持するために、BPA を使用することが推奨されます。

New console

> **Note**
>
> Amazon EMR コンソールは、再設計され、使いやすくなりました。古いコンソールと新しいコンソールエクスペリエンスの違いについては、「Amazon EMR コンソール」を参照してください。

新しいコンソールでブロックパブリックアクセスを設定する方法

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 上部のナビゲーションバーで、設定するリージョンを選択します (まだ選択されていない場合)。
3. 左側のナビゲーションペインの [EMR on EC2] で、[ブロックパブリックアクセス] を選択します。
4. [Block public access settings (パブリックアクセスブロック設定)] で、以下の手順を実行します。

実行方法	手順

実行方法	手順
パブリックアクセスブロックをオンまたはオフにする	[編集] を選択し、必要に応じて [有効にする] または [無効にする] を選択し、[保存] を選択します。
例外のリスト内のポートを編集する	1. [編集] を選択し、[ポート範囲の例外] セクションを見つけます。 2. 例外のリストにポートを追加するには、[Add a port range (ポート範囲の追加)] を選択し、新しいポートまたはポート範囲を入力します。追加するポートまたはポート範囲ごとにこの操作を繰り返します。 3. ポートまたはポート範囲を削除するには、ポート範囲のリストで、当該エントリの横にある [削除] を選択します。 4. [保存] を選択します。

Old console

古いコンソールでブロックパブリックアクセスを設定する方法

1. https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 上部のナビゲーションバーで、設定するリージョンが選択されていることを確認します。
3. [Block public access (パブリックアクセスブロック)] を選択します。
4. [Block public access settings (パブリックアクセスブロック設定)] で、以下の手順を実行します。

実行方法	手順
パブリックアクセスブロックをオンまたはオフにする	[変更] を選択し、必要に応じて [オン] または [オフ] を選択してから、チェックマークを選択して確定します。
例外のリスト内のポートを編集する	1. [例外] で、[Edit (編集)] を選択します。 2. 例外のリストにポートを追加するには、[Add a port range (ポート範囲の追加)] を選択し、新しいポートまたはポート範囲を入力します。追加するポートまたはポート範囲ごとにこの操作を繰り返します。 3. ポートまたはポート範囲を削除するには、[ポート範囲] リストのその エントリの横にある [x] を選択します。 4. [変更の保存] をクリックします。

AWS CLI

を使用してブロックパブリックアクセスを設定するには AWS CLI

- 以下の例に示すように、`aws emr put-block-public-access-configuration` コマンドを使用して、パブリックアクセスブロックを設定します。

実行方法	手順
パブリックアクセスブロックをオンにする	以下の例に示すように、BlockPublicSecurityGroupRules を true に設定します。クラスターを起動するには、クラスターに関連付けられているセキュリティグループのインバウンドルールで、パブリックアクセスを許可できません。 `aws emr put-block-public-access-configuration --block-public-access-configuration BlockPublicSecurityGroupRules=true`
パブリックアクセスブロックをオフにする	以下の例に示すように、BlockPublicSecurityGroupRules を false に設定します。クラスターに関連付けられているセキュリティグループのインバウンドルールで、任意のポートでのパブリックアクセスを許可できます。この設定はお勧めしません。 `aws emr put-block-public-access-configuration --block-public-access-configuration BlockPublicSecurityGroupRules=false`

実行方法	手順
パブリックアクセスブロックをオンにし、例外とするポートを指定する	以下の例では、パブリックアクセスブロックを有効にし、例外としてポート 22 とポート 100 ～ 101 を指定しています。これにより、関連付けられているセキュリティグループのインバウンドルールで、ポート 22、ポート 100、またはポート 101 でのパブリックアクセスを許可している場合は、クラスターを作成できます。 `aws emr put-block-public-access-configuration --block-public-access-configuration '{ "BlockPublicSecurityGroupRules": true, "PermittedPublicSecurityGroupRuleRanges": [ { "MinRange": 22, "MaxRange": 22 }, { "MinRange": 100, "MaxRange": 101 } ] }'`

セキュリティグループのルールを取り消すために Amazon EMR を設定する

Amazon EMR には、セキュリティグループのルールを取り消し、ブロックパブリックアクセス設定に準拠するためのアクセス許可が必要です。Amazon EMR に必要なアクセス許可を付与するには、次のいずれかの方法を使用できます。

- (推奨) `AmazonEMRServicePolicy_v2` 管理ポリシーをサービスロールにアタッチします。詳細については、「Amazon EMR のサービスロール (EMR ロール)」を参照してください。
- セキュリティグループに対する `ec2:RevokeSecurityGroupIngress` アクションを許可する新しいインラインポリシーを作成します。ロールアクセス許可ポリシーの変更方法の詳細については、「IAM ユーザーガイド」の IAM コンソール、AWS API、および AWS CLI を使用したロールアクセス許可ポリシーの変更を参照してください。

ブロックパブリックアクセス違反の解決

ブロックパブリックアクセス違反が発生した場合は、以下のいずれかのアクションを使用して軽減することができます。

- クラスター上のウェブインターフェイスにアクセスする場合は、「Amazon EMR クラスターでホストされているウェブインターフェイスを表示する」で説明されているオプションのいずれかを使用して SSH (ポート 22) 経由でインターフェイスにアクセスします。
- パブリック IP アドレスからではなく、特定の IP アドレスからクラスターへのトラフィックを許可するには、セキュリティグループルールを追加します。詳細については、「Amazon EC2 入門ガイド」の「セキュリティグループへのルールの追加」を参照してください。
- (非推奨) Amazon EMR BPA 例外を設定して、目的のポートまたはポート範囲を含めることができます。BPA 例外を指定すると、保護されていないポートにリスクが生じます。例外を指定する予定がある場合は、不要になった例外は即時に削除してください。詳細については、「パブリックアクセスブロックの設定」を参照してください。

セキュリティグループのルールに関連付けられたクラスターを特定する

特定のセキュリティグループルールに関連付けられているすべてのクラスターを特定したり、特定のクラスターのセキュリティグループルールを確認したりする必要がある場合があります。

- セキュリティグループがわかっている場合は、そのセキュリティグループのネットワークインターフェイスを見つければ、関連付けられているクラスターを特定できます。詳細については、「 AWS re:Post」で「How can I find the resources associated with an Amazon EC2 security group?」を参照してください。これらのネットワークインターフェイスにアタッチされている Amazon EC2 インスタンスには、そのインスタンスが属するクラスターの ID がタグ付けされます。
- 既知のクラスターのセキュリティグループを確認するには、「クラスターステータスと詳細の表示」のステップを実行します。クラスターのセキュリティグループは、コンソールの [ネットワークとセキュリティ] パネル、または AWS CLIの `Ec2InstanceAttributes` フィールドで確認できます。

Amazon EMR のコンプライアンス検証

サードパーティーの監査者は、複数の コンプライアンスプログラムの一環として Amazon EMR のセキュリティと AWS コンプライアンスを評価します。これらのプログラムには、SOC、PCI、FedRAMP、HIPAA などがあります。

特定のコンプライアンスプログラム AWS の対象となるサービスのリストについては、「コンプライアンスAWS プログラムによる対象範囲内の のサービス」を参照してください。一般的な情報については、「AWS コンプライアンスプログラム」を参照してください。

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、「 でのレポートのダウンロード AWS Artifact」を参照してください。

Amazon EMR を使用する際のお客様のコンプライアンス責任は、お客様のデータの機密性や貴社のコンプライアンス目的、適用可能な法律および規制によって決定されます。Amazon EMR の使用が HIPAA、PCI、FedRAMP などの標準に準拠していることを前提としている場合、 は以下に役立つリソース AWS を提供します。

- セキュリティとコンプライアンスのクイックスタートガイド – これらのデプロイガイドでは、アーキテクチャ上の考慮事項について説明し、セキュリティとコンプライアンスに重点を置いたベースライン環境を にデプロイする手順について説明します AWS。
- 「HIPAA セキュリティとコンプライアンスのアーキテクチャ」ホワイトペーパー – このホワイトペーパーでは、企業が AWS を使用して HIPAA 準拠のアプリケーションを作成する方法について説明します。
- AWS コンプライアンスリソース — このワークブックとガイドのコレクションは、お客様の業界や地域に適用される場合があります。
- AWS Config – この AWS サービスは、リソース設定が社内プラクティス、業界ガイドライン、および規制にどの程度準拠しているかを評価します。
- AWS Security Hub – この AWS サービスは、 内のセキュリティ状態を包括的に把握 AWS し、セキュリティ業界標準とベストプラクティスへの準拠を確認するのに役立ちます。

Amazon EMR の耐障害性

AWS グローバルインフラストラクチャは、 AWS リージョンとアベイラビリティーゾーンを中心に構築されています。 AWS リージョンは、低レイテンシー、高スループット、および高度に冗長なネットワークで接続された、物理的に分離された複数のアベイラビリティーゾーンを提供します。

アベイラビリティーゾーンでは、アベイラビリティーゾーン間で中断せずに、自動的にフェイルオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性、耐障害性、およびスケーラビリティが優れています。

AWS リージョンとアベイラビリティーゾーンの詳細については、AWS 「 グローバルインフラストラクチャ」を参照してください。

Amazon EMR は、 AWS グローバルインフラストラクチャに加えて、データの耐障害性とバックアップのニーズをサポートするのに役立ついくつかの機能を提供しています。

- EMRFS を利用した Amazon S3 との統合
- 複数のマスターノードのサポート

Amazon EMR でのインフラストラクチャセキュリティ

マネージドサービスである Amazon EMR 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 AWS 環境を設計するには、「セキュリティの柱 AWS Well-Architected Framework」の「Infrastructure Protection」を参照してください。

が AWS 公開した API コールを使用して、ネットワーク経由で Amazon EMR 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 は必須で TLS 1.3 がお勧めです。
- 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。

また、リクエストには、アクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または、AWS Security Token Service (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

トピック

- インターフェイス VPC エンドポイントを使用して Amazon EMR に接続する

インターフェイス VPC エンドポイントを使用して Amazon EMR に接続する

インターネット経由で接続する代わりに、Virtual Private Cloud (VPCAWS PrivateLink) のインターフェイス VPC エンドポイント () を使用して Amazon EMR に直接接続できます。インターフェイス VPC エンドポイントを使用する場合、VPC と Amazon EMR 間の通信は、完全に AWS ネットワーク内で行われます。各 VPC エンドポイントは、VPC サブネット内のプライベート IP アドレスを持つ 1 つ以上の Elastic Network Interface (ENI) で表されます。

インターフェイス VPC エンドポイントは、インターネットゲートウェイ、NAT デバイス、VPN 接続、または AWS Direct Connect 接続なしで VPC を Amazon EMR に直接接続します。VPC のインスタンスは、パブリック IP アドレスがなくても Amazon EMR API と通信できます。

VPC 経由で Amazon EMR を使用するには、VPC 内にあるインスタンスから接続するか、Amazon Virtual Private Network (VPN) または AWS Direct Connectを使用してプライベートネットワークを VPC に接続する必要があります。Amazon VPN については、「Amazon Virtual Private Cloud ユーザーガイド」の「VPN 接続」を参照してください。の詳細については AWS Direct Connect、「 ユーザーガイド」の「接続の作成AWS Direct Connect 」を参照してください。

コンソール AWS または AWS Command Line Interface （ AWS CLI) コマンドを使用して、インターフェイス VPC エンドポイントを作成して Amazon EMR に接続できます。詳細については、「インターフェイスエンドポイントの作成」を参照してください。

インターフェイス VPC エンドポイントを作成した後、エンドポイントのプライベート DNS ホスト名を有効にすると、デフォルトの Amazon EMR エンドポイントはお客様の VPC エンドポイントに解決されます。Amazon EMR のデフォルトのサービス名エンドポイントは、次の形式です。

```
elasticmapreduce.Region.amazonaws.com
```

プライベート DNS ホスト名を有効にしない場合は、Amazon VPC が以下の形式で使用できる DNS エンドポイント名を提供します。

```
VPC_Endpoint_ID.elasticmapreduce.Region.vpce.amazonaws.com
```

詳細については、「Amazon VPC 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。

Amazon EMR は、VPC 内のすべての API アクションへの呼び出しをサポートしています。

VPC エンドポイントポリシーを VPC エンドポイントにアタッチして、IAM プリンシパルのアクセスを制御できます。また、セキュリティグループを VPC エンドポイントに関連付けて、ネットワークトラフィックの送信元と送信先 (IP アドレスの範囲など) に基づいてインバウンドとアウトバウンドのアクセスを制御することもできます。詳細については、「VPC エンドポイントによるサービスのアクセスコントロール」を参照してください。

Amazon EMR の VPC エンドポイントポリシーの作成

Amazon EMR の Amazon VPC エンドポイントに対するポリシーを作成して、以下を指定することができます。

- アクションを実行できるプリンシパルまたは実行できないプリンシパル
- 実行可能なアクション
- アクションを実行できるリソース

詳細については、「Amazon VPC ユーザーガイド」の「VPC エンドポイントによるサービスのアクセスコントロール」を参照してください。

Example – 指定された AWS アカウントからのすべてのアクセスを拒否する VPC エンドポイントポリシー

次の VPC エンドポイントポリシーは、 AWS アカウント *123456789012* がエンドポイントを使用して リソースへのすべてのアクセスを拒否します。

```
{
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        },
        {
            "Action": "*",
            "Effect": "Deny",
            "Resource": "*",
            "Principal": {
                "AWS": [
                    "123456789012"
                ]
```

```
            }
        }
    ]
}
```

Example - 指定した IAM プリンシパル (ユーザー) への VPC アクセスのみを許可する VPC エンドポイントポリシー

次の VPC エンドポイントポリシーは、 AWS アカウント *123456789012* のユーザー *lijuan* へのフルアクセスのみを許可します。他のすべての IAM プリンシパルは、エンドポイントを使用したアクセスを拒否されます。

```
{
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::123456789012:user/lijuan"
                ]
            }
        }]
}
```

Example - 読み取り専用の EMR オペレーションを許可する VPC エンドポイントポリシー

次の VPC エンドポイントポリシーでは、 AWS アカウント *123456789012* のみが指定された Amazon EMR アクションを実行できます。

指定されたアクションは、Amazon EMR の読み取り専用アクセスに相当します。指定されたアカウントでは、VPC 上の他のすべてのアクションが拒否されます。他のすべてのアカウントは、すべてのアクセスを拒否されます。Amazon EMR アクションのリストについては、「Amazon EMR 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。

```
{
    "Statement": [
        {
            "Action": [
                "elasticmapreduce:DescribeSecurityConfiguration",
```

```
                "elasticmapreduce:GetBlockPublicAccessConfiguration",
                "elasticmapreduce:ListBootstrapActions",
                "elasticmapreduce:ViewEventsFromAllClustersInConsole",
                "elasticmapreduce:ListSteps",
                "elasticmapreduce:ListInstanceFleets",
                "elasticmapreduce:DescribeCluster",
                "elasticmapreduce:ListInstanceGroups",
                "elasticmapreduce:DescribeStep",
                "elasticmapreduce:ListInstances",
                "elasticmapreduce:ListSecurityConfigurations",
                "elasticmapreduce:DescribeEditor",
                "elasticmapreduce:ListClusters",
                "elasticmapreduce:ListEditors"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Principal": {
                "AWS": [
                    "123456789012"
                ]
            }
        }
    ]
}
```

Example - 指定したクラスターへのアクセスを拒否する VPC エンドポイントポリシー

次の VPC エンドポイントポリシーでは、すべてのアカウントとプリンシパルにフルアクセスを許可しますが、クラスター ID *j-A1B2CD34EF5G* の Amazon EMR クラスターで実行されるアクションへの AWS アカウント *123456789012* のアクセスは拒否します。クラスターのリソースレベルのアクセス許可をサポートしないその他の Amazon EMR アクションは、引き続き許可されます。Amazon EMR アクションのリストとそれに対応するリソースタイプについては、「Amazon EMR 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。

```
{
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        },
        {
```

```
            "Action": "*",
            "Effect": "Deny",
            "Resource": "arn:aws:elasticmapreduce:us-west-2:123456789012:cluster/j-
A1B2CD34EF5G",
            "Principal": {
                "AWS": [
                    "123456789012"
                ]
            }
        }
    ]
}
```

クラスターを管理する

クラスターを起動したら、クラスターをモニタリングおよび管理できます。Amazon EMR には、クラスターに接続して制御するために使用できるいくつかのツールが用意されています。

トピック

クラスターに接続する

Amazon EMR クラスターを実行するときに必要な作業は、多くの場合、データを分析するアプリケーションを実行し、Amazon S3 バケットからの出力を収集することのみです。また、クラスターの実行中にプライマリノードを操作することもできます。例えば、プライマリノードに接続することによって、インタラクティブなクエリの実行、ログファイルの確認、クラスターの問題のデバッグ、プライマリノードで実行される Ganglia などのアプリケーションを使用したパフォーマンスのモニタリングなどを実行できます。以降のセクションでは、プライマリノードに接続する手法について説明します。

EMR クラスターでは、プライマリノードは、タスクノードとコアノードとして実行される EC2 インスタンスを調整する Amazon EC2 インスタンスです。プライマリノードは、自身への接続に使用できるパブリック DNS 名を公開しています。デフォルトでは、Amazon EMR がプライマリノード、コアノード、タスクノードのセキュリティグループルールを作成し、これらがノードへのアクセス方法を決定します。

> **Note**
>
> プライマリノードには、クラスターの実行中にしか接続できません。クラスターが終了すると、プライマリノードとして動作している EC2 インスタンスも終了し、使用できなくなります。プライマリノードに接続するには、クラスターも認証する必要があります。Kerberos

認証を使用するか、クラスターを起動するときに Amazon EC2 キーペアのプライベートキーを指定できます。Kerberos の設定および接続の詳細については、「Amazon EMR での認証に Kerberos を使用する」を参照してください。コンソールからクラスターを起動するときは、[クラスターの作成] ページの [セキュリティとアクセス] セクションで Amazon EC2 キーペアのプライベートキーを指定します。

デフォルトでは、 ElasticMapReduce-master セキュリティグループはインバウンド SSH アクセスを許可しません。アクセスするソースからの SSH アクセス (TCP ポート 22) を許可する、インバウンドルールを追加する必要がある場合があります。セキュリティグループルールの変更の詳細については、Amazon EC2 ユーザーガイド」の「セキュリティグループへのルールの追加」を参照してください。

Important

ElasticMapReduce-master セキュリティグループの残りのルールを変更しないでください。これらのルールを変更すると、クラスターのオペレーションに悪影響を及ぼす可能性があります。

トピック

接続する前に: インバウンドトラフィックを承認する

Amazon EMR クラスターに接続する前に、コンピュータの IP アドレスなどの、信頼できるクライアントからのインバウンド SSH トラフィック (ポート 22) を承認する必要があります。これを行うには、接続先のノードのマネージドセキュリティグループルールを編集します。例えば、次の手順は、デフォルトの ElasticMapReduceマスターセキュリティグループに SSH アクセスのインバウンドルールを追加する方法を示しています。

Amazon EMR でセキュリティグループを使用する詳細については、「セキュリティグループを使用してネットワークトラフィックを制御する」を参照してください。

New console

新しいコンソールを使用して、信頼できるソースにプライマリセキュリティグループへの SSH アクセス許可を付与するには

セキュリティグループを編集するには、クラスターが存在する VPC のセキュリティグループを管理する権限が必要です。詳細については、「IAM ユーザーガイド」の「IAM ユーザーのアクセス許可の変更」と、EC2 セキュリティグループを管理できるようにする「ポリシーの例」を参照してください。

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で [クラスター] を選択し、更新するクラスターを選択します。選択すると、クラスターの詳細ページが開きます。このページの [プロパティ] タブは事前に選択されます。
3. [プロパティ] タブの [ネットワーク] で、[EC2 セキュリティグループ (ファイアウォール)] の横にある矢印を選択してこのセクションを展開します。[プライマリノード] で、セキュリティグループリンクを選択します。これにより、EC2 コンソールが開きます。
4. [インバウンドルール] タブを選択してから、[インバウンドルールの編集] を選択します。
5. 次の設定で、パブリックアクセスを許可するインバウンドルールを確認します。存在する場合は、[Delete] (削除) をクリックして削除します。

   - タイプ

     SSH
   - [ポート]

     22
   - ソース

     Custom 0.0.0.0/0

> **Warning**
>
> 2020 年 12 月以前は、 ElasticMapReduce-master セキュリティグループには、すべてのソースからのインバウンドトラフィックをポート 22 で許可する事前設定済みのルールがありました。このルールは、プライマリノードへの最初の SSH 接続を簡素

化するために作成されたものです。このインバウンドルールを削除して、信頼できる送信元のみにトラフィックを制限することを強くお勧めします。

6. ルールのリストの下部までスクロールし、[Add Rule] (ルールの追加) を選択します。
7. [Type (タイプ)] で、[SSH] を選択します。SSH を選択すると、[プロトコル] に [TCP] が、[ポート範囲] に [22] が自動的に入力されます。
8. [source] (送信元) には、[My IP] (マイ IP) を選択して、IP アドレスを送信元アドレスとして自動的に追加します。[Custom] (カスタム) で信頼済みクライアントの IP アドレスの範囲を追加することも、他のクライアントに追加のルールを作成することもできます。多くのネットワーク環境では IP アドレスを動的に割り当てるため、将来的に信頼済みクライアントの IP アドレスを更新することが必要になる場合があります。
9. [保存] を選択します。
10. 必要に応じてステップ 3 に戻り、[コアノードとタスクノード] を選択し、ステップ 4 ~ 8 を繰り返します。これにより、コアノードとタスクノードに SSH によるクライアントアクセス許可が付与されます。

Old console

コンソールを使用して、信頼できるソースにプライマリセキュリティグループへの SSH アクセスを許可するには

セキュリティグループを編集するには、クラスターが存在する VPC のセキュリティグループを管理する権限が必要です。詳細については、「IAM ユーザーガイド」の「IAM ユーザーのアクセス許可の変更」と、EC2 セキュリティグループを管理できるようにする「ポリシーの例」を参照してください。

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. [Clusters] を選択します。変更するクラスターの ID を選択します。
3. ネットワークとセキュリティペインで、EC2 セキュリティグループ (ファイアウォール) ドロップダウンを展開します。
4. プライマリノード で、セキュリティグループを選択します。
5. [Edit inbound rules] (インバウンドルールの編集) を選択します。
6. 次の設定で、パブリックアクセスを許可するインバウンドルールを確認します。存在する場合は、[Delete] (削除) をクリックして削除します。

- タイプ

  SSH

- [ポート]

  22

- ソース

  Custom 0.0.0.0/0

> **Warning**
>
> 2020 年 12 月以前は、すべてのソースからのインバウンドトラフィックをポート 22 で許可する事前設定されたルールがありました。このルールは、プライマリノードへの最初の SSH 接続を簡素化するために作成されたものです。このインバウンドルールを削除して、信頼できる送信元のみにトラフィックを制限することを強くお勧めします。

7. ルールのリストの下部までスクロールし、[Add Rule] (ルールの追加) を選択します。
8. [Type (タイプ)] で、[SSH] を選択します。

   SSH を選択すると、[Protocol] (プロトコル) に [TCP] が、[Port Range] (ポート範囲) に [22] が自動的に入力されます。
9. [source] (送信元) には、[My IP] (マイ IP) を選択して、IP アドレスを送信元アドレスとして自動的に追加します。[Custom] (カスタム) で信頼済みクライアントの IP アドレスの範囲を追加することも、他のクライアントに追加のルールを作成することもできます。多くのネットワーク環境では IP アドレスを動的に割り当てるため、将来的に信頼済みクライアントの IP アドレスを更新することが必要になる場合があります。
10. [保存] を選択します。
11. 必要に応じて、ネットワークとセキュリティペインの Core ノードとタスクノードで他のセキュリティグループを選択し、上記のステップを繰り返して、SSH クライアントがコアノードとタスクノードにアクセスできるようにします。

SSH を使用してプライマリノードに接続する

Secure Shell (SSH) とは、リモートコンピュータとの安全な接続を確立するために使用できるネットワークプロトコルです。接続後、ローカルコンピュータ上のターミナルは、リモートコンピュータで実行されているかのように動作します。ローカルで発行したコマンドがリモートコンピュータで実行され、リモートコンピュータからのコマンドの出力はターミナルウィンドウに表示されます。

で SSH を使用する場合 AWS、クラウドで実行されている仮想サーバーである EC2 インスタンスに接続します。Amazon EMR で作業するときの SSH の最も一般的な用途は、クラスターのプライマリノードとして動作する EC2 インスタンスへの接続です。

SSH を使用してプライマリノードに接続すると、クラスターをモニタリングし、操作できます。例えば、プライマリノードで Linux コマンドを発行したり、Hive や Pig などのアプリケーションをインタラクティブに実行したりできます。また、ディレクトリのブラウズやログファイルの読み取りなども可能です。SSH 接続にトンネルを作成して、プライマリノードでホストされるウェブインターフェイスを表示することもできます。詳細については、「Amazon EMR クラスターでホストされているウェブインターフェイスを表示する」を参照してください。

SSH を使用してプライマリノードに接続するには、プライマリノードのパブリック DNS 名が必要です。また、プライマリノードに関連付けられるセキュリティグループには、SSH 接続元のクライアントを含むソースからの SSH (TCP ポート 22) トラフィックを許可するインバウンドルールがある必要があります。クライアントからの SSH 接続を許可するルールの追加が必要になる場合があります。セキュリティグループルールの変更の詳細については、「Amazon EC2 ユーザーガイド」のセキュリティグループを使用してネットワークトラフィックを制御する「」および「セキュリティグループへのルールの追加」を参照してください。 https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-network-security.html Amazon EC2

プライマリノードのパブリック DNS 名を取得する

プライマリパブリック DNS 名は、Amazon EMR コンソールと AWS CLIを使用して取得できます。

> **Note**
>
> Amazon EMR コンソールは、再設計され、使いやすくなりました。古いコンソールと新しいコンソールエクスペリエンスの違いについては、「Amazon EMR コンソール」を参照してください。

New console

新しいコンソールを使用してプライマリノードのパブリック DNS 名を取得するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で [クラスター] を選択し、パブリック DNS 名を取得するクラスターを選択します。
3. クラスターの詳細ページの [概要] セクションに表示される [プライマリノードのパブリック DNS] 値を書き留めます。

Old console

古いコンソールを使用してプライマリノードのパブリック DNS 名を取得するには

1. 新しい Amazon EMR コンソールに移動し、サイドナビゲーションから [古いコンソールに切り替え] を選択します。古いコンソールに切り替えたときの動作の詳細については、「Using the old console」を参照してください。
2. [Cluster List] ページで、クラスターのリンクを選択します。
3. [クラスターの詳細] ページの [概要] セクションに表示される [マスターパブリック DNS] 値を書き留めます。

> **Note**
>
> [SSH] リンクを選択して、プライマリノードへの SSH 接続を作成する手順を表示することもできます。

CLI

を使用してプライマリノードのパブリック DNS 名を取得するには AWS CLI

1. クラスター識別子を取得するには、次のコマンドを入力します。

```
aws emr list-clusters
```

出力には、クラスター ID を含むクラスターのリストが表示されます。接続しているクラスターのクラスター ID を書き留めます。

```
"Status": {
    "Timeline": {
        "ReadyDateTime": 1408040782.374,
        "CreationDateTime": 1408040501.213
    },
    "State": "WAITING",
    "StateChangeReason": {
        "Message": "Waiting after step completed"
    }
},
"NormalizedInstanceHours": 4,
"Id": "j-2AL4XXXXXX5T9",
"Name": "My cluster"
```

2. クラスターのパブリック DNS 名を含むクラスターインスタンスのリストを表示するには、次のいずれかのコマンドを入力します。*j-2AL4XXXXXX5T9* の部分を、前のコマンドで返されたクラスター ID に置き換えてください。

```
aws emr list-instances --cluster-id j-2AL4XXXXXX5T9
```

または:

```
aws emr describe-cluster --cluster-id j-2AL4XXXXXX5T9
```

出力には、DNS 名と IP アドレスを含むクラスターインスタンスのリストが表示されます。PublicDnsName の値を書き留めます。

```
"Status": {
    "Timeline": {
        "ReadyDateTime": 1408040779.263,
        "CreationDateTime": 1408040515.535
    },
    "State": "RUNNING",
    "StateChangeReason": {}
},
"Ec2InstanceId": "i-e89b45e7",
"PublicDnsName": "ec2-###-##-##-###.us-west-2.compute.amazonaws.com"

"PrivateDnsName": "ip-###-##-##-###.us-west-2.compute.internal",
"PublicIpAddress": "##.###.###.##",
```

```
"Id": "ci-12XXXXXXXXFMH",
"PrivateIpAddress": "###.##.#.###"
```

詳細については、「AWS CLIの Amazon EMR コマンド」を参照してください。

Linux、Unix、または Mac OS X で SSH と Amazon EC2 プライベートキーを使用してプライマリノードに接続する

プライベートキーファイルで認証された SSH 接続を作成するには、クラスター起動時に Amazon EC2 キーペアのプライベートキーを指定する必要があります。キーペアへのアクセスの詳細については、Amazon EC2 ユーザーガイド」の「Amazon EC2 キーペアAmazon EC2」を参照してください。

Linux コンピュータには、デフォルトで SSH クライアントが含まれている可能性があります。たとえば、ほとんどの Linux、Unix、および Mac OS オペレーティングシステムには OpenSSH がインストールされています。SSH クライアントがあるかどうかを確認するには、コマンドラインで ssh と入力します。ご使用のコンピュータでこのコマンドが認識されない場合、プライマリノードに接続するために SSH クライアントをインストールします。OpenSSH プロジェクトが、SSH ツールの完全なスイートの無料実装を提供しています。詳細については、OpenSSH のウェブサイトを参照してください。

次の手順は、Linux、Unix、および Mac OS X で、Amazon EMR プライマリノードへの SSH 接続を開く方法です。

キーペアのプライベートキーのファイルアクセス許可を設定するには

Amazon EC2 キーペアのプライベートキーを使用して SSH 接続を作成する前に、キー所有者のみがファイルへのアクセス権限を持つように .pem ファイルに対するアクセス許可を設定しておく必要があります。これは、ターミナルまたは を使用して SSH 接続を作成するために必要です AWS CLI。

1. インバウンド SSH トラフィックを許可していることを確認します。手順については、「接続する前に: インバウンドトラフィックを承認する」を参照してください。
2. .pem ファイルを見つけます。この手順では、ファイル名が mykeypair.pem であり、現在のユーザーのホームディレクトリに保存されていることを想定しています。
3. 次のコマンドを入力してアクセス許可を設定します。*~/mykeypair.pem* の部分を、キーペアのプライベートキーファイルの完全修飾パスとファイル名に置き換えてください。例えば、C:/Users/<username>/.ssh/mykeypair.pem です。

```
chmod 400 ~/mykeypair.pem
```

.pem ファイルに対するアクセス許可を設定していない場合、キーファイルが保護されておらず、キーが拒否されることを示すエラーが表示されます。接続するためにキーペアのプライベートキーファイルに対するアクセス許可を設定する必要があるのは、このファイルを最初に使用するときだけです。

ターミナルを使用してプライマリノードに接続するには

1. ターミナルウィンドウを開きます。Mac OS X で、[Applications] > [Utilities] > [Terminal] を選択します。他の Linux ディストリビューションでは、ターミナルは通常、[Applications] > [Accessories] > [Terminal] にあります。
2. プライマリノードへの接続を確立するには、次のコマンドを入力します。*ec2-###-##-##-###.compute-1.amazonaws.com* はクラスターのプライマリパブリック DNS 名に、*~/mykeypair.pem* は .pem ファイルの完全修飾パスとファイル名に置き換えます。例えば「C:/Users/<username>/.ssh/mykeypair.pem」のようにです。

```
ssh hadoop@ec2-###-##-##-###.compute-1.amazonaws.com -i ~/mykeypair.pem
```

> **Important**
>
> Amazon EMR プライマリノードに接続するときは、ログイン名 hadoop を使用する必要があります。このログイン名を使用しない場合、Server refused our key のようなエラーが表示されることがあります。

3. 警告は、接続先ホストの正当性を検証できないことを示しています。yes を入力して、操作を続けます。
4. プライマリノードに対する操作が終了したら、次のコマンドを入力して SSH 接続を閉じます。

```
exit
```

SSH を使用したプライマリノードへの接続に問題がある場合は、「インスタンスへの接続に関するトラブルシューティング」を参照してください。

Windows で SSH を使用してプライマリノードに接続する

Windows ユーザーは、PuTTY などの SSH クライアントを使用して、プライマリノードに接続できます。Amazon EMR プライマリノードに接続する前に、PuTTY と PuTTYgen をダウンロードしてインストールしてください。これらのツールは、PuTTY のダウンロードページからダウンロードできます。

PuTTY は、Amazon EC2 によって生成されるキーペアのプライベートキーファイルの形式 (`.pem`) をネイティブでサポートしていません。PuTTYgen を使用して、キーファイルを必要な PuTTY 形式 (`.ppk`) に変換します。PuTTY を使用してプライマリノードへの接続を試みる前に、キーをこの形式 (`.ppk`) に変換する必要があります。

キーの変換の詳細については、Amazon EC2 ユーザーガイド」のPuTTYgen を使用したプライベートキーの変換」を参照してください。

PuTTY を使用してプライマリノードに接続するには

1. インバウンド SSH トラフィックを許可していることを確認します。手順については、「接続する前に: インバウンドトラフィックを承認する」を参照してください。
2. `putty.exe` を開きます。Windows のプログラムの一覧から PuTTY を起動することもできます。
3. 必要に応じて、[Category] リストで、[Session] を選択します。
4. ホスト名 (または IP アドレス) には、hadoop@*MasterPublicDNS* #入力します。たとえば、hadoop@*ec2-###-##-##-###.compute-1.amazonaws.com* です。
5. [Category] リストで、[Connection]、[SSH] の順に選択し、[Auth] を選択します。
6. [Private key file for authentication] では、[Browse] をクリックし、以前に生成した `.ppk` ファイルを選択します。
7. [Open] を選択し、[Yes] をクリックして PuTTY のセキュリティ警告を閉じます。

> **Important**
>
> プライマリノードにログインするときに、ユーザー名の入力を求められた場合は、「hadoop」と入力します。

8. プライマリノードに対する操作が終了したら、PuTTY を閉じることで SSH 接続を閉じることができます。

> **Note**
>
> SSH 接続のタイムアウト防止には、[Category] リストで [Connection] をクリックし、[Enable TCP_keepalives] オプションを選択します。PuTTY にアクティブな SSH セッションがある場合は、PuTTY のタイトルバーのコンテキストを開き (右クリック)、[Change Settings] を選択することで設定を変更できます。

SSH を使用したプライマリノードへの接続に問題がある場合は、「インスタンスへの接続に関するトラブルシューティング」を参照してください。

AWS CLIを使用してプライマリノードに接続する

Windows AWS CLI および Linux、Unix、Mac OS X で を使用して、プライマリノードとの SSH 接続を作成できます。プラットフォームに関係なく、プライマリノードのパブリック DNS 名と Amazon EC2 キーペアのプライベートキーが必要です。Linux、Unix、または Mac OS X AWS CLI で を使用している場合は、 に示すように、プライベートキー (`.pem` または `.ppk`) ファイルに対するアクセス許可も設定する必要がありますキーペアのプライベートキーのファイルアクセス許可を設定するには。

を使用してプライマリノードに接続するには AWS CLI

1. インバウンド SSH トラフィックを許可していることを確認します。手順については、「接続する前に: インバウンドトラフィックを承認する」を参照してください。
2. クラスター識別子を取得するには、次のように入力します。

   ```
   aws emr list-clusters
   ```

   出力には、クラスター ID を含むクラスターのリストが表示されます。接続しているクラスターのクラスター ID を書き留めます。

   ```
   "Status": {
       "Timeline": {
           "ReadyDateTime": 1408040782.374,
           "CreationDateTime": 1408040501.213
       },
       "State": "WAITING",
       "StateChangeReason": {
   ```

```
        "Message": "Waiting after step completed"
    }
},
"NormalizedInstanceHours": 4,
"Id": "j-2AL4XXXXXX5T9",
"Name": "AWS CLI cluster"
```

3. プライマリノードへの SSH 接続を開くには、次のコマンドを入力します。次の例で、*j-2AL4XXXXXX5T9* をクラスター ID に、*~/mykeypair.key* を .pem ファイル (Linux、Unix、および Mac OS X の場合) または .ppk ファイル (Windows の場合) の完全修飾パスとファイル名に置き換えてください。例えば、C:\Users\<username>\.ssh\mykeypair.pem です。

```
aws emr ssh --cluster-id j-2AL4XXXXXX5T9 --key-pair-file ~/mykeypair.key
```

4. プライマリノードでの作業が完了したら、 AWS CLI ウィンドウを閉じます。

詳細については、「AWS CLIの Amazon EMR コマンド」を参照してください。SSH を使用したプライマリノードへの接続に問題がある場合は、「インスタンスへの接続に関するトラブルシューティング」を参照してください。

Amazon EMR サービスポート

> **Note**
>
> Amazon EMR のコンポーネントのインターフェイスとサービスポートを次に示します。これはサービスポートの完全なリストではありません。SSL ポートや各種プロトコルなど、デフォルト以外のサービスは記載していません。

> **Important**
>
> セキュリティグループのルールを編集してポートを開くときには注意が必要です。ワークロードの実行に必要なプロトコルとポートには、必ず信頼できる認証済みのクライアントからのトラフィックのみを許可するルールを追加してください。

コンポーネント	サービスの説明	サービスのデフォルトでの実行	[ポート]	設定キー
Hadoop	HTTP KMS REST API	あり	9600	hadoop.kms.http.port
HDFS	Namenode ウェブ UI	あり	9870	dfs.namenode.http-address
	Namenode RPC	あり	8020	dfs.namenode.rpc-address
	DataNode ウェブ UI	あり	9864	dfs.datanode.http.address
	データ転送用の Datanode HTTP	あり	9866	dfs.datanode.address
	データ転送用の Datanode RPC	あり	9867	dfs.datanode.ipc.address
[Hive]	HiveServer2 ドリフト	あり	10000	hive.server2.thrift.port
	HiveServer2 HTTP	なし	10001	hive.server2.thrift.http.port
	HiveServer2 ウェブ UI	あり	10002	hive.server2.webui.port
	Hive メタストア	あり	9083	hive.metastore.port/metastore.thrift.port
	WebHCat	なし	50111	templeton.port
	LLAP デーモン管理サービス (RPC)	なし	15004	hive.llap.management.rpc.port

コンポーネント	サービスの説明	サービスのデフォルトでの実行	[ポート]	設定キー
	LLAP デーモンがホストするシャッフル用の YARN シャッフルポート	なし	15551	hive.llap.daemon.yarn.shuffle.port
	LLAP デーモン RPC	なし	動的	hive.llap.daemon.rpc.port
	LLAP デーモンウェブ UI	なし	15002	hive.llap.daemon.web.port
	LLAP デーモン出力サービス	なし	15003	hive.llap.daemon.output.service.port
Oozie		あり	11000	
Tez	Tez UI	あり	8080	
YARN	シャッフル	あり	13562	mapreduce.shuffle.port
	Localizer RPC	あり	8040	yarn.nodemanager.localizer.address
		あり	8041	
	NM ウェブアプリケーションアドレス	あり	8042	yarn.nodemanager.webapp.address
	RM ウェブアプリケーション	あり	8088	yarn.resourcemanager.webapp.address

コンポーネント	サービスの説明	サービスのデフォルトでの実行	[ポート]	設定キー
		あり	8025	
	スケジューラー	あり	8030	yarn.resourcemanager.scheduler.address
	アプリケーションマネージャーインターフェイス	あり	8032	yarn.resourcemanager.address
	RM 管理インターフェイス	あり	8033	yarn.resourcemanager.admin.address
	JobHistory サーバーウェブ UI	あり	19888	mapreduce.jobhistory.webapp.address
	JobHistory サーバー管理ウェブ UI	あり	10033	mapreduce.jobhistory.admin.address
	JobHistory サーバー (RPC)	あり	10020	mapreduce.jobhistory.address
	アプリケーションタイムラインサーバー (RPC)	あり	10200	yarn.timeline-service.address
	アプリケーションタイムラインサーバー HTTP ウェブ UI	あり	8188	yarn.timeline-service.webapp.address

コンポーネント	サービスの説明	サービスのデフォルトでの実行	[ポート]	設定キー
	アプリケーションタイムラインサーバー HTTPS ウェブ UI	なし	8190	yarn.timeline-service.webapp.https.address
		あり	20888	
Zookeeper	クライアントポート	あり	2181	
		あり	37301	
		あり	8341	

Amazon EMR クラスターでホストされているウェブインターフェイスを表示する

> **Important**
>
> カスタムセキュリティグループを設定して、これらのウェブインターフェイスへのインバウンドアクセスを許可することができます。インバウンドトラフィックを許可するポートでは、セキュリティ脆弱性が生じる可能性があることに注意してください。カスタムセキュリティグループを注意深く確認して、脆弱性を最小限に抑えます。詳細については、「セキュリティグループを使用してネットワークトラフィックを制御する」を参照してください。

EMR クラスターにインストールする Hadoop やその他のアプリケーションは、プライマリノードでホストされるウェブサイトとしてユーザーインターフェイスを公開します。セキュリティ上の理由のため、Amazon EMR マネージドセキュリティグループを使用する場合、これらのウェブサイトは、プライマリノードのローカルウェブサーバー上のみで使用できます。そのため、ウェブインターフェイスを表示するにはプライマリノードに接続する必要があります。詳細については、「SSH を使用してプライマリノードに接続する」を参照してください。Hadoop は、コアおよびタスクノードでホストされるウェブサイトとしてもユーザーインターフェイスを公開します。これらのウェブサイトも、ノードのローカルウェブサーバーでのみ利用できます。

次の表は、クラスターインスタンスで表示できるウェブインターフェイスを示しています。これらの Hadoop インターフェイスは、すべてのクラスターで使用できます。マスターインスタンスインターフェイスの場合は、 を Amazon EMR コンソールのクラスター概要タブにリストされているマスターパブリック DNS *master-public-dns-name*に置き換えます。 コアインスタンスインターフェイスとタスクインスタンスインターフェイスの場合は、 をインスタンスにリストされているパブリック DNS 名*coretask-public-dns-name*に置き換えます。インスタンスの [パブリック DNS 名] を見つけるには、Amazon EMR コンソールでリストからクラスターを選択し、[ハードウェア] タブを選択します。次に、接続するインスタンスを含むインスタンスグループの [ID] を選択し、インスタンスに対してリストされた [パブリック DNS 名] を書き留めます。

インターフェイスの名前	[URI]
Flink History Server (EMR バージョン 5.33 以降)	http://*master-public-dns-name* :8082/
Ganglia	http://*master-public-dns-name* /ganglia/
Hadoop HDFS NameNode (6.x より前の EMR バージョン)	https://*master-public-dns-name* :50470/
Hadoop HDFS NameNode	http://*master-public-dns-name* :50070/
Hadoop HDFS DataNode	http://*coretask-public-dns-name* :50075/
Hadoop HDFS NameNode (EMR バージョン 6.x)	https://*master-public-dns-name* :9870/
Hadoop HDFS DataNode (6.x より前の EMR バージョン)	https://*coretask-public-dns-name* :50475/
Hadoop HDFS DataNode (EMR バージョン 6.x)	https://*coretask-public-dns-name* :9865/
HBase	http://*master-public-dns-name* :16010/
Hue	http://*master-public-dns-name* :8888/
JupyterHub	https://*master-public-dns-name* :9443/

インターフェイスの名前	[URI]
Livy	http://*master-public-dns-name* :8998/
Spark HistoryServer	http://*master-public-dns-name* :18080/
Tez	http://*master-public-dns-name* :8080/tez-ui
YARN NodeManager	http://*coretask-public-dns-name* :8042/
YARN ResourceManager	http://*master-public-dns-name* :8088/
Zeppelin	http://*master-public-dns-name* :8890/

プライマリノードで利用できるアプリケーション固有のインターフェイスの中には、コアノードとタスクノードでは利用できないものもあるため、このドキュメントの手順は Amazon EMR プライマリノードに特化しています。コアノードおよびタスクノードのウェブインターフェイスには、プライマリノードのウェブインターフェイスにアクセスする場合と同じ手順でアクセスできます。

プライマリノードのウェブインターフェイスにアクセスするには、いくつかの方法があります。最も簡単な方法は、SSH を使用してプライマリノードに接続し、テキストベースのブラウザ Lynx を使用して SSH クライアントでウェブサイトを表示する方法です。ただし、Lynx はユーザーインターフェイスが制限されたテキストベースのブラウザであり、グラフィックを表示できません。次の例は、Lynx を使用して Hadoop ResourceManager インターフェイスを開く方法を示しています (SSH を使用してプライマリノードにログインすると、Lynx URLs も提供されます) 。

```
lynx http://ip-###-##-##-###.us-west-2.compute.internal:8088/
```

すべてのブラウザ機能を提供する、プライマリノードのウェブインターフェイスにアクセスするには、あと 2 つの方法があります。以下のうちのひとつを選択します。

- オプション 1 (技術的知識のあるユーザー向け): SSH クライアントを使用してプライマリノードに接続し、ローカルポートフォワーディングを使って SSH トンネリングを設定します。次に、インターネットブラウザを使用してプライマリノードでホストされるウェブインターフェイスを開きます。この方法では、SOCKS プロキシを使用せずに、ウェブインターフェイスへのアクセスを設定できます。

- オプション 2 (新規ユーザーに推奨): SSH クライアントを使用してプライマリノードに接続し、動的ポート転送で SSH トンネリングを設定し、Firefox FoxyProxy の場合は 、Chrome SwitchyOmega の場合は などのアドオンを使用して SOCKS プロキシ設定を管理するようにインターネットブラウザを設定します。この方法では、テキストパターンに基づいて自動的に URL をフィルタリングし、プロキシの設定をプライマリノードの DNS 名の形式に合致するドメインに限定できます。Firefox および Google Chrome FoxyProxy 用に を設定する方法の詳細については、「」を参照してくださいオプション 2、パート 2: プライマリノードでホストされるウェブサイトを表示するようにプロキシを設定する。

> **Note**
>
> クラスター設定経由でアプリケーションが実行されるポートを変更すると、Amazon EMR コンソールでポートへのハイパーリンクが更新されません。これは、コンソールに `server.port` 設定を読み取る機能がないためです。

Amazon EMR バージョン 5.25.0 以降では、SSH 接続を介してウェブプロキシを設定しなくても、コンソールから Spark 履歴サーバー UI にアクセスできます。詳細については、「永続 Spark History Server へのワンクリックアクセス」を参照してください。

トピック

オプション 1: ローカルポートフォワーディングを使用してプライマリノードへの SSH トンネルをセットアップする

プライマリノードのローカルウェブサーバーに接続するために、コンピュータとプライマリノードの間に SSH トンネルを作成します。これはポートフォワーディングとも呼ばれます。SOCKS プロキシを使用しない場合は、ローカルポートフォワーディングを使用してプライマリノードへの SSH トンネルをセットアップすることができます。ローカルポートフォワーディングでは、トラフィックを

プライマリノードのローカルウェブサーバーにある特定のリモートポートに転送するために使用する、未使用のローカルポートを指定します。

ローカルポートフォワーディングを使用して SSH トンネルをセットアップするには、プライマリノードのパブリック DNS 名と、キーペアのプライベートキーファイルが必要です。マスターパブリック DNS 名を特定する方法については、「古いコンソールを使用してプライマリノードのパブリック DNS 名を取得するには」を参照してください。キーペアへのアクセスの詳細については、Amazon EC2 ユーザーガイド」の「Amazon EC2 キーペアAmazon EC2」を参照してください。プライマリノード上で表示するサイトの詳細については、「Amazon EMR クラスターでホストされているウェブインターフェイスを表示する」を参照してください。

OpenSSH でローカルポートフォワーディングを使用してプライマリノードへの SSH トンネルをセットアップする

ターミナルでローカルポートフォワーディングを使用して SSH トンネルをセットアップするには

1. インバウンド SSH トラフィックを許可していることを確認します。手順については、「接続する前に: インバウンドトラフィックを承認する」を参照してください。
2. ターミナルウィンドウを開きます。Mac OS X で、[Applications] > [Utilities] > [Terminal] を選択します。他の Linux ディストリビューションでは、ターミナルは通常、[Applications] > [Accessories] > [Terminal] にあります。
3. 次のコマンドを入力して、ローカルマシンで SSH トンネルを開きます。このコマンド例では、ローカルポート 8157 (ランダムに選択された未使用のローカルポート) のトラフィックをマスターノードのローカルResourceManagerウェブサーバーのポート 8088 に転送することで、ウェブインターフェイスにアクセスします。

   コマンドの、*~/mykeypair.pem* を .pem ファイルの場所とファイル名に、*ec2-###-###-###.compute-1.amazonaws.com* をクラスターのマスターパブリック DNS 名に置き換えます。別のウェブインターフェイスにアクセスするには、8088 を該当するポート番号に置き換えます。たとえば、Zeppelin インターフェイスの場合、8088 を 8890 に置き換えます。

   ```
   ssh -i ~/mykeypair.pem -N -L 8157:ec2-###-##-##-###.compute-1.amazonaws.com:8088 hadoop@ec2-###-##-##-###.compute-1.amazonaws.com
   ```

   -L はローカルポートフォワーディングを使用することを示します。これにより、マスターノードのローカルウェブサーバーの指定したリモートポートにデータを転送するために使用するローカルポートを指定できます。

このコマンドを発行すると、ターミナルは開いたままになり、応答を返しません。

4. ブラウザで ResourceManager ウェブインターフェイスを開くには、アドレスバーhttp://localhost:*8157*/に と入力します。
5. プライマリノードのウェブインターフェイスに対する操作が終了したら、ターミナルウィンドウを閉じます。

オプション 2、パート 1: ダイナミックポートフォワーディングを使用してプライマリノードへの SSH トンネルをセットアップする

プライマリノードのローカルウェブサーバーに接続するために、コンピュータとプライマリノードの間に SSH トンネルを作成します。これはポートフォワーディングとも呼ばれます。ダイナミックポートフォワーディングを使用して SSH トンネルを作成する場合、指定した未使用のローカルポートにルーティングされたすべてのトラフィックが、プライマリノードのローカルウェブサーバーに転送されます。これにより、SOCKS プロキシが作成されます。その後、 FoxyProxy や などのアドオンを使用して SOCKS プロキシ設定を管理する SwitchyOmega ようにインターネットブラウザを設定できます。

プロキシ管理のアドオンを使用すると、テキストパターンに基づいて自動的に URL をフィルタリングし、プロキシの設定をプライマリノードのパブリック DNS 名の形式に合致するドメインに限定することができます。プライマリノードでホストされているウェブサイトとインターネットのウェブサイトの表示を切り替えると、ブラウザのアドオンで自動的にプロキシのオンとオフが切り替えられます。

作業を始める前に、プライマリノードのパブリック DNS 名とキーペアのプライベートキーファイルが必要です。プライマリパブリック DNS 名を特定する方法については、「古いコンソールを使用してプライマリノードのパブリック DNS 名を取得するには」を参照してください。キーペアへのアクセスの詳細については、Amazon EC2 ユーザーガイド」の「Amazon EC2 キーペアAmazon EC2」を参照してください。プライマリノード上で表示するサイトの詳細については、「Amazon EMR クラスターでホストされているウェブインターフェイスを表示する」を参照してください。

OpenSSH でダイナミックポートフォワーディングを使用してプライマリノードへの SSH トンネルをセットアップする

OpenSSH でダイナミックポートフォワーディングを使用して SSH トンネルをセットアップするには

1. インバウンド SSH トラフィックを許可していることを確認します。手順については、「接続する前に: インバウンドトラフィックを承認する」を参照してください。

2. ターミナルウィンドウを開きます。Mac OS X で、[Applications] > [Utilities] > [Terminal] を選択します。他の Linux ディストリビューションでは、ターミナルは通常、[Applications] > [Accessories] > [Terminal] にあります。
3. ローカルコンピュータで SSH トンネルを開くには、次のコマンドを入力します。*~/mykeypair.pem* を .pemファイルの場所とファイル名に置き換え、*8157* を未使用のローカルポート番号に置き換え、*ec2-###-##-##-###.compute-1.amazonaws.com* をクラスターのプライマリパブリック DNS 名に置き換えます。

   ```
   ssh -i ~/mykeypair.pem -N -D 8157 hadoop@ec2-###-##-##-###.compute-1.amazonaws.com
   ```

   このコマンドを発行すると、ターミナルは開いたままになり、応答を返しません。

   > **Note**
   >
   > -D はダイナミックポートフォワーディングを使用することを示し、プライマリノードのローカルウェブサーバーのすべてのリモートポートにデータを転送するために使用するローカルポートを指定できます。ダイナミックポートフォワーディングでは、コマンドで指定されたポートでローカル SOCKS プロキシのリスニングが実行されます。

4. トンネルがアクティブになった後、ブラウザの SOCKS プロキシを設定します。詳細については、「オプション 2、パート 2: プライマリノードでホストされるウェブサイトを表示するようにプロキシを設定する」を参照してください。
5. プライマリノードのウェブインターフェイスに対する操作が終了したら、ターミナルウィンドウを閉じます。

で動的ポート転送を使用して SSH トンネルを設定する AWS CLI

Windows AWS CLI では 、Linux、Unix、および Mac OS X では を使用して、プライマリノードとの SSH 接続を作成できます。Linux、Unix、または Mac OS X AWS CLI で を使用している場合は、「」に示すように、 .pem ファイルのアクセス許可を設定する必要がありますキーペアのプライベートキーのファイルアクセス許可を設定するには。Windows AWS CLI で を使用している場合は、パス環境変数に PuTTY が表示される必要があります。表示されていない場合は、OpenSSH や PuTTY が使用できないなどのエラーが表示されることがあります。

で動的ポート転送を使用して SSH トンネルを設定するには AWS CLI

1. インバウンド SSH トラフィックを許可していることを確認します。手順については、「接続する前に: インバウンドトラフィックを承認する」を参照してください。
2. 「AWS CLIを使用してプライマリノードに接続する」に示されているように、プライマリノードへの SSH 接続を作成します。
3. クラスター識別子を取得するには、次のように入力します。

```
aws emr list-clusters
```

出力には、クラスター ID を含むクラスターのリストが表示されます。接続しているクラスターのクラスター ID を書き留めます。

```
"Status": {
    "Timeline": {
        "ReadyDateTime": 1408040782.374,
        "CreationDateTime": 1408040501.213
    },
    "State": "WAITING",
    "StateChangeReason": {
        "Message": "Waiting after step completed"
    }
},
"NormalizedInstanceHours": 4,
"Id": "j-2AL4XXXXXX5T9",
"Name": "AWS CLI cluster"
```

4. 次のコマンドを入力して、ダイナミックポートフォワーディングを使用してプライマリノードへの SSH トンネルを開きます。次の例で、*j-2AL4XXXXXX5T9* をクラスター ID に、*~/mykeypair.key* を .pem ファイル (Linux、Unix、および Mac OS X の場合) または .ppk ファイル (Windows の場合) の場所とファイル名に置き換えてください。

```
aws emr socks --cluster-id j-2AL4XXXXXX5T9 --key-pair-file ~/mykeypair.key
```

> **Note**
>
> socks コマンドは、自動的にポート 8157 でダイナミックポートフォワーディングを設定します。現在、この設定は変更できません。

5. トンネルがアクティブになった後、ブラウザの SOCKS プロキシを設定します。詳細については、「オプション 2、パート 2: プライマリノードでホストされるウェブサイトを表示するようにプロキシを設定する」を参照してください。
6. プライマリノードのウェブインターフェイスの操作が完了したら、 AWS CLI ウィンドウを閉じます。

   での Amazon EMR コマンドの使用の詳細については AWS CLI、「」を参照してくださいいhttps://docs.aws.amazon.com/cli/latest/reference/emr。

PuTTY を使用してプライマリノードへの SSH トンネルをセットアップする

Windows ユーザーは、PuTTY などの SSH クライアントを使用して、プライマリノードへの SSH トンネルを作成できます。Amazon EMR プライマリノードに接続する前に、PuTTY と PuTTYgen をダウンロードしてインストールしてください。これらのツールは、PuTTY のダウンロードページからダウンロードできます。

PuTTY は、Amazon EC2 によって生成されるキーペアのプライベートキーファイルの形式 (`.pem`) をネイティブでサポートしていません。PuTTYgen を使用して、キーファイルを必要な PuTTY 形式 (`.ppk`) に変換します。PuTTY を使用してプライマリノードへの接続を試みる前に、キーをこの形式 (`.ppk`) に変換する必要があります。

キーの変換の詳細については、Amazon EC2 ユーザーガイド」のPuTTYgen を使用したプライベートキーの変換」を参照してください。

PuTTY でダイナミックポートフォワーディングを使用して SSH トンネルをセットアップするには

1. インバウンド SSH トラフィックを許可していることを確認します。手順については、「接続する前に: インバウンドトラフィックを承認する」を参照してください。
2. `putty.exe` をダブルクリックして PuTTY を起動します。Windows のプログラムの一覧から PuTTY を起動することもできます。

   > **Note**
   >
   > 既にプライマリノードとのアクティブな SSH セッションがある場合は、PuTTY のタイトルバーを右クリックし、[Change Settings] を選択することでトンネルを追加できます。

3. 必要に応じて、[Category] リストで、[Session] を選択します。

4. ホスト名 フィールドに、**hadoop@*MasterPublicDNS*** #入力します。たとえば、**hadoop@*ec2-###-##-##-###.compute-1.amazonaws.com*** です。
5. [Category] リストで、[Connection]、[SSH] の順に展開し、[Auth] を選択します。
6. [Private key file for authentication] では、[Browse] をクリックし、以前に生成した .ppk ファイルを選択します。

> **Note**
>
> PuTTY は、Amazon EC2 によって生成されるキーペアのプライベートキーファイルの形式 (.pem) をネイティブでサポートしていません。PuTTYgen を使用して、キーファイルを必要な PuTTY 形式 (.ppk) に変換します。PuTTY を使用してプライマリノードへの接続を試みる前に、キーをこの形式 (.ppk) に変換する必要があります。

7. [Category] リストで、[Connection]、[SSH] の順に展開し、[Tunnels] を選択します。
8. [Source port] フィールドに 8157 (未使用のローカルポート) を入力し、[Add] を選択します。
9. [Destination] フィールドは空白のままにしておきます。
10. [Dynamic] オプションと [Auto] オプションを選択します。
11. 開く をクリックします。
12. [Yes] を選択して、PuTTY セキュリティ警告を閉じます。

> **Important**
>
> プライマリノードにログインするときに、ユーザー名の入力を求められた場合は、「hadoop」と入力します。

13. トンネルがアクティブになった後、ブラウザの SOCKS プロキシを設定します。詳細については、「オプション 2、パート 2: プライマリノードでホストされるウェブサイトを表示するようにプロキシを設定する」を参照してください。
14. プライマリノードのウェブインターフェイスに対する操作が終了したら、PuTTY ウィンドウを閉じます。

オプション 2、パート 2: プライマリノードでホストされるウェブサイトを表示するようにプロキシを設定する

ダイナミックポートフォワーディングによる SSH トンネルを使用する場合、SOCKS プロキシ管理アドオンを使用して、ブラウザでプロキシ設定を管理する必要があります。SOCKS プロキシ管理

ツールを使用すると、テキストパターンに基づいて自動的に URL をフィルタリングし、プロキシの設定をプライマリノードのパブリック DNS 名の形式に合致するドメインに限定することができます。プライマリノードでホストされているウェブサイトとインターネットのウェブサイトの表示を切り替えると、ブラウザのアドオンで自動的にプロキシのオンとオフが切り替えられます。プロキシ設定を管理するには、 FoxyProxy や などのアドオンを使用するようにブラウザを設定します SwitchyOmega。

SSH トンネルの作成の詳細については、「オプション 2、パート 1: ダイナミックポートフォワーディングを使用してプライマリノードへの SSH トンネルをセットアップする」を参照してください。利用可能なウェブインターフェイスの詳細については、「Amazon EMR クラスターでホストされているウェブインターフェイスを表示する」を参照してください。

プロキシアドオンをセットアップするときに、次の設定を含めてください。

- ホストアドレスとして localhost を使用します。
- 「オプション 2、パート 1: ダイナミックポートフォワーディングを使用してプライマリノードへの SSH トンネルをセットアップする」でプライマリノードの SSH トンネルを確立するために選択したのと同じローカルポート番号を使用します。たとえば、ポート *8157* です。このポートは、接続に使用する PuTTY またはその他のターミナルエミュレータで使用するポート番号とも一致する必要があります。
- SOCKS v5 プロトコルを指定します。SOCKS v5 では、オプションでユーザー認証を設定できます。
- URL パターン

  次の URL パターンを許可リストに登録し、ワイルドカードパターンタイプで指定する必要があります。

  - US リージョンのクラスターのパブリック DNS 名と照合する *ec2*.*compute*.amazonaws.com* パターンと *10*.amazonaws.com* パターン。
  - 他のすべてのリージョンのクラスターのパブリック DNS 名と照合する *ec2*.compute* パターンと *10*.compute* パターン。
  - Hadoop のJobTracker ログファイルへのアクセスを提供する 10.* パターン。ネットワークアクセスプランと競合する場合は、このフィルターを変更します。
  - `us-east-1` リージョン、および他のすべてのリージョンのクラスターのプライベート (内部) DNS 名とそれぞれ照合する *.ec2.internal* パターンと *.compute.internal* パターン。

例: FoxyProxyFirefox 用に を設定する

次の例は、Mozilla Firefox の FoxyProxy 標準 (バージョン 7.5.1) 設定を示しています。

FoxyProxy は、プロキシ管理ツールのセットを提供します。これにより、Amazon EMR クラスター内の Amazon EC2 インスタンスで使用されるドメインに対応するパターンと一致する URL についてプロキシサーバーを使用できます。

Mozilla Firefox FoxyProxy を使用してインストールおよび設定するには

1. Firefox で、https://addons.mozilla.org/ に移動し、標準 を検索 FoxyProxyし、手順に従って Firefox FoxyProxy に追加します。
2. テキストエディタを使用し、次の設定例から `foxyproxy-settings.json` という名前の JSON ファイルを作成します。

```
{
  "k20d21508277536715": {
    "active": true,
    "address": "localhost",
    "port": 8157,
    "username": "",
    "password": "",
    "type": 3,
    "proxyDNS": true,
    "title": "emr-socks-proxy",
    "color": "#0055E5",
    "index": 9007199254740991,
    "whitePatterns": [
      {
        "title": "*ec2*.*compute*.amazonaws.com*",
        "active": true,
        "pattern": "*ec2*.*compute*.amazonaws.com*",
        "importedPattern": "*ec2*.*compute*.amazonaws.com*",
        "type": 1,
        "protocols": 1
      },
      {
        "title": "*ec2*.compute*",
        "active": true,
        "pattern": "*ec2*.compute*",
        "importedPattern": "*ec2*.compute*",
        "type": 1,
        "protocols": 1
```

```
      },
      {
        "title": "10.*",
        "active": true,
        "pattern": "10.*",
        "importedPattern": "http://10.*",
        "type": 1,
        "protocols": 2
      },
      {
        "title": "*10*.amazonaws.com*",
        "active": true,
        "pattern": "*10*.amazonaws.com*",
        "importedPattern": "*10*.amazonaws.com*",
        "type": 1,
        "protocols": 1
      },
      {
        "title": "*10*.compute*",
        "active": true,
        "pattern": "*10*.compute*",
        "importedPattern": "*10*.compute*",
        "type": 1,
        "protocols": 1
      },
      {
        "title": "*.compute.internal*",
        "active": true,
        "pattern": "*.compute.internal*",
        "importedPattern": "*.compute.internal*",
        "type": 1,
        "protocols": 1
      },
      {
        "title": "*.ec2.internal* ",
        "active": true,
        "pattern": "*.ec2.internal*",
        "importedPattern": "*.ec2.internal*",
        "type": 1,
        "protocols": 1
      }
    ],
    "blackPatterns": []
  },
```

```
  "logging": {
    "size": 100,
    "active": false
  },
  "mode": "patterns",
  "browserVersion": "68.12.0",
  "foxyProxyVersion": "7.5.1",
  "foxyProxyEdition": "standard"
}
```

3. Firefox で [拡張機能の管理] ページを開きます ([about:addons] に移動し、[拡張機能] を選択します)。
4. FoxyProxy 標準 を選択し、その他のオプションボタン (省略記号のようなボタン) を選択します。
5. ドロップダウンから [Options] を選択します。
6. 左のメニューから、[Import Settings] を選択します。
7. 「設定のインポート」ページで、 FoxyProxy 「6.0 以降の設定のインポート」で「設定のインポート」を選択し、作成したfoxyproxy-settings.jsonファイルの場所を参照し、ファイルを選択し、「 を開く」を選択します。
8. 既存の設定を上書きして新しい設定を保存するかどうかを確認するメッセージが表示されたら [OK] を選択します。

例: Chrome SwitchyOmega の設定

次の例は、複数のプロキシを設定、管理、切り替える Google Chrome. SwitchyOmega lets の拡張機能を設定する SwitchyOmega方法を示しています。

Google Chrome SwitchyOmega を使用してインストールおよび設定するには

1. https://chrome.google.com/webstore/category/extensions に移動し、プロキシ SwitchyOmegaを検索して Chrome に追加します。
2. [New profile] 選択し、プロファイル名として emr-socks-proxy を入力します。
3. [PAC profile] を選択し、[Create] を選択します。プロキシ自動設定 (Proxy Auto-Configuration, PAC) ファイルは、ウェブプロキシサーバーに転送する必要があるブラウザリクエストの許可リストを定義するのに役立ちます。

4. [PAC Script] フィールドのコンテンツを、ウェブプロキシサーバー経由で転送する URL を定義する次のスクリプトに置き換えます。SSH トンネルのセットアップ時に別のポート番号を指定した場合は、*8157* を指定のポート番号に置き換えます。

```
function FindProxyForURL(url, host) {
    if (shExpMatch(url, "*ec2*.*compute*.amazonaws.com*")) return 'SOCKS5
 localhost:8157';
    if (shExpMatch(url, "*ec2*.compute*")) return 'SOCKS5 localhost:8157';
    if (shExpMatch(url, "http://10.*")) return 'SOCKS5 localhost:8157';
    if (shExpMatch(url, "*10*.compute*")) return 'SOCKS5 localhost:8157';
    if (shExpMatch(url, "*10*.amazonaws.com*")) return 'SOCKS5 localhost:8157';
    if (shExpMatch(url, "*.compute.internal*")) return 'SOCKS5 localhost:8157';
    if (shExpMatch(url, "*ec2.internal*")) return 'SOCKS5 localhost:8157';
    return 'DIRECT';
}
```

5. [Actions] の下で、[Apply changes] を選択して、プロキシ設定を保存します。
6. Chrome ツールバーで、`emr-socks-proxy`プロファイル SwitchyOmega を選択して選択します。

ブラウザでウェブインターフェイスにアクセスする

ウェブインターフェイスを開くには、ブラウザのアドレスバーにプライマリノードまたはコアノードのパブリック DNS 名を入力し、その後に選択したインターフェイスのポート番号を入力します。次の例は、Spark に接続するために入力する URL を示しています HistoryServer。

```
http://master-public-dns-name:18080/
```

ノードのパブリック DNS 名を取得する手順については、「プライマリノードのパブリック DNS 名を取得する」を参照してください。ウェブインターフェイス URL の詳細なリストについては、「Amazon EMR クラスターでホストされているウェブインターフェイスを表示する」を参照してください。

クラスターへの作業の送信

このセクションでは、Amazon EMR クラスターに作業を送信する際に使用できる方法について説明します。作業を送信するには、ステップを追加するか、Hadoop ジョブをプライマリノードにインタラクティブに送信します。

クラスターにステップを送信するときは、以下のステップの動作に関するルールを考慮してください。

- ステップ ID の長さは最大 256 文字です。
- 1 つのクラスターで使用できる保留中および実行中のステップの最大数は 256 です。
- クラスターで実行しているアクティブなステップが 256 ある場合でも、プライマリノードにジョブをインタラクティブに送信できます。長時間稼働するクラスターが存続する間は、送信できるステップの数に制限はありませんが、任意の時点で実行中または保留中にできるステップは 256 ステップまでです。
- バージョン 5.0.0 を除く Amazon EMR バージョン 4.8.0 以降では、保留中のステップをキャンセルできます。詳細については、「ステップのキャンセル」を参照してください。
- Amazon EMR バージョン 5.28.0 以降では、保留中のステップと実行中のステップの両方をキャンセルできます。また、複数のステップを並行して実行して、クラスターの使用率を改善し、コストを削減することもできます。詳細については、「複数のステップを並行して実行する際の考慮事項」を参照してください。

> **Note**
>
> 最高のパフォーマンスを得るには、Amazon EMR で使用するカスタムブートストラップアクション、スクリプト、およびその他のファイルを、クラスター AWS リージョン と同じ にある Amazon S3 バケットに保存することをお勧めします。

トピック

- Amazon EMR マネジメントコンソールを使用してクラスターにステップを追加する
- を使用したクラスターへのステップの追加 AWS CLI
- 複数のステップを並行して実行する際の考慮事項
- ステップの表示
- ステップのキャンセル

Amazon EMR マネジメントコンソールを使用してクラスターにステップを追加する

AWS Management Consoleを使用して、クラスターにステップを追加するには、次の手順を実行します。特定のビッグデータアプリケーションのステップを送信する方法の詳細については、「Amazon EMR リリースガイド」の以下のセクションを参照してください。

- Submit a custom JAR step
- Submit a Hadoop streaming step
- Submit a Spark step
- Submit a Pig step
- Run a command or script as a step
- Pass values into steps to run Hive scripts

クラスターの作成中にステップを追加する

から AWS Management Console、クラスターの作成時にステップを追加できます。

> **Note**
>
> Amazon EMR コンソールは、再設計され、使いやすくなりました。古いコンソールと新しいコンソールエクスペリエンスの違いについては、「Amazon EMR コンソール」を参照してください。

New console

新しいコンソールを使用してクラスター作成時にステップを追加するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で、[クラスター] を選択し、[クラスターの作成] を選択します
3. [ステップ] で [ステップの追加] を選択します。[ステップの追加] ダイアログボックスの各フィールドに適切な値を入力します。ステップ引数の形式の詳細については、「ステップ引

数を追加する」を参照してください。オプションは、ステップタイプによって異なります。ステップを追加してダイアログを終了するには、[ステップの追加] を選択します。

4. クラスターに適用するその他のオプションを選択します。
5. クラスターを起動するには、[クラスターの作成] を選択します。

Old console

古いコンソールを使用してクラスターの作成時にステップを追加するには

1. https://console.aws.amazon.com/elasticmapreduce/home で Amazon EMR コンソールを開きます。[クラスターの作成]、[詳細オプション] の順に選択します。
2. [Step 1: Software and Steps (ステップ 1: ソフトウェアとステップ)] ページの [Step (optional) (ステップ (オプション))] で、[Run multiple steps in parallel to improve cluster utilization and save cost (複数のステップを並行して実行し、クラスターの使用率を改善してコストを削減する)] を選択します。同時実行レベルのデフォルト値は 10 です。並行して実行できるステップ数は、2～256 の範囲で選択できます。

   > **Note**
   >
   > 複数のステップの並行実行は、Amazon EMR バージョン 5.28.0 以降でのみサポートされています。

3. [After last step completes (最後のステップの完了後)] で、[Cluster enters waiting state (クラスターが待機状態に入る)] または [Auto-terminate the cluster (クラスターの自動終了)] を選択します。
4. [ステップタイプ]、[Add step (ステップを追加)] の順に選択します。
5. [Add Step (ステップを追加)] ダイアログボックスの各フィールドに適切な値を入力します。ステップ引数の形式の詳細については、「ステップ引数を追加する」を参照してください。オプションは、ステップタイプによって異なります。[複数のステップを並行して実行し、クラスターの使用率を改善してコストを削減する] を有効にした場合、[失敗時の操作] で使用できるオプションは [続行] のみです。次に、[追加] を選択します。

実行中のクラスターにステップを追加する

では AWS Management Console、自動終了オプションを無効にしてクラスターにステップを追加できます。

New console

新しいコンソールを使用して実行中のクラスターにステップを追加するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で [クラスター] を選択し、更新するクラスターを選択します。
3. クラスターの詳細ページの [ステップ] タブで、[ステップの追加] を選択します。既存のステップを複製するには、[アクション] ドロップダウンメニューを選択し、[ステップを複製] を選択します。
4. [ステップの追加] ダイアログボックスの各フィールドに適切な値を入力します。オプションは、ステップタイプによって異なります。ステップを追加してダイアログを終了するには、[ステップの追加] を選択します。

Old console

古いコンソールを使用して実行中のクラスターにステップを追加するには

1. https://console.aws.amazon.com/elasticmapreduce/home で Amazon EMR コンソールを開きます。[Cluster List] ページで、クラスターのリンクを選択します。
2. [クラスターの詳細] ページで、[ステップ] タブを選択します。
3. [ステップ] タブで、[Add step (ステップを追加)] を選択します。
4. [Add Step (ステップを追加)] ダイアログボックスの各フィールドに適切な値を入力し、[Add (追加)] をクリックします。オプションは、ステップタイプによって異なります。

実行中のクラスターでステップの同時実行レベルを変更する

では AWS Management Console、実行中のクラスターのステップ同時実行レベルを変更できます。

> **Note**
>
> 複数のステップの並行実行は、Amazon EMR バージョン 5.28.0 以降でのみ可能です。

New console

新しいコンソールを使用して、実行中のクラスターでステップの同時実行を変更するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で [クラスター] を選択し、更新するクラスターを選択します。同時実行の属性を変更するには、クラスターが実行中である必要があります。
3. クラスターの詳細ページの [ステップ] タブで、[属性] セクションを見つけます。[編集] を選択して同時実行を変更します。1 ~ 256 の値を入力します。

Old console

古いコンソールを使用して、実行中のクラスターでステップの同時実行を変更するには

1. https://console.aws.amazon.com/elasticmapreduce/home で Amazon EMR コンソールを開きます。[Cluster List] ページで、クラスターのリンクを選択します。
2. [クラスターの詳細] ページで、[ステップ] タブを選択します。
3. [同時実行] の [変更] を選択します。ステップ同時実行レベルの新しい値を選択して保存します。

ステップ引数を追加する

を使用してクラスター AWS Management Console にステップを追加する場合、引数フィールドでそのステップの引数を指定できます。引数は、空白で区切り、文字と空白で構成される文字列引数は引用符で囲む必要があります。

Example : 正しい引数

次の引数は、最後の文字列引数を引用符で囲んで AWS Management Console、 に対して正しくフォーマットされています。

```
bash -c "aws s3 cp s3://DOC-EXAMPLE-BUCKET/my-script.sh ."
```

次の例のように、読みやすくするために、各引数を別個の行に置くこともできます。

```
bash
```

```
-c
"aws s3 cp s3://DOC-EXAMPLE-BUCKET/my-script.sh ."
```

Example : 正しくない引数

次の引数の例は、 AWS Management Console用に正しい形式で設定されていません。最後の文字列引数 `aws s3 cp s3://DOC-EXAMPLE-BUCKET/my-script.sh .` には、空白文字が含まれていますが、引用符で囲まれていません。

```
bash -c aws s3 cp s3://DOC-EXAMPLE-BUCKET/my-script.sh .
```

を使用したクラスターへのステップの追加 AWS CLI

以下の手順は、 AWS CLIを使用して、新しく作成されるクラスターと実行中のクラスターにステップを追加する方法を示しています。どちらの例でも、クラスターにステップを追加するために `--steps` サブコマンドを使用します。

クラスターの作成中にステップを追加するには

- 以下のコマンドを入力し、クラスターを作成して Apache Pig ステップを追加します。*myKey* を Amazon EC2 キーペアの名前に置き換えてください。

```
aws emr create-cluster --name "Test cluster" \
--applications Name=Spark \
--use-default-roles \
--ec2-attributes KeyName=myKey \
--instance-groups InstanceGroupType=PRIMARY,InstanceCount=1,InstanceType=m5.xlarge
 InstanceGroupType=CORE,InstanceCount=2,InstanceType=m5.xlarge \
--steps '[{"Args":["spark-submit","--deploy-mode","cluster","--
class","org.apache.spark.examples.SparkPi","/usr/lib/spark/examples/jars/spark-
examples.jar","5"],"Type":"CUSTOM_JAR","ActionOnFailure":"CONTINUE","Jar":"command-
runner.jar","Properties":"","Name":"Spark application"}]'
```

> **Note**
>
> 引数のリストはステップのタイプによって異なります。

デフォルトでは、ステップの同時実行レベルは 1 です。クラスターの作成時に StepConcurrencyLevel パラメータを使用して、ステップの同時実行レベルを設定できます。

出力は、次のようなクラスター識別子です。

```
{
    "ClusterId": "j-2AXXXXXXGAPLF"
}
```

実行中のクラスターにステップを追加するには

- 以下のコマンドを入力し、実行中のクラスターにステップを追加します。*j-2AXXXXXXGAPLF* は独自のクラスター ID に置き換えます。

```
aws emr add-steps --cluster-id j-2AXXXXXXGAPLF \
--steps '[{"Args":["spark-submit","--deploy-mode","cluster","--
class","org.apache.spark.examples.SparkPi","/usr/lib/spark/examples/jars/spark-
examples.jar","5"],"Type":"CUSTOM_JAR","ActionOnFailure":"CONTINUE","Jar":"command-
runner.jar","Properties":"","Name":"Spark application"}]'
```

出力は、次のようなステップ識別子です。

```
{
    "StepIds": [
 "s-Y9XXXXXXAPMD"
    ]
}
```

実行中のクラスター StepConcurrencyLevel で を変更するには

1. 実行中のクラスターでは、ModifyCluster API を使用して StepConcurrencyLevel を変更できます。例えば、StepConcurrencyLevel を 10 に増やすには、次のコマンドを入力します。*j-2AXXXXXXGAPLF* はクラスター ID に置き換えます。

```
aws emr modify-cluster --cluster-id j-2AXXXXXXGAPLF --step-concurrency-level 10
```

2. 出力は以下のようになります。

```
{
"StepConcurrencyLevel": 10
}
```

での Amazon EMR コマンドの使用の詳細については AWS CLI、AWS CLI「 コマンドリファレンス」を参照してください。

複数のステップを並行して実行する際の考慮事項

- 並行して実行されるステップは任意の順序で完了する可能性がありますが、キューに保留中のステップは、送信された順序で実行状態に移行します。
- クラスターのステップ同時実行レベルを選択するときは、プライマリノードインスタンスタイプがユーザーワークロードのメモリ要件を満たしているかどうかを考慮する必要があります。メインステップのエグゼキュータープロセスは、各ステップのプライマリノードで実行されます。複数のステップを並行して実行するには、一度に 1 つのステップを実行するよりも、プライマリノードからのメモリと CPU の使用率が増えます。
- 同時ステップの複雑なスケジューリングとリソース管理を実現するには、FairScheduler や CapacityScheduler などの YARN スケジューリング機能を使用できます。たとえば、queueMaxAppsDefault セットで FairScheduler を使用すると、特定の数を超えるジョブが一度に実行されないようにできます。
- ステップの同時実行レベルは、リソースマネージャの設定によって異なります。たとえば、YARN が 5 の並列処理のみで設定されている場合、StepConcurrencyLevel が 10 に設定されていても、並列して実行できる YARN アプリケーションは 5 つだけです。リソースマネージャーの設定に関する詳細は、「Amazon EMR リリースガイド」の「アプリケーションの設定」を参照してください。
- クラスターのステップ同時実行レベルが 1 より大きいときは、CONTINUE 以外の ActionOnFailure を使用してステップを追加することはできません。
- クラスターのステップ同時実行レベルが 1 より大きい場合、ステップの ActionOnFailure 機能はアクティブになりません。
- クラスターのステップ同時実行レベルが 1 であるが、複数の実行ステップがある場合、TERMINATE_CLUSTER ActionOnFailure はアクティブにできますが、CANCEL_AND_WAIT ActionOnFailure はできません。このエッジケースは、クラスタース

テップの同時実行レベルが 1 より大きいが、複数ステップの実行中に下げられたときに発生します。

- EMR 自動スケーリングを使用し、YARN リソースに基づいてスケールアップおよびスケールダウンして、リソースの競合を防止できます。詳細については、「Amazon EMR 管理ガイド」の「カスタムポリシーによる自動スケーリングをインスタンスグループに使用する」を参照してください。
- ステップの同時レベルを下げると、EMR で、ステップ数を減らす前に実行中のすべてのステップを完了できます。クラスターで実行されている同時ステップが多すぎるためにリソースがなくなった場合は、実行中のステップを手動でキャンセルして、リソースを解放することをお勧めします。

ステップの表示

Amazon EMR が過去 7 日以内に完了したステップは最大 10,000 個まで確認できます。Amazon EMR がいつでも完了した 1,000 個のステップを表示することもできます。この合計数には、ユーザーが送信したステップとシステムステップが含まれます。

クラスターが 1,000 ステップのレコード制限に達すると、Amazon EMR はステータスが COMPLETED、CANCELLED、または FAILED の非アクティブなユーザー送信ステップを 7 日以上削除します。10,000 ステップレコードの制限を超えるステップを送信すると、Amazon EMR は非アクティブな期間に関係なく、非アクティブなユーザー送信ステップレコードを削除します。Amazon EMR は、これらのレコードをログファイルから削除しません。Amazon EMR はそれらを AWS コンソールから削除し、 AWS CLI または API を使用してクラスター情報を取得しても返されません。システムステップレコードは削除されません。

表示できるステップ情報は、クラスター情報の取得で使用するメカニズムによって異なります。以下の表は、使用可能な各オプションで返されるステップ情報を示しています。

オプション	DescribeJobFlow または --describe --jobflow	ListSteps または list-steps
SDK	256 ステップ	最大 10,000 ステップ
Amazon EMR CLI	256 ステップ	該当なし
AWS CLI	該当なし	最大 10,000 ステップ
API	256 ステップ	最大 10,000 ステップ

ステップのキャンセル

保留中および実行中のステップは AWS Management Console、、 AWS CLI、または Amazon EMR API からキャンセルできます。

> **Note**
>
> Amazon EMR コンソールは、再設計され、使いやすくなりました。古いコンソールと新しいコンソールエクスペリエンスの違いについては、「Amazon EMR コンソール」を参照してください。

New console

新しいコンソールを使用してステップをキャンセルするには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で [クラスター] を選択し、更新するクラスターを選択します。
3. クラスターの詳細ページの [ステップ] タブで、キャンセルするステップの横にあるチェックボックスをオンにします。[アクション] ドロップダウンメニューを選択し、[ステップをキャンセル] を選択します。
4. [ステップをキャンセル] ダイアログで、ステップをキャンセルして終了するのを待つか、ステップをキャンセルして強制終了するかを選択します。[Confirm] (確認) を選択します。
5. [ステップ] テーブルのステップのステータスが `CANCELLED` に変わります。

Old console

古いコンソールを使用してステップをキャンセルするには

1. 新しい Amazon EMR コンソールに移動し、サイドナビゲーションから [古いコンソールに切り替え] を選択します。古いコンソールに切り替えたときの動作の詳細については、「Using the old console」を参照してください。
2. [Cluster Details] ページで、[Steps] セクションを展開します。
3. キャンセルするステップごとに、[ステップ] のリストからステップを選択します。次に、[Cancel step (ステップをキャンセル)] を選択します。

4. [Cancel step (ステップをキャンセル)] ダイアログで、デフォルトのオプション [Cancel the step and wait for it to exit (ステップをキャンセルし、終了するまで待機する)] を維持します。プロセスの完了を待機せずに、すぐにステップを終了する場合は、[Cancel the step and force it to exit (ステップをキャンセルして強制終了する)] を選択します。
5. [Cancel step (ステップをキャンセル)] を選択します。

CLI

を使用して で をキャンセルするには AWS CLI

- クラスターとキャンセルするステップを指定して、`aws emr cancel-steps` コマンドを使用します。次の例では、2 つのステップをキャンセルする AWS CLI コマンドについて説明します。

```
aws emr cancel-steps --cluster-id j-2QUAXXXXXXXXX \
--step-ids s-3M8DXXXXXXXXX s-3M8DXXXXXXXXX \
--step-cancellation-option SEND_INTERRUPT
```

Amazon EMR バージョン 5.28.0 では、ステップをキャンセルするときに、`StepCancellationOption` パラメータで次の 2 つのキャンセルオプションのいずれかを選択できます。

- `SEND_INTERRUPT`– これがデフォルトのオプションです。ステップキャンセルリクエストを受信すると、EMR はステップに `SIGTERM` シグナルを送信し、このシグナルをキャッチして子孫ステップのプロセスを終了するか、プロセスが完了するのを待つ `SIGTERM` シグナルハンドラをステップロジックに追加します。
- `TERMINATE_PROCESS`— このオプションを選択すると、EMR はステップとそのすべての子孫プロセスをすぐに終了するプロセスに対して `SIGKILL` シグナルを送信します。

ステップのキャンセルの考慮事項

- 実行中のステップまたは保留中のステップをキャンセルすると、アクティブなステップ数からそのステップが削除されます。
- 実行中のステップをキャンセルしても、`stepConcurrencyLevel` には変更がないとみなされるため、保留中のステップの実行開始が許可されることはありません。
- 実行中のステップをキャンセルしてもステップ `ActionOnFailure` はトリガーされません。

- EMR 5.32.0 以降では、SEND_INTERRUPT StepCancellationOption はステップの子プロセスに SIGTERM シグナルを送信します。この信号を監視して、クリーンアップとシャットダウンを正常に実行する必要があります。TERMINATE_PROCESS StepCancellationOption は、ステップの子プロセスとそのすべての子孫プロセスに SIGKILL シグナルを送信しますが、非同期プロセスは影響を受けません。

クラスターを表示し、モニタリングする

Amazon EMR には、クラスターに関する情報を収集するために使用できるいくつかのツールが用意されています。クラスターに関する情報には、コンソールや CLI から、またはプログラムでアクセスできます。プライマリノードで、標準の Hadoop ウェブインターフェイスおよびログファイルを利用できます。また、 CloudWatch や Ganglia などのモニタリングサービスを使用して、クラスターのパフォーマンスを追跡することもできます。

アプリケーション履歴は、Amazon EMR 5.25.0 以降の Spark 履歴サーバーの「永続」アプリケーション UI を使用してコンソールから入手することもできます。Amazon EMR 6.x では、永続 YARN タイムラインサーバー、および Tez ユーザーインターフェイスも使用できます。これらのサービスはクラスター外でホストされるため、クラスターの終了後 30 日間は、SSH 接続やウェブプロキシを必要とすることなく、アプリケーション履歴にアクセスできます。「アプリケーションの履歴を表示する」を参照してください。

トピック

- クラスターステータスと詳細の表示
- デバッグの詳細ステップ
- アプリケーションの履歴を表示する
- ログファイルを表示する
- Amazon EC2 でクラスターインスタンスを表示する
- CloudWatch イベントとメトリクス
- Ganglia でクラスターアプリケーションメトリクスを表示する
- での Amazon EMR API コールのログ記録 AWS CloudTrail

クラスターステータスと詳細の表示

クラスターを作成すると、終了した後でもそのステータスを監視し、その実行と発生した可能性のあるエラーに関する詳細情報を取得できます。Amazon EMR は、終了したクラスターに関するメタ

データを 2 か月間参照用に保存します。その後、メタデータは削除されます。クラスター履歴からクラスターを削除することはできませんが、 AWS Management Consoleで [Filter (フィルター)] とAWS CLIを使用し、オプションと `list-clusters` コマンドを使って対象のクラスターを注視できます。

クラスターに保存されたアプリケーション履歴は、クラスターが実行中であるか終了されたかにかかわらず、記録後 1 週間アクセスできます。さらに、永続アプリケーションユーザーインターフェイスは、クラスターが終了してから 30 日間、クラスター外にアプリケーション履歴を保存します。「アプリケーションの履歴を表示する」を参照してください。

待機中、実行中などの、クラスターの状態の詳細については、「クラスターライフサイクルについて」を参照してください。

AWS Management Consoleを使用してクラスターの詳細を表示する

https://console.aws.amazon.com/emr のクラスターリストには、終了したクラスターを含む、アカウントと AWS リージョン内のすべてのクラスターが一覧表示されます。このリストには、各クラスターについて、[名前] と [ID]、[ステータス] と [ステータスの詳細]、[作成時刻]、クラスターが実行されていた [経過時間]、クラスター内のすべての EC2 インスタンスで計上された [正規化インスタンス時間] が表示されます。このリストは、クラスターのステータスをモニタリングするための開始点であり、分析とトラブルシューティングのため、各クラスターの詳細にドリルダウンできるよう設計されています。

> **Note**
>
> Amazon EMR コンソールは、再設計され、使いやすくなりました。古いコンソールと新しいコンソールエクスペリエンスの違いについては、「Amazon EMR コンソール」を参照してください。

New console

新しいコンソールを使用してクラスター情報を表示するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で [クラスター] を選択し、表示するクラスターを選択します。

3. [概要] パネルを使用して、クラスターのステータス、Amazon EMR がクラスターにインストールしたオープンソースアプリケーション、クラスターの作成に使用した Amazon EMR のバージョンなど、クラスター設定の基本情報を表示します。以下の表で説明するように、[概要] 配下の各タブを使用して情報を表示します。

Old console

古いコンソールを使用してクラスター情報を表示するには

1. 新しい Amazon EMR コンソールに移動し、サイドナビゲーションから [古いコンソールに切り替え] を選択します。古いコンソールに切り替えたときの動作の詳細については、「Using the old console」を参照してください。
2. クラスター情報の概要を表示するには、[名前] の下にあるクラスターのリンクの横にある下向き矢印を選択します。クラスターの列が展開され、クラスター、ハードウェア、ステップ、およびブートストラップアクションに関する詳細情報が表示されます。このセクションにあるリンクを使用して、詳細を表示します。たとえば、[Steps] のリンクをクリックすると、ステップログファイルへのアクセス、ステップに関連付けられた JAR の表示、ステップのジョブとタスクの詳細の表示、ログファイルへのアクセスができます。
3. クラスター情報の詳細を表示するには、[名前] の下にあるクラスターのリンクを選択して、クラスターの詳細ページを開きます。古いコンソールのクラスターの詳細ページでは、以下の情報を確認できます。

タブ (古いコンソール)	説明 (古いコンソール)
プロパティ	このタブを使用して、クラスターのオペレーティングシステム、クラスターの終了とセキュリティ設定、VPC とサブネットの情報、Amazon S3 のログの保存場所を表示します。
ブートストラップアクション	このタブを使用して、起動時にクラスターが実行するブートストラップアクションのステータスを表示します。ブートストラップアクションは、カスタムソフトウェアのインストールと詳細な設定に使用されます。詳細については、「追加のソフトウェアをインス

タブ (古いコンソール)	説明 (古いコンソール)
	トールするためのブートストラップアクションの作成」を参照してください。
モニタリング	このタブを使用して、クラスターの動作に関する主要なメトリクスを表示します。I/O およびデータストレージに関するクラスターレベルのデータ、ノードレベルのデータ、および情報を表示できます。
インスタンス	このタブを使用して、EC2 インスタンスの ID、DNS 名、EBS ボリュームなど、クラスター内のノードに関する情報を表示します。
ステップ	このタブを使用して、送信したステップのステータスを表示し、ログファイルにアクセスします。ステップの詳細については、「クラスターへの作業の送信」を参照してください。
アプリケーション	このタブを使用して、クラスター外の永続 YARN タイムラインサーバーと Tez UI アプリケーションの詳細を表示します。インストール済みのアプリケーション、クラスター構成、インスタンスグループに関する情報を表示することもできます。クラスターの実行中は、クラスター上のアプリケーションユーザーインターフェイスを使用できます。
イベント	このタブを使用して、クラスターのイベントログを表示します。詳細については、「を使用した Amazon EMR イベントのモニタリング CloudWatch」を参照してください。
タグ	このタブを使用して、クラスターに適用したタグをすべて表示します。

を使用してクラスターの詳細を表示する AWS CLI

以下の例では、 AWS CLIを使用してクラスターの詳細を取得する方法を示します。使用可能なコマンドの詳細については、「Amazon EMR のAWS CLI コマンドリファレンス」を参照してください。ステータス、ハードウェアやソフトウェアの設定、VPC 設定、ブートストラップアクション、インスタンスグループなど、クラスターレベルの詳細を表示するには、describe-cluster コマンドを使用します。クラスターの状態の詳細については、「クラスターライフサイクルについて」を参照してください。次の例は、`describe-cluster` コマンドの使用と、それに続けて list-clusters コマンドの例を示しています。

Example クラスターステータスの表示

`describe-cluster` コマンドを使用するには、クラスター ID が必要です。この例では、特定の日付範囲内に作成されたクラスターの一覧を取得し、返されたクラスター ID の 1 つを使用して、個別のクラスターのステータスに関する詳細情報を表示する方法を示します。

次のコマンドは、クラスター ID で置き換えるクラスター *`j-1K48XXXXXXHCB`* を示します。

```
aws emr describe-cluster --cluster-id j-1K48XXXXXXHCB
```

コマンドの出力は、次のようになります。

```
{
    "Cluster": {
        "Status": {
            "Timeline": {
                "ReadyDateTime": 1438281058.061,
                "CreationDateTime": 1438280702.498
            },
            "State": "WAITING",
            "StateChangeReason": {
                "Message": "Waiting for steps to run"
            }
        },
        "Ec2InstanceAttributes": {
            "EmrManagedMasterSecurityGroup": "sg-cXXXXX0",
            "IamInstanceProfile": "EMR_EC2_DefaultRole",
            "Ec2KeyName": "myKey",
            "Ec2AvailabilityZone": "us-east-1c",
            "EmrManagedSlaveSecurityGroup": "sg-example"
        },
        "Name": "Development Cluster",
```

```
        "ServiceRole": "EMR_DefaultRole",
        "Tags": [],
        "TerminationProtected": false,
        "ReleaseLabel": "emr-4.0.0",
        "NormalizedInstanceHours": 16,
        "InstanceGroups": [
            {
                "RequestedInstanceCount": 1,
                "Status": {
                    "Timeline": {
                        "ReadyDateTime": 1438281058.101,
                        "CreationDateTime": 1438280702.499
                    },
                    "State": "RUNNING",
                    "StateChangeReason": {
                        "Message": ""
                    }
                },
                "Name": "CORE",
                "InstanceGroupType": "CORE",
                "Id": "ig-2EEXAMPLEXXP",
                "Configurations": [],
                "InstanceType": "m5.xlarge",
                "Market": "ON_DEMAND",
                "RunningInstanceCount": 1
            },
            {
                "RequestedInstanceCount": 1,
                "Status": {
                    "Timeline": {
                        "ReadyDateTime": 1438281023.879,
                        "CreationDateTime": 1438280702.499
                    },
                    "State": "RUNNING",
                    "StateChangeReason": {
                        "Message": ""
                    }
                },
                "Name": "MASTER",
                "InstanceGroupType": "MASTER",
                "Id": "ig-2A1234567XP",
                "Configurations": [],
                "InstanceType": "m5.xlarge",
                "Market": "ON_DEMAND",
```

```
                "RunningInstanceCount": 1
            }
        ],
        "Applications": [
            {
                "Version": "1.0.0",
                "Name": "Hive"
            },
            {
                "Version": "2.6.0",
                "Name": "Hadoop"
            },
            {
                "Version": "0.14.0",
                "Name": "Pig"
            },
            {
                "Version": "1.4.1",
                "Name": "Spark"
            }
        ],
        "BootstrapActions": [],
        "MasterPublicDnsName": "ec2-X-X-X-X.compute-1.amazonaws.com",
        "AutoTerminate": false,
        "Id": "j-jobFlowID",
        "Configurations": [
            {
                "Properties": {
                    "hadoop.security.groups.cache.secs": "250"
                },
                "Classification": "core-site"
            },
            {
                "Properties": {
                    "mapreduce.tasktracker.reduce.tasks.maximum": "5",
                    "mapred.tasktracker.map.tasks.maximum": "2",
                    "mapreduce.map.sort.spill.percent": "90"
                },
                "Classification": "mapred-site"
            },
            {
                "Properties": {
                    "hive.join.emit.interval": "1000",
                    "hive.merge.mapfiles": "true"
```

```
                },
                "Classification": "hive-site"
            }
        ]
    }
}
```

Example 作成日に基づくクラスターの一覧表示

特定の日付範囲内に作成されたクラスターを取得するには、`list-clusters` コマンドと `--created-after` パラメータおよび `--created-before` パラメータを使用します。

次のコマンドは、2019 年 10 月 9 日 ~ 2019 年 10 月 12 日に作成されたすべてのクラスターを一覧表示します。

```
aws emr list-clusters --created-after 2019-10-09T00:12:00 --created-
before 2019-10-12T00:12:00
```

Example 状態に基づくクラスターの一覧表示

状態に基づいてクラスターの一覧を表示するには、`list-clusters` コマンドを使用し、`--cluster-states` パラメータを指定します。クラスターの有効な状態は、STARTING、BOOTSTRAPPING、RUNNING、WAITING、TERMINATING、TERMINATED、および TERMINATED_WITH_ERRORS です。

```
aws emr list-clusters --cluster-states TERMINATED
```

また、次のショートカットパラメータを使用して、指定された状態のすべてのクラスターを一覧表示することもできます。

- `--active` は、状態が STARTING、BOOTSTRAPPING、RUNNING、WAITING、または TERMINATING であるクラスターだけをフィルタリングします。
- `--terminated` は、状態が TERMINATED であるクラスターだけをフィルタリングします。
- `--failed` は、状態が TERMINATED_WITH_ERRORS であるクラスターだけをフィルタリングします。

次のコマンドは同じ結果を返します。

```
aws emr list-clusters --cluster-states TERMINATED
```

```
aws emr list-clusters --terminated
```

クラスターの状態の詳細については、「クラスターライフサイクルについて」を参照してください。

デバッグの詳細ステップ

Amazon EMR ステップが失敗し、バージョン 5.x 以降の AMI で API オペレーションを使用して作業を送信した場合、Amazon EMR は、場合によってはステップ障害の根本原因を識別して、関連するログファイル名と、API によるアプリケーションスタック追跡の一部を返すことができます。たとえば、以下の障害を識別できます。

- 出力ディレクトリがすでに存在する、入力ディレクトリが存在しない、またはアプリケーションがメモリ不足になる、といった一般的な Hadoop エラー。
- アプリケーションが互換性がないバージョンの Java でコンパイルされている、または、見つからないメインクラスで実行されている、といった Java のエラー。
- Amazon S3 に格納されたオブジェクトへのアクセスの問題。

この情報は、 DescribeStepおよび ListSteps API オペレーションを使用して使用できます。これらのオペレーションによってStepSummary返される の FailureDetails フィールド。 FailureDetails 情報にアクセスするには、 AWS CLI、コンソール、または AWS SDK を使用します。

> **Note**
>
> Amazon EMR コンソールは、再設計され、使いやすくなりました。古いコンソールと新しいコンソールエクスペリエンスの違いについては、「Amazon EMR コンソール」を参照してください。

New console

新しい Amazon EMR コンソールにはステップのデバッグ機能はありません。ただし、クラスターの終了に関する詳細を次の手順で確認できます。

新しいコンソールを使用して失敗の詳細を表示するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。

2. 左側のナビゲーションペインの [EMR on EC2] で [クラスター] を選択し、表示するクラスターを選択します。
3. クラスターの詳細ページの [概要] セクションに表示される [ステータス] 値を書き留めます。ステータスが [エラーで終了しました] の場合は、テキストにカーソルを合わせるとクラスターの障害の詳細が表示されます。

Old console

古いコンソールを使用して失敗の詳細を表示するには

1. 新しい Amazon EMR コンソールに移動し、サイドナビゲーションから [古いコンソールに切り替え] を選択します。古いコンソールに切り替えたときの動作の詳細については、「Using the old console」を参照してください。
2. クラスターリストを選択し、クラスターを選択します。
3. 詳細を表示するには、各ステップの隣にある矢印アイコンを選択します。ステップが失敗して Amazon EMR が根本原因を識別できる場合、失敗の詳細が表示されます。

CLI

を使用して失敗の詳細を表示するには AWS CLI

- を使用してステップの失敗の詳細を取得するには AWS CLI、 `describe-step` コマンドを使用します。

```
aws emr describe-step --cluster-id j-1K48XXXXXHCB --step-id s-3QM0XXXXXM1W
```

出力は以下のようになります。

```
{
  "Step": {
    "Status": {
      "FailureDetails": {
        "LogFile": "s3://myBucket/logs/j-1K48XXXXXHCB/steps/s-3QM0XXXXXM1W/
stderr.gz",
        "Message": "org.apache.hadoop.mapred.FileAlreadyExistsException: Output
 directory s3://myBucket/logs/beta already exists",
        "Reason": "Output directory already exists."
      },
```

```
      "Timeline": {
        "EndDateTime": 1469034209.143,
        "CreationDateTime": 1469033847.105,
        "StartDateTime": 1469034202.881
      },
      "State": "FAILED",
      "StateChangeReason": {}
    },
    "Config": {
      "Args": [
        "wordcount",
        "s3://myBucket/input/input.txt",
        "s3://myBucket/logs/beta"
      ],
      "Jar": "s3://myBucket/jars/hadoop-mapreduce-examples-2.7.2-amzn-1.jar",
      "Properties": {}
    },
    "Id": "s-3QM0XXXXXM1W",
    "ActionOnFailure": "CONTINUE",
    "Name": "ExampleJob"
  }
}
```

アプリケーションの履歴を表示する

コンソールのクラスターの詳細ページで、Spark 履歴サーバーや YARN タイムラインサーバーアプリケーションの詳細を表示できます。Amazon EMR のアプリケーション履歴を使用すると、アクティブなジョブとジョブ履歴のトラブルシューティングや分析を簡単に行うことができます。

> **Note**
>
> Amazon EMR で使用するオフコンソールアプリケーションのセキュリティを強化するために、アプリケーションホスティングドメインはパブリックサフィックスリスト (PSL) に登録されます。これらのホスティングドメインの例には以下が含まれます: `emrstudio-prod.us-east-1.amazonaws.com`、`emrnotebooks-prod.us-east-1.amazonaws.com`、`emrappui-prod.us-east-1.amazonaws.com`セキュリティ強化のため、デフォルトのドメイン名に機密性の高い Cookie を設定する必要がある場合は、`__Host-` プレフィックスの付いた Cookie を使用することをお勧めします。これは、ク

ロスサイトリクエストフォージェリ (CSRF) 攻撃からドメインを防ぐ際に役立ちます。詳細については、「Mozilla 開発者ネットワーク」の「Set-Cookie」ページを参照してください。

[アプリケーション] タブの [アプリケーションユーザーインターフェイス] セクションには、クラスターの状態とクラスターにインストール済みのアプリケーションに応じて、いくつかの表示オプションが用意されています。

- 永続アプリケーションユーザーインターフェイスへのクラスター外アクセス - Amazon EMR バージョン 5.25.0 以降、Spark UI と Spark 履歴サービスで永続アプリケーションユーザーインターフェイスリンクを使用できます。Amazon EMR バージョン 5.30.1 以降では、Tez UI と YARN タイムラインサーバーにも永続アプリケーションユーザーインターフェイスがあります。YARN タイムラインサーバーと Tez UI は、アクティブなクラスターと終了したクラスターについてのメトリクスを提供するオープンソースアプリケーションです。Spark ユーザーインターフェイスには、スケジューラのステージとタスク、RDD サイズとメモリ使用量、環境に関する情報、動作中の実行プログラムに関する情報が表示されます。永続アプリケーション UI はクラスター外で実行されるため、クラスター情報とログは、アプリケーションの終了後 30 日間使用できます。クラスター上のアプリケーションユーザーインターフェイスとは異なり、永続アプリケーション UI では、SSH 接続を介してウェブプロキシをセットアップする必要はありません。
- クラスター上のアプリケーションユーザーインターフェイス - クラスターで実行できる、さまざまなアプリケーション履歴ユーザーインターフェイスがあります。クラスター上のユーザーインターフェイスはマスターノードでホストされるため、ウェブサーバーへの SSH 接続をセットアップする必要があります。クラスター上のアプリケーションのユーザーインターフェイスは、アプリケーションの終了後、アプリケーションの履歴を 1 週間保持します。SSH トンネルの設定の詳細と手順については、「Amazon EMR クラスターでホストされているウェブインターフェイスを表示する」を参照してください。

  Spark History Server、YARN Timeline Server、Hive アプリケーションを除き、クラスター上のアプリケーション履歴はクラスターの実行中のみ表示できます。

永続アプリケーションユーザーインターフェイスの表示

Amazon EMR バージョン 5.25.0 以降では、クラスターの [概要] ページまたはコンソールの [Application user interfaces] (アプリケーションユーザーインターフェイス) タブを使用して、クラスター外でホストされる永続 Spark History Server アプリケーションの詳細に接続できます。Tez UI および YARN Timeline Server の永続アプリケーションインターフェイスは、Amazon EMR バージョ

ン 5.30.1 以降で利用可能です。永続的なアプリケーション履歴へのワンクリックリンクアクセスには、次の利点があります。

- SSH 接続を介してウェブプロキシを設定することなく、アクティブなジョブとジョブ履歴をすばやく分析してトラブルシューティングできます。
- アクティブなクラスターと終了したクラスターのアプリケーション履歴および関連ログファイルにアクセスできます。ログは、アプリケーションの終了後 30 日間利用できます。

コンソールでクラスターの詳細に移動し、[アプリケーション] タブを選択します。クラスターが起動したら、必要なアプリケーション UI を選択します。アプリケーション UI が新しいブラウザタブで開きます。詳細については、「Monitoring and Instrumentation」を参照してください。

YARN コンテナログは、Spark 履歴サーバー、YARN タイムラインサーバー、および Tez UI 上のリンクから表示できます。

> **Note**
>
> Spark History Server、YARN Timeline Server、および Tez UI から YARN コンテナログにアクセスするには、Amazon S3 へのクラスターのログ記録を有効にする必要があります。ログ記録を有効にしない場合、YARN コンテナログへのリンクは機能しません。

ログの収集

永続アプリケーションユーザーインターフェイスへのワンクリックアクセスを有効にするため、Amazon EMR は次の 2 種類のログを収集します。

- アプリケーションイベントログは、EMR システムバケットに収集されます。イベントログは、Amazon S3 管理のキー (SSE-S3) を使用したサーバー側の暗号化を使用して、保存時に暗号化されます。クラスターにプライベートサブネットを使用している場合は、必ずプライベートサブネットの Amazon S3 ポリシーのリソースリストに “`arn:aws:s3:::prod.MyRegion.appinfo.src/*`” を含めてください。詳細については、「プライベートサブネットの最小 Amazon S3 ポリシー」を参照してください。
- YARN コンテナログは、お客様が所有する Amazon S3 バケットに収集されます。YARN コンテナログにアクセスするには、クラスターのログ記録を有効にする必要があります。詳細については、「クラスターのログ記録とデバッグを設定する」を参照してください。

プライバシー上の理由でこの機能を無効にする必要がある場合は、以下の例に示すように、クラスターを作成するときにブートストラップスクリプトを使用してデーモンを停止できます。

```
aws emr create-cluster --name "Stop Application UI Support" --release-label emr-7.1.0 \
--applications Name=Hadoop Name=Spark --ec2-attributes KeyName=<myEMRKeyPairName> \
--instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m3.xlarge
 InstanceGroupType=CORE,InstanceCount=1,InstanceType=m3.xlarge
 InstanceGroupType=TASK,InstanceCount=1,InstanceType=m3.xlarge \
--use-default-roles --bootstrap-actions Path=s3://region.elasticmapreduce/bootstrap-
actions/run-if,Args=["instance.isMaster=true","echo Stop Application UI | sudo tee /
etc/apppusher/run-apppusher; sudo systemctl stop apppusher || exit 0"]
```

このブートストラップスクリプトを実行すると、Amazon EMR は Spark History Server または YARN Timeline Server のイベントログを EMR システムバケットに収集しません。[Application user interfaces (アプリケーションユーザーインターフェイス)] タブにはアプリケーション履歴情報が表示されず、コンソールからすべてのアプリケーションユーザーインターフェイスにアクセスできなくなります。

大規模な Spark イベントログファイル

Spark ストリーミングなどの長時間実行する Spark ジョブや、Spark SQL クエリなどの大規模なジョブでは、大きなイベントログが生成される場合があります。大きなイベントログが生成されると、コンピューティングインスタンスのディスク容量がすぐに使い果たされ、永続 UI を読み込む際に `OutOfMemory` エラーが発生する可能性があります。このような問題を回避するには、Spark イベントログのローリングと圧縮機能を有効にすることをお勧めします。この機能は、Amazon EMR バージョン emr-6.1.0 以降で使用できます。ローリングと圧縮の詳細については、Spark ドキュメントの「Applying compaction on rolling event log files」を参照してください。

Spark イベントログのローリングおよび圧縮機能を有効にするには、以下の Spark 構成設定を有効にします。

- `spark.eventLog.rolling.enabled` - サイズに基づくイベントログのローリングを有効にします。この設定はデフォルトでは無効になっています。
- `spark.eventLog.rolling.maxFileSize` - ローリングが有効な場合、ロールオーバーする前のイベントログファイルの最大サイズを指定します。デフォルトでは 128 MB です。
- `spark.history.fs.eventLog.rolling.maxFilesToRetain` - 保持する圧縮されていないイベントログファイルの最大数を指定します。デフォルトでは、すべてのイベントログファイルが保持されます。小さい値に設定すると、古いイベントログが圧縮されます。最小値は 1 です。

圧縮では、次のような古いイベントログファイルを含むイベントを除外しようとするため注意してください。イベントが破棄されると、Spark 履歴サーバー UI には表示されなくなります。

- 終了したジョブのイベント、および関連するステージやタスクのイベント
- 終了したエグゼキューターのイベント
- 完了した SQL 問い合わせのイベント、および関連するジョブ、ステージ、タスクのイベント

ローリングと圧縮を有効にしてクラスターを起動するには

1. 次の設定で spark-configuration.json ファイルを作成します。

```
[
   {
     "Classification": "spark-defaults",
        "Properties": {
           "spark.eventLog.rolling.enabled": true,
           "spark.history.fs.eventLog.rolling.maxFilesToRetain": 1
        }
   }
]
```

2. 以下のように Spark のローリング圧縮設定を使用してクラスターを作成します。

```
aws emr create-cluster \
--release-label emr-6.6.0 \
--instance-type m4.large \
--instance-count 2 \
--use-default-roles \
--configurations file://spark-configuration.json
```

考慮事項と制約事項

現在、永続アプリケーションユーザーインターフェイスへのワンクリックアクセスには、次の制限があります。

- Spark History Server UI にアプリケーションの詳細が表示されると、少なくとも 2 分間の遅延が発生します。
- この機能は、アプリケーションのイベントログディレクトリが HDFS にある場合にのみ有効です。デフォルトでは、Amazon EMR は HDFS のディレクトリにイベントログを保存します。デ

フォルトのディレクトリを Amazon S3 などの別のファイルシステムに変更すると、この機能が動作しません。

- この機能は、現在、複数のマスターノードを持つ EMR クラスター、または AWS Lake Formation と統合された EMR クラスターでは利用できません。
- 永続アプリケーションユーザーインターフェイスへのワンクリックアクセスを有効にするには、Amazon EMR の `DescribeCluster` アクションに対するアクセス許可が必要です。このアクションに対する IAM プリンシパルのアクセス許可を拒否した場合、アクセス許可の変更が反映されるまでに 5 分ほどかかります。
- 実行中のクラスターでアプリケーションを再設定すると、アプリケーション UI からアプリケーション履歴を使用できなくなります。
- ごとに AWS アカウント、アクティブなアプリケーション UIsは 200 です。
- 次の では AWS リージョン、Amazon EMR 6.14.0 以降を使用してコンソールからアプリケーション UIs にアクセスできます。
  - アジアパシフィック (ジャカルタ) (ap-southeast-3)
  - 欧州 (スペイン) (eu-south-2)
  - アジアパシフィック (メルボルン) (ap-southeast-4)
  - イスラエル (テルアビブ) (il-central-1)
  - 中東 (UAE) (me-central-1)
- 次の では AWS リージョン、Amazon EMR 5.25.0 以降を使用してコンソールからアプリケーション UIs にアクセスできます。
  - 米国東部 (バージニア北部) (us-east-1)
  - 米国西部 (オレゴン) (us-west-2)
  - アジアパシフィック (ムンバイ) (ap-south-1)
  - アジアパシフィック (ソウル) (ap-northeast-2)
  - アジアパシフィック (シンガポール) (ap-southeast-1)
  - アジアパシフィック (シドニー) (ap-southeast-2)
  - アジアパシフィック (東京) (ap-northeast-1)
  - カナダ (中部) (ca-central-1)
  - 南米 (サンパウロ) (sa-east-1)
  - ヨーロッパ (フランクフルト) (eu-central-1)
  - 欧州 (アイルランド) (eu-west-1)
  - ヨーロッパ (ロンドン) (eu-west-2)

- 欧州 (パリ) (eu-west-3)
- 欧州 (ストックホルム) (eu-north-1)
- 中国 (北京) (cn-north-1)
- 中国 (寧夏) (cn-northwest-1)

アプリケーション履歴の概要を表示する

> **Note**
>
> ユーザーエクスペリエンスを向上させるため、アプリケーションの履歴を最大 30 日間保持する永続アプリケーションインターフェイスを使用することをお勧めします。このページで説明しているアプリケーション履歴の概要機能は、新しい Amazon EMR コンソール (https://console.aws.amazon.com/emr) では利用できません。詳細については、「永続アプリケーションユーザーインターフェイスの表示」を参照してください。

Amazon EMR リリース 5.8.0 から 5.36.0、および 6.8.0 までの 6.x リリースでは、古い Amazon EMR コンソールの [アプリケーションユーザーインターフェイス] タブで、アプリケーション履歴の概要を表示できます。Amazon EMR の [アプリケーションユーザーインターフェイス] では、アプリケーションの使用を終了してから 7 日間アプリケーション履歴の要約を保持します。

考慮事項と制約事項

古い Amazon EMR コンソールの [アプリケーションユーザーインターフェイス] タブを使用するときは、次の制限事項を考慮してください。

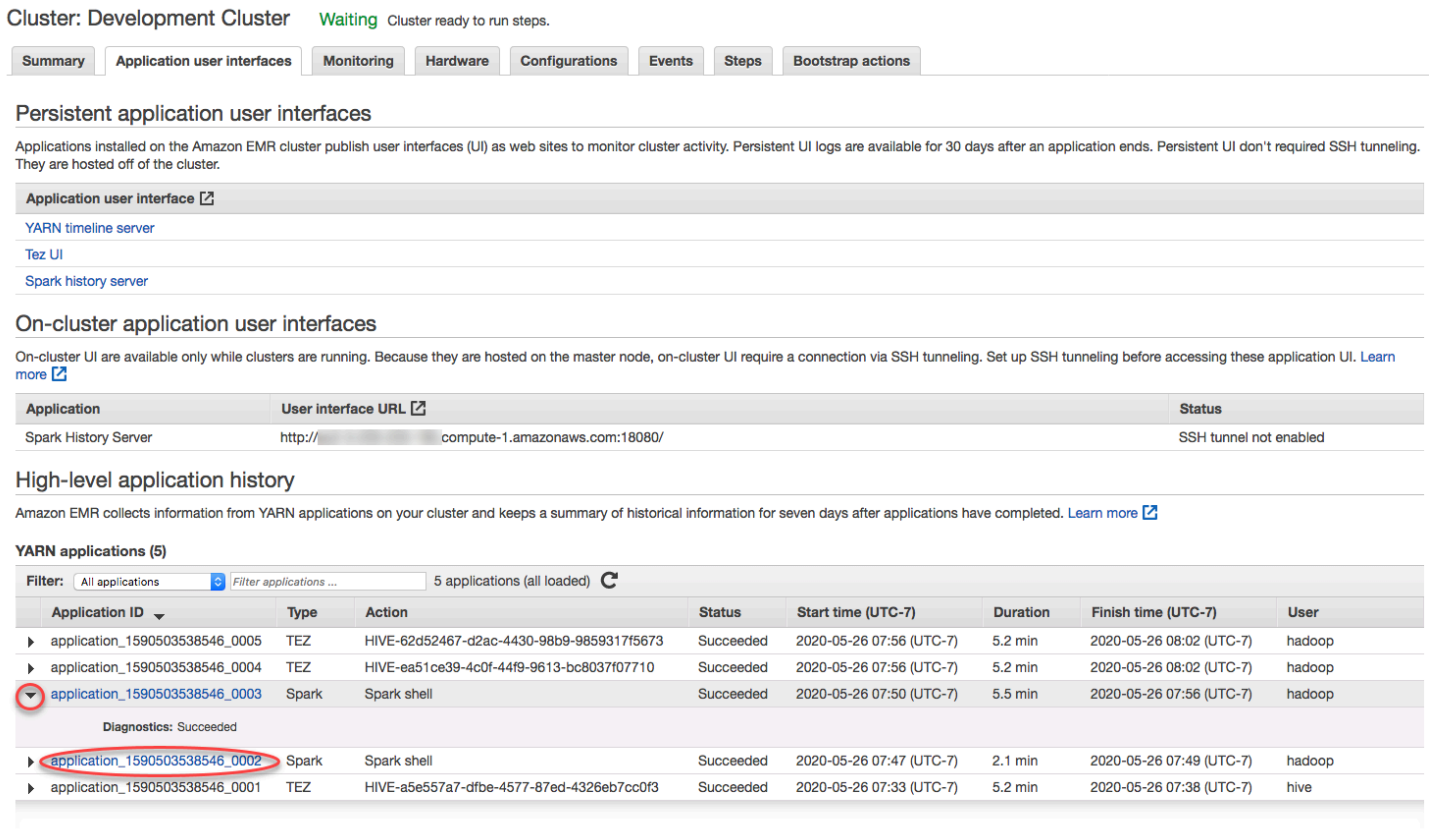
- アプリケーション履歴の概要機能にアクセスできるのは、Amazon EMR リリース 5.8.0 から 5.36.0、および 6.8.0 までの 6.x リリースを使用している場合のみです。2023 年 1 月 23 日をもって、Amazon EMR はすべてのバージョンでアプリケーション履歴の概要機能を廃止します。Amazon EMR バージョン 5.25.0 以降を使用している場合は、代わりに永続アプリケーションユーザーインターフェイスを使用することをお勧めします。
- アプリケーション履歴の概要機能は、Spark ストリーミングアプリケーションをサポートしていません。
- 現在、複数のマスターノードを備えた Amazon EMR クラスターや AWS Lake Formationと統合された Amazon EMR クラスターでは、永続アプリケーションユーザーインターフェイスにワンクリックでアクセスすることはできません。

例: アプリケーション履歴の概要を表示する

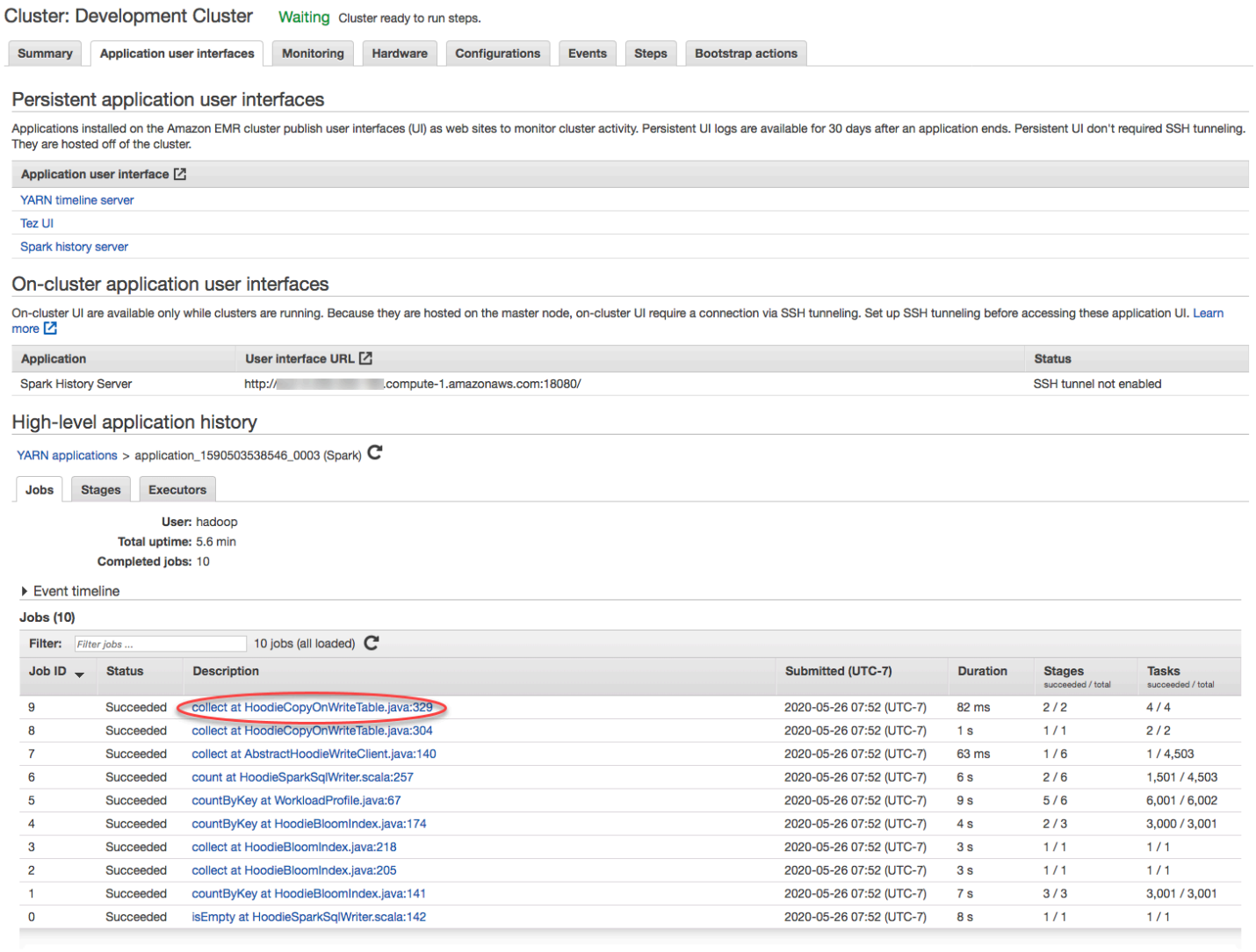
次の手順は、古いコンソールでクラスターの詳細ページの [アプリケーションユーザーインターフェイス] タブを使用して、Spark アプリケーションまたは YARN アプリケーションからジョブの詳細をドリルダウンする方法を示しています。

クラスターの詳細を表示するには、[クラスター] リストからクラスターの [名前] を選択します。YARN コンテナログに関する情報を表示するには、クラスターのログ記録を有効にする必要があります。詳細については、「クラスターのログ記録とデバッグを設定する」を参照してください。Spark アプリケーション履歴について、概要の一覧に表示される情報は、Spark 履歴サーバー UI に表示される情報のみです。

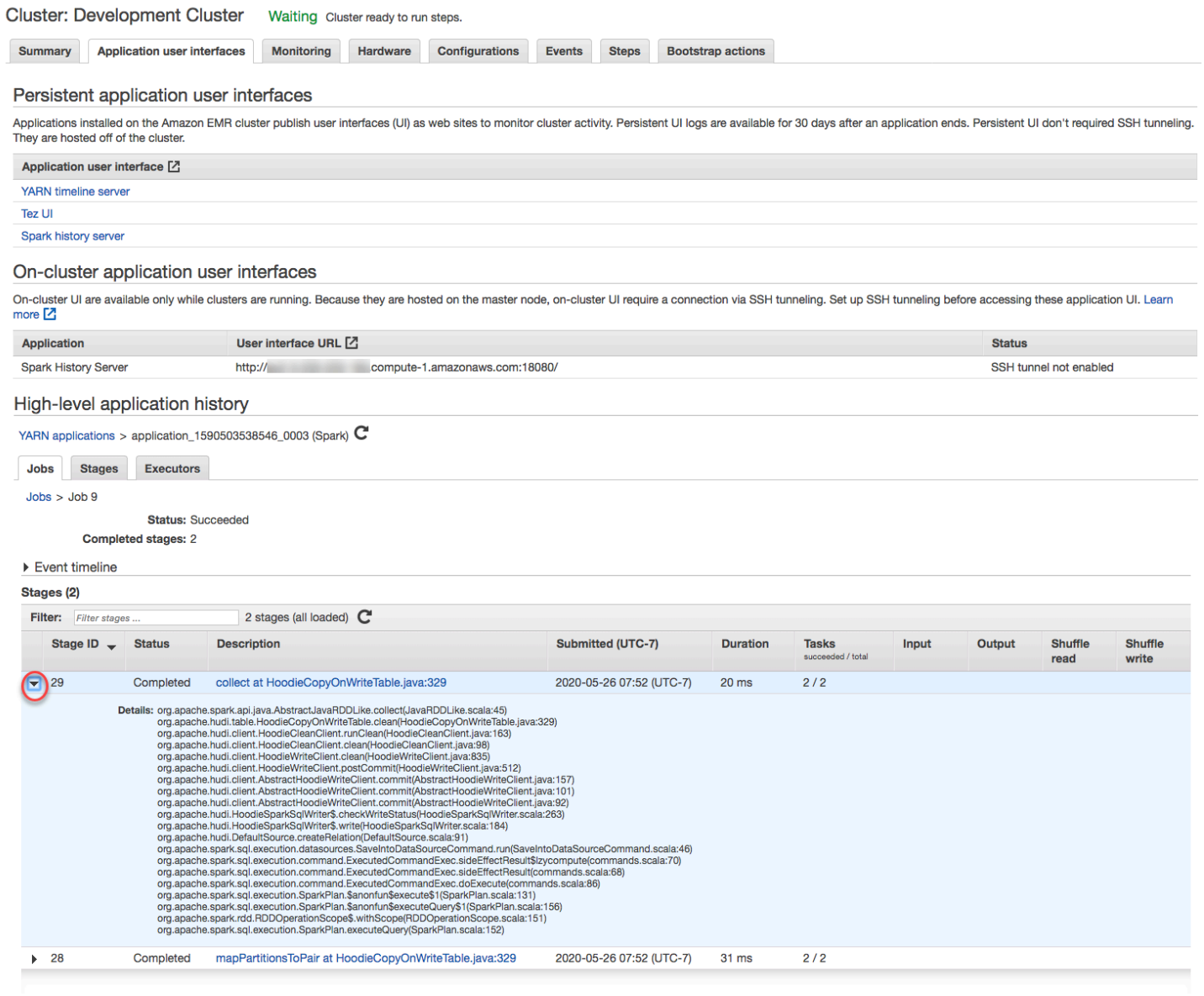
[High-level application history] (アプリケーションの概要履歴) の下にある [Application user interfaces] (アプリケーションユーザーインターフェイス) タブでは、行を展開して Spark アプリケーションの診断サマリーを表示することも、[アプリケーション ID] リンクを選択して別のアプリケーションの詳細を表示することもできます。



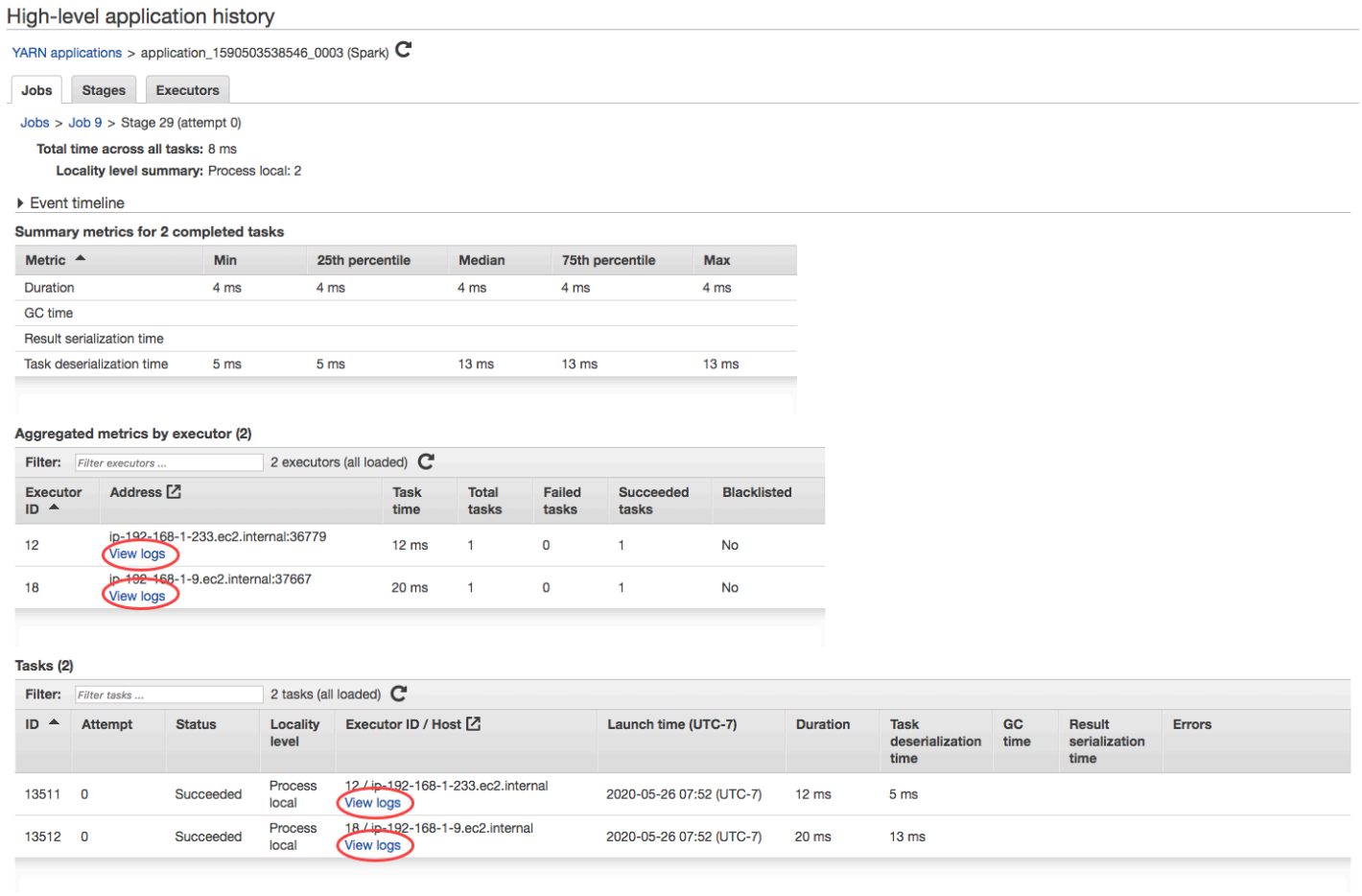
[アプリケーション ID] リンクを選択すると、UI が変更されて、そのアプリケーションの [YARN アプリケーション] の詳細が表示されます。[YARN アプリケーション] の詳細の [ジョブ] タブで、そのジョブの [説明] リンクを選択して、そのジョブの詳細を表示できます。



ジョブの詳細ページで、個々のジョブステージに関する情報を展開し、[説明] リンクを選択して、ステージの詳細を表示できます。



ステージの詳細ページで、ステージタスクとエグゼキューターの主要なメトリクスを表示できます。タスクログとエグゼキューターログは、[ログの表示] リンクを使用して表示することもできます。



ログファイルを表示する

Amazon EMR と Hadoop はいずれも、クラスターの状態を報告するログファイルを生成します。デフォルトで、ログファイルは `/mnt/var/log/` ディレクトリのプライマリノードに出力されます。クラスターの起動時に設定した方法に応じて、これらのログは Amazon S3 にもアーカイブされることがあり、グラフィカルなデバッグツールを使用して表示できる場合があります。

プライマリノードには、さまざまな種類のログが書き込まれます。Amazon EMR は、ステップ、ブートストラップアクション、およびインスタンスの状態の各ログを書き込みます。Apache Hadoop は、ジョブ、タスク、およびタスク試行の処理を報告するログを出力します。また、Hadoop は、デーモンのログも記録します。Hadoop によって書き込まれるログの詳細については、http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/ClusterSetup.html を参照してください。

プライマリノードのログファイルを表示する

次の表は、プライマリノードで確認できるログファイルの一部を示しています。

ロケーション	説明
/emr/instance-controller/log/bootstrap-actions	ブートストラップアクションの処理中に出力されるログ。
/mnt/var/log/hadoop-state-pusher	Hadoop の状態プッシャープロセスで出力されるログ。
/emr/instance-controller/log	インスタンスコントローラログ。
/emr/instance-state	インスタンス状態ログ。ノードの CPU、メモリの状態、およびガベージコレクタースレッドに関する情報が含まれます。
/emr/service-nanny	サービスナニープロセスで出力されるログ
/mnt/var/log/*application*	Hadoop、Spark、Hive などのアプリケーションに固有のログ。
mnt/var/log/hadoop/steps/*N*	ステップの処理に関する情報が含まれステップログ。*N* の値は、Amazon EMR によって割り当てられた stepId を示します。たとえば、クラスターに 2 つのステップ、`s-1234ABCDEFGH` と `s-5678IJKLMNOP` があるとします。最初のステップは `/mnt/var/log/hadoop/steps/s-1234ABCDEFGH/` に配置され、2 番目のステップは `/mnt/var/log/hadoop/steps/s-5678IJKLMNOP/` に配置されます。 Amazon EMR によって書き込まれるステップログは次のとおりです。 • controller— ステップの処理に関する情報。ロード中にステップが失敗した場合は、このログでスタックトレースを見つけることができます。

ロケーション	説明
	• syslog— ステップでの Hadoop ジョブの実行を記録します。 • stderr— Hadoop がステップを処理している間の Hadoop の標準エラーチャネル。 • stdout— Hadoop がステップを処理している間の Hadoop の標準出力チャネル。

AWS CLIでプライマリノードのログファイルを表示するには

1. 「SSH を使用してプライマリノードに接続する」の説明に従って、SSH を使用してプライマリノードに接続します。
2. 表示するログファイル情報が保存されているディレクトリに移動します。前述の表は、使用できるログファイルの種類とその保存場所の一覧です。次の例は、ID が `s-1234ABCDEFGH` のステップログに移動するコマンドを示しています。

   ```
   cd /mnt/var/log/hadoop/steps/s-1234ABCDEFGH/
   ```

3. 任意のファイルビューワーを使用してログファイルを表示します。次の例では、Linux の `less` コマンドを使用して `controller` ログファイルを表示します。

   ```
   less controller
   ```

Amazon S3 にアーカイブされたログファイルを表示する

デフォルトでは、コンソールを使用して起動した Amazon EMR クラスターは自動的に Amazon S3 にログファイルをアーカイブします。独自のログのパスを指定したり、コンソールが自動的にログのパスを生成することを許可したりできます。CLI または API を使用して起動されるクラスターについては、Amazon S3 ログのアーカイブを手動で設定する必要があります。

ログファイルを Amazon S3 にアーカイブするように Amazon EMR を設定すると、*/cluster-id/* フォルダ内の指定した S3 の場所にファイルが保存されます。ここで、*cluster-id* はクラスター ID です。

Amazon S3 で見つかるログファイルの一部を次の表に示します。

ロケーション	説明
/*cluster-id* /node/	ブートストラップアクション、インスタンスの状態、ノードのアプリケーションログを含むノードログ。各ノードのログは、そのノードの EC2 インスタンス識別子が名前に付いたフォルダーに保存されます。
/*cluster-id* /node/*instance-id* /*application*	アプリケーションに関連付けられた各アプリケーションまたはデーモンにより作成されたログ。たとえば、Hive サーバーログは *cluster-id* /node/*instance-id* /hive/hive-server.log にあります。
/*cluster-id* /steps/*step-id*/	ステップの処理に関する情報が含まれステップログ。*step-id* の値は、Amazon EMR によって割り当てられたステップ ID を示します。たとえば、クラスターに 2 つのステップ、s-1234ABCDEFGH と s-5678IJKLMNOP があるとします。最初のステップは /mnt/var/log/hadoop/steps/s-1234ABCDEFGH/ に配置され、2 番目のステップは /mnt/var/log/hadoop/steps/s-5678IJKLMNOP/ に配置されます。 Amazon EMR によって書き込まれるステップログは次のとおりです。 • controller— ステップの処理に関する情報。ロード中にステップが失敗した場合は、このログでスタックトレースを見つけることができます。 • syslog— ステップでの Hadoop ジョブの実行を記録します。 • stderr— Hadoop がステップを処理している間の Hadoop の標準エラーチャネル。

ロケーション	説明
	• stdout— Hadoop がステップを処理している間の Hadoop の標準出力チャネル。
/*cluster-id* /containers	アプリケーションコンテナログ。各 YARN アプリケーションのログは、これらの場所に保存されます。
/*cluster-id* /hadoop-mapreduce/	ジョブの設定の詳細とジョブ履歴 MapReduce に関する情報を含むログ。

Amazon S3 コンソールを使用して Amazon S3 にアーカイブされたログファイルを表示するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
2. Amazon S3 にログファイルをアーカイブするようにクラスターを設定したときに指定した S3 バケットを開きます。
3. 表示する情報を含むログファイルに移動します。前述の表は、使用できるログファイルの種類とその保存場所の一覧です。
4. ログファイルオブジェクトをダウンロードして表示します。手順については、「オブジェクトのダウンロード」を参照してください。

デバッグツールでログファイルを表示する

Amazon EMR では、デバッグツールは自動的に有効になりません。クラスターの起動時にこれを設定する必要があります。新しい Amazon EMR コンソールにはデバッグツールがないことに注意してください。

古いコンソールを使用してクラスターログを表示するには

1. 新しい Amazon EMR コンソールに移動し、サイドナビゲーションから [古いコンソールに切り替え] を選択します。古いコンソールに切り替えたときの動作の詳細については、「Using the old console」を参照してください。
2. クラスターリストページで、表示するクラスターの横にある詳細アイコンを選択します。

これにより、[クラスターの詳細] ページが開きます。[ステップ] セクションの各ステップの右にあるリンクに、そのステップで使用できるさまざまなタイプのログが表示されます。これらのログは Amazon EMR によって生成されます。

3. 特定のステップに関連付けられた Hadoop ジョブのリストを表示するには、ステップの右の [ジョブの表示] リンクを選択します。
4. 特定のジョブに関連付けられた Hadoop タスクのリストを表示するには、ジョブの右の [ジョブの表示] リンクを選択します。
5. 特定のタスクが完了しようとして実行した試行のリストを表示するには、タスクの右の [試行の表示] リンクを選択します。
6. タスク試行によって生成されたログを表示するには、タスク試行の右の [stderr]、[stdout]、および [syslog] の各リンクを選択します。

Amazon EMR がログファイルを Amazon S3 のバケットにアップロードすると、デバッグツールに、そのログファイルへのリンクが表示されます。ログファイルは 5 分ごとに Amazon S3 にアップロードされるため、ステップが完了してからログファイルのアップロードが完了するまでには、数分間かかることがあります。

Amazon EMR は、デバッグツールに表示される Hadoop のジョブ、タスク、およびタスク試行の状態を定期的に更新します。デバッグペインでリストの更新をクリックすると、これらの項目 up-to-date のステータスが最大になります。

Amazon EC2 でクラスターインスタンスを表示する

リソース管理を補助するために、Amazon EC2 ではタグ形式でメタデータをリソースに割り当てることができます。Amazon EC2 の各タグは、キーと値から構成されます。タグを使用すると、Amazon EC2 リソースを目的、所有者、環境などさまざまな方法で分類することができます。

タグに基づいてリソースを検索およびフィルタリングできます。 AWS アカウントを通じてリソースに割り当てるタグは、ユーザーのみが使用できます。同じリソースを共有している他のアカウントから、そのアカウントのタグを見ることはできません。

Amazon EMR では、キー/値ペアを使用して起動する各 EC2 インスタンスに自動的にタグが付けられます。キーによって、クラスターとインスタンスが属するインスタンスグループが識別されます。これにより、EC2 インスタンスのフィルタリングが行いやすくなります。例えば、特定のクラスターに属するインスタンスのみを表示したり、タスクのインスタンスグループで現在実行中のすべてのインスタンスを表示したりできます。これは、複数のクラスターを同時に実行する場合や、大量の EC2 インスタンスを管理する場合に非常に便利です。

Amazon EMR によって割り当てられる事前に定義されたキー/値ペアを次に示します。

キー	値	値の定義
aws:elasticmapreduce:job-flow-id	*job-flow-identifier*	インスタンスがプロビジョニングされているクラスターの ID。`j-XXXXXXXXXXXXX` の形式で表示され、最大 256 文字です。
aws:elasticmapreduce:instance-group-role	*group-role*	インスタンスグループのタイプで、`master`、`core`、または `task` のいずれかの値で入力します。

Amazon EMR が追加するタグに基づいて表示およびフィルタリングできます。詳細については、「Amazon EC2 ユーザーガイド」の「タグの使用」を参照してください。 Amazon EC2 Amazon EMR が設定するタグはシステムタグであり、編集も削除もできないため、タグの表示とフィルタリングに関するセクションが最も役立ちます。

> **Note**
>
> Amazon EMR は、EC2 インスタンスのステータスが [実行中] に更新されるときにタグを追加します。EC2 インスタンスがプロビジョニングされてから、ステータスが [実行中] に設定されるまでの間にレイテンシーがある場合、Amazon EMR によって設定されたタグは、インスタンスが開始すると表示されます。タグが表示されない場合は、数分待ってからビューを更新します。

CloudWatch イベントとメトリクス

イベントとメトリクスを使用して、Amazon EMR クラスターのアクティビティと状態を追跡します。イベントは、クラスター内の特定の発生 (クラスターが状態を開始から実行中に変更するときなど) をモニタリングするのに役立ちます。メトリクスは、特定の値 (HDFS がクラスター内で使用する使用可能なディスク容量の割合など) をモニタリングするのに役立ちます。

CloudWatch イベントの詳細については、「Amazon CloudWatch Events ユーザーガイド」を参照してください。メトリクスの詳細については CloudWatch、「Amazon CloudWatch ユーザーガイド」

の「Amazon メトリクスの使用」および CloudWatch「Amazon アラームの作成」を参照してください。 CloudWatch

トピック

- を使用した Amazon EMR メトリクスのモニタリング CloudWatch
- を使用した Amazon EMR イベントのモニタリング CloudWatch
- CloudWatch イベントへの対応

を使用した Amazon EMR メトリクスのモニタリング CloudWatch

メトリクスは 5 分ごとに更新され、Amazon EMR クラスター CloudWatch ごとに自動的に収集され、 にプッシュされます。この間隔は設定できません。で報告された Amazon EMR メトリクスには料金はかかりません CloudWatch。これらの 5 分のデータポイントメトリクスは 63 日間アーカイブされ、その期間が経過したデータは破棄されます。

Amazon EMR のメトリクスを使用するには、どうすればよいですか?

Amazon EMR で報告されるメトリクスの一般的な使用法を次の表に示します。ここで紹介するのは使用開始するための提案事項であり、総括的な一覧ではありません。Amazon EMR によって報告されるメトリクスの詳細なリストについては、「で Amazon EMR によってレポートされるメトリクス CloudWatch」を参照してください。

どうすればよいか?	関連するメトリクス
クラスターの進行を追跡する	`RunningMapTasks` 、`Remaining MapTasks` 、`RunningReduceTasks` 、および `RemainingReduceTasks` メトリクスを確認します。
アイドル状態のクラスターを検出する	`IsIdle` メトリクスは、クラスターが現在実行されていないライブのタスクかどうかを追跡します。クラスターのアイドル状態の時間が指定した長さ (たとえば、30 分) に達した場合に通知されるようアラームを設定できます。
ノードのストレージがいつ使い果たされるかを検出する	`MRUnhealthyNodes` メトリクスは、1 つ以上のコアノードまたはタスクノードが

どうすればよいか?	関連するメトリクス
	ローカルディスクストレージを使い果たし、UNHEALTHY YARN 状態に移行するタイミングを追跡します。例えば、コアノードやタスクノードのディスク容量が少ないと、タスクを実行できなくなります。
クラスターのストレージがいつ使い果たされるかを検出する	HDFSUtilization メトリクスは、クラスターの合計 HDFS 容量をモニタリングします。クラスターのサイズを変更してコアノードを追加するよう求める場合があります。例えば、HDFS の使用率が高いと、ジョブやクラスターの状態に影響する可能性があります。
クラスターが低容量で稼働している場合に検出する	MRLostNodes メトリクスは、1 つ以上のコアノードまたはタスクノードがマスターノードと通信できなくなった状態を追跡します。例えば、マスターノードがコアノードまたはタスクノードにアクセスできない場合です。

詳細については、NO_SLAVE_LEFT でクラスターが終了し、FAILED_BY_MASTER でコアノードが終了する「」およびAWSSupport「-AnalyzeEMRLogs」を参照してください。

Amazon EMR のアクセス CloudWatch メトリクス

Amazon EMR コンソールまたは コンソール CloudWatch を使用して、Amazon EMR がレポートするメトリクスを表示できますCloudWatch 。 CloudWatch CLI コマンドmon-get-statsまたは CloudWatch GetMetricStatistics API を使用してメトリクスを取得することもできます。を使用した Amazon EMR のメトリクスの表示または取得の詳細については CloudWatch、「Amazon CloudWatch ユーザーガイド」を参照してください。

> **Note**
>
> Amazon EMR コンソールは、再設計され、使いやすくなりました。古いコンソールと新しいコンソールエクスペリエンスの違いについては、「Amazon EMR コンソール」を参照してください。

New console

新しいコンソールを使用してメトリクスを表示するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で [クラスター] を選択し、メトリクスを表示するクラスターを選択します。選択すると、クラスターの詳細ページが開きます。
3. クラスターの詳細ページの [モニタリング] タブを選択します。[クラスターステータス]、[ノードのステータス] または [入力と出力] オプションのいずれか 1 つを選択して、クラスターの進行状況と状態に関するレポートをロードします。
4. 表示するメトリクスを選択したら、各グラフを拡大します。グラフの時間枠を絞り込むには、あらかじめ入力されているオプションを選択するか、[カスタム] を選択します。

Old console

古いコンソールを使用してメトリクスを表示するには

1. Amazon EMR コンソール (https://console.aws.amazon.com/elasticmapreduce/) を開きます。
2. クラスターのメトリクスを表示するには、クラスターを選択して、[Summary] ペインを表示します。
3. [Monitoring] を選択して、クラスターに関する情報を表示します。[クラスターステータス]、[マップ/低減]、[ノードステータス]、または [IO] というタブのいずれか 1 つを選択して、クラスターの進行状況と状態に関するレポートをロードします。
4. 表示するメトリクスを選択したら、グラフサイズを選択できます。[Start] および [End] フィールドを編集して、特定の時間枠にメトリクスを絞り込みます。

で Amazon EMR によってレポートされるメトリクス CloudWatch

次の表に、Amazon EMR がコンソールでレポートし、 にプッシュするメトリクスを示します CloudWatch。

Amazon EMR のメトリクス

Amazon EMR は、複数のメトリクスのデータを に送信します CloudWatch。すべての Amazon EMR クラスターが、5 分ごとにメトリクスを自動送信します。2 週間分のメトリクスがアーカイブされ、その期間が経過したデータは破棄されます。

`AWS/ElasticMapReduce` 名前空間には、次のメトリクスが含まれます。

> **Note**
>
> Amazon EMR は、クラスターからメトリクスを取得します。クラスターにアクセスできなくなると、そのクラスターが再度利用できるようになるまでメトリクスは報告されません。

次のメトリクスは、Hadoop 2.x バージョンを実行しているクラスターで使用できます。

メトリクス	説明
クラスターステータス	
IsIdle	クラスターが作業を行っていないが、まだ有効で課金されていることを示します。タスクもジョブも実行されていない場合は 1 に設定され、それ以外の場合は 0 に設定されます。この値は 5 分間隔で確認され、値が 1 の場合は、確認時にクラスターがアイドル状態だったことのみを示します。5 分間ずっとアイドル状態だったことを示すわけではありません。誤検出を避けるには、5 分ごとの確認で、複数回連続して値が 1 の場合に通知するようアラームを指定する必要があります。たとえば、30 分間にわたってこの値が 1 だった場合に通知するようアラームを指定できます。 ユースケース:クラスターのパフォーマンスを監視する 単位: ブール
ContainerAllocated	によって割り当てられたリソースコンテナの数ResourceManager。 ユースケース:クラスターの進捗状況を監視する

メトリクス	説明
	単位: Count
ContainerReserved	予約されているコンテナの数。 ユースケース:クラスターの進捗状況を監視する 単位: Count
ContainerPending	キュー内にあり、まだ割り当てられていないコンテナの数。 ユースケース:クラスターの進捗状況を監視する 単位: Count
ContainerPendingRatio	割り当てられたコンテナに対する保留中のコンテナの比率 (ContainerPendingRatio = ContainerPending / ContainerAllocated ）。 ContainerAllocated = 0 の場合、 ContainerPendingRatio = ですContainerPending。の値は、パーセンテージではなく数値 ContainerPendingRatio を表します。この値は、コンテナ割り当て動作に基づくクラスターリソースのスケーリングに役立ちます。 単位: Count
AppsCompleted	YARN に送信され、完了したアプリケーションの数。 ユースケース:クラスターの進捗状況を監視する 単位: Count
AppsFailed	YARN に送信され、完了できなかったアプリケーションの数。 ユースケース:クラスターの進捗状況を監視し、クラスターの状態を監視する 単位: Count

メトリクス	説明
AppsKilled	YARN に送信され、強制終了されたアプリケーションの数。 ユースケース:クラスターの進捗状況を監視し、クラスターの状態を監視する 単位: Count
AppsPending	YARN に送信され、保留状態になっているアプリケーションの数。 ユースケース:クラスターの進捗状況を監視する 単位: Count
AppsRunning	YARN に送信され、実行中であるアプリケーションの数。 ユースケース:クラスターの進捗状況を監視する 単位: Count
AppsSubmitted	YARN に送信されたアプリケーションの数。 ユースケース:クラスターの進捗状況を監視する 単位: Count
ノードのステータス	
CoreNodesRunning	動作中のコアノードの数。このメトリクスのデータポイントは、対応するインスタンスグループが存在する場合にのみ報告されます。 ユースケース:クラスターの状態を監視する 単位: Count

メトリクス	説明
CoreNodesPending	割り当て待機中のコアノードの数。リクエストされたすべてのコアノードが直ちに使用可能になるとは限りません。このメトリクスでは、保留中のリクエストを報告します。このメトリクスのデータポイントは、対応するインスタンスグループが存在する場合にのみ報告されます。 ユースケース:クラスターの状態を監視する 単位: Count
LiveDataNodes	Hadoop から処理を受け取るデータノードの割合。 ユースケース:クラスターの状態を監視する 単位: パーセント
MRTotalNodes	MapReduce ジョブで現在使用可能なノードの数。YARN メトリクス `mapred.resourcemanager.TotalNodes` と同等。 ユースケース:クラスターの進捗状況を監視する 単位: Count
MRActiveNodes	現在実行中の MapReduce タスクまたはジョブのノード数。YARN メトリクス `mapred.resourceman ager.NoOfActiveNodes` と同等。 ユースケース:クラスターの進捗状況を監視する 単位: Count

メトリクス	説明
MRLostNodes	LOST 状態でマーク MapReduce された に割り当てられたノードの数。YARN メトリクス `mapred.resourceman ager.NoOfLostNodes` と同等。 ユースケース:クラスターの状態を監視し、クラスターの進捗状況を監視する 単位: Count
MRUnhealthyNodes	UNHEALTHY 状態でマークされた MapReduce ジョブで使用できるノードの数。YARN メトリクス `mapred.re sourcemanager.NoOfUnhealthyNodes` と同等。 ユースケース:クラスターの進捗状況を監視する 単位: Count
MRDecommissionedNodes	DECOMMISSIONED 状態でマークされた MapReduce アプリケーションに割り当てられたノードの数。YARN メトリクス `mapred.resourcemanager.NoOfDecommiss ionedNodes` と同等。 ユースケース:クラスターの状態を監視し、クラスターの進捗状況を監視する 単位: Count
MRRebootedNodes	REBOOTED 状態で再起動およびマーク MapReduce されたで使用可能なノードの数。YARN メトリクス `mapred.re sourcemanager.NoOfRebootedNodes` と同等。 ユースケース:クラスターの状態を監視し、クラスターの進捗状況を監視する 単位: Count

メトリクス	説明
MultiMasterInstanceGroupNodesRunning	実行中のマスターノードの数。 ユースケース: マスターノードの障害のモニタリングと置換 単位: Count
MultiMasterInstanceGroupNodesRunningPercentage	リクエストされたマスターノードインスタンス数で実行中のマスターノードの割合 ユースケース: マスターノードの障害のモニタリングと置換 単位: パーセント
MultiMasterInstanceGroupNodesRequested	リクエストされたマスターノードの数。 ユースケース: マスターノードの障害のモニタリングと置換 単位: Count
IO	
S3BytesWritten	Amazon S3 に書き込まれたバイト数。このメトリクスは MapReduce ジョブのみを集約し、Amazon EMR の他のワークロードには適用されません。 ユースケース:クラスターのパフォーマンスを分析し、クラスターの進捗状況を監視する 単位: Count
S3BytesRead	Amazon S3 から読み込まれたバイト数。このメトリクスはMapReduce ジョブのみを集約し、Amazon EMR の他のワークロードには適用されません。 ユースケース:クラスターのパフォーマンスを分析し、クラスターの進捗状況を監視する 単位: Count

メトリクス	説明
HDFSUtilization	現在使用されている HDFS ストレージの割合。 ユースケース:クラスターのパフォーマンスを分析する 単位: パーセント
HDFSBytesRead	HDFS からの読み込みバイト数。このメトリクスは MapReduce ジョブのみを集約し、Amazon EMR の他のワークロードには適用されません。 ユースケース:クラスターのパフォーマンスを分析し、クラスターの進捗状況を監視する 単位: Count
HDFSBytesWritten	HDFS への書き込みバイト数。このメトリクスはMapReduce ジョブのみを集約し、Amazon EMR の他のワークロードには適用されません。 ユースケース:クラスターのパフォーマンスを分析し、クラスターの進捗状況を監視する 単位: Count
MissingBlocks	HDFS にレプリカがないブロックの数。これらのブロックは破損している可能性があります。 ユースケース:クラスターの状態を監視する 単位: Count
CorruptBlocks	HDFS が破損しているものと報告するブロックの数。 ユースケース:クラスターの状態を監視する 単位: Count

メトリクス	説明
TotalLoad	同時データ転送数の合計。 ユースケース:クラスターの状態を監視する 単位: Count
MemoryTotalMB	クラスターでのメモリの合計量。 ユースケース:クラスターの進捗状況を監視する 単位: Count
MemoryReservedMB	予約されているメモリの量。 ユースケース:クラスターの進捗状況を監視する 単位: Count
MemoryAvailableMB	割り当てに使用できるメモリの量。 ユースケース:クラスターの進捗状況を監視する 単位: Count
YARNMemoryAvailablePercentage	YARN で使用できる残りのメモリの割合 (YARN MemoryAvailablePercentage = MemoryAvailableMB/ MemoryTotalMB)。この値は、YARN のメモリの使用状況に基づくクラスターリソースのスケーリングに役立ちます。 単位: パーセント
MemoryAllocatedMB	クラスターに割り当てられているメモリの量。 ユースケース:クラスターの進捗状況を監視する 単位: Count

メトリクス	説明
PendingDeletionBlocks	削除用にマークされているブロックの数。 ユースケース:クラスターの進捗状況を監視し、クラスターの状態を監視する 単位: Count
UnderReplicatedBlocks	1 回以上レプリケートする必要があるブロックの数。 ユースケース:クラスターの進捗状況を監視し、クラスターの状態を監視する 単位: Count
DfsPendingReplicationBlocks	ブロックレプリケーションのステータス: レプリケーション中のブロック、レプリケーションリクエストの有効期間、および成功しなかったレプリケーションリクエスト。 ユースケース:クラスターの進捗状況を監視し、クラスターの状態を監視する 単位: Count
CapacityRemainingGB	残りの HDFS ディスク容量の合計。 ユースケース:クラスターの進捗状況を監視し、クラスターの状態を監視する 単位: Count

次に示すのは Hadoop の 1 メトリクスです。

メトリクス	説明
クラスターステータス	
IsIdle	クラスターが作業を行っていないが、まだ有効で課金されていることを示します。タスクもジョブも実行されていな

メトリクス	説明
	い場合は 1 に設定され、それ以外の場合は 0 に設定されます。この値は 5 分間隔で確認され、値が 1 の場合は、確認時にクラスターがアイドル状態だったことのみを示します。5 分間ずっとアイドル状態だったことを示すわけではありません。誤検出を避けるには、5 分ごとの確認で、複数回連続して値が 1 の場合に通知するようアラームを指定する必要があります。たとえば、30 分間にわたってこの値が 1 だった場合に通知するようアラームを指定できます。 ユースケース:クラスターのパフォーマンスを監視する 単位: ブール
JobsRunning	クラスターの、現在実行中のジョブの数。 ユースケース:クラスターの状態を監視する 単位: Count
JobsFailed	クラスターの、失敗したジョブの数。 ユースケース:クラスターの状態を監視する 単位: Count
マップ/低減	
MapTasksRunning	各ジョブの実行マップタスクの数。スケジューラがインストールされ、複数のジョブが実行中の場合は、複数のグラフが生成されます。 ユースケース:クラスターの進捗状況を監視する 単位: Count

メトリクス	説明
MapTasksRemaining	各ジョブの残存マップタスクの数。スケジューラがインストールされ、複数のジョブが実行中の場合は、複数のグラフが生成されます。残存マップタスクとは、実行中、強制終了済み、完了済み以外の状態のものをいいます。 ユースケース:クラスターの進捗状況を監視する 単位: Count
MapSlotsOpen	未使用のマップタスクの容量。指定されたクラスターのマップタスクの最大数から、そのクラスターで現在実行中のマップタスクの合計数を差し引いて算出します。 ユースケース:クラスターのパフォーマンスを分析する 単位: Count
RemainingMapTasksPerSlot	合計残存マップタスク数と、クラスター内で使用できる合計マップスロット数の比率。 ユースケース:クラスターのパフォーマンスを分析する 単位: Ratio
ReduceTasksRunning	各ジョブの実行削減タスクの数。スケジューラがインストールされ、複数のジョブが実行中の場合は、複数のグラフが生成されます。 ユースケース:クラスターの進捗状況を監視する 単位: Count
ReduceTasksRemaining	各ジョブの残存削減タスクの数。スケジューラがインストールされ、複数のジョブが実行中の場合は、複数のグラフが生成されます。 ユースケース:クラスターの進捗状況を監視する 単位: Count

メトリクス	説明
ReduceSlotsOpen	未使用の削減タスクの容量。指定されたクラスターの削減タスクの最大キャパシティーから、そのクラスターで現在実行中の削減タスクの数を差し引いて算出します。 ユースケース:クラスターのパフォーマンスを分析する 単位: Count
ノードのステータス	
CoreNodesRunning	動作中のコアノードの数。このメトリクスのデータポイントは、対応するインスタンスグループが存在する場合にのみ報告されます。 ユースケース:クラスターの状態を監視する 単位: Count
CoreNodesPending	割り当て待機中のコアノードの数。リクエストされたすべてのコアノードが直ちに使用可能になるとは限りません。このメトリクスでは、保留中のリクエストを報告します。このメトリクスのデータポイントは、対応するインスタンスグループが存在する場合にのみ報告されます。 ユースケース:クラスターの状態を監視する 単位: Count
LiveDataNodes	Hadoop から処理を受け取るデータノードの割合。 ユースケース:クラスターの状態を監視する 単位: パーセント

メトリクス	説明
TaskNodesRunning	動作中のタスクノードの数。このメトリクスのデータポイントは、対応するインスタンスグループが存在する場合にのみ報告されます。 ユースケース:クラスターの状態を監視する 単位: Count
TaskNodesPending	割り当て待機中のタスクノードの数。リクエストされたすべてのタスクノードが直ちに使用可能になるとは限りません。このメトリクスでは、保留中のリクエストを報告します。このメトリクスのデータポイントは、対応するインスタンスグループが存在する場合にのみ報告されます。 ユースケース:クラスターの状態を監視する 単位: Count
LiveTaskTrackers	機能しているタスクトラッカーの割合。 ユースケース:クラスターの状態を監視する 単位: パーセント
IO	
S3BytesWritten	Amazon S3 に書き込まれたバイト数。このメトリクスは MapReduce ジョブのみを集約し、Amazon EMR の他のワークロードには適用されません。 ユースケース:クラスターのパフォーマンスを分析し、クラスターの進捗状況を監視する 単位: Count

メトリクス	説明
S3BytesRead	Amazon S3 から読み込まれたバイト数。このメトリクスはMapReduce ジョブのみを集約し、Amazon EMR の他のワークロードには適用されません。 ユースケース:クラスターのパフォーマンスを分析し、クラスターの進捗状況を監視する 単位: Count
HDFSUtilization	現在使用されている HDFS ストレージの割合。 ユースケース:クラスターのパフォーマンスを分析する 単位: パーセント
HDFSBytesRead	HDFS からの読み込みバイト数。 ユースケース:クラスターのパフォーマンスを分析し、クラスターの進捗状況を監視する 単位: Count
HDFSBytesWritten	HDFS への書き込みバイト数。 ユースケース:クラスターのパフォーマンスを分析し、クラスターの進捗状況を監視する 単位: Count
MissingBlocks	HDFS にレプリカがないブロックの数。これらのブロックは破損している可能性があります。 ユースケース:クラスターの状態を監視する 単位: Count

メトリクス	説明
TotalLoad	クラスター DataNodes 内のすべての によって報告された現在のリーダーとライターの合計数。 ユースケース: 高い I/O がジョブ実行パフォーマンスの低下につながる可能性の度合いを診断します。 DataNode デーモンを実行しているワーカーノードもマップを実行し、タスクを減らす必要があります。時間の経過とともに TotalLoad 値が永続的に高い場合は、I/O が高いことがパフォーマンスの低下の一因である可能性があります。この値の一時的なスパイクは一般的なものであり、通常問題を示すものではありません。 単位: Count

クラスター容量メトリクス

次のメトリクスは、クラスターの現在の容量またはターゲットの容量を示します。これらのメトリクスは、マネージドスケーリングまたは自動終了が有効になっているときにのみ使用できます。

インスタンスフリートで構成されるクラスターの場合、クラスター容量のメトリクスは Units 単位で測定されます。インスタンスグループで構成されるクラスターの場合、クラスター容量のメトリクスは、マネージドスケーリングポリシーで使用される単位タイプに基づき、Nodes 単位または VCPU 単位で測定されます。詳細については、「Amazon EMR 管理ガイド」の「EMR マネージドスケーリングの使用」を参照してください。

メトリクス	説明
• TotalUnitsRequested • TotalNodesRequested • TotalVCPURequested	マネージドスケーリングによって決定された、クラスター内の単位/ノード/vCPU の合計ターゲット数。 単位: Count
• TotalUnitsRunning	実行中のクラスターで使用可能な単位/ノード/vCPU の現在の合計数。クラスターのサイズ変更がリクエストされる

メトリクス	説明
• TotalNodesRunning • TotalVCPURunning	と、クラスターに新しいインスタンスが追加または削除された後に、このメトリクスが更新されます。 単位: Count
• CoreUnitsRequested • CoreNodesRequested • CoreVCPURequested	マネージドスケーリングによって決定された、クラスター内の CORE 単位/ノード/vCPU のターゲット数。 単位: Count
• CoreUnitsRunning • CoreNodesRunning • CoreVCPURunning	クラスターで実行されている CORE 単位/ノード/vCPU の現在の数。 単位: Count
• TaskUnitsRequested • TaskNodesRequested • TaskVCPURequested	マネージドスケーリングによって決定された、クラスター内の TASK 単位/ノード/vCPU のターゲット数。 単位: Count
• TaskUnitsRunning • TaskNodesRunning • TaskVCPURunning	クラスターで実行されている TASK 単位/ノード/vCPU の現在の数。 単位: Count

Amazon EMR は、自動終了ポリシーを使用して自動終了を有効にすると、次のメトリクスを 1 分単位で出力します。一部のメトリクスは Amazon EMR バージョン 6.4.0 以降でのみ利用できます。自動終了の詳細については、「自動終了ポリシーを使用する」を参照してください。

メトリクス	説明
TotalNotebookKernels	クラスターで実行中とアイドル状態のノートブックカーネルの合計数。 このメトリクスは Amazon EMR バージョン 6.4.0 以降でのみ利用できます。
AutoTerminationIsClusterIdle	クラスターが使用中かどうかを示します。 値 0 は、クラスターが次のコンポーネントのいずれかによってアクティブに使用されていることを示します。 • YARN アプリケーション • HDFS • ノートブック • Spark History Server などのクラスター上の UI 値 1 は、クラスターがアイドル状態であることを示します。Amazon EMR は、クラスターの連続的なアイドル状態 (AutoTerminationIsClusterIdle = 1) を検査します。クラスターのアイドル時間が自動終了ポリシーの IdleTimeout 値と等しくなると、Amazon EMR はクラスターを終了します。

Amazon EMR メトリクスのディメンション

Amazon EMR データは、次の表のいずれかのディメンションを使用してフィルタリングできます。

ディメンション	説明
JobFlowId	クラスター ID と同一で、j-XXXXXXXXXXXXX という形式のクラスターの一意の識別子です。この値を見つけるには、Amazon EMR コンソールでクラスターをクリックします。

を使用した Amazon EMR イベントのモニタリング CloudWatch

Amazon EMR はイベントを追跡し、最大 7 日間、そのイベントに関する情報を Amazon EMR コンソールで保持します。Amazon EMR は、クラスター、インスタンスグループ、インスタンスフリート、自動スケーリングポリシー、またはステップの状態に変化があった場合にイベントを記録します。イベントは、イベントの発生日時、影響を受けた要素の詳細、その他の重要なデータポイントを取得します。

次の表は、Amazon EMR のイベントを、イベントが示す状態や状態の変化、イベントの重大度、イベントタイプ、イベントコード、およびイベントメッセージと共にリストしたものです。Amazon EMR はイベントを JSON オブジェクトで表し、自動的にイベントストリームに送信します。JSON オブジェクトは、 CloudWatch イベントを使用してイベント処理のルールを設定するときに重要です。ルールは JSON オブジェクトのパターンを照合するためです。詳細については、「Amazon Events ユーザーガイド」の「イベントとイベントパターン」および「Amazon EMR イベント」を参照してください。 https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/EventTypes.html#emr_event_type CloudWatch

> **Note**
>
> 最も適切な情報を提供できるよう、エラーメッセージは継続的に改善されています。そのため、ワークフローの次のアクションを開始するためにメッセージのテキストを解析しないようにすることをお勧めします。

クラスター起動のイベント

状態や状態の変化	緊急度	イベントタイプ	イベントコード	メッセージ
CREATING	WARN	Amazon EMR インスタンスフリートのプロビジョニング	EC2 provisioning - Insufficient Instance Capacity	インスタンスフリート InstanceFleetID 用の Amazon EMR クラスター ClusterId (ClusterName) を作成できません。Amazon EC2 でインスタンスタイプ [Instancetype1, Instancetype2] のスポット容量が不足し、アベイラビリティーゾーン [Instancetype3, Instancetype4] でインスタンスタイプ [AvailabilityZone1, Avaliabilityzone2] のオンデマン

状態や状態の変化	緊急度	イベントタイプ	イベントコード	メッセージ
				ド容量が不足しています。このイベントへの対応方法の詳細については、こちらのドキュメントを確認してください。

状態や状態の変化	緊急度	イベントタイプ	イベントコード	メッセージ
CREATING	WARN	Amazon EMR インスタンスグループのプロビジョニング	EC2 provisioning - Insufficient Instance Capacity	インスタンスグループ InstancegroupID 用の Amazon EMR クラスター ClusterId (ClusterName) を作成できません。Amazon EC2 でアベイラビリティーゾーン AvailabilityZone のインスタンスタイプ Instancetype の [Spot or On-Demand] 容量が不足しています。このイベントへの対応方法の詳細については、こちらのドキュメントを確認してください。

状態や状態の変化	緊急度	イベントタイプ	イベントコード	メッセージ
STARTING	INFO	EMR クラスターの状態の変更	なし	Amazon EMR クラスター ClusterId (ClusterName) は Time にリクエストされ、作成中です。

状態や状態の変化	緊急度	イベントタイプ	イベントコード	メッセージ
STARTING	INFO	EMR クラスターの状態の変更	なし	**Note** インスタンスフリートが構成されており、Amazon EC2 内で複数のアベイラビリティーゾーンが選択されているクラスターのみに適用されます。 Amazon EMR クラスター ClusterId (ClusterName) は、指定したアベイラビリティーゾーンオプションから選択されたアベイラビリティーゾー

状態や状態の変化	緊急度	イベントタイプ	イベントコード	メッセージ
				ン (AvailabilityZoneID) で作成中です。
STARTING	INFO	EMR クラスターの状態の変更	なし	Amazon EMR クラスター ClusterId (ClusterName) は Time にステップの実行を開始しました。

状態や状態の変化	緊急度	イベントタイプ	イベントコード	メッセージ
WAITING	INFO	EMR クラスターの状態の変更	なし	Amazon EMR クラスター ClusterId (ClusterName) は Time に作成され、使用する準備ができています。 ~ または ~ Amazon EMR クラスター ClusterId (ClusterName) は Time に保留中のすべてのステップの処理を完了しました。 **Note** WAITING 状態のクラスターは、引き続きジョブを処理している可能性があります。

> **Note**
>
> イベントコード `EC2 provisioning - Insufficient Instance Capacity` のイベントは、EMR クラスターでクラスターの作成またはサイズ変更操作中に、Amazon EC2 からインスタンスフリートやインスタンスグループの容量不足エラーが発生した場合に定期的に発行されます。これらのイベントへの対応方法の詳細については、「Amazon EMR クラスターのインスタンス容量不足のイベントに対応する」を参照してください。

クラスター終了のイベント

状態や状態の変化	緊急度	イベントタイプ	イベントコード	メッセージ
TERMINATED	重大度は次に示す状態変更の理由によって異なります。 • **CRITICAL** クラスターが次に示す状態変更の理由のいずれかによって終了した場合: INTERNAL_ERROR 、VALIDN_ERROR 、INSFAILURE 、BOO_FAILURE 、STURE 。 • **INFO** クラスターが次に示す状態変更の	EMR クラスターの状態の変更	なし	Amazon EMR クラスター ClusterId (ClusterName) は、Time に StateChangeReason: Code の理由で終了しました。

状態や状態の変化	緊急度	イベントタイプ	イベントコード	メッセージ
	理由のいずれかによって終了した場合: USER_REQUEST または ALL_STEPS_COMPLETED 。			
TERMINATED_WITH_ERRORS	CRITICAL	EMR クラスターの状態の変更	なし	Amazon EMR クラスター ClusterId (ClusterName) は、Time に StateChangeReason: Code の理由でエラーが発生して終了しました。

インスタンスフリートの状態変更イベント

> **Note**
>
> インスタンスフリート設定は、5.0.0 および 5.0.3 を除く Amazon EMR リリース 4.8.0 以降でのみ使用できます。

状態や状態の変化	緊急度	イベントタイプ	イベントコード	メッセージ
PROVISIONING から WAITING へ	INFO		なし	Amazon EMR クラスター ClusterId (ClusterName) 内のインスタンスフリート InstanceFleetID のプロビジョニングが完了しました。プロビジョニングは Time に開始され、Num 分かかりました。インスタンスフリートには、Num のオンデマンド容量があり、Num のスポット容量があります。ターゲットのオンデマンド容量は Num で、ターゲットのスポット容量は Num でした。
WAITING から RESIZING へ	INFO		なし	Amazon EMR クラスター ClusterId (ClusterName) 内のイン

状態や状態の変化	緊急度	イベントタイプ	イベントコード	メッセージ
				スタンスフリート InstanceFleetID のサイズ変更は、Time に開始されました。インスタンスフリートは、オンデマンド容量 Num からターゲットの Num へ、スポット容量 Num からターゲットの Num へサイズ変更されています。

状態や状態の変化	緊急度	イベントタイプ	イベントコード	メッセージ
RESIZING から WAITING へ	INFO		なし	Amazon EMR クラスター ClusterId （Cluster Name） 内のインスタンスフリート InstanceFleetID のサイズ変更操作が完了しました。サイズ変更は Time に開始し、Num 分かかりました。インスタンスフリートには、Num のオンデマンド容量があり、Num のスポット容量があります。ターゲットのオンデマンド容量は Num で、ターゲットのスポット容量は Num でした。

状態や状態の変化	緊急度	イベントタイプ	イベントコード	メッセージ
RESIZING から WAITING へ	INFO		なし	Amazon EMR クラスター ClusterId （Cluster Name） 内のインスタンスフリート InstanceFleetID のサイズ変更操作がタイムアウトになり、停止しました。サイズ変更は Time に開始し、Num 分後に停止しました。インスタンスフリートには、Num のオンデマンド容量があり、Num のスポット容量があります。ターゲットのオンデマンド容量は Num で、ターゲットのスポット容量は Num でした。

状態や状態の変化	緊急度	イベントタイプ	イベントコード	メッセージ
SUSPENDED	ERROR		なし	Amazon EMR クラスター ClusterId (ClusterName) 内のインスタンスフリート InstanceFleetID は、Time に ReasonDesc の理由で停止されました。
RESIZING	WARNING		なし	Amazon EMR クラスター ClusterId (ClusterName) 内のインスタンスフリート InstanceFleetID のサイズ変更操作は、ReasonDesc の理由で進行していません。

状態や状態の変化	緊急度	イベントタイプ	イベントコード	メッセージ
WAITING または Running	INFO		なし	Amazon EMR クラスター ClusterId （Cluster Name） 内のインスタンスフリート InstanceFleetID のサイズ変更操作は、Amazon EMR がアベイラビリティーゾーン AvailabilityZone にスポット容量を追加している間に完了できませんでした。スポット容量の追加プロビジョニングのリクエストをキャンセルしました。推奨アクションについては「インスタンスとアベイラビリティーゾーンの柔軟性に関するベストプラクティス」を確認し、もう一度試してください。

状態や状態の変化	緊急度	イベントタイプ	イベントコード	メッセージ
WAITING または Running	INFO		なし	Amazon EMR クラスター ClusterId （Cluster Name） 内のインスタンスフリート InstanceFleetID のサイズ変更操作は、Entity によって Time に開始されました。

インスタンスフリートのサイズ変更のイベント

イベントタイプ	緊急度	イベントコード	メッセージ
Amazon EMR インスタンスフリートのサイズ変更	ERROR	Spot Provisioning timeout	Amazon EMR クラスター ClusterId （Cluster Name） 内のインスタンスフリート InstanceFleetID のサイズ変更操作は、AZ AvailabilityZone でのスポット容量の取得中に完了できませんでした。リクエストをキャンセルし、追加のスポット容量のプ

イベントタイプ	緊急度	イベントコード	メッセージ
			ロビジョニングを中止しました。インスタンスフリートはスポット容量 num をプロビジョニングしました。ターゲットのスポット容量は num でした。詳細と推奨アクションについてはこちらのドキュメントページを確認し、もう一度試してください。

イベントタイプ	緊急度	イベントコード	メッセージ
Amazon EMR インスタンスフリートのサイズ変更	ERROR	On-Demand Provisioning timeout	Amazon EMR クラスター ClusterId （Cluster Name） 内のインスタンスフリート InstanceFleetID のサイズ変更操作は、AZ AvailabilityZone でのオンデマンド容量の取得中に完了できませんでした。リクエストをキャンセルし、追加のオンデマンド容量のプロビジョニングを中止しました。インスタンスフリートはオンデマンド容量 num をプロビジョニングしました。ターゲットのオンデマンド容量は num でした。詳細と推奨アクションについてはこちらのドキュメントページを確認し、もう一度試してください。

イベントタイプ	緊急度	イベントコード	メッセージ
Amazon EMR インスタンスフリートのサイズ変更	WARNING	EC2 provisioning - Insufficient Instance Capacity	EMR クラスター ClusterId (ClusterName) 内のインスタンスフリート InstanceFleetID のサイズ変更操作を完了できません。Amazon EC2 でインスタンスタイプ [Instancetype1, Instancetype2] のスポット容量が不足し、アベイラビリティーゾーン [Instancetype3, Instancetype4] でインスタンスタイプ [AvailabilityZone1] のオンデマンド容量が不足しています。現在、インスタンスフリートは num のオンデマンド容量をプロビジョニング済みで、ターゲットのオンデマンド容量は num でした。プロビジョニング済みのスポット容量は num で、ターゲットのスポット容量は num でした。このイベント

イベントタイプ	緊急度	イベントコード	メッセージ
			への対応方法の詳細については、こちらのドキュメントを確認してください。

イベントタイプ	緊急度	イベントコード	メッセージ
Amazon EMR インスタンスフリートのサイズ変更	WARNING	Spot Provisioning Timeout - Continuing Resize	AZ AvailabilityZone の [Instancetype1, Instancetype2] の Amazon EMR クラスター ClusterId (ClusterName) 内のインスタンスフリート ID InstanceFleetID に対して time に開始されたインスタンスフリートのサイズ変更操作で、スポット容量を引き続きプロビジョニングしています。time に開始された前回のサイズ変更操作では、タイムアウト期間が終了したため、Amazon EMR はリクエストされた num 個のインスタンスのうち num 個をインスタンスフリートに追加した後、スポット容量のプロビジョニングを停止しました。詳細については、こちらのドキュメントページを確認してください。

イベントタイプ	緊急度	イベントコード	メッセージ
Amazon EMR インスタンスフリートのサイズ変更	WARNING	On-Demand Provisioning Timeout - Continuing Resize	AZ AvailabilityZone の [Instancetype1, Instancetype2] の Amazon EMR クラスター ClusterId (ClusterName) 内のインスタンスフリート ID InstanceFleetID に対して time に開始されたインスタンスフリートのサイズ変更操作で、オンデマンド容量を引き続きプロビジョニングしています。time に開始された前回のサイズ変更操作では、タイムアウト期間が終了したため、Amazon EMR はリクエストされた num 個のインスタンスのうち num 個をインスタンスフリートに追加した後、オンデマンド容量のプロビジョニングを停止しました。詳細については、こちらのドキュメントページを確認してください。

> **Note**
>
> プロビジョニングのタイムアウトイベントは、タイムアウト時間が過ぎた後に Amazon EMR がフリートのスポット容量またはオンデマンド容量のプロビジョニングを停止した場合に発生します。これらのイベントへの対応方法の詳細については、「Amazon EMR クラスターのインスタンスフリートのサイズ変更タイムアウトイベントに対応する」を参照してください。

インスタンスグループのイベント

イベントタイプ	緊急度	イベントコード	メッセージ
RESIZING から Running へ	INFO	なし	Amazon EMR クラスター ClusterId （ClusterName） 内のインスタンスグループ InstanceGroupID のサイズ変更操作は完了しました。これで Num 個のインスタンスがあります。サイズ変更は Time に開始され、完了するまでに Num 分かかりました。
RUNNING から RESIZING へ	INFO	なし	Amazon EMR クラスター ClusterId (ClusterName) 内のインスタンスグループ InstanceGroupID のサイズ変更は、Time に開始されました。インスタンス数 Num 個から

イベントタイプ	緊急度	イベントコード	メッセージ
			Num 個にサイズ変更をしています。
SUSPENDED	ERROR	なし	Amazon EMR クラスター ClusterId (ClusterName) 内のインスタンスグループ InstanceGroupID は、Time に ReasonDesc の理由で停止されました。
RESIZING	WARNING	なし	Amazon EMR クラスター ClusterId (Cluster Name) 内のインスタンスグループ InstanceGroupID のサイズ変更操作は、ReasonDesc の理由で進行していません。

イベントタイプ	緊急度	イベントコード	メッセージ
Amazon EMR インスタンスグループのサイズ変更	WARNING	EC2 provisioning - Insufficient Instance Capacity	Amazon EC2 でアベイラビリティーゾーン [AvailabilityZone1] のインスタンスタイプ [Instancetype] の Spot/On Demand 容量が不足しているため、EMR クラスター ClusterId (ClusterName) 内のインスタンスグループ InstanceGroupID で time に開始したサイズ変更操作を完了できません。現在、インスタンスグループの実行インスタンス数は num で、リクエストされたインスタンス数は num でした。このイベントへの対応方法の詳細については、こちらのドキュメントを確認してください。

イベントタイプ	緊急度	イベントコード	メッセージ
RUNNING から RESIZING へ	INFO	なし	Amazon EMR クラスター ClusterId (ClusterName) 内のインスタンスグループ InstanceGroupID のサイズ変更は、Entity によって Time に開始されました。

> **Note**
>
> Amazon EMR バージョン 5.21.0 以降では、実行中のクラスター内のインスタンスグループごとに、クラスター設定を上書きして追加の設定分類を指定できます。これを行うには、Amazon EMR コンソール、 AWS Command Line Interface（AWS CLI）、または AWS SDK を使用します。詳細については、「実行中のクラスターのインスタンスグループの設定を指定する」を参照してください。

Amazon EMR の再設定オペレーションのイベントを、イベントが示す状態や状態の変化、イベントの重大度、およびイベントメッセージとともに次の表に示します。

状態や状態の変化	緊急度	メッセージ
RUNNING	INFO	Amazon EMR クラスター ClusterId (ClusterName) 内のインスタンスグループ InstanceGroupID の再設定は、ユーザーによって Time に開始されました。リクエストされた設定のバージョンは、Num です。

状態や状態の変化	緊急度	メッセージ
RECONFIGURING から Running へ	INFO	Amazon EMR クラスター ClusterId (ClusterName) 内のインスタンスグループ InstanceGroupID の再設定操作は完了しました。再設定は Time に開始され、完了まで Num 分かかりました。現在の設定のバージョンは Num です。
RUNNING から RECONFIGURING へ in	INFO	Amazon EMR クラスター ClusterId (ClusterName) 内のインスタンスグループ InstanceGroupID の再設定は、Time に開始されました。バージョン番号 Num からバージョン番号 Num に設定されます。
RESIZING	INFO	Amazon EMR クラスター ClusterId (ClusterName) 内のインスタンスグループ InstanceGroupID の設定バージョン Num に対する再設定操作は、インスタンスグループが State であるため、Time に一時的にブロックされます。

状態や状態の変化	緊急度	メッセージ
RECONFIGURING	INFO	Amazon EMR クラスター ClusterId (ClusterName) 内のインスタンスグループ InstanceGroupID のインスタンス数 Num に対するサイズ変更操作は、インスタンスグループが State であるため、Time に一時的にブロックされます。
RECONFIGURING	WARNING	Amazon EMR クラスター ClusterId (ClusterName) 内のインスタンスグループ InstanceGroupID の再設定操作は、Time に失敗し、失敗までに Num 分間かかりました。失敗した設定のバージョンは Num です。
RECONFIGURING	INFO	Amazon EMR クラスター ClusterId (ClusterName) 内のインスタンスグループ InstanceGroupID の設定を前に正常に稼働していたバージョン番号 Num に戻す操作は、Time に実行されます。新しい設定バージョンは Num です。

状態や状態の変化	緊急度	メッセージ
RECONFIGURING から Running へ	INFO	Amazon EMR クラスター ClusterId (ClusterName) 内のインスタンスグループ InstanceGroupID の設定を前に正常に稼働していたバージョン Num に戻す操作は、Time に正常に終了しました。新しい設定バージョンは Num です。
RECONFIGURING から SUSPENDED へ	CRITICAL	Amazon EMR クラスター ClusterId (ClusterName) 内のインスタンスグループ InstanceGroupID の設定を前に正常に稼働していたバージョン Num に戻す操作は、Time に失敗しました。

自動スケーリングポリシーのイベント

状態や状態の変化	緊急度	メッセージ
PENDING	INFO	自動スケーリングポリシーは、Amazon EMR クラスター ClusterId (Cluster Name) 内のインスタンスグループ InstanceGroupID に Time に追加されました。ポリシーはアタッチメントを保留しています。 ~ または ~ Amazon EMR クラスター ClusterId (Cluster

状態や状態の変化	緊急度	メッセージ
		Name） 内のインスタンスグループ InstanceGroupID の自動スケーリングポリシーは、Time に更新されました。ポリシーはアタッチメントを保留しています。
ATTACHED	INFO	Amazon EMR クラスター ClusterId （Cluster Name） 内のインスタンスグループ InstanceGroupID の自動スケーリングポリシーは、Time にアタッチされました。
DETACHED	INFO	Amazon EMR クラスター ClusterId （Cluster Name） 内のインスタンスグループ InstanceGroupID の自動スケーリングポリシーは、Time にデタッチされました。

状態や状態の変化	緊急度	メッセージ
FAILED	ERROR	Amazon EMR クラスター ClusterId （Cluster Name） 内のインスタンスグループ InstanceGroupID の自動スケーリングポリシーは、アタッチできず、Time に失敗しました。 ~ または ~ Amazon EMR クラスター ClusterId （Cluster Name） 内のインスタンスグループ InstanceGroupID の自動スケーリングポリシーは、デタッチできず、Time に失敗しました。

ステップイベント

状態や状態の変化	緊急度	メッセージ
PENDING	INFO	ステップ StepID （StepName） は Amazon EMR クラスター ClusterId （ClusterName） に Time に追加され、実行待ちです。
CANCEL_PENDING	WARN	Amazon EMR クラスター ClusterId （ClusterName） のステップ StepID （StepName） は Time にキャンセルされ、キャンセル待ちです。

状態や状態の変化	緊急度	メッセージ
RUNNING	INFO	Amazon EMR クラスター ClusterId (ClusterName) のステップ StepID (StepName) は Time に実行を開始しました。
COMPLETED	INFO	Amazon EMR クラスター ClusterId (ClusterName) のステップ StepID (StepName) は Time に実行を完了しました。ステップの実行は Time に開始され、完了するまでに Num 分かかりました。
CANCELLED	WARN	Amazon EMR クラスター ClusterId (ClusterName) 内のクラスターステップ StepID (StepName) に対するキャンセルリクエストは、Time に正常終了し、ステップがキャンセルされました。
FAILED	ERROR	Amazon EMR クラスター ClusterId (ClusterName) のステップ StepID (StepName) は Time に失敗しました。

異常なノード交換イベント

イベントタイプ	緊急度	イベントコード	メッセージ
Amazon EMR の異常なノード交換	INFO	異常なコアノードが検出されました	Amazon EMR は、Amazon EMR クラスター[instance ID (Instance Name)] 内の InstanceGroup/Fleet のコアインスタンス clusterID (ClusterName) が であることを特定しましたUNHEALTHY 。Amazon EMR は、UNHEALTHY インスタンスの復旧または正常な置き換えを試みます。
Amazon EMR の異常なノード交換	INFO	コアノードの異常 - 置換が無効	Amazon EMR は、Amazon EMR クラスター[instance ID (Instance Name)] 内の InstanceGroup/Flee

イベントタイプ	緊急度	イベントコード	メッセージ	
			t のコアインスタンス {clusterID} (ClusterName) が であることを特定しましたUNHEALTHY 。クラスターで正常な異常のあるコアノードの交換を有効にして、復旧できない場合にAmazon EMRがUNHEALTHY インスタンスを適切に置き換えるようにします。	

イベントタイプ	緊急度	イベントコード	メッセージ
Amazon EMR の異常なノード交換	WARN	異常なコアノードが置き換えられていない	Amazon EMR は、理由 clusterID (ClusterName) により、Amazon EMR クラスター内の [instanceID (InstanceName)] InstanceGroup/Fleet のUNHEALTHY コアインスタンスを置き換えることができません。 Note Amazon EMR がコアノードを置き換えられない理由は、シナリオによって異なります。例えば、Amazon EMR がノードを

イベントタイプ	緊急度	イベントコード	メッセージ
			削除できない理由の 1 つは、クラスターに残りのコアノードがないためです。
Amazon EMR の異常なノード交換	INFO	異常なコアノードの復旧	Amazon EMR が Amazon EMR クラスター[instance ID (Instance Name)] の InstanceGroup/Fleet でUNHEALTHY コアインスタンスを復旧しました clusterID (ClusterName)

異常のあるノードの交換の詳細については、「異常のあるノードの交換」を参照してください。

Amazon EMR コンソールを使用してイベントを表示する

各クラスターについて、イベントの簡単なリストを詳細ペインに表示できます。これは発生の降順にイベントを表示します。また、リージョンでのすべてのクラスターのすべてのイベントを、発生の降順に表示することもできます。

ユーザーにリージョンでのすべてのクラスターのイベントを表示しないようにするには、"Effect": "Deny" アクションのアクセス権限を拒否するステートメント (elasticmapreduce:ViewEventsFromAllClustersInConsole) を、ユーザーにアタッチされているポリシーに追加します。

> **Note**
>
> Amazon EMR コンソールは、再設計され、使いやすくなりました。古いコンソールと新しいコンソールエクスペリエンスの違いについては、「Amazon EMR コンソール」を参照してください。

New console

新しいコンソールを使用して、リージョン内のすべてのクラスターのイベントを表示するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で [イベント] を選択します。

新しいコンソールを使用して特定のクラスターのイベントを表示するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で、[クラスター] を選択し、クラスターを選択します。
3. すべてのイベントを表示するには、クラスターの詳細ページの [イベント] タブを選択します。

Old console

古いコンソールを使用して、リージョン内のすべてのクラスターのイベントを表示するには

1. Amazon EMR コンソール (https://console.aws.amazon.com/elasticmapreduce/) を開きます。
2. [Events] を選択します。

古いコンソールを使用して特定のクラスターのイベントを表示するには

1. Amazon EMR コンソール (https://console.aws.amazon.com/elasticmapreduce/) を開きます。
2. [Cluster List] を選択し、クラスターを選択してから、[View details] を選択します。
3. クラスター詳細ペインで、[Events] を選択します。

CloudWatch イベントへの対応

このセクションでは、Amazon EMR がイベントCloudWatch メッセージ として出力する実用的なイベントに応答するさまざまな方法について説明します。

トピック

- を使用した Amazon EMR イベントのルールの作成 CloudWatch
- CloudWatch メトリクスにアラームを設定する
- Amazon EMR クラスターのインスタンス容量不足のイベントに対応する
- Amazon EMR クラスターのインスタンスフリートのサイズ変更タイムアウトイベントに対応する

を使用した Amazon EMR イベントのルールの作成 CloudWatch

Amazon EMR は、イベントを CloudWatch イベントストリームに自動的に送信します。指定パターンに応じてイベントに一致するルールを作成し、メールで通知を送信するなどのアクションを取るようイベントをターゲットにルーティングします。パターンはイベント JSON オブジェクトに対してマッチングされます。Amazon EMR イベントの詳細については、「Amazon Events ユーザーガイド」の「Amazon EMR CloudWatch イベント」を参照してください。

CloudWatch イベントルールの設定については、「イベント でトリガーする CloudWatch ルールの作成」を参照してください。

CloudWatch メトリクスにアラームを設定する

Amazon EMR はメトリクスを Amazon にプッシュします CloudWatch。これに応じて、CloudWatch を使用して Amazon EMR メトリクスにアラームを設定できます。例えば、HDFS 使用率が 80% を超えるたびに E メールを送信 CloudWatch するように でアラームを設定できます。詳細な手順については、「Amazon ユーザーガイド」の CloudWatch 「アラームの作成または編集」を参照してください。 CloudWatch

Amazon EMR クラスターのインスタンス容量不足のイベントに対応する

概要

Amazon EMR クラスターは、選択されたアベイラビリティーゾーンにクラスターの起動またはサイズ変更のリクエストに対応するのに十分な容量がない場合にイベントコード `EC2 provisioning - Insufficient Instance Capacity` を返します。Amazon EMR で容量不足の例外が繰り返し発生し、クラスターの起動またはクラスターのサイズ変更操作のプロビジョニングリクエストに対応できない場合、インスタンスグループとインスタンスフリートの両方でイベントが定期的に発生します。

このページでは、EMR クラスターでこのイベントタイプが発生した場合に最も適切に対応する方法について説明します。

容量不足イベントへの推奨対応

容量不足のイベントには、次のいずれかの方法で対応することをお勧めします。

- 容量が回復するまで待ちます。容量は頻繁に変化するため、容量不足の例外は自然に解消される可能性があります。Amazon EC2 の容量が利用可能になり次第、クラスターが起動またはサイズ変更が完了します。
- または、クラスターを終了して、インスタンスタイプの設定を変更し、更新されたクラスター設定のリクエストを使用して新しいクラスターを作成します。詳細については、「インスタンスとアベイラビリティーゾーンの柔軟性に関するベストプラクティス」を参照してください。

次のセクションで説明するように、容量不足のイベントに対するルールや自動応答を設定することも可能です。

容量不足のイベントからの自動回復

`EC2 provisioning - Insufficient Instance Capacity` などのイベントコードを含む Amazon EMR イベントに対応する自動化機能を構築できます。例えば、次の AWS Lambda 関数は、オンデマンドインスタンスを使用するインスタンスグループで EMR クラスターを終了し、元のリクエストとは異なるインスタンスタイプを含むインスタンスグループで新しい EMR クラスターを作成します。

以下の条件によって自動化プロセスが開始されます。

- プライマリノードまたはコアノードで容量不足のイベントが 20 分を超えて発生しています。

- クラスターは [準備完了] または [待機中] 状態ではありません。EMR クラスターの状態の詳細については、「クラスターライフサイクルについて」を参照してください。

> **Note**
>
> 容量不足の例外に対する自動化プロセスを構築するときは、容量不足のイベントは回復可能である点を考慮してください。容量は頻繁に変化し、Amazon EC2 の容量が利用可能になり次第、クラスターはサイズ変更を再開したり、操作を開始したりします。

Example 容量不足のイベントに対応する関数

```
// Lambda code with Python 3.10 and handler is lambda_function.lambda_handler
// Note: related IAM role requires permission to use Amazon EMR

import json
import boto3
import datetime
from datetime import timezone

INSUFFICIENT_CAPACITY_EXCEPTION_DETAIL_TYPE = "EMR Instance Group Provisioning"
INSUFFICIENT_CAPACITY_EXCEPTION_EVENT_CODE = (
    "EC2 provisioning - Insufficient Instance Capacity"
)
ALLOWED_INSTANCE_TYPES_TO_USE = [
    "m5.xlarge",
    "c5.xlarge",
    "m5.4xlarge",
    "m5.2xlarge",
    "t3.xlarge",
]
CLUSTER_START_ACCEPTABLE_STATES = ["WAITING", "RUNNING"]
CLUSTER_START_SLA = 20

CLIENT = boto3.client("emr", region_name="us-east-1")

# checks if the incoming event is 'EMR Instance Fleet Provisioning' with eventCode 'EC2
 provisioning - Insufficient Instance Capacity'
def is_insufficient_capacity_event(event):
    if not event["detail"]:
        return False
```

```
    else:
        return (
            event["detail-type"] == INSUFFICIENT_CAPACITY_EXCEPTION_DETAIL_TYPE
            and event["detail"]["eventCode"]
            == INSUFFICIENT_CAPACITY_EXCEPTION_EVENT_CODE
        )


# checks if the cluster is eligible for termination
def is_cluster_eligible_for_termination(event, describeClusterResponse):
    # instanceGroupType could be CORE, MASTER OR TASK
    instanceGroupType = event["detail"]["instanceGroupType"]
    clusterCreationTime = describeClusterResponse["Cluster"]["Status"]["Timeline"][
        "CreationDateTime"
    ]
    clusterState = describeClusterResponse["Cluster"]["Status"]["State"]

    now = datetime.datetime.now()
    now = now.replace(tzinfo=timezone.utc)
    isClusterStartSlaBreached = clusterCreationTime < now - datetime.timedelta(
        minutes=CLUSTER_START_SLA
    )

    # Check if instance group receiving Insufficient capacity exception is CORE or
 PRIMARY (MASTER),
    # and it's been more than 20 minutes since cluster was created but the cluster
 state and the cluster state is not updated to RUNNING or WAITING
    if (
        (instanceGroupType == "CORE" or instanceGroupType == "MASTER")
        and isClusterStartSlaBreached
        and clusterState not in CLUSTER_START_ACCEPTABLE_STATES
    ):
        return True
    else:
        return False


# Choose item from the list except the exempt value
def choice_excluding(exempt):
    for i in ALLOWED_INSTANCE_TYPES_TO_USE:
        if i != exempt:
            return i
```

```
# Create a new cluster by choosing different InstanceType.
def create_cluster(event):
    # instanceGroupType cloud be CORE, MASTER OR TASK
    instanceGroupType = event["detail"]["instanceGroupType"]

    # Following two lines assumes that the customer that created the cluster already
 knows which instance types they use in original request
    instanceTypesFromOriginalRequestMaster = "m5.xlarge"
    instanceTypesFromOriginalRequestCore = "m5.xlarge"

    # Select new instance types to include in the new createCluster request
    instanceTypeForMaster = (
        instanceTypesFromOriginalRequestMaster
        if instanceGroupType != "MASTER"
        else choice_excluding(instanceTypesFromOriginalRequestMaster)
    )
    instanceTypeForCore = (
        instanceTypesFromOriginalRequestCore
        if instanceGroupType != "CORE"
        else choice_excluding(instanceTypesFromOriginalRequestCore)
    )

    print("Starting to create cluster...")
    instances = {
        "InstanceGroups": [
            {
                "InstanceRole": "MASTER",
                "InstanceCount": 1,
                "InstanceType": instanceTypeForMaster,
                "Market": "ON_DEMAND",
                "Name": "Master",
            },
            {
                "InstanceRole": "CORE",
                "InstanceCount": 1,
                "InstanceType": instanceTypeForCore,
                "Market": "ON_DEMAND",
                "Name": "Core",
            },
        ]
    }
    response = CLIENT.run_job_flow(
        Name="Test Cluster",
        Instances=instances,
```

```
        VisibleToAllUsers=True,
        JobFlowRole="EMR_EC2_DefaultRole",
        ServiceRole="EMR_DefaultRole",
        ReleaseLabel="emr-6.10.0",
    )

    return response["JobFlowId"]


# Terminated the cluster using clusterId received in an event
def terminate_cluster(event):
    print("Trying to terminate cluster, clusterId: " + event["detail"]["clusterId"])
    response = CLIENT.terminate_job_flows(JobFlowIds=[event["detail"]["clusterId"]])
    print(f"Terminate cluster response: {response}")


def describe_cluster(event):
    response = CLIENT.describe_cluster(ClusterId=event["detail"]["clusterId"])
    return response


def lambda_handler(event, context):
    if is_insufficient_capacity_event(event):
        print(
            "Received insufficient capacity event for instanceGroup, clusterId: "
            + event["detail"]["clusterId"]
        )

        describeClusterResponse = describe_cluster(event)

        shouldTerminateCluster = is_cluster_eligible_for_termination(
            event, describeClusterResponse
        )
        if shouldTerminateCluster:
            terminate_cluster(event)

            clusterId = create_cluster(event)
            print("Created a new cluster, clusterId: " + clusterId)
        else:
            print(
                "Cluster is not eligible for termination, clusterId: "
                + event["detail"]["clusterId"]
            )
```

```
    else:
        print("Received event is not insufficient capacity event, skipping")
```

Amazon EMR クラスターのインスタンスフリートのサイズ変更タイムアウトイベントに対応する

概要

Amazon EMR クラスターは、インスタンスフリートクラスターのサイズ変更操作の実行中にイベントを発行します。プロビジョニングのタイムアウトイベントは、タイムアウト時間が過ぎた後に Amazon EMR がフリートのスポット容量またはオンデマンド容量のプロビジョニングを停止した場合に発生します。タイムアウト時間は、インスタンスフリートのサイズ変更仕様の一部としてユーザーが設定できます。同じインスタンスフリートに対して連続でサイズ変更するシナリオでは、Amazon EMR は現在のサイズ変更操作のタイムアウト時間が過ぎると、`Spot provisioning timeout - continuing resize` または `On-Demand provisioning timeout - continuing resize` イベントを発行します。その後、フリートの次のサイズ変更操作のために容量のプロビジョニングを開始します。

インスタンスフリートのサイズ変更のタイムアウトイベントに対応する

プロビジョニングのタイムアウトイベントには、次のいずれかの方法で対応することをお勧めします。

- サイズ変更の仕様を再確認し、サイズ変更操作を再試行します。容量は頻繁に変化するため、Amazon EC2 の容量が利用可能になり次第、クラスターは正常にサイズ変更されます。厳格な SLA が求められるジョブでは、タイムアウト時間を低い値に設定することをお勧めします。
- または、次のいずれかの方法を取ります。
  - インスタンスやアベイラビリティーゾーンの柔軟性に関するベストプラクティスに基づいて、さまざまなインスタンスタイプを使用して新しいクラスターを起動、または
  - オンデマンド容量でクラスターを起動
- provisioning timeout - continuing resize イベントでは、サイズ変更操作が処理されるのをさらに待つことができます。Amazon EMR は設定されたサイズ変更仕様に従いながら、フリートに対してトリガーされたサイズ変更操作を順次処理し続けます。

次のセクションで説明するように、このイベントに対するルールや自動応答を設定することも可能です。

プロビジョニングのタイムアウトイベントからの自動回復

Spot Provisioning timeout イベントコードを含む Amazon EMR イベントに対応する自動化機能を構築できます。例えば、次の AWS Lambda 関数は、タスクノードにスポットインスタンスを使用するインスタンスフリートを持つ EMR クラスターを終了し、元のリクエストよりも多様なインスタンスタイプを含むインスタンスフリートで新しい EMR クラスターを作成します。この例では、タスクノードに対する Spot Provisioning timeout イベントが発行されると、Lambda 関数の実行がトリガーされます。

Example **Spot Provisioning timeout** イベントに対応する関数の例

```
// Lambda code with Python 3.10 and handler is lambda_function.lambda_handler
// Note: related IAM role requires permission to use Amazon EMR

import json
import boto3
import datetime
from datetime import timezone

SPOT_PROVISIONING_TIMEOUT_EXCEPTION_DETAIL_TYPE = "EMR Instance Fleet Resize"
SPOT_PROVISIONING_TIMEOUT_EXCEPTION_EVENT_CODE = (
    "Spot Provisioning timeout"
)

CLIENT = boto3.client("emr", region_name="us-east-1")

# checks if the incoming event is 'EMR Instance Fleet Resize' with eventCode 'Spot
 provisioning timeout'
def is_spot_provisioning_timeout_event(event):
    if not event["detail"]:
        return False
    else:
        return (
            event["detail-type"] == SPOT_PROVISIONING_TIMEOUT_EXCEPTION_DETAIL_TYPE
            and event["detail"]["eventCode"]
            == SPOT_PROVISIONING_TIMEOUT_EXCEPTION_EVENT_CODE
        )


# checks if the cluster is eligible for termination
def is_cluster_eligible_for_termination(event, describeClusterResponse):
    # instanceFleetType could be CORE, MASTER OR TASK
    instanceFleetType = event["detail"]["instanceFleetType"]
```

```
    # Check if instance fleet receiving Spot provisioning timeout event is TASK
    if (instanceFleetType == "TASK"):
        return True
    else:
        return False


# create a new cluster by choosing different InstanceType.
def create_cluster(event):
    # instanceFleetType cloud be CORE, MASTER OR TASK
    instanceFleetType = event["detail"]["instanceFleetType"]

    # the following two lines assumes that the customer that created the cluster
 already knows which instance types they use in original request
    instanceTypesFromOriginalRequestMaster = "m5.xlarge"
    instanceTypesFromOriginalRequestCore = "m5.xlarge"

    # select new instance types to include in the new createCluster request
    instanceTypesForTask = [
        "m5.xlarge",
        "m5.2xlarge",
        "m5.4xlarge",
        "m5.8xlarge",
        "m5.12xlarge"
    ]

    print("Starting to create cluster...")
    instances = {
        "InstanceFleets": [
            {
                "InstanceFleetType":"MASTER",
                "TargetOnDemandCapacity":1,
                "TargetSpotCapacity":0,
                "InstanceTypeConfigs":[
                    {
                        'InstanceType': instanceTypesFromOriginalRequestMaster,
                        "WeightedCapacity":1,
                    }
                ]
            },
            {
                "InstanceFleetType":"CORE",
                "TargetOnDemandCapacity":1,
```

```
                "TargetSpotCapacity":0,
                "InstanceTypeConfigs":[
                    {
                        'InstanceType': instanceTypesFromOriginalRequestCore,
                        "WeightedCapacity":1,
                    }
                ]
            },
            {
                "InstanceFleetType":"TASK",
                "TargetOnDemandCapacity":0,
                "TargetSpotCapacity":100,
                "LaunchSpecifications":{},
                "InstanceTypeConfigs":[
                    {
                        'InstanceType': instanceTypesForTask[0],
                        "WeightedCapacity":1,
                    },
                    {
                        'InstanceType': instanceTypesForTask[1],
                        "WeightedCapacity":2,
                    },
                    {
                        'InstanceType': instanceTypesForTask[2],
                        "WeightedCapacity":4,
                    },
                    {
                        'InstanceType': instanceTypesForTask[3],
                        "WeightedCapacity":8,
                    },
                    {
                        'InstanceType': instanceTypesForTask[4],
                        "WeightedCapacity":12,
                    }
                ],
                "ResizeSpecifications": {
                    "SpotResizeSpecification": {
                        "TimeoutDurationMinutes": 30
                    }
                }
            }
        ]
    }
    response = CLIENT.run_job_flow(
```

```
        Name="Test Cluster",
        Instances=instances,
        VisibleToAllUsers=True,
        JobFlowRole="EMR_EC2_DefaultRole",
        ServiceRole="EMR_DefaultRole",
        ReleaseLabel="emr-6.10.0",
    )

    return response["JobFlowId"]


# terminated the cluster using clusterId received in an event
def terminate_cluster(event):
    print("Trying to terminate cluster, clusterId: " + event["detail"]["clusterId"])
    response = CLIENT.terminate_job_flows(JobFlowIds=[event["detail"]["clusterId"]])
    print(f"Terminate cluster response: {response}")


def describe_cluster(event):
    response = CLIENT.describe_cluster(ClusterId=event["detail"]["clusterId"])
    return response


def lambda_handler(event, context):
    if is_spot_provisioning_timeout_event(event):
        print(
            "Received spot provisioning timeout event for instanceFleet, clusterId: "
            + event["detail"]["clusterId"]
        )

        describeClusterResponse = describe_cluster(event)

        shouldTerminateCluster = is_cluster_eligible_for_termination(
            event, describeClusterResponse
        )
        if shouldTerminateCluster:
            terminate_cluster(event)

            clusterId = create_cluster(event)
            print("Created a new cluster, clusterId: " + clusterId)
        else:
            print(
                "Cluster is not eligible for termination, clusterId: "
                + event["detail"]["clusterId"]
```

```
            )

    else:
        print("Received event is not spot provisioning timeout event, skipping")
```

Ganglia でクラスターアプリケーションメトリクスを表示する

Ganglia は、4.2 から 6.15 までの Amazon EMR リリースで使用できます。Ganglia はスケーラブルなオープンソースプロジェクトで、パフォーマンスへの影響を最小限に抑えながら、クラスターやグリッドをモニタリングできるように設計されています。クラスターで Ganglia を有効にすると、レポートを生成し、クラスター全体のパフォーマンスを表示するだけでなく、個別のノードインスタンスのパフォーマンスを調べることができます。また、Ganglia は、Hadoop および Spark メトリクスを取り込み、視覚化するように設定されています。詳細については、「Amazon EMR リリースガイド」の「Ganglia」を参照してください。

での Amazon EMR API コールのログ記録 AWS CloudTrail

Amazon EMR は AWS CloudTrail、Amazon EMR のユーザー、ロール、または 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。 は、Amazon EMR のすべての API コールをイベントとして CloudTrail キャプチャします。キャプチャされる呼び出しには、Amazon EMR コンソールからの呼び出しと、Amazon EMR API 操作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、Amazon EMR の CloudTrail イベントなど、Amazon S3 バケットへのイベントの継続的な配信を有効にすることができます。 Amazon S3 証跡を設定しない場合でも、CloudTrail コンソールのイベント履歴 で最新のイベントを表示できます。によって収集された情報を使用して CloudTrail、Amazon EMR に対するリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

の詳細については CloudTrail、「 AWS CloudTrail ユーザーガイド」を参照してください。

の Amazon EMR 情報 CloudTrail

CloudTrail AWS アカウントを作成すると、 がアカウントで有効になります。Amazon EMR でアクティビティが発生すると、そのアクティビティは CloudTrail イベント履歴 の他の AWS サービスイベントとともに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、「イベント履歴 を含む CloudTrail イベントの表示」を参照してください。

Amazon EMR のイベントなど、 AWS アカウント内のイベントの継続的な記録については、証跡を作成します。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ログに記録し、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、それに基づいて行動するように他の 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:

- 追跡を作成するための概要
- CloudTrail がサポートするサービスと統合
- の Amazon SNS 通知の設定 CloudTrail
- 複数のリージョンからの CloudTrail ログファイルの受信と複数のアカウントからの CloudTrail ログファイルの受信

すべての Amazon EMR アクションは によってログに記録 CloudTrail され、Amazon EMR API リファレンス に記載されています。例えば、、、および `DescribeCluster`アクションを呼び出すと`ListCluster`と`RunJobFlow`、 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、誰がリクエストを生成したかという情報が含まれます。アイデンティティ情報は、以下を判別するのに役立ちます:

- リクエストが root または AWS Identity and Access Management (IAM) ユーザーの認証情報を使用して行われたかどうか。
- 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テンポラリ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。

ユーザーではなくプロセスがクラスターを作成する場合は、`principalId` 識別子を使用して、クラスターの作成に関連付けられているユーザーを特定できます。詳細については、CloudTrail userIdentity 要素」を参照してください。

例: Amazon EMR ログファイルのエントリ

証跡は、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できるようにする設定です。 CloudTrail ログファイルには 1 つ以上のログエントリが含まれます。イベントは任意のソース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。

次の例は、 RunJobFlowアクションを示す CloudTrail ログエントリを示しています。


```
{
 "Records": [
 {
         "eventVersion":"1.01",
         "userIdentity":{
            "type":"IAMUser",
            "principalId":"EX_PRINCIPAL_ID",
            "arn":"arn:aws:iam::123456789012:user/temporary-user-xx-7M",
            "accountId":"123456789012",
            "userName":"temporary-user-xx-7M"
         },
         "eventTime":"2018-03-31T17:59:21Z",
         "eventSource":"elasticmapreduce.amazonaws.com",
         "eventName":"RunJobFlow",
         "awsRegion":"us-west-2",
         "sourceIPAddress":"192.0.2.1",
         "userAgent":"aws-sdk-java/unknown-version Linux/xx Java_HotSpot(TM)_64-
Bit_Server_VM/xx",
         "requestParameters":{
            "tags":[
               {
                  "value":"prod",
                  "key":"domain"
               },
               {
                  "value":"us-west-2",
                  "key":"realm"
               },
               {
                  "value":"VERIFICATION",
                  "key":"executionType"
               }
            ],
            "instances":{
               "slaveInstanceType":"m5.xlarge",
               "ec2KeyName":"emr-integtest",
               "instanceCount":1,
               "masterInstanceType":"m5.xlarge",
               "keepJobFlowAliveWhenNoSteps":true,
               "terminationProtected":false
            },
```

```
            "visibleToAllUsers":false,
            "name":"MyCluster",
            "ReleaseLabel":"emr-5.16.0"
         },
         "responseElements":{
            "jobFlowId":"j-2WDJCGEG4E6AJ"
         },
         "requestID":"2f482daf-b8fe-11e3-89e7-75a3d0e071c5",
         "eventID":"b348a38d-f744-4097-8b2a-e68c9b424698"
      },
  ...additional entries
   ]
}
```

クラスターのスケーリングを使用する

需要が変動するワークロードに応じて、Amazon EMR クラスターで使用できる Amazon EC2 インスタンスの数を自動または手動で調整できます。自動スケーリングを使用するには、2 つのオプションがあります。Amazon EMR Managed Scaling を有効にすることも、カスタムの自動スケーリングポリシーを作成することもできます。以下の表では、2 つのオプションの違いについて説明しています。

	Amazon EMR Managed Scaling	カスタム自動スケーリング
スケーリングポリシーとルール	ポリシーは必要ありません。Amazon EMR は、クラスターメトリクスを継続的に評価し、最適化されたスケーリング決定を行うことにより、自動スケーリングアクティビティを管理します。	スケーリングアクティビティ、評価期間、クールダウン期間をトリガーする特定の条件などの、自動スケーリングポリシーとルールを定義して管理する必要があります。
サポートされている Amazon EMR リリース	Amazon EMR バージョン 5.30.0 以降 (Amazon EMR バージョン 6.0.0 を除く)	Amazon EMR バージョン 4.0.0 以降

	Amazon EMR Managed Scaling	カスタム自動スケーリング
サポートされているクラスター構成	インスタンスグループまたはインスタンスフリート	インスタンスグループのみ
スケーリング制限の設定	スケーリング制限は、クラスター全体に対して設定されます。	スケーリング制限は、各インスタンスグループに対してのみ設定できます。
メトリクス評価頻度	5 ~ 10秒ごと メトリクスの評価を頻繁に行うことで、Amazon EMR によるスケーリング決定の精度が高くなります。	評価期間は 5 分単位でのみ定義できます。
サポートされているアプリケーション	Spark、Hadoop、Hive、Flink などの YARN アプリケーションのみがサポートされています。Amazon EMR Managed Scaling は、Presto や HBase などの YARN に基づいていないアプリケーションをサポートしていません。	自動スケーリングルールを定義するときに、サポート対象とするアプリケーションを選択できます。

考慮事項

- Amazon EMR クラスターは、常に 1 つまたは 3 つのプライマリノードで構成されます。クラスターを最初に設定すると、コアノードとタスクノードのみをスケールできます。クラスターのプライマリノードの数をスケールすることはできません。
- インスタンスグループの場合、再設定操作とサイズ変更操作は同時ではなく順番に行われます。インスタンスグループのサイズ変更中に再設定を開始すると、インスタンスグループで実行中のサイズ変更が完了次第、再設定が開始されます。逆も同様で、インスタンスグループの再設定中にサイズ変更操作を開始すると、再設定後にサイズ変更が開始されます。

Amazon EMR でマネージドスケーリングを使用する

> **Important**
>
> マネージドスケーリングには、最新の Amazon EMR リリース (Amazon EMR 7.1.0) を使用することを強くお勧めします。一部の初期のリリースでは、断続的なアプリケーション障害やスケーリングの遅延が発生する可能性があります。Amazon EMR では、5.x リリース 5.30.2、5.31.1、5.32.1、5.33.1 以降、および 6.x リリース 6.1.1、6.2.1、6.3.1 以降でこの問題を解決しました。リージョンと提供リリースの詳細については、「マネージドスケーリングの提供状況」を参照してください。

概要

Amazon EMR バージョン 5.30.0 以降 (Amazon EMR 6.0.0 を除く) では、Amazon EMR Managed Scaling を有効にできます。マネージドスケーリングを使用すると、ワークロードに基づいてクラスター内のインスタンスやユニットの数を自動的に増減できます。Amazon EMR は引き続きクラスターのメトリクスを評価し、クラスターのコストと速度を最適化するためのスケーリングを決定します。マネージドスケーリングは、インスタンスグループとインスタンスフリートのいずれかで構成されるクラスターで使用できます。

マネージドスケーリングの提供状況

- 次の では AWS リージョン、Amazon EMR マネージドスケーリングは Amazon EMR 6.14.0 以降で使用できます。
  - アジアパシフィック (ハイデラバード) (ap-south-2)
  - アジアパシフィック (ジャカルタ) (ap-southeast-3)
  - 欧州 (スペイン) (eu-south-2)
- 次の では AWS リージョン、Amazon EMR マネージドスケーリングは Amazon EMR 5.30.0 および 6.1.0 以降で使用できます。
  - 米国東部 (バージニア北部) (us-east-1)
  - 米国東部 (オハイオ) (us-east-2)
  - 米国西部 (オレゴン) (us-west-2)
  - 米国西部 (北カリフォルニア) (us-west-1)
  - アフリカ (ケープタウン) (af-south-1)
  - アジアパシフィック (香港) (ap-east-1)

- アジアパシフィック (ムンバイ) (ap-south-1)
- アジアパシフィック (ソウル) (ap-northeast-2)
- アジアパシフィック (シンガポール) (ap-southeast-1)
- アジアパシフィック (シドニー) (ap-southeast-2)
- アジアパシフィック (東京) (ap-northeast-1)
- カナダ (中部) (ca-central-1)
- 南米 (サンパウロ) (sa-east-1)
- ヨーロッパ (フランクフルト) (eu-central-1)
- 欧州 (アイルランド) (eu-west-1)
- ヨーロッパ (ロンドン) (eu-west-2)
- 欧州 (ミラノ) (eu-south-1)
- 欧州 (パリ) (eu-west-3)
- 欧州 (ストックホルム) (eu-north-1)
- 中国 (北京) (cn-north-1)
- 中国 (寧夏) (cn-northwest-1)
- AWS GovCloud （ 米国東部) (us-gov-east-1)
- AWS GovCloud （ 米国西部) (us-gov-west-1)

- Amazon EMR Managed Scaling は、Spark、Hadoop、Hive、Flink などの YARN アプリケーションでのみ機能します。Presto や HBase など、YARN に基づいていないアプリケーションはサポートされていません。

マネージドスケーリングのパラメータ

マネージドスケーリングでは、以下のパラメータを設定する必要があります。制限は、コアノードとタスクノードのみに適用されます。初期設定後にプライマリノードをスケールすることはできません。

- 最小(`MinimumCapacityUnits`) — 1 クラスターに許可される EC2 容量の下限。これは、インスタンスグループについては仮想中央処理装置 (vCPU) コアまたはインスタンスで測定されます。インスタンスフリートについてはユニットで測定されます。
- 最大(`MaximumCapacityUnits`) — 1 クラスターに許可される EC2 容量の上限。これは、インスタンスグループについては仮想中央処理装置 (vCPU) コアまたはインスタンスで測定されます。インスタンスフリートについてはユニットで測定されます。

- オンデマンド制限(MaximumOnDemandCapacityUnits) (オプション) — クラスター内のオンデマンドマーケットタイプに許可される EC2 容量の上限。このパラメータが指定されていない場合、デフォルトは MaximumCapacityUnits の値になります。
  - このパラメータは、オンデマンドインスタンスとスポットインスタンスの間で容量割り当てを分割するために使用します。例えば、最小パラメータを 2 インスタンス、最大パラメータを 100 インスタンス、オンデマンド制限を 10 インスタンスに設定した場合、Amazon EMR Managed Scaling は最大 10 のオンデマンドインスタンスにスケールアップし、残りの容量をスポットインスタンスに割り当てます。詳細については、「ノード割り当てシナリオ」を参照してください。
- 最大コアノード(MaximumCoreCapacityUnits) (オプション) — クラスターのコアノードタイプに許可される EC2 キャパシティの上限。このパラメータが指定されていない場合、デフォルトは MaximumCapacityUnits の値になります。
  - このパラメータは、コアノードとタスクノードの間で容量割り当てを分割するために使用します。例えば、最小パラメータを 2 インスタンス、最大を 100 インスタンス、最大コアノードを 17 インスタンスに設定した場合、Amazon EMR Managed Scaling は最大 17 のコアノードにスケールアップし、残りの 83 インスタンスをタスクノードに割り当てます。詳細については、「ノード割り当てシナリオ」を参照してください。

マネージドスケーリングパラメータの詳細については、「ComputeLimits」を参照してください。

Amazon EMR Managed Scaling に関する考慮事項

- マネージドスケーリングは、 AWS リージョン および Amazon EMR の限定リリースでサポートされています。詳細については、「マネージドスケーリングの提供状況」を参照してください。
- Amazon EMR Managed Scaling の必須パラメータを設定する必要があります。詳細については、「マネージドスケーリングのパラメータ」を参照してください。
- マネージドスケーリングを使用するには、metrics-collector プロセスが API Gateway のマネージドスケーリング用のパブリック API エンドポイントに接続できる必要があります。でプライベート DNS 名を使用する場合 Amazon Virtual Private Cloud、マネージドスケーリングは正しく機能しません。マネージドスケーリングが動作するようにするには、次のアクションのうち 1 つを実行することをお勧めします。
  - Amazon VPC から API Gateway のインターフェイス VPC エンドポイントを削除します。
  - 「VPC から API Gateway API に接続するときに「HTTP 403 Forbidden」エラーが発生するのはなぜですか?」の手順に従い、プライベート DNS 名の設定を無効にします。

  - 代わりに、クラスターをプライベートサブネットで起動します。詳細については、「プライベートサブネット」のトピックを参照してください。
- スケールダウン中に YARN ジョブが断続的に遅く、YARN リソースマネージャーのログに、その間にほとんどのノードが拒否リストに記載されたことが示された場合は、廃止タイムアウトしきい値を調整できます。

  `spark.blacklist.decommissioning.timeout` を 1 時間から 1 分に減らして、他の保留中のコンテナがタスク処理の続行にノードを使用できるようにします。

  最長の「Spark タスク」がノード上でまだ実行されている間に、Amazon EMR がノードを強制終了しないように、`YARN.resourcemanager.nodemanager-graceful-decommission-timeout-secs` もより大きな値に設定する必要があります。現在のデフォルトは 60 分です。つまり、YARN は、ノードが廃止状態になってから 60 分後にコンテナを強制終了します。

  以下は YARN リソースマネージャーログの行の例で、ノードが廃止状態に追加されたことを示しています。

```
2021-10-20 15:55:26,994 INFO
 org.apache.hadoop.YARN.server.resourcemanager.DefaultAMSProcessor
 (IPC Server handler 37 on default port 8030): blacklist are updated in
 Scheduler.blacklistAdditions: [ip-10-10-27-207.us-west-2.compute.internal,
 ip-10-10-29-216.us-west-2.compute.internal, ip-10-10-31-13.us-
west-2.compute.internal, ... , ip-10-10-30-77.us-west-2.compute.internal],
 blacklistRemovals: []
```

  Amazon EMR がノードの廃止時に YARN の拒否リストと連携する方法、Amazon EMR のノードが拒否リストに追加されるケース、Spark ノードの廃止動作の設定について詳細を確認してください。
- EBS ボリュームの使用率が高すぎると、マネージドスケーリングの問題が発生する可能性があります。EBS ボリュームの使用率を 90% 未満にすることをお勧めします。詳細については、「インスタンスストレージ」を参照してください。
- Amazon CloudWatch メトリクスは、Amazon EMR マネージドスケーリングを運用するために不可欠です。Amazon CloudWatch メトリクスを注意深くモニタリングして、データが欠落していないことを確認することをお勧めします。欠落しているメトリクスを検出するように CloudWatch アラームを設定する方法の詳細については、「Amazon CloudWatch アラームの使用」を参照してください。

- Presto がインストールされていない 5.30.0 クラスターおよび 5.30.1 クラスターでマネージドスケーリング操作を実行すると、特にスケールダウン操作の後にすぐ、スケールアップ操作が実行されたときに、アプリケーション障害が発生するか、ユニフォームインスタンスグループまたはインスタンスフリートが ARRESTED 状態のままになる場合があります。

  回避策として、ジョブに Presto が必要ない場合でも、Amazon EMR リリース 5.30.0 および 5.30.1 を使用してクラスターを作成するときに、インストールするアプリケーションとして Presto を選択します。

- Amazon EMR Managed Scaling の最大コアノードとオンデマンド制限を設定するときは、インスタンスグループとインスタンスフリートの違いを考慮してください。各インスタンスグループは、オンデマンドインスタンスとスポットインスタンスに対して、同じインスタンスタイプと同じ購入オプションで構成されています。各インスタンスフリートについては、オンデマンドインスタンスとスポットインスタンスとしてプロビジョニングできる 5 つのインスタンスタイプを指定できます。詳細については、「インスタンスフリートまたはユニフォームインスタンスグループを使用してクラスターを作成する」、「インスタンスフリートオプション」、および「ノード割り当てシナリオ」を参照してください。

- Amazon EMR 5.30.0 以降で、マスターセキュリティグループのデフォルトの [すべて許可] アウトバウンドルールを削除して 0.0.0.0/ にした場合、サービスアクセス用のセキュリティグループへのアウトバウンド TCP 接続をポート 9443 で許可するルールを追加する必要があります。サービスアクセス用のセキュリティグループで、マスターセキュリティグループからのインバウンド TCP トラフィックをポート 9443 で許可する必要もあります。セキュリティグループの設定の詳細については、「Amazon EMR-managed security group for the primary instance (private subnets))」を参照してください。

- マネージドスケーリングは YARN ノードラベル 機能をサポートしていません。マネージドスケーリングを使用するクラスターではノードラベルを使用しないでください。例えば、エグゼキューターをタスクノードでのみ実行できるようにしないでください。Amazon EMR クラスターでノードラベルを使用すると、クラスターがスケールアップされず、アプリケーションが遅くなる可能性があります。

- AWS CloudFormation を使用して、Amazon EMR マネージドスケーリングを設定できます。詳細については、「 ユーザーガイドAWS::EMR::Cluster」のAWS CloudFormation 「」を参照してください。

機能履歴

この表に、Amazon EMR Managed Scaling 機能の更新を示します。

リリース日	機能	Amazon EMR のバージョン
2024 年 3 月 31 日	マネージドスケーリングは、ap-south-2 アジアパシフィック (ハイデラバード) リージョンで利用できます。	6.14.0 以降
2024 年 2 月 13 日	マネージドスケーリングは、eu-south-2 欧州 (スペイン) リージョンで利用できます。	6.14.0 以降
2023 年 10 月 10 日	マネージドスケーリングが ap-southeast-3 アジアパシフィック (ジャカルタ) リージョンで利用可能になりました。	6.14.0 以降
2023 年 7 月 28 日	マネージドスケーリングが強化され、Amazon EMR で現在のインスタンスグループでスケールアップに遅延が生じた場合、スケールアップ時に別のタスクインスタンスグループに切り替わるようになりました。	5.34.0 以降、6.4.0 以降
2023 年 6 月 16 日	マネージドスケーリングが強化され、アプリケーションマスターを実行しているノードを認識し、そのノードがスケールダウンされないようになりました。詳細については、「ノード割り当て戦略とシナリオを把握する」を参照してください。	5.34.0 以降、6.4.0 以降

リリース日	機能	Amazon EMR のバージョン
2022 年 3 月 21 日	クラスターのスケールダウン時に使用される Spark シャッフルデータの認識機能が追加されました。Apache Spark とマネージドスケーリング機能が有効になっている Amazon EMR クラスターの場合、Amazon EMR は Spark エグゼキューターと中間シャッフルデータのロケーションを継続的にモニタリングします。Amazon EMR はこの情報を使用して、使用率の高いシャッフルデータを含まない、使用率の低いインスタンスのみをスケールダウンします。これにより、損失したシャッフルデータの再計算が発生しなくなり、コスト削減とジョブパフォーマンスの向上につながります。詳細については、Spark のプログラミングガイドを参照してください。	5.34.0 以降、6.4.0 以降

Amazon EMR のマネージドスケーリングを設定する

以下のセクションでは、、、 AWS SDK for Javaまたは でマネージドスケーリングを使用する EMR AWS Management Consoleクラスターを起動する方法について説明します AWS Command Line Interface。

トピック

- AWS Management Console を使用してマネージドスケーリングを設定する
- AWS CLI を使用してマネージドスケーリングを設定する
- AWS SDK for Java を使用してマネージドスケーリングを設定する

AWS Management Console を使用してマネージドスケーリングを設定する

Amazon EMR コンソールを使用して、クラスターの作成時にマネージドスケーリングを設定したり、実行中のクラスターのマネージドスケーリングポリシーを変更したりできます。

New console

新しいコンソールを使用して、クラスターの作成時にマネージドスケーリングを設定するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で、[クラスター] を選択し、[クラスターの作成] を選択します
3. Amazon EMR リリース [emr-5.30.0] 以降を選択します。ただし、バージョン [emr-6.0.0] は除きます。
4. [クラスターのスケーリングとプロビジョニングのオプション] で [EMR マネージドスケーリングを使用] を選択します。インスタンスの [最小] 数と [最大] 数、[最大コアノード] インスタンス、[最大オンデマンドインスタンス] を指定します。
5. クラスターに適用するその他のオプションを選択します。
6. クラスターを起動するには、[クラスターの作成] を選択します。

新しいコンソールを使用して既存のクラスターにマネージドスケーリングを設定するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で [クラスター] を選択し、更新するクラスターを選択します。
3. クラスターの詳細ページの [インスタンス] タブで、[インスタンスグループ設定] セクションを見つけます。[クラスタースケーリングを編集] を選択し、インスタンスの [最小] 数と [最大] 数、および [オンデマンド] 制限に新しい値を指定します。

Old console

古いコンソールでクラスターを作成する場合は、クイックオプションまたは詳細なクラスター設定オプションを使用してマネージドスケーリングを設定できます。[概要] ページまたは [ハードウェア] ページで [マネージドスケーリング] 設定を変更することで、実行中のクラスターに対してマネージドスケーリングポリシーを作成または変更することもできます。

古いコンソールでクラスターの作成時にクイックオプションを使用してマネージドスケーリングを設定するには

1. Amazon EMR コンソールを開き、[クラスターの作成] を選択して、[クラスターの作成 - クイックオプション] を開きます。
2. [クラスターのスケーリングとプロビジョニングのオプション] の横の [ハードウェア構成] セクションで、チェックボックスをオンにして [ワークロードに基づいてクラスターノードをスケーリングする] を有効にします。
3. [Core and task units] (コアユニットとタスクユニット) の下で、コアインスタンスとタスクインスタンスの数の [最小] と [最大] を指定します。

古いコンソールでクラスターの作成時に詳細オプションを使用してマネージドスケーリングを設定するには

1. Amazon EMR コンソールで、[クラスターの作成] を選択し、[詳細オプションに移動する] を選択します。次に、[ステップ 1: ソフトウェアおよびステップ] のオプションを選択し、[ステップ 2: ハードウェア構成] に移動します。
2. [Cluster composition (クラスターの構築)] セクションで、[インスタンスフリート] または [Uniform インスタンスグループ] を選択します。
3. [クラスターのスケーリングとプロビジョニングのオプション] で [クラススケーリングを有効にする] を選択します。次に、[Use EMR managed scaling] (EMR マネージドスケーリングを使用する) を選択します。[Core and task units] (コアユニットとタスクユニット) の下で、インスタンスまたはインスタンスフリートユニットの数の [最小] と [最大]、[On-Demand limit] (オンデマンド制限) と [Maximum Core Node] (最大コアノード) の数を指定します。

   インスタンスグループで構成されるクラスターで、インスタンスグループごとにカスタムの自動スケーリングポリシーを定義する場合は、[Create a custom Auto Scaling policy (カスタムの自動スケーリングポリシーを作成する)] を選択することもできます。詳細については、「カスタムポリシーによる自動スケーリングをインスタンスグループに使用する」を参照してください。

古いコンソールを使用して既存のクラスターのマネージドスケーリングを変更するには

1. Amazon EMR コンソールを開き、クラスターリストからクラスターを選択して、[ハードウェア] タブを選択します。

2. [クラスターのスケーリングとプロビジョニングのオプション] セクションで、Amazon EMR Managed Scaling に対する [編集] を選択します。
3. [クラスターのスケーリングとプロビジョニングのオプション] セクションで、インスタンスの [最小] 数と [最大] 数、および [オンデマンド制限] に新しい値を指定します。

AWS CLI を使用してマネージドスケーリングを設定する

Amazon EMR の AWS CLI コマンドを使用して、クラスターの作成時にマネージドスケーリングを設定できます。短縮構文を使用して、関連コマンドのインラインで JSON 設定を指定したり、設定 JSON を含むファイルを参照したりできます。また、既存のクラスターにマネージドスケーリングポリシーを適用して、以前に適用したマネージドスケーリングポリシーを削除することもできます。さらに、スケーリングポリシーの詳細設定を実行中のクラスターから取得できます。

クラスター起動時にマネージドスケーリングを有効化する

マネージドスケーリングは、次の例で示すように、クラスターの起動時に有効にできます。

```
aws emr create-cluster \
 --service-role EMR_DefaultRole \
 --release-label emr-7.1.0 \
 --name EMR_Managed_Scaling_Enabled_Cluster \
 --applications Name=Spark Name=Hbase \
 --ec2-attributes KeyName=keyName,InstanceProfile=EMR_EC2_DefaultRole \
 --instance-groups InstanceType=m4.xlarge,InstanceGroupType=MASTER,InstanceCount=1
 InstanceType=m4.xlarge,InstanceGroupType=CORE,InstanceCount=2 \
 --region us-east-1 \
 --managed-scaling-policy
 ComputeLimits='{MinimumCapacityUnits=2,MaximumCapacityUnits=4,UnitType=Instances}'
```

を使用する場合は、-- managed-scaling-policy オプションを使用してマネージドポリシー設定を指定することもできますcreate-cluster。

既存のクラスターへのマネージドスケーリングポリシーの適用

マネージドスケーリングポリシーは、次の例で示すように、既存のクラスターに適用できます。

```
aws emr put-managed-scaling-policy
--cluster-id j-123456
--managed-scaling-policy ComputeLimits='{MinimumCapacityUnits=1,
MaximumCapacityUnits=10,  MaximumOnDemandCapacityUnits=10, UnitType=Instances}'
```

aws emr put-managed-scaling-policy コマンドを使用して、マネージドスケーリングポリシーを既存のクラスターに適用することもできます。次の例では、マネージドスケーリングポリシー設定を指定する JSON ファイル managedscaleconfig.json への参照を使用します。

```
aws emr put-managed-scaling-policy --cluster-id j-123456 --managed-scaling-policy
 file://./managedscaleconfig.json
```

次の例は、マネージドスケーリングポリシーを定義する managedscaleconfig.json ファイルの内容を示しています。

```
{
    "ComputeLimits": {
        "UnitType": "Instances",
        "MinimumCapacityUnits": 1,
        "MaximumCapacityUnits": 10,
        "MaximumOnDemandCapacityUnits": 10
    }
}
```

マネージドスケーリングポリシー設定の取得

GetManagedScalingPolicy コマンドを使用すると、ポリシー設定を取得できます。たとえば、次のコマンドは、クラスター ID j-123456 を持つクラスターの設定を取得します。

```
aws emr get-managed-scaling-policy --cluster-id j-123456
```

このコマンドでは、次のサンプルアウトプットが生成されます。

```
{
   "ManagedScalingPolicy": {
      "ComputeLimits": {
         "MinimumCapacityUnits": 1,
         "MaximumOnDemandCapacityUnits": 10,
         "MaximumCapacityUnits": 10,
         "UnitType": "Instances"
      }
   }
}
```

での Amazon EMR コマンドの使用の詳細については AWS CLI、「」を参照してくださいhttps://docs.aws.amazon.com/cli/latest/reference/emr。

マネージドスケーリングポリシーの削除

RemoveManagedScalingPolicy コマンドを使用すると、ポリシー設定を削除できます。たとえば、次のコマンドでは、クラスター ID j-123456 を持つクラスターの設定を削除します。

```
aws emr remove-managed-scaling-policy --cluster-id j-123456
```

AWS SDK for Java を使用してマネージドスケーリングを設定する

以下のプログラム抜粋では、 AWS SDK for Javaを使用してマネージドスケーリングを設定する方法を示します。

```
package com.amazonaws.emr.sample;

import java.util.ArrayList;
import java.util.List;

import com.amazonaws.AmazonClientException;
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduce;
import com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduceClientBuilder;
import com.amazonaws.services.elasticmapreduce.model.Application;
import com.amazonaws.services.elasticmapreduce.model.ComputeLimits;
import com.amazonaws.services.elasticmapreduce.model.ComputeLimitsUnitType;
import com.amazonaws.services.elasticmapreduce.model.InstanceGroupConfig;
import com.amazonaws.services.elasticmapreduce.model.JobFlowInstancesConfig;
import com.amazonaws.services.elasticmapreduce.model.ManagedScalingPolicy;
import com.amazonaws.services.elasticmapreduce.model.RunJobFlowRequest;
import com.amazonaws.services.elasticmapreduce.model.RunJobFlowResult;

public class CreateClusterWithManagedScalingWithIG {

 public static void main(String[] args) {
  AWSCredentials credentialsFromProfile = getCreadentials("AWS-Profile-Name-Here");

  /**
   * Create an Amazon EMR client with the credentials and region specified in order to
 create the cluster
   */
  AmazonElasticMapReduce emr = AmazonElasticMapReduceClientBuilder.standard()
```

```
    .withCredentials(new AWSStaticCredentialsProvider(credentialsFromProfile))
    .withRegion(Regions.US_EAST_1)
    .build();

  /**
   * Create Instance Groups - Primary, Core, Task
   */
  InstanceGroupConfig instanceGroupConfigMaster = new InstanceGroupConfig()
    .withInstanceCount(1)
    .withInstanceRole("MASTER")
    .withInstanceType("m4.large")
    .withMarket("ON_DEMAND");

  InstanceGroupConfig instanceGroupConfigCore = new InstanceGroupConfig()
   .withInstanceCount(4)
   .withInstanceRole("CORE")
   .withInstanceType("m4.large")
   .withMarket("ON_DEMAND");

  InstanceGroupConfig instanceGroupConfigTask = new InstanceGroupConfig()
   .withInstanceCount(5)
   .withInstanceRole("TASK")
   .withInstanceType("m4.large")
   .withMarket("ON_DEMAND");

  List<InstanceGroupConfig> igConfigs = new ArrayList<>();
  igConfigs.add(instanceGroupConfigMaster);
  igConfigs.add(instanceGroupConfigCore);
  igConfigs.add(instanceGroupConfigTask);

        /**
         *  specify applications to be installed and configured when Amazon EMR creates
the cluster
         */
  Application hive = new Application().withName("Hive");
  Application spark = new Application().withName("Spark");
  Application ganglia = new Application().withName("Ganglia");
  Application zeppelin = new Application().withName("Zeppelin");

  /**
   * Managed Scaling Configuration -
         * Using UnitType=Instances for clusters composed of instance groups
   *
         * Other options are:
```

```
        * UnitType = VCPU ( for clusters composed of instance groups)
        * UnitType = InstanceFleetUnits ( for clusters composed of instance fleets)
        **/
 ComputeLimits computeLimits = new ComputeLimits()
   .withMinimumCapacityUnits(1)
   .withMaximumCapacityUnits(20)
   .withUnitType(ComputeLimitsUnitType.Instances);

 ManagedScalingPolicy managedScalingPolicy = new ManagedScalingPolicy();
 managedScalingPolicy.setComputeLimits(computeLimits);

 // create the cluster with a managed scaling policy
 RunJobFlowRequest request = new RunJobFlowRequest()
         .withName("EMR_Managed_Scaling_TestCluster")
         .withReleaseLabel("emr-7.1.0")          // Specifies the version label for
the Amazon EMR release; we recommend the latest release
         .withApplications(hive,spark,ganglia,zeppelin)
         .withLogUri("s3://path/to/my/emr/logs")  // A URI in S3 for log files is
required when debugging is enabled.
         .withServiceRole("EMR_DefaultRole")      // If you use a custom IAM service
role, replace the default role with the custom role.
         .withJobFlowRole("EMR_EC2_DefaultRole")  // If you use a custom Amazon EMR
role for EC2 instance profile, replace the default role with the custom Amazon EMR
role.
         .withInstances(new JobFlowInstancesConfig().withInstanceGroups(igConfigs)
             .withEc2SubnetId("subnet-123456789012345")
             .withEc2KeyName("my-ec2-key-name")
             .withKeepJobFlowAliveWhenNoSteps(true))
         .withManagedScalingPolicy(managedScalingPolicy);
   RunJobFlowResult result = emr.runJobFlow(request);

   System.out.println("The cluster ID is " + result.toString());
}

public static AWSCredentials getCredentials(String profileName) {
 // specifies any named profile in .aws/credentials as the credentials provider
 try {
  return new ProfileCredentialsProvider("AWS-Profile-Name-Here")
    .getCredentials();
       } catch (Exception e) {
           throw new AmazonClientException(
                   "Cannot load credentials from .aws/credentials file. " +
                   "Make sure that the credentials file exists and that the profile
name is defined within it.",
```

```
                    e);
            }
    }

    public CreateClusterWithManagedScalingWithIG() { }
}
```

ノード割り当て戦略とシナリオを把握する

このセクションでは、Amazon EMR Managed Scaling で使用できるノード割り当て戦略と一般的なスケーリングシナリオの概要について説明します。

ノード割り当て戦略

Amazon EMR Managed Scaling は、次のスケールアップ戦略とスケールダウン戦略に基づいて、コアノードとタスクノードを割り当てます。

スケールアップ戦略

- Amazon EMR Managed Scaling では、まずコアノードに容量を追加し、次に最大許容容量に達するまで、または目的のスケールアップのターゲット容量が達成されるまでタスクノードに容量を追加します。
- Amazon EMR で現在のインスタンスグループでスケールアップに遅延が発生すると、マネージドスケーリングを使用するクラスターは自動的に別のタスクインスタンスグループに切り替わります。
- `MaximumCoreCapacityUnits` パラメータが設定されている場合、Amazon EMR はコアユニットが最大許容制限に達するまで、コアノードをスケーリングします。残りの容量はすべてタスクノードに追加されます。
- `MaximumOnDemandCapacityUnits` パラメータが設定されている場合、Amazon EMR は、オンデマンドユニットが最大許容制限に達するまで、オンデマンドインスタンスを使用してクラスターをスケーリングします。残りの容量はすべて、スポットインスタンスを使用して追加されます。
- `MaximumCoreCapacityUnits` と `MaximumOnDemandCapacityUnits` の両方のパラメータが設定されている場合、Amazon EMR はスケーリング中に両方の制限を考慮します。

  例えば、`MaximumCoreCapacityUnits` が `MaximumOnDemandCapacityUnits` 未満の場合、Amazon EMR はまず、コア容量の制限に達するまでコアノードの容量をスケールします。残りの容量については、Amazon EMR はまずオンデマンドインスタンスを使用してオンデマンドの制限に達するまでタスクノードをスケールし、次にスポットインスタンスを使用してタスクノードをスケールします。

スケールダウン戦略

- Amazon EMR バージョン 5.34.0 以降、および Amazon EMR バージョン 6.4.0 以降では、Spark のシャッフルデータ (Spark が特定の操作を実行するためにパーティション間で再配布するデータ) を認識するマネージドスケーリングをサポートしています。シャッフル操作の詳細については、Spark のプログラミングガイドを参照してください。マネージドスケーリングでは、使用率の高いシャッフルデータを含まない、使用率の低いインスタンスのみをスケールダウンします。このインテリジェントなスケーリングにより、意図しないシャッフルデータの損失を防ぐことができ、ジョブを再試行したり、中間データを再計算したりする必要がなくなります。
- Amazon EMR Managed Scaling は、まずタスクノードを削除し、次に目的のスケールダウンのターゲット容量が達成されるまでコアノードを削除します。クラスターのスケーリングは、マネージドスケーリングポリシーの最小制約を下回ることはありません。
- 各ノードタイプ (コアノードまたはタスクノード) 内で、Amazon EMR Managed Scaling は、まずスポットインスタンスを削除し、次にオンデマンドインスタンスを削除します。
- Amazon EMR 5.x リリース 5.34.0 以降、および 6.x リリース 6.4.0 以降で起動されたクラスターの場合、Amazon EMR Managed Scaling は、Apache Spark 用の `ApplicationMaster` が実行されているノードをスケールダウンしません。これにより、ジョブの失敗や再試行が最小限に抑えられ、ジョブのパフォーマンスが向上し、コストが削減されます。クラスター内のどのノードで `ApplicationMaster` が実行されているかを確認するには、Spark 履歴サーバーにアクセスし、Spark アプリケーション ID の [Executors] タブでドライバーをフィルタリングします。

クラスターに負荷がない場合、Amazon EMR は前の評価で提案された新しいインスタンスの追加をキャンセルし、スケールダウン操作を実行します。クラスターの負荷が大きい場合、Amazon EMR はインスタンスの削除をキャンセルし、スケールアップ操作を実行します。

ノード割り当てに関する考慮事項

スポットの再利用時に HDFS データが失われないように、コアノードに対してオンデマンド購入オプションを使用することをお勧めします。タスクノードに対してスポット購入オプションを使用すると、タスクノードにスポットインスタンスを追加するときのコストが削減され、ジョブ実行が高速になります。

ノード割り当てシナリオ

最大、最小、オンデマンド制限、および最大コアノードの各パラメータをさまざまな組み合わせで、セットアップすることで、ニーズに基づいてさまざまなスケーリングシナリオを作成できます。

シナリオ 1: コアノードのみをスケーリングする

コアノードのみをスケーリングするには、マネージドスケーリングパラメータが次の要件を満たしている必要があります。

- オンデマンド制限が最大限度と等しいこと。
- 最大コアノードが最大限度と等しいこと。

オンデマンド制限と最大コアノードのパラメータが指定されていないときは、両方のパラメータがデフォルトで最大限度に設定されます。

コアノードのみをスケーリングするシナリオの例を次に示します。

クラスターの初期状態	スケーリングパラメータ	スケーリングの動作
インスタンスグループ コア:1 オンデマンド タスク:1 オンデマンドと 1 スポット	UnitType: Instances MinimumCapacityUnits : 1 MaximumCapacityUnits : 20 MaximumOnDemandCapacityUnits : 20 MaximumCoreCapacityUnits : 20	オンデマンドタイプを使用して、コアノードで 1 ~ 20 個のインスタンスまたはインスタンスフリートユニットをスケーリングします。タスクノードのスケーリングはありません。
インスタンスフリート コア:1 オンデマンド タスク:1 オンデマンドと 1 スポット	UnitType: InstanceFleetUnits MinimumCapacityUnits : 1 MaximumCapacityUnits : 20 MaximumOnDemandCapacityUnits : 20 MaximumCoreCapacityUnits : 20	

シナリオ 2: タスクノードのみをスケーリングする

タスクノードのみをスケーリングするには、マネージドスケーリングパラメータが次の要件を満たしている必要があります。

- 最大コアノードが最小限度と等しいこと。

タスクノードのみをスケーリングするシナリオの例を次に示します。

クラスターの初期状態	スケーリングパラメータ	スケーリングの動作
インスタンスグループ コア:2 オンデマンド タスク:1 スポット	UnitType: Instances MinimumCapacityUnits : 2 MaximumCapacityUnits : 20 MaximumCoreCapacityUnits : 2	コアノードを 2 個に固定し、0 から 18 インスタンスまたはインスタンスフリートユニットの間のタスクノードのみをスケーリングします。最小限度と最大限度の間の容量が、タスクノードのみに追加されます。
インスタンスフリート コア:2 オンデマンド タスク:1 スポット	UnitType: InstanceFleetUnits MinimumCapacityUnits : 2 MaximumCapacityUnits : 20 MaximumCoreCapacityUnits : 2	

シナリオ 3: クラスター内のオンデマンドインスタンスのみ

オンデマンドインスタンスのみを使用するには、クラスターとマネージドスケーリングパラメータが次の要件を満たしている必要があります。

- オンデマンド制限が最大限度と等しいこと。

  オンデマンド制限が指定されていないときは、パラメータ値がデフォルトで最大限度に設定されます。デフォルト値は、Amazon EMR がオンデマンドインスタンスのみをスケーリングすることを示します。

最大コアノードが最大限度未満の場合、最大コアノードパラメータを使用して、コアノードとタスクノード間で容量割り当てを分割できます。

インスタンスグループで構成されるクラスターでこのシナリオを有効にするには、初期構成時にクラスター内のすべてのノードグループがオンデマンドマーケットタイプを使用する必要があります。

クラスター全体にオンデマンドインスタンスを持つシナリオの例を次に示します。

クラスターの初期状態	スケーリングパラメータ	スケーリングの動作
インスタンスグループ コア:1 オンデマンド タスク:1 オンデマンド	UnitType: Instances MinimumCapacityUnits : 1 MaximumCapacityUnits : 20 MaximumOnDemandCapacityUnits : 20 MaximumCoreCapacityUnits : 12	オンデマンドタイプを使用して、コアノードで 1 ~ 12 個のインスタンスまたはインスタンスフリートユニットをスケーリングします。オンデマンドを使用して、タスクノードで残りの容量をスケーリングします。スポットインスタンスを使用したスケーリングはありません。
インスタンスフリート コア:1 オンデマンド タスク:1 オンデマンド	UnitType: InstanceFleetUnits MinimumCapacityUnits : 1 MaximumCapacityUnits : 20 MaximumOnDemandCapacityUnits : 20 MaximumCoreCapacityUnits : 12	

シナリオ 4: クラスター内のスポットインスタンスのみ

スポットインスタンスのみを使用するには、マネージドスケーリングパラメータが次の要件を満たしている必要があります。

- オンデマンド制限が 0 に設定されていること。

最大コアノードが最大限度未満の場合、最大コアノードパラメータを使用して、コアノードとタスクノード間で容量割り当てを分割できます。

インスタンスグループで構成されるクラスターでこのシナリオを有効にするには、コアインスタンスグループが初期設定時にスポット購入オプションを使用する必要があります。タスクインスタンスグループにスポットインスタンスがない場合、Amazon EMR Managed Scaling は、必要に応じてスポットインスタンスを使用してタスクグループを作成します。

クラスター全体にスポットインスタンスを持つシナリオの例を次に示します。

クラスターの初期状態	スケーリングパラメータ	スケーリングの動作
インスタンスグループ コア:1 スポット タスク:1 スポット	UnitType: Instances MinimumCapacityUnits : 1 MaximumCapacityUnits : 20 MaximumOnDemandCapacityUnits : 0	スポットを使用して、コアノードで 1 ~ 20 個のインスタンスまたはインスタンスフリートユニットをスケーリングします。オンデマンドタイプを使用したスケーリングはありません。
インスタンスフリート コア:1 スポット タスク:1 スポット	UnitType: InstanceFleetUnits MinimumCapacityUnits : 1 MaximumCapacityUnits : 20 MaximumOnDemandCapacityUnits : 0	

シナリオ 5: コアノードのオンデマンドインスタンスと、タスクノードのスポットインスタンスをスケーリングする

コアノードのオンデマンドインスタンスと、タスクノードのスポットインスタンスをスケーリングするには、マネージドスケーリングパラメータが次の要件を満たしている必要があります。

- オンデマンドの制限が最大コアノードに等しいこと。

- オンデマンド制限と最大コアノードの両方が最大限度未満であること。

インスタンスグループで構成されるクラスターでこのシナリオを有効にするには、コアノードグループがオンデマンド購入オプションを使用する必要があります。

コアノードのオンデマンドインスタンスと、タスクノードのスポットインスタンスをスケーリングするシナリオの例を次に示します。

クラスターの初期状態	スケーリングパラメータ	スケーリングの動作
インスタンスグループ コア:1 オンデマンド タスク:1 オンデマンドと 1 スポット	UnitType: Instances MinimumCapacityUnits : 1 MaximumCapacityUnits : 20 MaximumOnDemandCapacityUnits : 7 MaximumCoreCapacityUnits : 7	タスクノードにすでに 1 つのオンデマンドユニットがあり、オンデマンドの最大制限が 7 であるため、コアノードで 6 個のオンデマンドユニットまでスケールアップします。次に、タスクノードで 13 個のスポットユニットまでスケールアップします。
インスタンスフリート コア:1 オンデマンド タスク:1 オンデマンドと 1 スポット	UnitType: InstanceFleetUnits MinimumCapacityUnits : 1 MaximumCapacityUnits : 20 MaximumOnDemandCapacityUnits : 7 MaximumCoreCapacityUnits : 7	

マネージドスケーリングメトリクスについて

Amazon EMR は、クラスターでマネージドスケーリングが有効になっているとき、1 分単位の精度のデータで高解像度のメトリクスを公開します。Amazon EMR コンソールまたは Amazon コンソールを使用して、マネージドスケーリングによって制御されるすべてのサイズ変更の開始と完了のイ

ベントを表示できます CloudWatch 。 CloudWatch メトリクスは、Amazon EMR マネージドスケーリングが動作するために重要です。 CloudWatch メトリクスを注意深くモニタリングして、データが欠落していないことを確認することをお勧めします。欠落しているメトリクスを検出するように CloudWatch アラームを設定する方法の詳細については、「Amazon CloudWatch アラームの使用」を参照してください。Amazon EMR での CloudWatch イベントの使用の詳細については、「イベントのモニタリング CloudWatch」を参照してください。

次のメトリクスは、クラスターの現在の容量またはターゲットの容量を示します。これらのメトリクスは、マネージドスケーリングが有効になっている場合にのみ使用できます。インスタンスフリートで構成されるクラスターの場合、クラスター容量のメトリクスは Units 単位で測定されます。インスタンスグループで構成されるクラスターの場合、クラスター容量のメトリクスは、マネージドスケーリングポリシーで使用される単位タイプに基づき、Nodes 単位または vCPU 単位で測定されます。

メトリクス	説明
• TotalUnitsRequested • TotalNodesRequested • TotalVCPURequested	マネージドスケーリングによって決定された、クラスター内の単位/ノード/vCPU の合計ターゲット数。 単位: Count
• TotalUnitsRunning • TotalNodesRunning • TotalVCPURunning	実行中のクラスターで使用可能な単位/ノード/vCPU の現在の合計数。クラスターのサイズ変更がリクエストされると、クラスターに新しいインスタンスが追加または削除された後に、このメトリクスが更新されます。 単位: Count
• CoreUnitsRequested • CoreNodesRequested • CoreVCPURequested	マネージドスケーリングによって決定された、クラスター内の CORE 単位/ノード/vCPU のターゲット数。 単位: Count

メトリクス	説明
• CoreUnitsRunning • CoreNodesRunning • CoreVCPURunning	クラスターで実行されている CORE 単位/ノード/vCPU の現在の数。 単位: Count
• TaskUnitsRequested • TaskNodesRequested • TaskVCPURequested	マネージドスケーリングによって決定された、クラスター内の TASK 単位/ノード/vCPU のターゲット数。 単位: Count
• TaskUnitsRunning • TaskNodesRunning • TaskVCPURunning	クラスターで実行されている TASK 単位/ノード/vCPU の現在の数。 単位: Count

次のメトリクスは、クラスターとアプリケーションの使用状況を示します。これらのメトリクスは、すべての Amazon EMR 機能で使用できますが、クラスターでマネージドスケーリングが有効になっているときは、1 分単位の精度のデータで高解像度のメトリクスが公開されます。以下のメトリクスを前の表のクラスター容量メトリクスと関連付けることで、マネージドスケーリングの決定について理解することができます。

メトリクス	説明
AppsCompleted	YARN に送信され、完了したアプリケーションの数。 ユースケース:クラスターの進捗状況を監視する 単位: Count

メトリクス	説明
AppsPending	YARN に送信され、保留状態になっているアプリケーションの数。 ユースケース:クラスターの進捗状況を監視する 単位: Count
AppsRunning	YARN に送信され、実行中であるアプリケーションの数。 ユースケース:クラスターの進捗状況を監視する 単位: Count
ContainerAllocated	によって割り当てられたリソースコンテナの数ResourceManager。 ユースケース:クラスターの進捗状況を監視する 単位: Count
ContainerPending	キュー内にあり、まだ割り当てられていないコンテナの数。 ユースケース:クラスターの進捗状況を監視する 単位: Count
ContainerPendingRatio	割り当てられたコンテナに対する保留中のコンテナの比率 (ContainerPendingRatio = ContainerPending / ContainerAllocated ）。 ContainerAllocated = 0 の場合、 ContainerPendingRatio = ですContainerPending。の値は、パーセンテージではなく数値 ContainerPendingRatio を表します。この値は、コンテナ割り当て動作に基づくクラスターリソースのスケーリングに役立ちます。 単位: Count

メトリクス	説明
HDFSUtilization	現在使用されている HDFS ストレージの割合。 ユースケース:クラスターのパフォーマンスを分析する 単位: パーセント
IsIdle	クラスターが作業を行っていないが、まだ有効で課金されていることを示します。タスクもジョブも実行されていない場合は 1 に設定され、それ以外の場合は 0 に設定されます。この値は 5 分間隔で確認され、値が 1 の場合は、確認時にクラスターがアイドル状態だったことのみを示します。5 分間ずっとアイドル状態だったことを示すわけではありません。誤検出を避けるには、5 分ごとの確認で複数回連続してこの値が 1 である場合に通知するように、アラームを指定する必要があります。たとえば、30 分間にわたってこの値が 1 だった場合に通知するようアラームを指定できます。 ユースケース:クラスターのパフォーマンスを監視する 単位: ブール
MemoryAvailableMB	割り当てに使用できるメモリの量。 ユースケース:クラスターの進捗状況を監視する 単位: Count
MRActiveNodes	現在実行中の MapReduce タスクまたはジョブのノード数。YARN メトリクス mapred.resourceman ager.NoOfActiveNodes と同等。 ユースケース:クラスターの進捗状況を監視する 単位: Count

メトリクス	説明
YARNMemoryAvailablePercentage	YARN で使用できる残りのメモリの割合 (YARN MemoryAvailablePercentage = MemoryAvailableMB/MemoryTotalMB)。この値は、YARN のメモリの使用状況に基づくクラスターリソースのスケーリングに役立ちます。 単位: パーセント

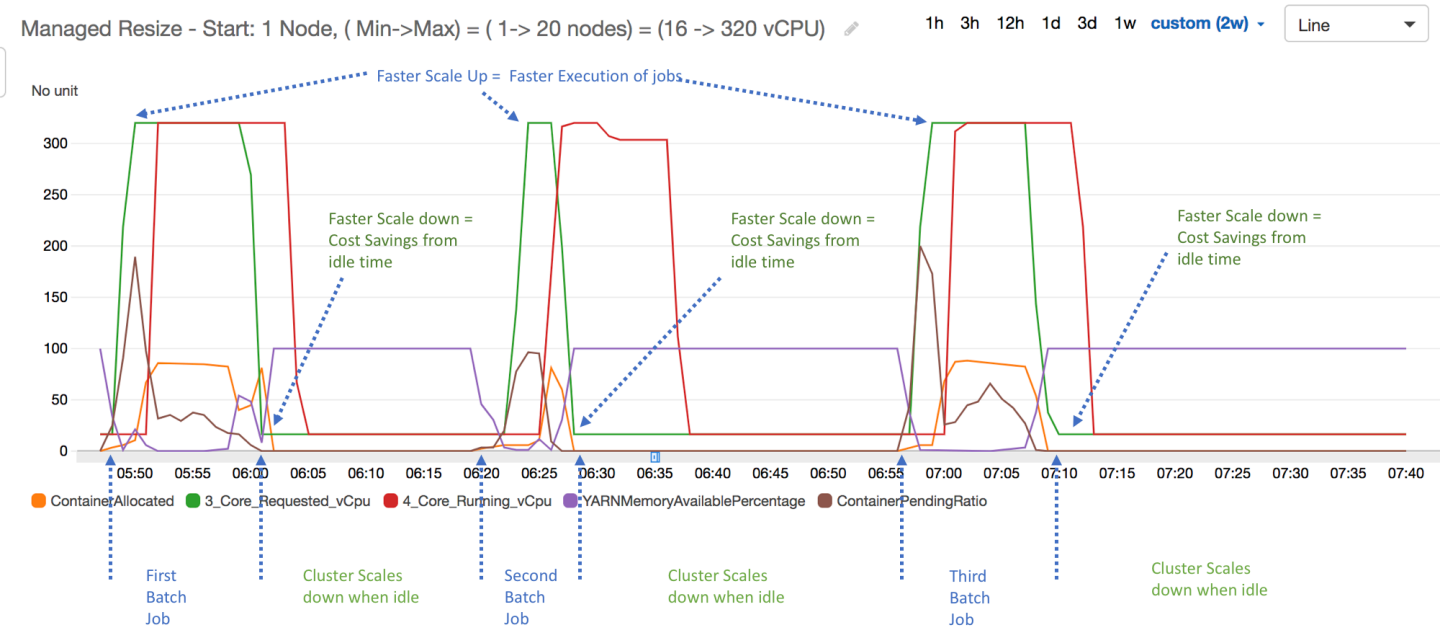
マネージドスケーリングメトリクスをグラフ化する

以下の手順で示すように、メトリクスをグラフ化することにより、クラスターのワークロードパターンと、Amazon EMR Managed Scaling によって行われた対応するスケーリング決定を視覚化できます。

CloudWatch コンソールでマネージドスケーリングメトリクスをグラフ化するには

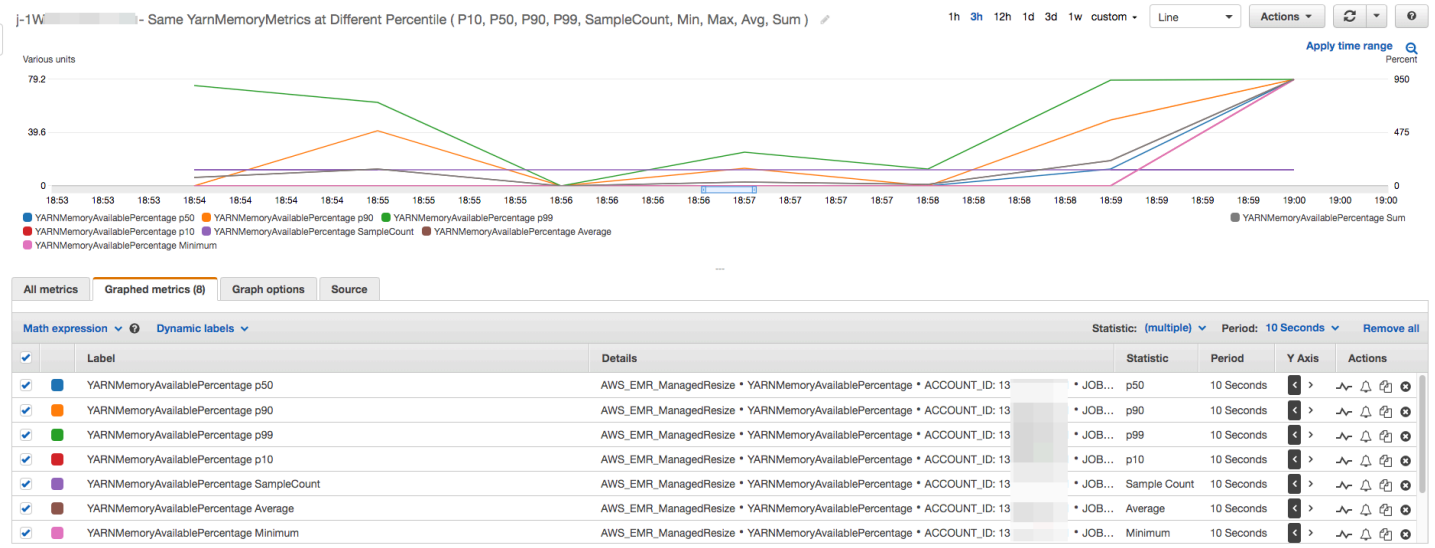
1. CloudWatch コンソールを開きます。
2. ナビゲーションペインで [Amazon EMR] を選択します。モニタリングするクラスターのクラスター識別子を検索できます。
3. グラフ化するメトリクスまでスクロールダウンします。グラフを表示するメトリクスを開きます。
4. 1 つ以上のメトリクスをグラフ化するには、各メトリクスの横にあるチェックボックスを選択します。

次の例は、クラスターの Amazon EMR Managed Scaling のアクティビティを示しています。グラフには、アクティブ度の低いワークロードがある場合にコストを節約する、3 つの自動スケールダウン期間が示されています。



すべてのクラスターの容量および使用率のメトリクスが、1 分間隔で公開されます。各 1 分間データには追加の統計情報も関連付けられており、Percentiles、Min、Max、Sum、Average、SampleCount などさまざまな関数に使用できます。

たとえば、次のグラフでは、Sum、Average、Min、SampleCount とともに、同じ YARNMemoryAvailablePercentage メトリクスが異なるパーセンタイル (P10、P50、P90、P99) で描画されます。



カスタムポリシーによる自動スケーリングをインスタンスグループに使用する

Amazon EMR リリース 4.0 以降のカスタムポリシーを使用した自動スケーリングでは、スケーリングポリシー で指定した CloudWatch メトリクスやその他のパラメータに基づいて、コアノードとタスクノードをプログラムでスケールアウトおよびスケールインできます。 カスタムポリシーによる自動スケーリングは、インスタンスグループ設定を使用するときに利用できます。インスタンスフリートを使用するときには利用できません。インスタンスグループとインスタンスフリートの詳細については、「インスタンスフリートまたはユニフォームインスタンスグループでクラスターを作成する」を参照してください。

> **Note**
>
> Amazon EMR でカスタムポリシー機能による自動スケーリングを使用するには、クラスターの作成時に `VisibleToAllUsers` パラメータに `true` を設定する必要があります。詳細については、「」を参照してくださいSetVisibleToAllUsers。

スケーリングポリシーは、インスタンスグループ設定の一部です。インスタンスグループの初期設定時にポリシーを指定するか、インスタンスグループがアクティブになった後でも、既存のクラスターのインスタンスグループを変更して、ポリシーを指定できます。プライマリインスタンスグループを除くクラスター内の各インスタンスグループは、独自のスケーリングポリシーを持つことができます。スケーリングポリシーはスケールアウトルールとスケールインルールで構成されます。スケー

ルアウトルールとスケールインルールは、それぞれに異なるパラメータを用いて、個別に設定できます。

スケーリングポリシーは、 AWS Management Console、、 AWS CLIまたは Amazon EMR API を使用して設定できます。 AWS CLI または Amazon EMR API を使用する場合は、スケーリングポリシーを JSON 形式で指定します。さらに、 AWS CLI または Amazon EMR API を使用する場合、カスタム CloudWatch メトリクスを指定できます。カスタムのメトリクスは、 AWS Management Consoleでの選択には使用できません。最初にコンソールを使用してスケーリングポリシーを作成する場合は、まず、多数のアプリケーションが事前設定されているデフォルトのポリシーが適しています。デフォルトのルールは削除したり変更できます。

自動スケーリングでは EMR クラスターの容量 を調整できますが on-the-fly、ベースラインワークロード要件を検討し、ノードとインスタンスグループの設定を計画する必要があります。詳細については、「クラスター設定のガイドライン」を参照してください。

> **Note**
>
> ほとんどのワークロードで、リソースの活用を最適化するには、スケールイン、スケールアウトの両方のルールを設定することが理想となります。一方を設定せずにどちらかのルールのみを設定すると、規模の拡大や縮小の後に、インスタンスカウントを手動でサイズ調整する必要があります。つまりこの方法では、手動でのリセットを伴う「一方通行の」自動スケールアウトまたはスケールインポリシーを設定することになります。

自動スケーリングの IAM ロールを作成する

Amazon EMR での自動スケーリングには、スケーリングアクティビティがトリガーされたときにインスタンスを追加および削除する権限がある IAM ロールが必要です。デフォルトロールである `EMR_AutoScaling_DefaultRole` は、適切なロールポリシーと信頼ポリシーで設定されており、この目的に使用できます。で初めてスケーリングポリシーを使用してクラスターを作成すると AWS Management Console、Amazon EMR はデフォルトのロールを作成し、アクセス許可のデフォルトの管理ポリシー をアタッチします`AmazonElasticMapReduceforAutoScalingRole`。

で自動スケーリングポリシーを使用してクラスターを作成する場合は AWS CLI、まずデフォルトの IAM ロールが存在するか、適切なアクセス許可を提供するポリシーがアタッチされたカスタム IAM ロールがあることを確認する必要があります。デフォルトロールを作成するには、クラスターを作成する前に `create-default-roles` コマンドを実行します。その後、クラスターの作成時に `--auto-scaling-role EMR_AutoScaling_DefaultRole` オプションを指定します。または、カスタムの自動スケーリングロール (例: `--auto-scaling-role` *`MyEMRAutoScalingRole`*) を作成

して、クラスター作成時に指定することもできます。カスタマイズされた自動スケーリングロールを Amazon EMR 向けに作成する場合は、管理ポリシーに基づいたカスタムロールのアクセス許可ポリシーをベースにすることをお勧めします。詳細については、「AWS のサービスおよびリソースへのアクセス許可を Amazon EMR に付与する IAM サービスロールの設定」を参照してください。

自動スケーリングルールについて

スケールアウトルールがインスタンスグループのスケーリングをトリガーするときに、ルールに従って Amazon EC2 インスタンスがインスタンスグループに追加されます。Amazon EC2 インスタンスが `InService` 状態になるとすぐに、Apache Spark、Apache Hive、Presto などのアプリケーションで新しいノードを使用できます。インスタンスを終了し、ノードを削除するスケールインルールを設定することもできます。自動的にスケーリングする Amazon EC2 インスタンスのライフサイクルの詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Auto Scaling のライフサイクル」を参照してください。

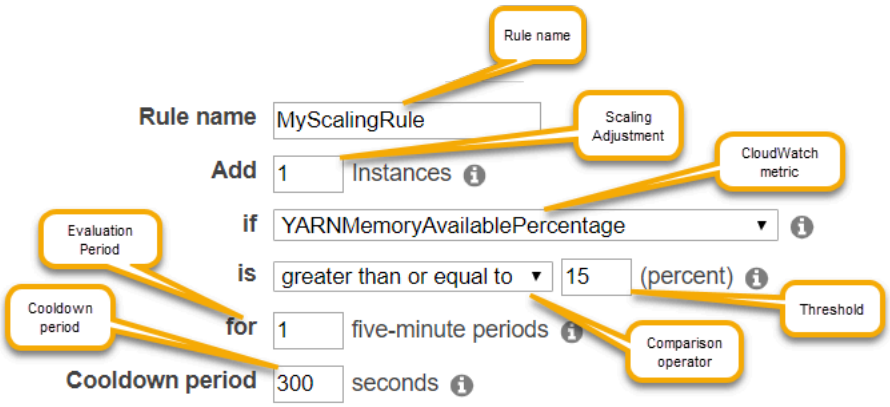
クラスターが Amazon EC2 インスタンスを削除する方法を設定できます。請求の Amazon EC2 インスタンス時間の境界と、タスクの完了時のどちらで削除するかを選択できます。この設定は自動スケーリングと手動でのサイズ変更オペレーションの両方に適用されます。この設定の詳細については、「クラスターのスケールダウン」を参照してください。

ポリシー内の各ルールのうち、次のパラメータで自動スケーリングの動作を決定します。

> **Note**
>
> ここに記載されているパラメータは、Amazon EMR AWS Management Console の に基づいています。 AWS CLI または Amazon EMR API を使用する場合、追加の詳細設定オプションを使用できます。詳細オプションの詳細については、「Amazon EMR API リファレンスSimpleScalingPolicyConfiguration」の「」を参照してください。

- 最大インスタンスおよび最小インスタンス。[最大インスタンス] という制約は、インスタンスグループに含めることができる Amazon EC2 インスタンスの最大数を指定します。これはすべてのスケールアウトルールに適用されます。同様に、[最小インスタンス] という制約は、Amazon EC2 インスタンスの最小数を指定します。これはすべてのスケールインルールに適用されます。
- ルール名は、ポリシー内で一意である必要があります。
- スケーリング調整は、ルールによりトリガーされた規模の拡大や縮小の間に追加 (スケールアウトルールの場合)、または終了 (スケールインルールの場合) する EC2 インスタンスの数を決定します。

- アラーム状態を監視されるCloudWatch メトリクス 。
- 比較演算子 。メトリクスをCloudWatch しきい値と比較し、トリガー条件を決定するために使用されます。
- 評価期間 。5 分単位で、スケーリングアクティビティがトリガーされる前に CloudWatch メトリクスがトリガー状態である必要があります。
- クールダウン期間 (秒単位) は、何らかのルールによって開始されるスケーリングアクティビティと、次に開始されるスケーリングアクティビティとの間で経過する必要がある時間です。インスタンスグループがスケーリングアクティビティを完了し、スケーリング後の状態に達すると、クールダウン期間によって、後続のスケーリングアクティビティが安定する可能性がある CloudWatch メトリクスの機会が提供されます。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Auto Scaling のクールダウン」を参照してください。



考慮事項と制約事項

- Amazon CloudWatch メトリクスは、Amazon EMR の自動スケーリングを運用するために不可欠です。Amazon CloudWatch メトリクスを注意深くモニタリングして、データが欠落していないことを確認することをお勧めします。欠落しているメトリクスを検出するように Amazon CloudWatch アラームを設定する方法の詳細については、「Amazon CloudWatch アラームの使用」を参照してください。
- EBS ボリュームの使用率が高すぎると、マネージドスケーリングの問題が発生する可能性があります。EBS ボリュームの使用率を注意深く監視して、EBS ボリュームの使用率が 90% 未満であることを確認することをお勧めします。追加の EBS ボリュームを指定する方法については、「インスタンスストレージ」を参照してください。
- Amazon EMR リリース 5.18 から 5.28 のカスタムポリシーを使用した自動スケーリングでは、Amazon CloudWatch メトリクスにデータが断続的に欠落しているためにスケーリングが失敗することがあります。自動スケーリングを向上させるために、最新の Amazon EMR バージョンを

使用することをお勧めします。5.18 から 5.28 の間の Amazon EMR リリースを使用する必要がある場合は、パッチの使用について、AWS サポートにお問い合わせいただくこともできます。

AWS Management Console を使用した自動スケーリングの設定

クラスターを作成する際、高度なクラスター設定オプションを使用して、インスタンスグループにスケーリングポリシーを設定します。既存のクラスターのハードウェア設定でインスタンスグループを変更することにより、実行中のインスタンスグループのスケーリングポリシーを作成または変更することもできます。

1. 新しい Amazon EMR コンソールに移動し、サイドナビゲーションから [古いコンソールに切り替え] を選択します。古いコンソールに切り替えたときの動作の詳細については、「Using the old console」を参照してください。
2. クラスターを作成する場合、Amazon EMR コンソールで、[クラスターの作成] を選択し、[詳細オプションに移動する] を選択します。次に、[ステップ 1: ソフトウェアおよびステップ] のオプションを選択し、[ステップ 2: ハードウェア構成] に移動します。

   - または -

   実行中のクラスターのインスタンスグループを変更する場合、クラスターリストからクラスターを選択し、その後 [ハードウェア] セクションを展開します。
3. [クラスターのスケーリングとプロビジョニングのオプション] セクションで [クラスタースケーリングを有効にする] を選択します。次に、[Create a custom Auto Scaling policy (カスタムの自動スケーリングポリシーを作成する)] を選択します。

   [Custom automatic scaling policies (カスタムの自動スケーリングポリシー)] の表で、設定するインスタンスグループの行に表示されている鉛筆アイコンをクリックします。Auto Scaling ルールの画面が開きます。
4. インスタンスグループをスケールアウトした後に含める最大インスタンスを入力するか、インスタンスグループをスケールインした後に含める最小インスタンスを入力します。
5. ルールのパラメータを編集するには鉛筆をクリックします。ポリシーからルールを削除するには [X] を、ルールを追加するには [Add rule] をクリックします。
6. このトピックで先に記載したとおり、ルールのパラメータを選択します。Amazon EMR で使用できる CloudWatch メトリクスの詳細については、「Amazon ユーザーガイド」の「Amazon EMR メトリクスとディメンション CloudWatch 」を参照してください。

AWS CLI を使用した自動スケーリングの設定

Amazon EMR の AWS CLI コマンドを使用して、クラスターの作成時およびインスタンスグループの作成時に自動スケーリングを設定できます。短縮構文を使用して、関連コマンドのインラインで JSON 設定を指定したり、設定 JSON を含むファイルを参照したりできます。既存のインスタンスグループに自動スケーリングポリシーを適用したり、以前適用されていた自動スケーリングポリシーを削除することもできます。さらに、スケーリングポリシーの詳細設定を実行中のクラスターから取得できます。

> **Important**
>
> 自動スケーリングポリシーを持つクラスターを作成するときは、`--auto-scaling-role` *`MyAutoScalingRole`* コマンドを使用して、自動スケーリング用の IAM ロールを指定する必要があります。デフォルトロールは、*`EMR_AutoScaling_DefaultRole`* で、`create-default-roles` コマンドを使用して作成できます。ロールはクラスターが作成された後にのみ追加でき、既存のクラスターには追加できません。

自動スケーリングポリシーを設定するときに使用できるパラメータの詳細については、Amazon EMR API リファレンスPutAutoScalingPolicyの「」を参照してください。

インスタンスグループに適用する自動スケーリングポリシーを持つクラスターを作成する

`aws emr create-cluster` コマンドの `--instance-groups` オプション内で自動スケーリング設定を指定できます。次の例は、create-cluster コマンドを示しています。このコマンドには、インラインにコアインスタンスグループの自動スケーリングポリシーがあります。コマンドは、 AWS Management Console Amazon EMR の で自動スケーリングポリシーを作成するときに表示されるデフォルトのスケールアウトポリシーに相当するスケーリング設定を作成します。簡潔にするために、スケールインポリシーは表示されません。スケールインルールなしで、スケールアウトルールを作成するのは推奨されていません。

```
aws emr create-cluster --release-label emr-5.2.0 --service-role
 EMR_DefaultRole --ec2-attributes InstanceProfile=EMR_EC2_DefaultRole
 --auto-scaling-role EMR_AutoScaling_DefaultRole  --instance-groups
 Name=MyMasterIG,InstanceGroupType=MASTER,InstanceType=m5.xlarge,InstanceCount=1
 'Name=MyCoreIG,InstanceGroupType=CORE,InstanceType=m5.xlarge,InstanceCount=2,AutoScalingPolicy
scale-out,Description=Replicates the default scale-out rule in the
 console.,Action={SimpleScalingPolicyConfiguration={AdjustmentType=CHANGE_IN_CAPACITY,ScalingAd
```

```
ElasticMapReduce,Period=300,Statistic=AVERAGE,Threshold=15,Unit=PERCENT,Dimensions=[{Key=JobFlo
```

次のコマンドは、コマンドラインを使用して、*instancegroupconfig.json* という名前のインスタンスグループ設定ファイルの一部としての自動スケーリングポリシー定義を提供する方法を示しています。

```
aws emr create-cluster --release-label emr-5.2.0 --service-role EMR_DefaultRole --ec2-
attributes InstanceProfile=EMR_EC2_DefaultRole --instance-groups file://your/path/to/
instancegroupconfig.json --auto-scaling-role EMR_AutoScaling_DefaultRole
```

次の構成ファイルのコンテンツ:

```
[
{
  "InstanceCount": 1,
  "Name": "MyMasterIG",
  "InstanceGroupType": "MASTER",
  "InstanceType": "m5.xlarge"
},
{
  "InstanceCount": 2,
  "Name": "MyCoreIG",
  "InstanceGroupType": "CORE",
  "InstanceType": "m5.xlarge",
  "AutoScalingPolicy":
    {
     "Constraints":
      {
       "MinCapacity": 2,
       "MaxCapacity": 10
      },
     "Rules":
     [
      {
       "Name": "Default-scale-out",
       "Description": "Replicates the default scale-out rule in the console for YARN
 memory.",
       "Action":{
        "SimpleScalingPolicyConfiguration":{
          "AdjustmentType": "CHANGE_IN_CAPACITY",
          "ScalingAdjustment": 1,
```

```
          "CoolDown": 300
        }
       },
       "Trigger":{
        "CloudWatchAlarmDefinition":{
          "ComparisonOperator": "LESS_THAN",
          "EvaluationPeriods": 1,
          "MetricName": "YARNMemoryAvailablePercentage",
          "Namespace": "AWS/ElasticMapReduce",
          "Period": 300,
          "Threshold": 15,
          "Statistic": "AVERAGE",
          "Unit": "PERCENT",
          "Dimensions":[
             {
               "Key" : "JobFlowId",
               "Value" : "${emr.clusterId}"
             }
          ]
        }
       }
      }
     ]
   }
}
]
```

自動スケーリングポリシーを持つインスタンスグループをクラスターに追加する

create-cluster を使用するときと同じ方法で、--instance-groups オプションを使用して、add-instance-groups コマンドでスケーリングポリシーの設定を指定できます。次の例では、インスタンスグループ設定がある JSON ファイル *instancegroupconfig.json* への参照を使用しています。

```
aws emr add-instance-groups --cluster-id j-1EKZ3TYEVF1S2 --instance-groups file://your/path/to/instancegroupconfig.json
```

既存のインスタンスグループに自動スケーリングポリシーを適用するか、適用されたポリシーを変更する

aws emr put-auto-scaling-policy コマンドを使用して自動スケーリングポリシーを既存のインスタンスグループに適用します。インスタンスグループは、自動スケーリング IAM ロールを使用

するクラスターの一部である必要があります。次の例では、自動スケーリングポリシー設定を指定する JSON ファイル *autoscaleconfig.json* への参照を使用します。

```
aws emr put-auto-scaling-policy --cluster-id j-1EKZ3TYEVF1S2 --instance-group-
id ig-3PLUZBA6WLS07 --auto-scaling-policy file://your/path/to/autoscaleconfig.json
```

autoscaleconfig.json ファイルのコンテンツは、前の例と同じスケールアウトルールを定義するもので、次に示されています。

```
{
          "Constraints": {
                  "MaxCapacity": 10,
                  "MinCapacity": 2
          },
          "Rules": [{
                  "Action": {
                          "SimpleScalingPolicyConfiguration": {
                                  "AdjustmentType": "CHANGE_IN_CAPACITY",
                                  "CoolDown": 300,
                                  "ScalingAdjustment": 1
                          }
                  },
                  "Description": "Replicates the default scale-out rule in the console
 for YARN memory",
                  "Name": "Default-scale-out",
                  "Trigger": {
                          "CloudWatchAlarmDefinition": {
                                  "ComparisonOperator": "LESS_THAN",
                                  "Dimensions": [{
                                          "Key": "JobFlowId",
                                          "Value": "${emr.clusterID}"
                                  }],
                                  "EvaluationPeriods": 1,
                                  "MetricName": "YARNMemoryAvailablePercentage",
                                  "Namespace": "AWS/ElasticMapReduce",
                                  "Period": 300,
                                  "Statistic": "AVERAGE",
                                  "Threshold": 15,
                                  "Unit": "PERCENT"
                          }
                  }
          }]
```

```
  }
```

自動スケーリングポリシーをインスタンスグループから削除する

```
aws emr remove-auto-scaling-policy --cluster-id j-1EKZ3TYEVF1S2 --instance-group-
id ig-3PLUZBA6WLS07
```

自動スケーリングポリシー設定を取得する

describe-cluster コマンドは、 InstanceGroup ブロック内のポリシー設定を取得します。たとえば、次のコマンドは、クラスター ID j-1CWOHP4PI30VJ を持つクラスターの設定を取得します。

```
aws emr describe-cluster --cluster-id j-1CWOHP4PI30VJ
```

このコマンドでは、次のサンプルアウトプットが生成されます。

```
{
    "Cluster": {
        "Configurations": [],
        "Id": "j-1CWOHP4PI30VJ",
        "NormalizedInstanceHours": 48,
        "Name": "Auto Scaling Cluster",
        "ReleaseLabel": "emr-5.2.0",
        "ServiceRole": "EMR_DefaultRole",
        "AutoTerminate": false,
        "TerminationProtected": true,
        "MasterPublicDnsName": "ec2-54-167-31-38.compute-1.amazonaws.com",
        "LogUri": "s3n://aws-logs-232939870606-us-east-1/elasticmapreduce/",
        "Ec2InstanceAttributes": {
            "Ec2KeyName": "performance",
            "AdditionalMasterSecurityGroups": [],
            "AdditionalSlaveSecurityGroups": [],
            "EmrManagedSlaveSecurityGroup": "sg-09fc9362",
            "Ec2AvailabilityZone": "us-east-1d",
            "EmrManagedMasterSecurityGroup": "sg-0bfc9360",
            "IamInstanceProfile": "EMR_EC2_DefaultRole"
        },
        "Applications": [
            {
```

```
                "Name": "Hadoop",
                "Version": "2.7.3"
            }
        ],
        "InstanceGroups": [
            {
                "AutoScalingPolicy": {
                    "Status": {
                        "State": "ATTACHED",
                        "StateChangeReason": {
                            "Message": ""
                        }
                    },
                    "Constraints": {
                        "MaxCapacity": 10,
                        "MinCapacity": 2
                    },
                    "Rules": [
                        {
                            "Name": "Default-scale-out",
                            "Trigger": {
                                "CloudWatchAlarmDefinition": {
                                    "MetricName": "YARNMemoryAvailablePercentage",
                                    "Unit": "PERCENT",
                                    "Namespace": "AWS/ElasticMapReduce",
                                    "Threshold": 15,
                                    "Dimensions": [
                                        {
                                            "Key": "JobFlowId",
                                            "Value": "j-1CWOHP4PI30VJ"
                                        }
                                    ],
                                    "EvaluationPeriods": 1,
                                    "Period": 300,
                                    "ComparisonOperator": "LESS_THAN",
                                    "Statistic": "AVERAGE"
                                }
                            },
                            "Description": "",
                            "Action": {
                                "SimpleScalingPolicyConfiguration": {
                                    "CoolDown": 300,
                                    "AdjustmentType": "CHANGE_IN_CAPACITY",
                                    "ScalingAdjustment": 1
```

```
                        }
                    }
                },
                {
                    "Name": "Default-scale-in",
                    "Trigger": {
                        "CloudWatchAlarmDefinition": {
                            "MetricName": "YARNMemoryAvailablePercentage",
                            "Unit": "PERCENT",
                            "Namespace": "AWS/ElasticMapReduce",
                            "Threshold": 75,
                            "Dimensions": [
                                {
                                    "Key": "JobFlowId",
                                    "Value": "j-1CWOHP4PI30VJ"
                                }
                            ],
                            "EvaluationPeriods": 1,
                            "Period": 300,
                            "ComparisonOperator": "GREATER_THAN",
                            "Statistic": "AVERAGE"
                        }
                    },
                    "Description": "",
                    "Action": {
                        "SimpleScalingPolicyConfiguration": {
                            "CoolDown": 300,
                            "AdjustmentType": "CHANGE_IN_CAPACITY",
                            "ScalingAdjustment": -1
                        }
                    }
                }
            ]
        },
        "Configurations": [],
        "InstanceType": "m5.xlarge",
        "Market": "ON_DEMAND",
        "Name": "Core - 2",
        "ShrinkPolicy": {},
        "Status": {
            "Timeline": {
                "CreationDateTime": 1479413437.342,
                "ReadyDateTime": 1479413864.615
            },
```

```
                    "State": "RUNNING",
                    "StateChangeReason": {
                        "Message": ""
                    }
                },
                "RunningInstanceCount": 2,
                "Id": "ig-3M16XBE8C3PH1",
                "InstanceGroupType": "CORE",
                "RequestedInstanceCount": 2,
                "EbsBlockDevices": []
            },
            {
                "Configurations": [],
                "Id": "ig-OP62I28NSE8M",
                "InstanceGroupType": "MASTER",
                "InstanceType": "m5.xlarge",
                "Market": "ON_DEMAND",
                "Name": "Master - 1",
                "ShrinkPolicy": {},
                "EbsBlockDevices": [],
                "RequestedInstanceCount": 1,
                "Status": {
                    "Timeline": {
                        "CreationDateTime": 1479413437.342,
                        "ReadyDateTime": 1479413752.088
                    },
                    "State": "RUNNING",
                    "StateChangeReason": {
                        "Message": ""
                    }
                },
                "RunningInstanceCount": 1
            }
        ],
        "AutoScalingRole": "EMR_AutoScaling_DefaultRole",
        "Tags": [],
        "BootstrapActions": [],
        "Status": {
            "Timeline": {
                "CreationDateTime": 1479413437.339,
                "ReadyDateTime": 1479413863.666
            },
            "State": "WAITING",
            "StateChangeReason": {
```

```
                "Message": "Cluster ready after last step completed."
            }
        }
    }
}
```

実行中のクラスターのサイズを手動で変更する

、、または Amazon EMR API を使用して AWS Management Console、実行中のクラスターのコアインスタンスグループとタスクインスタンスグループ AWS CLI、およびインスタンスフリートからインスタンスを追加および削除できます。クラスターがインスタンスグループを使用する場合、明示的にインスタンス数を変更します。クラスターでインスタンスフリートを使用する場合、オンデマンドインスタンスとスポットインスタンスのターゲットユニットを変更できます。次に、インスタンスフリートは新しいターゲットに合わせてインスタンスを追加または削除します。詳細については、「インスタンスフリートオプション」を参照してください。アプリケーションは、インスタンスが利用可能になったらすぐに、新しくプロビジョニングされた Amazon EC2 インスタンスを使用してノードをホストできます。インスタンスを削除すると、Amazon EMR はジョブを中断しない方法でタスクを終了し、データ損失を防止します。詳細については、「タスクの完了時に終了」を参照してください。

コンソールを使用してクラスターのサイズを変更する

Amazon EMR コンソールを使用して、実行中のクラスターのサイズを変更できます。

Console

新しいコンソールを使用して既存のクラスターのインスタンス数を変更するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で [クラスター] を選択し、更新するクラスターを選択します。クラスターは実行中である必要があります。プロビジョニングしているクラスターや終了したクラスターのサイズを変更することはできません。
3. クラスターの詳細ページの [インスタンス] タブで、[インスタンスグループ] パネルを表示します。
4. 既存のインスタンスグループのサイズを変更するには、サイズを変更するコアインスタンスグループまたはタスクインスタンスグループの横にあるラジオボタンを選択し、[インスタン

スグループのサイズを変更] を選択します。インスタンスグループの新しいインスタンス数を指定し、[サイズ変更] を選択します。

> **Note**
>
> 実行中のインスタンスグループのサイズを縮小することを選択した場合、Amazon EMR はデータ損失を最小限に抑えるため、グループから削除するインスタンスをインテリジェントに選択します。サイズ変更アクションをより細かく管理するには、インスタンスグループの [ID] を選択し、削除するインスタンスを選択して、[削除] オプションを使用します。インテリジェントなスケールダウン動作の詳細については、「クラスターのスケールダウン」を参照してください。

5. サイズ変更アクションをキャンセルする場合は、ステータスが [サイズ変更中] のインスタンスグループのラジオボタンを選択し、アクションのリストから [サイズ変更を停止] を選択します。
6. ワークロードの増加に応じて 1 つまたは複数のタスクインスタンスグループをクラスターに追加するには、リストアクションから [タスクインスタンスグループを追加] を選択します。Amazon EC2 インスタンスタイプを選択し、タスクグループのインスタンス数を入力して、[タスクインスタンスグループを追加] を選択すると、クラスターの [インスタンスグループ] パネルに戻ります。

ノードの数を変更すると、インスタンスグループの [Status (ステータス)] が更新されます。変更リクエストが完了すると、[Status (ステータス)] が [Running (実行中)] になります。

を使用してクラスターのサイズを変更する AWS CLI

を使用して AWS CLI 、実行中のクラスターのサイズを変更できます。タスクノードの数は増減することができ、実行中のクラスターのコアノードの数を増やすことができます。 AWS CLI または API を使用して、コアインスタンスグループのインスタンスをシャットダウンすることもできます。そうする場合は、注意が必要です。コアインスタンスグループのインスタンスをシャッドダウンすると、データが失われる可能性があり、インスタンスは自動的に置換されません。

コアグループおよびタスクグループのサイズの変更以外に、 AWS CLIを使用して、実行中のクラスターに 1 つ以上のタスクインスタンスグループを追加することもできます。

でインスタンス数を変更してクラスターのサイズを変更するには AWS CLI

コアグループまたはタスクグループにインスタンスを追加したり、 InstanceCount パラメータを使用して サブコマンドを使用して AWS CLI modify-instance-groupsタスクグループからインスタンスを削除したりできます。コアグループまたはタスクグループにインスタンスを追加するには、InstanceCount を増やします。タスクグループのインスタンス数を減らすには、InstanceCount を減らします。タスクグループのインスタンス数を 0 に変更すると、すべてのインスタンスが削除されますが、インスタンスグループは削除されません。

- タスクインスタンスグループのインスタンス数を 3 個から 4 個に増やすには、次のコマンドを入力し、*ig-31JXXXXXXBT0* をインスタンスグループ ID に置き換えます。

```
aws emr modify-instance-groups --instance-groups
 InstanceGroupId=ig-31JXXXXXXBT0,InstanceCount=4
```

InstanceGroupId を取得するには、describe-cluster サブコマンドを使用します。出力は、各インスタンスグループの ID が含まれている、Cluster という名前の JSON オブジェクトです。このコマンドを使用するには、クラスター ID が必要です (aws emr list-clusters コマンドまたはコンソールを使用して取得できます)。インスタンスグループ ID を取得するには、次のコマンドを入力し、*j-2AXXXXXXGAPLF* をクラスター ID に置き換えます。

```
aws emr describe-cluster --cluster-id j-2AXXXXXXGAPLF
```

では AWS CLI、 --modify-instance-groupsサブコマンドを使用してコアインスタンスグループのインスタンスを終了することもできます。

> **Warning**
>
> EC2InstanceIdsToTerminate の指定は注意して行う必要があります。インスタンスは、そこで実行中のアプリケーションのステータスにかかわらず即時削除され、自動的には置き換えられません。これは、クラスターの [Scale down behavior] 設定とは無関係です。この方法でインスタンスを削除すると、データ損失や、予測不可能なクラスター動作が発生する可能性があります。

特定のインスタンスを終了するには、インスタンスグループ ID (aws emr describe-cluster --cluster-id サブコマンドによって返されます) とインスタンス ID (aws emr list-instances --cluster-id サブコマンドによって返されます) が必要です。次のコマ

ンドを入力し、*ig-6RXXXXXX07SA* をインスタンスグループ ID に置き換え、*i-f9XXXXf2* をインスタンス ID に置き換えます。

```
aws emr modify-instance-groups --instance-groups
 InstanceGroupId=ig-6RXXXXXX07SA,EC2InstanceIdsToTerminate=i-f9XXXXf2
```

での Amazon EMR コマンドの使用の詳細については AWS CLI、「」を参照してくださいhttps://docs.aws.amazon.com/cli/latest/reference/emr。

でタスクインスタンスグループを追加してクラスターのサイズを変更するには AWS CLI

では AWS CLI、1～48 個のタスクインスタンスグループを `--add-instance-groups`サブコマンドでクラスターに追加できます。タスクインスタンスグループは、プライマリインスタンスグループとコアインスタンスグループが含まれるクラスターにのみ追加できます。を使用すると AWS CLI、`--add-instance-groups`サブコマンドを使用するたびに最大 5 つのタスクインスタンスグループを追加できます。

1. クラスターに 1 つのタスクインスタンスグループを追加するには、次のコマンドを入力し、*j-JXBXXXXXX37R* をクラスター ID に置き換えます。

   ```
   aws emr add-instance-groups --cluster-id j-JXBXXXXXX37R --instance-groups
    InstanceCount=6,InstanceGroupType=task,InstanceType=m5.xlarge
   ```

2. クラスターに複数のタスクインスタンスグループを追加するには、次のコマンドを入力し、*j-JXBXXXXXX37R* をクラスター ID に置き換えます。1 つのコマンドで最大 5 個のタスクインスタンスグループを追加できます。

   ```
   aws emr add-instance-groups --cluster-id j-JXBXXXXXX37R --instance-
   groups InstanceCount=6,InstanceGroupType=task,InstanceType=m5.xlarge
    InstanceCount=10,InstanceGroupType=task,InstanceType=m5.xlarge
   ```

   での Amazon EMR コマンドの使用の詳細については AWS CLI、「」を参照してくださいhttps://docs.aws.amazon.com/cli/latest/reference/emr。

サイズ変更を中断する

Amazon EMR バージョン 4.1.0 以降を使用して、既存のサイズ変更操作中に、サイズ変更を実行できます。さらに、既に提出されたサイズ変更リクエストを中止したり、新規リクエストを提

出して先のリクエストの処理が終了するのを待たずに上書きすることもできます。また、既存のサイズ変更をコンソールから中止することも、クラスターのターゲット数を現在の数とした ModifyInstanceGroups API 呼び出しを使用して中止することも可能です。

以下のスクリーンショットには、[Stop] を選択することで中止することのできる、サイズ変更中のタスクインスタンスグループが示されています。

を使用してサイズ変更を中断するには AWS CLI

AWS CLI を使用して、 modify-instance-groupsサブコマンドでサイズ変更を停止できます。インスタンスグループに 6 つのインスタンスがあり、これを 10 に増やしたいとします。そして、その後このリクエストをキャンセルしたいとします:

- 最初のリクエスト:

```
aws emr modify-instance-groups --instance-groups
 InstanceGroupId=ig-myInstanceGroupId,InstanceCount=10
```

最初のリクエストを中止する 2 番目のリクエスト:

```
aws emr modify-instance-groups --instance-groups
 InstanceGroupId=ig-myInstanceGroupId,InstanceCount=6
```

> **Note**
>
> この処理は非同期であるため、後から申請したリクエストが反映される前に、以前の API リクエストに関したインスタンスの数の変更が表示される場合があります。サイズを縮小する場合、現行のノードで作業中のインスタンスグループのサイズは、それらのノードにおける作業が完了するまで縮小されないことがあります。

停止状態

新しいクラスターノードを起動しようとしているときに、多数のエラーが発生すると、インスタンスグループが停止状態になります。たとえば、ブートストラップアクションの実行中に新しいノードが失敗した場合、インスタンスグループは SUSPENDED 状態になり、新しいノードのプロビジョニングが続行されなくなります。基本となる問題を解決したら、クラスターのインスタンスグループで必

要な数のノードをリセットしてください。その後、インスタンスグループはノードの割り当てを再開します。インスタンスグループを変更すると、Amazon EMR は再度ノードをプロビジョニングしようとします。実行中のノードについては再開または終了しません。

では AWS CLI、 `list-instances` サブコマンドは、 `describe-cluster`サブコマンドと同様に、すべてのインスタンスとその状態を返します。Amazon EMR によってインスタンスグループのエラーが検出されると、グループの状態が SUSPENDED に変更されます。

を使用して SUSPENDED 状態のクラスターをリセットするには AWS CLI

クラスター内のインスタンスの状態を表示するには、`describe-cluster` サブコマンドを入力し、`--cluster-id` パラメータを指定します。

- クラスター内のすべてのインスタンスとインスタンスグループについての情報を表示するには、次のコマンドを入力し、*j-3KVXXXXXXY7UG* をクラスター ID に置き換えます。

  ```
  aws emr describe-cluster --cluster-id j-3KVXXXXXXY7UG
  ```

  出力には、インスタンスグループとインスタンスの状態に関する情報が表示されます:

  ```
  {
      "Cluster": {
          "Status": {
              "Timeline": {
                  "ReadyDateTime": 1413187781.245,
                  "CreationDateTime": 1413187405.356
              },
              "State": "WAITING",
              "StateChangeReason": {
                  "Message": "Waiting after step completed"
              }
          },
          "Ec2InstanceAttributes": {
              "Ec2AvailabilityZone": "us-west-2b"
          },
          "Name": "Development Cluster",
          "Tags": [],
          "TerminationProtected": false,
          "RunningAmiVersion": "3.2.1",
          "NormalizedInstanceHours": 16,
          "InstanceGroups": [
              {
  ```

```
                "RequestedInstanceCount": 1,
                "Status": {
                    "Timeline": {
                        "ReadyDateTime": 1413187775.749,
                        "CreationDateTime": 1413187405.357
                    },
                    "State": "RUNNING",
                    "StateChangeReason": {
                        "Message": ""
                    }
                },
                "Name": "MASTER",
                "InstanceGroupType": "MASTER",
                "InstanceType": "m5.xlarge",
                "Id": "ig-3ETXXXXXXFYV8",
                "Market": "ON_DEMAND",
                "RunningInstanceCount": 1
            },
            {
                "RequestedInstanceCount": 1,
                "Status": {
                    "Timeline": {
                        "ReadyDateTime": 1413187781.301,
                        "CreationDateTime": 1413187405.357
                    },
                    "State": "RUNNING",
                    "StateChangeReason": {
                        "Message": ""
                    }
                },
                "Name": "CORE",
                "InstanceGroupType": "CORE",
                "InstanceType": "m5.xlarge",
                "Id": "ig-3SUXXXXXQ9ZM",
                "Market": "ON_DEMAND",
                "RunningInstanceCount": 1
            }
...
}
```

特定のインスタンスグループについての情報を表示するには、list-instances サブコマンドを入力し、--cluster-id および --instance-group-types パラメータを指定します。プライマリ、コアまたはタスクグループの情報を表示できます。

```
aws emr list-instances --cluster-id j-3KVXXXXXXY7UG --instance-group-types "CORE"
```

modify-instance-groups 状態のクラスターをリセットするには、--instance-groups サブコマンドを使用し、SUSPENDED パラメータを指定します。インスタンスグループ ID は、describe-cluster サブコマンドによって返されます。

```
aws emr modify-instance-groups --instance-groups
 InstanceGroupId=ig-3SUXXXXXXQ9ZM,InstanceCount=3
```

クラスターサイズを縮小する場合の考慮事項

実行中のクラスターのサイズを縮小する場合は、以下の Amazon EMR の動作とベストプラクティスについて考慮してください。

- 進行中のジョブへの影響を減らすため、Amazon EMR は削除するインスタンスをインテリジェントに選択します。クラスターのスケールダウン動作の詳細については、「Amazon EMR 管理ガイド」の「タスクの完了時に終了」を参照してください。
- クラスターのサイズをスケールダウンすると、Amazon EMR は削除したインスタンスから残っているインスタンスにデータをコピーします。グループに残っているインスタンスに、このデータを保存するのに十分なストレージ容量があることを確認してください。
- Amazon EMR は、グループ内のインスタンスの HDFS を廃止しようと試みます。クラスターのサイズを縮小する前に、HDFS への書き込み I/O を最小限に抑えることをお勧めします。
- クラスターのサイズ縮小時に最も細かく管理するには、コンソールでクラスターを表示して [インスタンス] タブに移動します。サイズを変更するインスタンスグループの [ID] を選択します。次に、削除する特定のインスタンスに対して [削除] オプションを使用します。

容量のプロビジョニングのタイムアウトを設定する

インスタンスフリートを使用すると、プロビジョニングのタイムアウトを設定できます。プロビジョニングのタイムアウトでは、クラスターの起動時やクラスターのスケーリング操作時に、クラスターが指定された時間のしきい値を超えた場合、インスタンス容量のプロビジョニングを停止するよう Amazon EMR に指示します。以下のトピックでは、クラスターの起動とクラスターのスケールアップ操作に対してプロビジョニングのタイムアウトを設定する方法について説明します。

トピック

- Amazon EMR でクラスター起動のプロビジョニングのタイムアウトを設定する
- Amazon EMR でクラスターサイズ変更のプロビジョニングのタイムアウト期間をカスタマイズする

Amazon EMR でクラスター起動のプロビジョニングのタイムアウトを設定する

クラスター内の各フリートにスポットインスタンスをプロビジョニングするためのタイムアウト期間を定義できます。Amazon EMR がスポット容量をプロビジョニングできない場合は、代わりにクラスターを終了させるか、オンデマンド容量をプロビジョニングするかを選択できます。クラスターのサイズ変更プロセス中にタイムアウト期間が終了すると、Amazon EMR はプロビジョニングされていないスポットリクエストをキャンセルします。プロビジョニングされていないスポットインスタンスは、オンデマンド容量に転送されません。

> **Note**
>
> 古いコンソールではプロビジョニングのタイムアウト期間をカスタマイズできません。古いコンソールと新しいコンソールエクスペリエンスの違いについては、「Amazon EMR コンソール」を参照してください。

Amazon EMR コンソールでクラスター起動のプロビジョニングのタイムアウト期間をカスタマイズするには、次の手順を実行します。

New console

新しいコンソールを使用してクラスターの作成時にプロビジョニングのタイムアウトを設定するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で、[クラスター] を選択し、[クラスターの作成] を選択します
3. [クラスターの作成] ページで [クラスター設定] に移動し、[インスタンスフリート] を選択します。
4. [クラスターのスケーリングとプロビジョニングのオプション] で、コアフリートとタスクフリートのスポットサイズを指定します。

5. [スポットタイムアウトの設定] で、[スポットタイムアウト後にクラスターを終了する] または [スポットタイムアウト後にオンデマンドに切り替える] のいずれかを選択します。次に、スポットインスタンスのプロビジョニングのタイムアウト期間を指定します。デフォルト値は 1 時間です。
6. クラスターに適用するその他のオプションを選択します。
7. 設定したタイムアウトを使用してクラスターを起動するには、[クラスターの作成] を選択します。

AWS CLI

create-cluster コマンドでプロビジョニングのタイムアウトを指定するには

```
aws emr create-cluster \
--release-label emr-5.35.0 \
--service-role EMR_DefaultRole \
--ec2-attributes '{"InstanceProfile":"EMR_EC2_DefaultRole","SubnetIds":["subnet-
XXXXX"]}' \
--instance-fleets
 '[{"InstanceFleetType":"MASTER","TargetOnDemandCapacity":1,"TargetSpotCapacity":0,"LaunchSp
{"OnDemandSpecification":{"AllocationStrategy":"lowest-
price"}},"InstanceTypeConfigs":[{"WeightedCapacity":1,"EbsConfiguration":
{"EbsBlockDeviceConfigs":[{"VolumeSpecification":
{"SizeInGB":32,"VolumeType":"gp2"},"VolumesPerInstance":2}]},"BidPriceAsPercentageOfOnDemand
 - 1"},
{"InstanceFleetType":"CORE","TargetOnDemandCapacity":1,"TargetSpotCapacity":1,"LaunchSpecifi
{"SpotSpecification":
{"TimeoutDurationMinutes":120,"TimeoutAction":"SWITCH_TO_ON_DEMAND"},"OnDemandSpecification"
{"AllocationStrategy":"lowest-price"}},"InstanceTypeConfigs":
[{"WeightedCapacity":1,"EbsConfiguration":{"EbsBlockDeviceConfigs":
[{"VolumeSpecification":
{"SizeInGB":32,"VolumeType":"gp2"},"VolumesPerInstance":2}]},"BidPriceAsPercentageOfOnDemand
 - 2"}]'
```

Amazon EMR でクラスターサイズ変更のプロビジョニングのタイムアウト期間をカスタマイズする

クラスター内の各フリートにスポットインスタンスをプロビジョニングするためのタイムアウト期間を定義できます。Amazon EMR はスポット容量をプロビジョニングできない場合、サイズ変更リクエストをキャンセルし、追加のスポット容量をプロビジョニングしようとする試みを停止します。ク

ラスター作成時にタイムアウトを設定できます。実行中のクラスターでは、タイムアウトを追加や更新できます。

タイムアウト期間が終了すると、Amazon EMR は自動的にイベントを Amazon CloudWatch Events ストリームに送信します。では CloudWatch、指定したパターンに従ってイベントを照合するルールを作成し、イベントをターゲットにルーティングしてアクションを実行できます。例えば、E メール通知を送信するようにルールを設定できます。ルールの作成方法の詳細については、「を使用した Amazon EMR イベントのルールの作成 CloudWatch」を参照してください。さまざまなイベントの詳細については、「インスタンスフリートの状態変更イベント」を参照してください。

クラスターのサイズ変更におけるプロビジョニングのタイムアウトの例

AWS CLIでサイズ変更のプロビジョニングのタイムアウトを指定する

次の例では、create-cluster コマンドを使用して、サイズ変更のプロビジョニングのタイムアウトを追加しています。

```
aws emr create-cluster \
--release-label emr-5.35.0 \
--service-role EMR_DefaultRole \
--ec2-attributes '{"InstanceProfile":"EMR_EC2_DefaultRole","SubnetIds":["subnet-
XXXXX"]}' \
--instance-fleets
 '[{"InstanceFleetType":"MASTER","TargetOnDemandCapacity":1,"TargetSpotCapacity":0,"InstanceTyp
[{"WeightedCapacity":1,"EbsConfiguration":{"EbsBlockDeviceConfigs":
[{"VolumeSpecification":
{"SizeInGB":32,"VolumeType":"gp2"},"VolumesPerInstance":2}]},"BidPriceAsPercentageOfOnDemandPri
 - 1"},
{"InstanceFleetType":"CORE","TargetOnDemandCapacity":1,"TargetSpotCapacity":1,"LaunchSpecificat
{"SpotSpecification":
{"TimeoutDurationMinutes":120,"TimeoutAction":"SWITCH_TO_ON_DEMAND"},"OnDemandSpecification":
{"AllocationStrategy":"lowest-price"}},"ResizeSpecifications":
{"SpotResizeSpecification":{"TimeoutDurationMinutes":20},"OnDemandResizeSpecification":
{"TimeoutDurationMinutes":25}},"InstanceTypeConfigs":
[{"WeightedCapacity":1,"EbsConfiguration":{"EbsBlockDeviceConfigs":
[{"VolumeSpecification":
{"SizeInGB":32,"VolumeType":"gp2"},"VolumesPerInstance":2}]},"BidPriceAsPercentageOfOnDemandPri
 - 2"}]'
```

次の例では、modify-instance-fleet コマンドを使用して、サイズ変更のプロビジョニングのタイムアウトを追加しています。

```
aws emr modify-instance-fleet \
--cluster-id j-XXXXXXXXXXXX \
--instance-fleet '{"InstanceFleetId":"if-XXXXXXXXXXX","ResizeSpecifications":
{"SpotResizeSpecification":{"TimeoutDurationMinutes":30},"OnDemandResizeSpecification":
{"TimeoutDurationMinutes":60}}}' \
--region us-east-1
```

次の例では、add-instance-fleet-command を使用して、サイズ変更のプロビジョニングのタイムアウトを追加しています。

```
aws emr add-instance-fleet \
--cluster-id j-XXXXXXXXXXXX \
--instance-fleet
 '{"InstanceFleetType":"TASK","TargetOnDemandCapacity":1,"TargetSpotCapacity":0,"InstanceTypeCo
[{"WeightedCapacity":1,"EbsConfiguration":{"EbsBlockDeviceConfigs":
[{"VolumeSpecification":
{"SizeInGB":32,"VolumeType":"gp2"},"VolumesPerInstance":2}]},"BidPriceAsPercentageOfOnDemandPri
{"SpotResizeSpecification":{"TimeoutDurationMinutes":30},"OnDemandResizeSpecification":
{"TimeoutDurationMinutes":35}}}' \
--region us-east-1
```

でサイズ変更と起動のプロビジョニングタイムアウトを指定する AWS CLI

次の例では、create-cluster コマンドを使用して、サイズ変更と起動のプロビジョニングのタイムアウトを追加しています。

```
aws emr create-cluster \
--release-label emr-5.35.0 \
--service-role EMR_DefaultRole \
--ec2-attributes '{"InstanceProfile":"EMR_EC2_DefaultRole","SubnetIds":["subnet-
XXXXX"]}' \
--instance-fleets
 '[{"InstanceFleetType":"MASTER","TargetOnDemandCapacity":1,"TargetSpotCapacity":0,"LaunchSpeci
{"OnDemandSpecification":{"AllocationStrategy":"lowest-price"}},"InstanceTypeConfigs":
[{"WeightedCapacity":1,"EbsConfiguration":{"EbsBlockDeviceConfigs":
[{"VolumeSpecification":
{"SizeInGB":32,"VolumeType":"gp2"},"VolumesPerInstance":2}]},"BidPriceAsPercentageOfOnDemandPri
 - 1"},
{"InstanceFleetType":"CORE","TargetOnDemandCapacity":1,"TargetSpotCapacity":1,"LaunchSpecificat
{"SpotSpecification":
{"TimeoutDurationMinutes":120,"TimeoutAction":"SWITCH_TO_ON_DEMAND"},"OnDemandSpecification":
{"AllocationStrategy":"lowest-price"}},"ResizeSpecifications":
```

```
{"SpotResizeSpecification":{"TimeoutDurationMinutes":20},"OnDemandResizeSpecification":
{"TimeoutDurationMinutes":25}},"InstanceTypeConfigs":
[{"WeightedCapacity":1,"EbsConfiguration":{"EbsBlockDeviceConfigs":
[{"VolumeSpecification":
{"SizeInGB":32,"VolumeType":"gp2"},"VolumesPerInstance":2}]},"BidPriceAsPercentageOfOnDemandPri
 - 2"}]'
```

サイズ変更のプロビジョニングのタイムアウトに関する考慮事項

インスタンスフリートに対してクラスターのプロビジョニングのタイムアウトを設定するときは、以下の動作を考慮してください。

- スポットインスタンスとオンデマンドインスタンスの両方でプロビジョニングのタイムアウトを設定できます。プロビジョニングの最小タイムアウトは 5 分です。プロビジョニングの最大タイムアウトは 7 日間です。
- プロビジョニングのタイムアウトは、インスタンスフリートを使用する EMR クラスターにのみ設定できます。コアフリートとタスクフリートはそれぞれ個別に設定する必要があります。
- クラスターの作成時に、プロビジョニングのタイムアウトを設定できます。実行中のクラスターに対してタイムアウトを追加したり、既存のタイムアウトを更新したりできます。
- 複数のサイズ変更操作を送信すると、Amazon EMR は各サイズ変更操作のプロビジョニングのタイムアウトを追跡します。例えば、クラスターのプロビジョニングのタイムアウトを *60* 分に設定します。*##### T1* にサイズ変更操作 *R1* を送信します。時間 *T2* に 2 つ目のサイズ変更操作 *R2* を送信します。R1 のプロビジョニングのタイムアウト期限は *T1 + 60 #*です。R2 のプロビジョニングのタイムアウト期限は *T2 + 60 #*です。
- タイムアウト時間が過ぎる前に新しいスケールアップのサイズ変更操作を送信した場合、Amazon EMR は EMR クラスターの容量をプロビジョニングし続けようとします。

クラスターのスケールダウン

> **Note**
>
> スケールダウン動作オプションは、Amazon EMR リリース 5.10.0 以降でサポートされなくなりました。Amazon EC2 に秒単位の請求が導入されたため、Amazon EMR クラスターのデフォルトのスケールダウン動作は、タスクの完了時に終了するようになりました。

Amazon EMR リリース 5.1.0 から 5.9.1 のスケールダウン動作には、Amazon EC2 請求のインスタンス時間の境界での終了と、タスク完了時の終了の 2 つのオプションがあります。Amazon EMR リリース 5.10.0 以降では、Amazon EC2 に秒単位の請求が導入されたため、インスタンス時間の境界での終了の設定は廃止されています。このオプションが使用できないバージョンでは、インスタンス時間の境界での終了を指定することはお勧めされていません。

> **Warning**
>
> を使用して `modify-instance-groups` で AWS CLI を発行する場合`EC2InstanceIdsToTerminate`、これらのインスタンスは、これらの設定を考慮せずに、実行中のアプリケーションのステータスに関係なく、すぐに終了します。この方法でインスタンスを削除すると、データ損失や、予測不可能なクラスター動作が発生する可能性があります。

タスク完了時の終了を指定すると、Amazon EMR は、Amazon EC2 インスタンスを削除する前にタスクを拒否リストに登録し、ノードから削除します。いずれの動作を指定しても、HDFS の破損につながる可能性があれば、Amazon EMR はコアインスタンスグループの Amazon EC2 インスタンスを削除しません。

タスクの完了時に終了

Amazon EMR では、ワークロードに影響を与えずにクラスターをスケールダウンできます。Amazon EMR は、データを失ったりジョブを中断したりすることなく、サイズ縮小処理中にコアノードとタスクノードの YARN、HDFS、およびその他のデーモンを適切に停止します。Amazon EMR は、グループに割り当てられた作業が完了し、アイドル状態の場合にのみインスタンスグループのサイズを縮小します。YARN NodeManager Graceful Decommission では、ノードが廃止を待機する時間を手動で調整できます。

この時間は、`YARN-site` 設定分類のプロパティを使用して設定します。5.12.0 以降 Amazon EMR リリースを使用する場合、`YARN.resourcemanager.nodemanager-graceful-decommission-timeout-secs` プロパティを指定します。以前の Amazon EMR リリースを使用する場合、`YARN.resourcemanager.decommissioning.timeout` プロパティを指定します。

停止時間がタイムアウトした時点で実行中のコンテナーまたは YARN アプリケーションがあった場合、そのノードは強制的に停止され、実行中のコンテナーは YARN によって他のノードで再スケジュールされます。デフォルト値は 3,600 秒 (1 時間) です。このタイムアウトを任意の高い値に設定することで、グレースフルな縮小を強制的に行い、長い時間待機させることができます。詳細につ

いては、Apache Hadoop ドキュメントの「Graceful Decommission of YARN Nodes」を参照してください。

タスクノードグループ

Amazon EMR は、ステップやアプリケーションに対して実行されているタスクがないインスタンスをインテリジェントに選択し、それらのインスタンスを最初にクラスターから削除します。クラスター内のすべてのインスタンスが使用されている場合、Amazon EMR はインスタンスのタスクが完了するの待ってからクラスターから削除します。デフォルトの待機時間は 1 時間です。この値は `YARN.resourcemanager.decommissioning.timeout` 設定で変更できます。Amazon EMR では、この新しい設定が動的に使用されます。この値を任意の大きな数に設定することで、Amazon EMR がタスクを終了することなく、クラスターのサイズを縮小できます。

コアノードグループ

コアノードでは、インスタンスグループを減らすために YARN デーモン NodeManager と HDFS DataNode デーモンの両方を廃止する必要があります。YARN では、グレースフルなサイズ縮小により、停止の対象となるノードは、保留中や未完了のコンテナまたはアプリケーションがない場合にのみ `DECOMMISSIONED` 状態に移行します。停止作業開始時においてノードでコンテナーが実行されていない場合、停止作業は即終了します。

HDFS では、グレースフルなサイズ縮小により、HDFS のターゲット容量にすべての既存ブロックが収まるよう十分な大きさが確保されます。ターゲット容量の大きさが十分でない場合、残りのノードが HDFS にある現在のデータを処理できるように、一部のコアインスタンスのみが停止されます。ノードが完全に停止されるよう、HDFS に十分な容量があるよう確認してください。また、インスタンスグループの削減を試みる前に、書き込み I/O を最小限に抑えるようにしてください。書き込み I/O が多すぎると、サイズ変更操作の完了が遅れる可能性があります。

もう 1 つの制限は、デフォルトのレプリケーション係数です。`dfs.replication`内部`/etc/hadoop/conf/hdfs-site`。Amazon EMR では、クラスターの作成時に、クラスター内のインスタンス数に基づいて値が設定されます。インスタンス数が 1～3 の場合は 1、4～9 の場合は 2、10 以上の場合は 3 となります。

> **Warning**
>
> 1. ノードが 4 つ未満のクラスターで `dfs.replication` を 1 に設定すると、単一ノードがダウンした場合に HDFS データが失われる可能性があります。本番環境のワークロードには、少なくとも 4 つのコアノードを持つクラスターを使用することをお勧めします。

2. Amazon EMR では、クラスターはコアノードを dfs.replication 未満にスケールすることはできません。例えば、dfs.replication = 2 の場合、コアノードの最小数は 2 です。
3. マネージドスケーリングや自動スケーリングを使用する場合や、クラスターのサイズを手動で変更する場合は、dfs.replication を 2 以上に設定することをお勧めします。

グレースフルな縮小では、コアノードを HDFS のレプリケーション係数未満に減らすことはできません。これは、レプリカが不十分な場合に HDFS がファイルを閉じることができるようにするためです。この制限を回避するには、レプリケーション係数を下げてNameNode デーモンを再起動します。

Amazon EMR のスケールダウン動作を設定します。

> **Note**
>
> インスタンス時間で終了するスケールダウン動作オプションは、Amazon EMR リリース 5.10.0 以降でサポートされなくなりました。次のスケールダウン動作オプションは、リリース 5.1.0 から 5.9.1 の Amazon EMR コンソールにのみ表示されます。

AWS Management Console、、または Amazon EMR API を使用して AWS CLI、クラスターの作成時にスケールダウン動作を設定できます。

Console

新しいコンソールを使用してスケールダウン動作を設定するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で、[クラスター] を選択し、[クラスターの作成] を選択します
3. クラスターのスケーリングとプロビジョニングオプションセクションで、カスタム自動スケーリングの使用を選択します。カスタム自動スケーリングポリシー で、プラスアクションボタンを選択して、ポリシーにスケールを追加します。スケールインポリシーとスケールアウトポリシーの両方を追加することをお勧めします。ポリシーのセットを 1 つだけ追加すると、Amazon EMR は一方向スケーリングのみを実行し、他のアクションを手動で実行する必要があります。

4. クラスターに適用するその他のオプションを選択します。
5. クラスターを起動するには、[クラスターの作成] を選択します。

AWS CLI

を使用してスケールダウン動作を設定するには AWS CLI

- `--scale-down-behavior` オプションを使用して、`TERMINATE_AT_INSTANCE_HOUR` または `TERMINATE_AT_TASK_COMPLETION` のいずれかを指定します。

クラスターを終了する

このセクションでは、クラスターを終了する方法について説明します。削除保護の有効化とクラスターの自動終了については、「クラスターの終了を制御する」を参照してください。STARTING、RUNNING、WAITING のいずれかの状態のクラスターを終了できます。WAITING 状態のクラスターは終了する必要があります。終了しないと、無制限に実行され、アカウントに対して料金が発生します。STARTING 状態から移行できないクラスター、またはステップを完了できないクラスターを終了できます。

終了保護が設定されているクラスターを終了する場合、クラスターを終了する前に終了保護を無効にしておく必要があります。クラスターは、コンソール、、 AWS CLIまたは `TerminateJobFlows` API を使用してプログラムで終了できます。

クラスターの設定によっては、クラスターが完全に終了し、割り当てられたリソース (EC2 インスタンスなど) が解放されるまでに、5～20 分かかる場合があります。

> **Note**
>
> 終了したクラスターは再起動できませんが、終了したクラスターのクローンを作成して、新しいクラスターでその構成を再利用できます。詳細については、「コンソールを使用してクラスターを複製する」を参照してください。

> **Important**
>
> Amazon EMR は Amazon EMR サービスロールと `AWSServiceRoleForEMRCleanup` ロールを使用して、アカウント内で使用しなくなったクラスターリソース (Amazon EC2 インス

タンスなど) をクリーンアップします。ロールポリシーにはリソースを削除または終了するアクションを含める必要があります。そうでない場合、Amazon EMR はこれらのクリーンアップアクションを実行できず、クラスターに残っている未使用のリソースに対して料金が発生する可能性があります。

コンソールを使用してクラスターを終了する

Amazon EMR コンソールを使用して 1 つ以上のクラスターを終了できます。コンソールのクラスターを終了する手順は、終了保護が有効かどうかによって異なります。保護されているクラスターを終了するには、まず終了保護を無効にする必要があります。

New console

新しいコンソールを使用してクラスターを終了するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. [クラスター] を選択し、終了するクラスターを選択します。
3. [アクション] ドロップダウンメニューで [クラスターを終了] を選択し、[クラスターを終了] プロンプトを開きます。
4. プロンプトで [終了] を選択します。クラスター設定によっては、終了に 5 ~ 10 分間かかる場合があります。Amazon EMR クラスターの終了に関する詳細は、「クラスターを終了する」を参照してください。

Old console

古いコンソールを使用して、終了保護が無効な状態でクラスターを終了するには

1. 新しい Amazon EMR コンソールに移動し、サイドナビゲーションから [古いコンソールに切り替え] を選択します。古いコンソールに切り替えたときの動作の詳細については、「Using the old console」を参照してください。
2. 終了するクラスターを選択します。複数のクラスターを同時に選択し、同時に終了することができます。
3. [Terminate] (終了) を選択します。
4. プロンプトが表示されたら、[Terminate (終了)] を選択します。

Amazon EMR はクラスターのインスタンスを削除し、ログデータの保存を停止します。

古いコンソールを使用して、終了保護が有効な状態でクラスターを終了するには

1. 新しい Amazon EMR コンソールに移動し、サイドナビゲーションから [古いコンソールに切り替え] を選択します。古いコンソールに切り替えたときの動作の詳細については、「Using the old console」を参照してください。
2. [Cluster List (クラスターリスト)] ページで終了するクラスターを選択します。複数のクラスターを同時に選択し、同時に終了することができます。
3. [Terminate] (終了) を選択します。
4. プロンプトが表示されたら、[Change] を選択して終了保護を無効にします。複数のクラスターを選択した場合は、[Turn off all] を選択して、すべてのクラスターの終了保護を同時に無効にします。
5. [Terminate clusters (クラスターの終了)] ダイアログで [Termination Protection (削除保護)] の [Off (オフ)] を選択し、チェックマークをクリックして確定します。
6. [Terminate (終了)] をクリックします。

   Amazon EMR はクラスターのインスタンスを削除し、ログデータの保存を停止します。

AWS CLIを使用してクラスターを終了する

を使用して保護されていないクラスターを終了するには AWS CLI

を使用して保護されていないクラスターを終了するには AWS CLI、--cluster-ids パラメータを指定して `terminate-clusters`サブコマンドを使用します。

- 1 つのクラスターを終了する次のコマンドを入力して、*j-3KVXXXXXXX7UG* をクラスター ID に置き換えます。

  ```
  aws emr terminate-clusters --cluster-ids j-3KVXXXXXXX7UG
  ```

  複数のクラスターを終了するには、次のコマンドを入力し、*j-3KVXXXXXXX7UG #* と *j-WJ2XXXXXX8EU* をクラスター ID に置き換えます。

  ```
  aws emr terminate-clusters --cluster-ids j-3KVXXXXXXX7UG j-WJ2XXXXXX8EU
  ```

での Amazon EMR コマンドの使用の詳細については AWS CLI、「」を参照してくださいhttps://docs.aws.amazon.com/cli/latest/reference/emr。

を使用して保護されたクラスターを終了するには AWS CLI

を使用して保護されたクラスターを終了するには AWS CLI、まず `--no-termination-protected`パラメータを指定して `modify-cluster-attributes`サブコマンドを使用して終了保護を無効にします。次に、`terminate-clusters` サブコマンドを `--cluster-ids` パラメータと共に使用してクラスターを終了します。

1. 次のコマンドを入力して終了保護を無効にし、*j-3KVTXXXXXX7UG* をクラスター ID に置き換えます。

   ```
   aws emr modify-cluster-attributes --cluster-id j-3KVTXXXXXX7UG --no-termination-protected
   ```

2. クラスターを終了するには、次のコマンドを入力して、*j-3KVXXXXXXX7UG* をクラスター ID に置き換えます。

   ```
   aws emr terminate-clusters --cluster-ids j-3KVXXXXXXX7UG
   ```

   複数のクラスターを終了するには、次のコマンドを入力し、*j-3KVXXXXXXX7UG #* と *j-WJ2XXXXXX8EU* をクラスター ID に置き換えます。

   ```
   aws emr terminate-clusters --cluster-ids j-3KVXXXXXXX7UG j-WJ2XXXXXX8EU
   ```

   での Amazon EMR コマンドの使用の詳細については AWS CLI、「」を参照してくださいhttps://docs.aws.amazon.com/cli/latest/reference/emr。

API を使用してクラスターを終了する

`TerminateJobFlows` オペレーションが、ステップの処理を終了し、Amazon EC2 から Amazon S3 にログデータをアップロードして (設定されている場合)、Hadoop クラスターを終了します。`KeepJobAliveWhenNoSteps` リクエストで `False` を `RunJobFlows` に設定すると、クラスターも自動的に終了します。

1 つのクラスターまたは複数クラスターのリストをクラスターの ID を指定して終了するには、このアクションを使用します。

に固有の入力パラメータの詳細については、TerminateJobFlows「」を参照してください TerminateJobFlows。リクエストの一般的なパラメータの詳細については、「リクエストの一般的なパラメータ」を参照してください。

コンソールを使用してクラスターを複製する

Amazon EMR コンソールを使用してクラスターを複製できます。これにより、新しいクラスターの基盤として使用する元のクラスターの設定のコピーを作成します。

> **Note**
>
> Amazon EMR コンソールは、再設計され、使いやすくなりました。新しいコンソールでは自動スケーリングを使用するクラスターをクローンできますが、新しいクラスターを作成できるのは、クラスターを手動でスケールする場合やマネージドスケーリングを使用する場合のみです。古いコンソールと新しいコンソールエクスペリエンスの違いの詳細については、「Amazon EMR コンソール」を参照してください。

New console

新しいコンソールを使用してクラスターのクローンを作成するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/emr で Amazon EMR コンソールを開きます。
2. 左側のナビゲーションペインの [EMR on EC2] で、[クラスター] を選択します。
3. クラスターリストからクラスターのクローンを作成するには
   a. 検索オプションやフィルターオプションを使用して、クローンを作成するクラスターをリストビューで探します。
   b. クローンを作成するクラスターの行の左側にあるチェックボックスをオンにします。
   c. リストビューの上部に [クローン] オプションが表示されるようになります。[クローン] を選択して、クローンの作成プロセスを開始します。クラスターにステップが設定されており、他のクラスター設定と一緒にステップもクローン作成する場合は、[ステップを含める] と [次へ] を選択します。

d. クローンされたクラスターからコピーした新しいクラスターの設定を確認します。必要に応じて設定を調整します。新しいクラスターの設定に問題がなければ、[クラスターの作成] を選択して新しいクラスターを起動します。

4. クラスターの詳細ページからクラスターのクローンを作成するには

   a. クローンを作成するクラスターの詳細ページに移動するには、クラスターのリストビューから [クラスター ID] を選択します。

   b. クラスターの詳細ページの上部にある [アクション] メニューから [クラスターを複製] を選択し、クローンの作成プロセスを開始します。クラスターにステップが設定されており、他のクラスター設定と一緒にステップもクローン作成する場合は、[ステップを含める] と [次へ] を選択します。

   c. クローンされたクラスターからコピーした新しいクラスターの設定を確認します。必要に応じて設定を調整します。新しいクラスターの設定に問題がなければ、[クラスターの作成] を選択して新しいクラスターを起動します。

Old console

古いコンソールを使用してクラスターのクローンを作成するには

1. 新しい Amazon EMR コンソールに移動し、サイドナビゲーションから [古いコンソールに切り替え] を選択します。古いコンソールに切り替えたときの動作の詳細については、「Using the old console」を参照してください。
2. [クラスターを作成] を選択します。
3. [Cluster List (クラスターリスト)] ページで、複製するクラスターをクリックします。
4. [Cluster Details (クラスターの詳細)] ページの上部で、[Clone (クローン)] をクリックします。

   クローンとして作成されたクラスターに元のクラスターのステップを含めるには、ダイアログボックスで [Yes (はい)] を選択します。ステップを含めずに元のクラスターの設定のクローンを作成するには、[No (いいえ)] を選択します。

   > **Note**
   >
   > AMI 3.1.1 以降 (Hadoop 2.x) または AMI 2.4.8 以降 (Hadoop 1.x) を使用して作成されたクラスターでは、ステップを含めてクラスターのクローンを作成すると、すべてのシステムステップ (Hive の設定など) が、ユーザーが送信したステップと共に

クローンとして作成されます。ステップの最大数は合計で 1,000 ステップです。コンソールのステップ履歴に表示されなくなった以前のステップは、クローンとして作成することはできません。以前の AMI では、クローンとして作成できるステップは 256 ステップまでです (システムステップを含む) 。詳細については、「クラスターへの作業の送信」を参照してください。

5. [Create Cluster (クラスターの作成)] ページが、元のクラスターの設定のコピーとともに表示されます。設定を確認し、必要な変更を加え、[Create Cluster (クラスターの作成)] をクリックします。

AWS Data Pipelineでクラスターを自動的に繰り返す

AWS Data Pipeline は、データの移動と変換を自動化するサービスです。これを使用して入力データの Amazon S3 への移動をスケジュールし、クラスターを起動してそのデータを処理するようにスケジュールできます。たとえば、トラフィックログを記録するウェブサーバーがあるとします。毎週クラスターを実行してトラフィックデータを分析する場合は、 を使用してそれらのクラスター AWS Data Pipeline をスケジュールできます。 AWS Data Pipeline はデータ駆動型のワークフローであるため、あるタスク (クラスターの起動) を別のタスク (入力データを Amazon S3 に移動) に依存させることができます。また、強力な再試行機能もあります。

の詳細については AWS Data Pipeline、「 AWS Data Pipeline デベロッパーガイド」、特に Amazon EMR に関するチュートリアルを参照してください。

- チュートリアル: Amazon EMR ジョブフローを起動する
- 開始方法: AWS Data Pipeline、Amazon EMR、および Hive を使用してウェブログを処理する
- チュートリアル: を使用した Amazon DynamoDB のインポートとエクスポート AWS Data Pipeline

クラスターのトラブルシューティングを行う

EMR クラスターは、オープンソースソフトウェア、カスタムアプリケーションコード、および で構成される複雑な エコシステムで実行されます AWS のサービス。こうした要素のいずれかで問題が発生すると、クラスターに障害が発生したり、処理の完了までに予想以上に時間がかかったりする可能性があります。クラスターの問題を特定し、修正するには、次のトピックが役立ちます。

トピック

- トラブルシューティングに利用可能なツール
- Amazon EMR とアプリケーションプロセス (デーモン) を表示して再起動する
- Amazon EMR の一般的なエラー
- 失敗したクラスターのトラブルシューティング
- 低速なクラスターのトラブルシューティング
- Lake Formation クラスターのトラブルシューティング

新しい Hadoop アプリケーションを開発するときは、デバッグを有効にして、小規模の代表的なデータのサブセットを処理して、アプリケーションをテストすることをお勧めします。また、アプリケーションを実行して各ステップを個別に step-by-step テストすることもできます。詳細については、「クラスターのログ記録とデバッグを設定する」および「ステップ 5: クラスターのステップバイステップテスト」を参照してください。

トラブルシューティングに利用可能なツール

クラスターのエラーを特定して修正するには、このページで取り上げるツールを利用すると良いでしょう。場合によっては、クラスターの起動時に一部のツールを初期化する必要があります。その他のツールは、デフォルトで、いずれのクラスターにも使用できます。

トピック

- EMR クラスターの詳細を表示する
- EMR クラスターエラーの詳細を表示する
- スクリプトを実行し、Amazon EMR プロセスを設定する
- ログファイルを表示する
- EMR クラスターのパフォーマンスを監視する

EMR クラスターの詳細を表示する

AWS Management Console、、または EMR API を使用して AWS CLI、EMR クラスターとジョブ実行に関する詳細情報を取得できます。 AWS Management Console および の使用の詳細については、 AWS CLI「」を参照してくださいクラスターステータスと詳細の表示。

Amazon EMR コンソールの詳細ペイン

Amazon EMR コンソールの [クラスター] リストに、アカウントと AWS リージョン内の各クラスターのステータスに関する概要情報が表示されます。リストには、過去 2 か月間に起動したアクティブなクラスターと終了させたクラスターがすべて表示されます。[Clusters] リストから、クラスターの [Name] を選択するとクラスターの詳細を表示できます。この情報は、簡単に操作できるよう別々のカテゴリに分類されています。

クラスターの詳細ページで利用できる [アプリケーションユーザーインターフェイス] は、トラブルシューティングに役立ちます。YARN アプリケーションのステータスが表示されるほか、Spark アプリケーションなど一部では、ジョブ、ステージ、エグゼキューターなど、さまざまなメトリクスやファセットの詳細を表示できます。詳細については、「アプリケーションの履歴を表示する」を参照してください。この機能は、Amazon EMR リリース 5.8.0 以降でのみ使用できます。

Amazon EMR コマンドラインインターフェイス

クラスターに関する詳細は、 `--describe`引数 AWS CLI を使用して から見つけることができます。

Amazon EMR API

`DescribeJobFlows` アクションを使用して、API からクラスターに関する情報を検索できます。

EMR クラスターエラーの詳細を表示する

EMR クラスターがエラーで終了すると、`DescribeCluster` と `ListClusters` の各 API からエラーコードとエラーメッセージが返ります。クラスターエラーによっては、`ErrorDetail` データ配列が障害のトラブルシューティングに役立つ場合があります。

`ErrorDetail` データを含むエラーコードのリストについては、情報を含む ErrorDetailエラーコードを参照してください。

> **Note**
>
> 適切な最新情報を得られるよう、エラーメッセージは継続的に改善されています。ErrorMessage のテキストは、変更される可能性があるため、解析はお勧めしません。

スクリプトを実行し、Amazon EMR プロセスを設定する

トラブルシューティングプロセスの一環として、クラスターでのカスタムスクリプトの実行や、クラスタープロセスの表示と設定が有効な場合があります。

アプリケーションプロセスを表示して再起動する

潜在的な問題を診断するために、クラスターで実行中のプロセスを表示すると役立つことがあります。クラスターのマスターノードに接続して、クラスタープロセスを停止および再起動できます。詳細については、「Amazon EMR とアプリケーションプロセス (デーモン) を表示して再起動する」を参照してください。

SSH 接続なしでコマンドとスクリプトを実行する

クラスターでコマンドまたはスクリプトをステップとして実行するために、マスターノードへの SSH 接続を確立せずに、command-runner.jar ツールまたは script-runner.jar ツールを使用できます。詳細については、「Amazon EMR クラスターでコマンドとスクリプトを実行する」を参照してください。

ログファイルを表示する

Amazon EMR と Hadoop は両方とも、クラスターの実行時にログファイルを生成します。クラスターを起動したときの設定に応じて、さまざまなツールからログファイルにアクセスできます。詳細については、「クラスターのログ記録とデバッグを設定する」を参照してください。

マスターノードのログファイル

すべてのクラスターは、マスターノードの /mnt/var/log/ ディレクトリにログファイルを発行します。ログファイルはクラスターが実行されている間のみ利用可能です。

Simple Storage Service (Amazon S3) にアーカイブされたログファイル

クラスターを起動して、Simple Storage Service (Amazon S3) のログパスを指定すると、クラスターはマスターノードの /mnt/var/log/ に保存されているログファイルを Simple Storage Service

(Amazon S3) に 5 分間隔でコピーします。これによって、クラスターが終了された後でも確実にログファイルにアクセスできます。ファイルは 5 分間隔でアーカイブされるため、突然終了したクラスターの最後の数分は利用できない可能性があります。

EMR クラスターのパフォーマンスを監視する

Amazon EMR には、クラスターのパフォーマンスを監視するためのいくつかのツールが用意されています。

Hadoop ウェブインターフェイス

すべてのクラスターは、クラスターに関する情報を含むマスターノードに Web インターフェイスー式を発行します。SSH トンネルを使用してマスターノードの Web ページに接続することによって、これらの Web ページにアクセスできます。詳細については、「Amazon EMR クラスターでホストされているウェブインターフェイスを表示する」を参照してください。

CloudWatch メトリクス

すべてのクラスターは、メトリクスを にレポートします CloudWatch。 CloudWatch は、メトリクスを追跡するウェブサービスであり、これらのメトリクスにアラームを設定するために使用できます。詳細については、「を使用した Amazon EMR メトリクスのモニタリング CloudWatch」を参照してください。

Amazon EMR とアプリケーションプロセス (デーモン) を表示して再起動する

クラスターのトラブルシューティング中に、実行中のプロセスのリストを確認する必要があることがあります。プロセスの停止または再起動が必要な場合もあるでしょう 例えば、設定を変更した後や、ログファイルとエラーメッセージの分析後に特定のプロセスの問題に気付いた後などにプロセスを再起動できます。

クラスターで実行されるプロセスには、Amazon EMR プロセス (インスタンスコントローラーやログプッシャーなど) と、クラスターにインストールされたアプリケーションに関連付けられたプロセス (hadoop-hdfs-namenodeや など) の 2 種類があります hadoop-yarn-resourcemanager。

クラスター上のプロセスを直接操作するには、まずマスターノードに接続する必要があります。詳細については、「クラスターに接続する」を参照してください。

実行中のプロセスの表示

クラスターで実行中のプロセスを表示するために使用する方法は、使用する Amazon EMR のバージョンによって異なります。

EMR 5.30.0 and 6.0.0 and later

Example : 実行中のすべてのプロセスを一覧表示する

次の例では systemctl を使用し、--type を指定して、すべてのプロセスを表示します。

```
systemctl --type=service
```

Example : 特定のプロセスを一覧表示する

次の例では、hadoop を含む名前を持つすべてのプロセスを一覧表示します。

```
systemctl --type=service | grep -i hadoop
```

出力例:

```
 hadoop-hdfs-namenode.service           loaded active running Hadoop namenode
 hadoop-httpfs.service                  loaded active running Hadoop httpfs
 hadoop-kms.service                     loaded active running Hadoop kms
 hadoop-mapreduce-historyserver.service loaded active running Hadoop historyserver
 hadoop-state-pusher.service            loaded active running Daemon process that
 processes and serves EMR metrics data.
 hadoop-yarn-proxyserver.service        loaded active running Hadoop proxyserver
 hadoop-yarn-resourcemanager.service    loaded active running Hadoop resourcemanager
 hadoop-yarn-timelineserver.service     loaded active running Hadoop timelineserver
```

Example : 特定のプロセスの詳細なステータスレポートを表示する

次の例では、hadoop-hdfs-namenode サービスの詳細なステータスレポートを表示します。

```
sudo systemctl status hadoop-hdfs-namenode
```

出力例:

```
hadoop-hdfs-namenode.service - Hadoop namenode
   Loaded: loaded (/etc/systemd/system/hadoop-hdfs-namenode.service; enabled; vendor
 preset: disabled)
```

```
   Active: active (running) since Wed 2021-08-18 21:01:46 UTC; 26min ago
 Main PID: 9733 (java)
    Tasks: 0
   Memory: 1.1M
   CGroup: /system.slice/hadoop-hdfs-namenode.service
           # 9733 /etc/alternatives/jre/bin/java -Dproc_namenode -Xmx1843m -server -
XX:OnOutOfMemoryError=kill -9 %p ...

Aug 18 21:01:37 ip-172-31-20-123 systemd[1]: Starting Hadoop namenode...
Aug 18 21:01:37 ip-172-31-20-123 su[9715]: (to hdfs) root on none
Aug 18 21:01:37 ip-172-31-20-123 hadoop-hdfs-namenode[9683]: starting namenode,
 logging to /var/log/hadoop-hdfs/ha...out
Aug 18 21:01:46 ip-172-31-20-123 hadoop-hdfs-namenode[9683]: Started Hadoop
 namenode:[  OK  ]
Aug 18 21:01:46 ip-172-31-20-123 systemd[1]: Started Hadoop namenode.
Hint: Some lines were ellipsized, use -l to show in full.
```

EMR 4.x - 5.29.0

Example : 実行中のすべてのプロセスを一覧表示する

次の例では、実行中のすべてのプロセスを一覧表示します。

```
initctl list
```

EMR 2.x - 3.x

Example : 実行中のすべてのプロセスを一覧表示する

次の例では、実行中のすべてのプロセスを一覧表示します。

```
ls /etc/init.d/
```

プロセスの停止と再起動

実行中のプロセスを確認してから、必要に応じてそれらを停止して再起動できます。

EMR 5.30.0 and 6.0.0 and later

Example : プロセスを停止する

次の例では、hadoop-hdfs-namenode プロセスを停止します。

```
sudo systemctl stop hadoop-hdfs-namenode
```

status に対してクエリを実行して、プロセスが停止したことを確認できます。

```
sudo systemctl status hadoop-hdfs-namenode
```

出力例:

```
hadoop-hdfs-namenode.service - Hadoop namenode
  Loaded: loaded (/etc/systemd/system/hadoop-hdfs-namenode.service; enabled; vendor
 preset: disabled)
  Active: failed (Result: exit-code) since Wed 2021-08-18 21:37:50 UTC; 8s ago
Main PID: 9733 (code=exited, status=143)
```

Example : プロセスを開始する

次の例では、hadoop-hdfs-namenode プロセスを開始します。

```
sudo systemctl start hadoop-hdfs-namenode
```

ステータスのクエリを実行して、プロセスが実行中になったことを確認できます。

```
sudo systemctl status hadoop-hdfs-namenode
```

出力例:

```
hadoop-hdfs-namenode.service - Hadoop namenode
   Loaded: loaded (/etc/systemd/system/hadoop-hdfs-namenode.service; enabled; vendor
 preset: disabled)
   Active: active (running) since Wed 2021-08-18 21:38:24 UTC; 2s ago
  Process: 13748 ExecStart=/etc/init.d/hadoop-hdfs-namenode start (code=exited,
 status=0/SUCCESS)
 Main PID: 13800 (java)
    Tasks: 0
   Memory: 1.1M
   CGroup: /system.slice/hadoop-hdfs-namenode.service
           # 13800 /etc/alternatives/jre/bin/java -Dproc_namenode -Xmx1843m -server
 -XX:OnOutOfMemoryError=kill -9 %p...
```

EMR 4.x - 5.29.0

Example : 実行中のプロセスを停止する

次の例では、hadoop-hdfs-namenode サービスを停止します。

```
sudo stop hadoop-hdfs-namenode
```

Example : 停止したプロセスを再起動する

次の例では、hadoop-hdfs-namenode サービスを再起動します。restart ではなく start コマンドを使用する必要があります。

```
sudo start hadoop-hdfs-namenode
```

Example : プロセスのステータスを確認する

次の例では、hadoop-hdfs-namenode のステータスを取得します。status コマンドを使用して、プロセスが停止または開始したことを確認できます。

```
sudo status hadoop-hdfs-namenode
```

EMR 2.x - 3.x

Example : アプリケーションプロセスを停止する

次の例では、クラスターにインストールされている Amazon EMR のバージョンに関連付けられている hadoop-hdfs-namenode サービスを停止します。

```
sudo /etc/init.d/hadoop-hdfs-namenode stop
```

Example : アプリケーションプロセスを再起動する

次のコマンド例では、hadoop-hdfs-namenode プロセスを再起動します。

```
sudo /etc/init.d/hadoop-hdfs-namenode start
```

Example : Amazon EMR プロセスを停止する

次の例では、クラスター上の Amazon EMR のバージョンに関連付けられていないインスタンスコントローラーなどのプロセスを停止します。

```
sudo /sbin/stop instance-controller
```

Example : Amazon EMR プロセスを再起動する

次の例では、クラスター上の Amazon EMR のバージョンに関連付けられていないインスタンスコントローラーなどのプロセスを再起動します。

```
sudo /sbin/start instance-controller
```

> **Note**
>
> `/sbin/start`, `stop` および `restart` コマンドは `/sbin/intictl` へのシンボリックリンクです。`initctl` の詳細については、コマンドプロンプトで `man initctl` と入力して、initctl マニュアルページを参照してください。

Amazon EMR の一般的なエラー

クラスターでは、障害や、データ処理の遅延が発生する場合があります。このセクションでは、一般的なクラスターの問題とその対処法を示します。

トピック

- 情報を含む ErrorDetailエラーコード
- リソースエラー
- 入力エラーと出力エラー
- 権限エラー
- Hive クラスターのエラー
- VPC エラー
- ストリーミングクラスターのエラー
- カスタム JAR クラスターのエラー
- AWS GovCloud （米国西部) エラー
- 見つからないクラスターを検索する

情報を含む ErrorDetailエラーコード

EMR クラスターがエラーで終了すると、DescribeCluster と ListClusters の各 API からエラーコードとエラーメッセージが返ります。クラスターエラーによっては、ErrorDetail データ配列が障害のトラブルシューティングに役立つ場合があります。

ErrorDetail 配列を持つエラーでは、次の情報を確認できます。

ErrorCode

プログラムによるアクセスに使用できる一意のエラーコード。

ErrorData

キーと値のペアを示す識別子のリスト。識別子は、プログラミングや手動検索に使用できる。エラーコード内の ErrorData 値の詳細については、エラーコードのトラブルシューティングページを参照してください。

ErrorMessage

エラーの説明。Amazon EMR ドキュメントの詳細情報へのリンクが記載されている。

> **Note**
>
> ErrorMessage のテキストは、変更される可能性があるため、解析はお勧めしません。

カテゴリ別のエラーコード

- ブートストラップ障害のエラーコード
- 内部エラーコード
- 検証失敗のエラーコード

ブートストラップ障害のエラーコード

次のセクションでは、ブートストラップ障害のエラーコードに対するトラブルシューティングについて説明します。

トピック

- BOOTSTRAP_FAILURE_PRIMARY_WITH_NON_ZERO_CODE

- BOOTSTRAP_FAILURE_BA_DOWNLOAD_FAILED_PRIMARY
- BOOTSTRAP_FAILURE_FILE_NOT_FOUND_PRIMARY

BOOTSTRAP_FAILURE_PRIMARY_WITH_NON_ZERO_CODE

概要

クラスターが BOOTSTRAP_FAILURE_PRIMARY_WITH_NON_ZERO_CODE エラーで終了した場合、プライマリインスタンスでブートストラップアクションが失敗したことを示しています。ブートストラップアクションの詳細については、追加のソフトウェアをインストールするためのブートストラップアクションの作成 を参照してください。

解決方法

このエラーを解決するには、返った API エラーの内容を確認して、ブートストラップアクションスクリプトを変更します。その後、更新したブートストラップアクションを使用して、クラスターを新規作成します。

障害が発生した EMR クラスターのトラブルシューティングを行うには、DescribeCluster と ListClusters の各 API から返った ErrorDetail の情報を参照してください。詳細については、「情報を含む ErrorDetailエラーコード」を参照してください。返った ErrorDetail 内の ErrorData 配列によって、次の情報を確認できます。

primary-instance-id

ブートストラップアクションが失敗したプライマリインスタンスの ID。

bootstrap-action

失敗したブートストラップアクションの序数。bootstrap-action 値 1 を持つスクリプトによって、そのインスタンスで最初のアクションが実行されます。

return-code

失敗したブートストラップアクションのリターンコード。

amazon-s3-path

ブートストラップアクションが失敗した、Amazon S3 の口ケーション。

public-doc

エラーコードドキュメントの公開 URL。

完了すべきステップ

ブートストラップアクションエラーの根本原因を特定して修正するには、次のステップを実行し、その後、新規クラスターを起動します。

1. Amazon S3 のブートストラップアクションログファイルを確認して、障害の根本原因を特定します。Amazon EMR ログの表示方法については、ログファイルを表示する を参照してください。
2. インスタンスの作成時にクラスターログを有効にした場合は、stdout ログで詳細を確認してください。ブートストラップアクションの stdout ログは、次に示す Amazon S3 のロケーションにあります。

   ```
   s3://EXAMPLE-BUCKET/logs/Your_Cluster_Id/node/Primary_Instance_Id/bootstrap-actions/Failed_Bootstrap_Action_Number/stdout.gz
   ```

   クラスターログの詳細については、「クラスターのログ記録とデバッグを設定する」を参照してください。
3. ブートストラップアクションが失敗したかどうかを判断するには、stdout ログ内の例外と return-code 内の値を確認します。ErrorData
4. 前のステップで得た情報に基づいてブートストラップアクションを修正し、例外の回避や、例外が発生した際の適切な処理を行えるようにします。
5. 更新したブートストラップアクションを使用して新規クラスターを起動します。

BOOTSTRAP_FAILURE_BA_DOWNLOAD_FAILED_PRIMARY

概要

クラスターが BOOTSTRAP_FAILURE_BA_DOWNLOAD_FAILED_PRIMARY エラーで終了した場合、プライマリインスタンスが指定された Amazon S3 のロケーションからブートストラップアクションスクリプトをダウンロードできなかったことを示しています。この場合、次の原因が考えられます。

- ブートストラップアクションスクリプトファイルが、指定した Amazon S3 のロケーションにない。
- クラスターの Amazon EC2 インスタンスに設定されたサービスロール (Amazon EMR の EC2 インスタンスプロファイルとも呼ばれます) に、ブートストラップアクションスクリプトのある Amazon S3 バケットへのアクセス権限がない。サービスロールの詳細については、「クラスター EC2 インスタンスのサービスロール (EC2 インスタンスプロファイル)」を参照してください。

ブートストラップアクションの詳細については、追加のソフトウェアをインストールするためのブートストラップアクションの作成 を参照してください。

解決方法

このエラーを解決するには、プライマリインスタンスがブートストラップアクションスクリプトに適切にアクセスできるようにします。

障害が発生した EMR クラスターのトラブルシューティングを行うには、DescribeCluster と ListClusters の各 API から返った ErrorDetail の情報を参照してください。詳細については、「情報を含む ErrorDetailエラーコード」を参照してください。返った ErrorDetail 内の ErrorData 配列によって、次の情報を確認できます。

primary-instance-id

ブートストラップアクションが失敗したプライマリインスタンスの ID。

bootstrap-action

失敗したブートストラップアクションの序数。bootstrap-action 値 1 を持つスクリプトによって、そのインスタンスで最初のアクションが実行されます。

amazon-s3-path

ブートストラップアクションが失敗した、Amazon S3 のロケーション。

public-doc

エラーコードドキュメントの公開 URL。

完了すべきステップ

ブートストラップアクションエラーの根本原因を特定して修正するには、次のステップを実行し、その後、新規クラスターを起動します。

トラブルシューティングのステップ

1. ErrorData 配列の amazon-s3-path 値を使用して、関連するブートストラップアクションスクリプトを Amazon S3 内で検索します。
2. インスタンスの作成時にクラスターログを有効にした場合は、stdout ログで詳細を確認してください。ブートストラップアクションの stdout ログは、次に示す Amazon S3 のロケーションにあります。

```
s3://EXAMPLE-BUCKET/logs/Your_Cluster_Id/node/Primary_Instance_Id/bootstrap-
actions/Failed_Bootstrap_Action_Number/stdout.gz
```

クラスターログの詳細については、「クラスターのログ記録とデバッグを設定する」を参照してください。

3. ブートストラップアクションが失敗したかどうかを判断するには、stdout ログ内の例外と return-code 内の値を確認します。ErrorData
4. 前のステップで得た情報に基づいてブートストラップアクションを修正し、例外の回避や、例外が発生した際の適切な処理を行えるようにします。
5. 更新したブートストラップアクションを使用して新規クラスターを起動します。

BOOTSTRAP_FAILURE_FILE_NOT_FOUND_PRIMARY

概要

BOOTSTRAP_FAILURE_FILE_NOT_FOUND_PRIMARY エラーは、プライマリインスタンスが、指定された Amazon S3 バケットから自分でダウンロードしたブートストラップアクションスクリプトを見つけられないことを示しています。

解決方法

このエラーを解決するには、プライマリインスタンスがブートストラップアクションスクリプトに適切にアクセスできるようにします。

障害が発生した EMR クラスターのトラブルシューティングを行うには、DescribeCluster と ListClusters の各 API から返った ErrorDetail の情報を参照してください。詳細については、「情報を含む ErrorDetailエラーコード」を参照してください。返った ErrorDetail 内の ErrorData 配列によって、次の情報を確認できます。

primary-instance-id

ブートストラップアクションが失敗したプライマリインスタンスの ID。

bootstrap-action

失敗したブートストラップアクションの序数。bootstrap-action 値 1 を持つスクリプトによって、そのインスタンスで最初のアクションが実行されます。

amazon-s3-path

ブートストラップアクションが失敗した、Amazon S3 のロケーション。

public-doc

エラーコードドキュメントの公開 URL。

完了すべきステップ

ブートストラップアクションエラーの根本原因を特定して修正するには、次のステップを実行し、その後、新規クラスターを起動します。

1. 関連するブートストラップアクションスクリプトを Amazon S3 内で検索するには、`ErrorData` 配列の `amazon-s3-path` 値を使用します。
2. Amazon S3 のブートストラップアクションログファイルを確認して、障害の根本原因を特定します。Amazon EMR ログの表示方法については、ログファイルを表示する を参照してください。

   > **Note**
   >
   > クラスターのログを有効にしていない場合は、同じ設定とブートストラップアクションを使用して、クラスターを新規作成する必要があります。クラスターログが有効になっていることを確認するには、「クラスターのログ記録とデバッグを設定する」を参照してください。

3. ブートストラップアクションの `stdout` ログを確認して、プライマリインスタンスの `/emr/instance-controller/lib/bootstrap-actions` フォルダにあるファイルを削除するカスタムプロセスがないことを確認します。ブートストラップアクションの `stdout` ログは、次に示す Amazon S3 のロケーションにあります。

   ```
   s3://EXAMPLE-BUCKET/logs/Your_Cluster_Id/node/Primary_Instance_Id/bootstrap-actions/Failed_Bootstrap_Action_Number/stdout.gz
   ```

4. 更新したブートストラップアクションを使用して新規クラスターを起動します。

内部エラーコード

次のセクションでは、内部エラーのコードに対するトラブルシューティングについて説明します。

トピック

INTERNAL_ERROR_EC2_INSUFFICIENT_CAPACITY_AZ

概要

クラスターが INTERNAL_ERROR_EC2_INSUFFICIENT_CAPACITY_AZ エラーで終了した場合、選択されたアベイラビリティーゾーンに Amazon EC2 インスタンスタイプのリクエストを満たすだけのキャパシティがなかったことを示しています。クラスターに選択したサブネットによって、アベイラビリティーゾーンが決まります。Amazon EMR のサブネットの詳細については、「ネットワークを設定する」を参照してください。

解決方法

このエラーを解決するには、インスタンスタイプの設定を変更し、更新済みのリクエストによって、クラスターを新規作成します。

障害が発生した EMR クラスターのトラブルシューティングを行うには、DescribeCluster と ListClusters の各 API から返った ErrorDetail の情報を参照してください。詳細については、「情報を含む ErrorDetailエラーコード」を参照してください。返った ErrorDetail 内の ErrorData 配列によって、次の情報を確認できます。

instance-type

容量が不足しているインスタンスタイプ。

availability-zone

サブネットが関連付けられたアベイラビリティーゾーン。

public-doc

エラーコードドキュメントの公開 URL。

完了すべきステップ

クラスター設定エラーの根本原因を特定して修正するには、次のステップを実行します。

- クラスター設定のベストプラクティスを確認します。「Amazon EMR 管理ガイド」の「クラスター設定のベストプラクティス」を参照してください。
- 起動に関する問題をトラブルシューティングし、設定を確認します。「Amazon EC2 ユーザーガイド」の「インスタンスの起動に関する問題のトラブルシューティングAmazon EC2」を参照してください。
- 更新したクラスター設定を使用して、新規クラスターを起動します。

INTERNAL_ERROR_SPOT_PRICE_INCREASE_PRIMARY

概要

クラスターが INTERNAL_ERROR_SPOT_PRICE_INCREASE_PRIMARY エラーで終了した場合、最大スポット料金以下でインスタンスを利用できず、Amazon EMR がプライマリノードのスポットインスタンス要求を処理できなかったことを示しています。詳細については、「Amazon EC2 ユーザーガイド」の「スポットインスタンス」を参照してください。

解決方法

このエラーを解決するには、クラスターのインスタンスタイプを対象料金の範囲内で指定するか、そのインスタンスタイプの料金の上限を引き上げてください。

障害が発生した EMR クラスターのトラブルシューティングを行うには、`DescribeCluster` と `ListClusters` の各 API から返った `ErrorDetail` の情報を参照してください。詳細については、「情報を含む ErrorDetailエラーコード」を参照してください。返った `ErrorDetail` 内の `ErrorData` 配列によって、次の情報を確認できます。

`primary-instance-id`

障害が発生したクラスターのプライマリインスタンス ID。

`instance-type`

容量が不足しているインスタンスタイプ。

`availability-zone`

サブネットが存在するアベイラビリティーゾーン。

`public-doc`

エラーコードドキュメントの公開 URL。

完了すべきステップ

次のステップを実行してクラスター設定戦略のトラブルシューティングを行い、新規クラスターを起動してください。

1. Amazon EC2 スポットインスタンスのベストプラクティスと、クラスター設定戦略の両方を確認します。詳細については、「Amazon EC2 ユーザーガイド」の「EC2 スポットのベストプラクティス」および「」を参照してくださいクラスター設定のベストプラクティス。 Amazon EC2
2. インスタンスタイプ設定またはアベイラビリティーゾーンを変更し、更新済みのリクエストでクラスターを新規作成します。
3. 問題が解決しない場合は、オンデマンドキャパシティをプライマリインスタンスに使用してください。

INTERNAL_ERROR_SPOT_NO_CAPACITY_PRIMARY

概要

クラスターが INTERNAL_ERROR_SPOT_NO_CAPACITY_PRIMARY エラーで終了した場合、プライマリノードのスポットインスタンスリクエストを満たすだけのキャパシティがなかったことを示しています。詳細については、「Amazon EC2 ユーザーガイド」の「スポットインスタンス」を参照してください。

解決方法

このエラーを解決するには、クラスターのインスタンスタイプを対象料金の範囲内で指定するか、そのインスタンスタイプの料金の上限を引き上げてください。

障害が発生した EMR クラスターのトラブルシューティングを行うには、`DescribeCluster` と `ListClusters` の各 API から返った `ErrorDetail` の情報を参照してください。詳細については、「情報を含む ErrorDetailエラーコード」を参照してください。返った `ErrorDetail` 内の `ErrorData` 配列によって、次の情報を確認できます。

`primary-instance-id`

障害が発生したクラスターのプライマリインスタンス ID。

`instance-type`

容量が不足しているインスタンスタイプ。

availability-zone

サブネットが関連付けられたアベイラビリティーゾーン。

public-doc

エラーコードドキュメントの公開 URL。

完了すべきステップ

次のステップを実行してクラスター設定戦略のトラブルシューティングを行い、新規クラスターを起動してください。

1. Amazon EC2 スポットインスタンスのベストプラクティスと、クラスター設定戦略の両方を確認します。詳細については、「Amazon EC2 ユーザーガイド」の「EC2 スポットのベストプラクティス」および「」を参照してくださいクラスター設定のベストプラクティス。 Amazon EC2
2. インスタンスタイプ設定を変更し、更新済みのリクエストでクラスターを新規作成します。
3. 問題が解決しない場合は、オンデマンドキャパシティをプライマリインスタンスに使用してください。

検証失敗のエラーコード

次のセクションでは、検証失敗のエラーコードに対するトラブルシューティングについて説明します。

トピック

- VALIDATION_ERROR_SUBNET_NOT_FROM_ONE_VPC
- VALIDATION_ERROR_SECURITY_GROUP_NOT_FROM_ONE_VPC
- VALIDATION_ERROR_INVALID_SSH_KEY_NAME
- VALIDATION_ERROR_INSTANCE_TYPE_NOT_SUPPORTED

VALIDATION_ERROR_SUBNET_NOT_FROM_ONE_VPC

概要

クラスターが `VALIDATION_ERROR_SUBNET_NOT_FROM_ONE_VPC` エラーで終了した場合、クラスターのサブネットと、クラスターの参照先サブネットが異なる Virtual Private Cloud (VPC) に属して

いることを示しています。Amazon EMR でクラスターを起動するには、VPC のサブネット全体でインスタンスフリート設定を使用します。インスタンスフリートの詳細については、「Amazon EMR 管理ガイド」の「インスタンスフリートを設定する」を参照してください。

解決方法

このエラーを解決するには、対象クラスターのサブネットが属する VPC のサブネットを使用します。

障害が発生した EMR クラスターのトラブルシューティングを行うには、`DescribeCluster` と `ListClusters` の各 API から返った `ErrorDetail` の情報を参照してください。詳細については、「情報を含む ErrorDetailエラーコード」を参照してください。返った `ErrorDetail` 内の `ErrorData` 配列によって、次の情報を確認できます。

`vpc`

各サブネット: VPC ペア、サブネットが属する VPC の ID。

`subnet`

各サブネット: VPC ペア、サブネット ID。

`public-doc`

エラーコードドキュメントの公開 URL。

完了すべきステップ

エラーを特定し、修正するには、次のステップを実行します。

1. `ErrorData` 配列にリストされているサブネット ID を確認し、それらが EMR クラスターを起動する VPC に属していることを確認します。
2. サブネットの設定を変更します。次のいずれかの方法を使用すると、VPC 内の利用可能なパブリックサブネットとプライベートサブネットをすべて検索できます。

   - Amazon VPC コンソールに移動します。サブネット を選択し、クラスターの 内に存在するすべてのサブネットを一覧表示 AWS リージョン します。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みを検索するには、[パブリック IPv4 アドレスの自動割り当て] フィルターを適用します。クラスターが使用する VPC 内のサブネットを検索して選択するには、[VPC でフィルタリング] オプションを使用します。サブネットの作成方法の詳細については、「Amazon Virtual Private Cloud ユーザーガイド」の「サブネットの作成」を参照してください。

- を使用して AWS CLI 、クラスターが使用する VPC で使用可能なすべてのパブリックサブネットとプライベートサブネットを検索します。詳細については、describe-subnets API を参照してください。VPC にサブネットを新規作成する方法については、create-subnet API を参照してください。

3. クラスターと同じ VPC のサブネットを使用して新規クラスターを起動します。

VALIDATION_ERROR_SECURITY_GROUP_NOT_FROM_ONE_VPC

概要

クラスターが `VALIDATION_ERROR_SECURITY_GROUP_NOT_FROM_ONE_VPC` エラーで終了した場合、クラスターのセキュリティグループと、クラスターに割り当てたセキュリティグループが異なる Virtual Private Cloud (VPC) に属していることを示しています。セキュリティグループの詳細については、「Amazon EMR マネージドセキュリティグループと追加セキュリティグループを指定する」と「セキュリティグループを使用してネットワークトラフィックを制御する」を参照してください。

解決方法

このエラーを解決するには、対象クラスターのセキュリティグループが属する VPC のセキュリティグループを使用します。

障害が発生した EMR クラスターのトラブルシューティングを行うには、`DescribeCluster` と `ListClusters` の各 API から返った `ErrorDetail` の情報を参照してください。詳細については、「情報を含む ErrorDetailエラーコード」を参照してください。返った `ErrorDetail` 内の `ErrorData` 配列によって、次の情報を確認できます。

`vpc`

セキュリティグループ: VPC ペア、セキュリティグループが属する VPC の ID。

`security-group`

セキュリティグループ: VPC ペア、セキュリティグループ ID。

`public-doc`

エラーコードドキュメントの公開 URL。

完了すべきステップ

エラーを特定し、修正するには、次のステップを実行します。

1. ErrorData 配列にリストされているセキュリティグループ ID を確認し、それらが EMR クラスターを起動する VPC に属していることを確認します。
2. Amazon VPC コンソールに移動します。[セキュリティグループ] を選択すると、選択したリージョン内のセキュリティグループがすべて一覧表示されます。クラスターと同じ VPC のセキュリティグループを検索し、セキュリティグループの設定を変更します。
3. クラスターと同じ VPC のセキュリティグループを使用して新規クラスターを起動します。

VALIDATION_ERROR_INVALID_SSH_KEY_NAME

概要

クラスターが VALIDATION_ERROR_INVALID_SSH_KEY_NAME エラーで終了した場合、プライマリインスタンスへの SSH 接続時に、有効でない Amazon EC2 キーペアが使用されたことを示しています。キーペア名が正しくないか、リクエストされた にキーペアが存在しない可能性がありますAWS リージョン。キーペアの詳細については、Amazon EC2 ユーザーガイド」の「Amazon EC2 キーペアと Linux インスタンスAmazon EC2」を参照してください。

解決方法

このエラーを解決するには、有効な SSH キーペア名を使用してクラスターを新規作成します。

障害が発生した EMR クラスターのトラブルシューティングを行うには、DescribeCluster と ListClusters の各 API から返った ErrorDetail の情報を参照してください。詳細については、「情報を含む ErrorDetailエラーコード」を参照してください。返った ErrorDetail 内の ErrorData 配列によって、次の情報を確認できます。

ssh-key

クラスターを作成したときに指定した SSH キーペア名。

public-doc

エラーコードドキュメントの公開 URL。

完了すべきステップ

エラーを特定し、修正するには、次のステップを実行します。

1. *keypair*.pem ファイルを確認し、Amazon EMR コンソールに表示される SSH キーの名前と一致することを確認します。

2. Amazon EC2 コンソールに移動します。使用した SSH キー名が、クラスター AWS リージョンが使用する で使用可能であることを確認します。アカウント ID の AWS リージョン 横の は、の上部にあります AWS Management Console。
3. 有効な SSH キー名を使用して新規クラスターを起動します。

VALIDATION_ERROR_INSTANCE_TYPE_NOT_SUPPORTED

概要

クラスターが `VALIDATION_ERROR_INSTANCE_TYPE_NOT_SUPPORTED` エラーで終了した場合、クラスターの AWS リージョン とアベイラビリティーゾーンが、指定したインスタンス (1 つ以上のインスタンスグループに属している) のタイプをサポートしていないことを示しています。リージョン内のアベイラビリティーゾーンによっては、インスタンスタイプが Amazon EMR のサポート対象となる場合とならない場合があります。リージョン内のアベイラビリティーゾーンは、クラスター用に選択したサブネットによって決定されます。Amazon EMR がサポートするインスタンスタイプとリージョンのリストについては、「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。

解決方法

このエラーを解決するには、クラスターをリクエストするリージョンとアベイラビリティーゾーンで Amazon EMR がサポートする、クラスターのインスタンスタイプを指定します。

障害が発生した EMR クラスターのトラブルシューティングを行うには、`DescribeCluster` と `ListClusters` の各 API から返った `ErrorDetail` の情報を参照してください。詳細については、「情報を含む ErrorDetailエラーコード」を参照してください。返った `ErrorDetail` 内の `ErrorData` 配列によって、次の情報を確認できます。

`instance-types`

サポートされていないインスタンスタイプのリスト。

`availability-zones`

サブネットが関連付けられたアベイラビリティーゾーンのリスト。

`public-doc`

エラーコードドキュメントの公開 URL。

完了すべきステップ

エラーを特定し、修正するには、次のステップを実行します。

1. を使用して AWS CLI 、アベイラビリティーゾーンで使用可能なインスタンスタイプを取得します。これを行うには、 `ec2 describe-instance-type-offerings` コマンドを使用して、使用可能なインスタンスタイプを場所 (AWS リージョン またはアベイラビリティーゾーン) でフィルタリングできます。例えば、次のコマンドを使用すると、指定した AZ (*us-east-2a*) で提供されているインスタンスタイプが返ります。

   ```
   aws ec2 describe-instance-type-offerings --location-type "availability-
   zone" --filters Name=location,Values=us-east-2a --region us-east-2 --query
    "InstanceTypeOfferings[*].[InstanceType]" --output text | sort
   ```

   利用可能なインスタンスタイプを確認にする方法については、「Amazon EC2 インスタンスタイプの検索」を参照してください。

2. 対象のクラスターと同じリージョンとアベイラビリティーゾーンで利用可能なインスタンスタイプを確認したら、次の解決策のいずれかを選択して実行します。

   a. クラスターを新規作成します。次に、選択したインスタンスタイプが利用可能で Amazon EMR によってサポートされているアベイラビリティーゾーンに属するクラスターのサブネットを選択します。

   b. 障害が発生したクラスターが属していたリージョンと Amazon EC2 サブネットに新しいクラスターを作成します。ただし、使用するインスタンスタイプは、Amazon EMR がその口ケーションでサポートしているタイプを選択します。

Amazon EMR がサポートするインスタンスタイプとリージョンのリストについては、「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。インスタンスタイプの機能を比較するには、「Amazon EC2 インスタンスタイプ」を参照してください。

リソースエラー

次は、クラスターのリソースの制約により引き起こされる一般的なエラーです。

トピック

- NO_SLAVE_LEFT でクラスターが終了し、FAILED_BY_MASTER でコアノードが終了する
- Cannot replicate block, only managed to replicate to zero nodes. (ブロックをレプリケートできません。ゼロノードにのみレプリケートできます。)
- EC2 QUOTA EXCEEDED
- Too many fetch-failures (フェッチの失敗が多すぎる)

- File could only be replicated to 0 nodes instead of 1 (ファイルは 1 ノードではなく 0 ノードにのみレプリケートできる)
- 拒否リストに登録されたノード
- スロットリングエラー
- サポートされていないインスタンスタイプ
- EC2 が容量不足
- HDFS レプリケーション係数エラー
- HDFS スペース不足エラー

NO_SLAVE_LEFT でクラスターが終了し、FAILED_BY_MASTER でコアノードが終了する

このエラーは通常、削除保護が無効になっており、すべてのコアノードが `yarn-site.xml` ファイルに対応する `yarn-site` 設定分類の最大使用率しきい値で指定したディスクストレージ容量を超過したことが原因で発生します。この値のデフォルト値は 90% です。コアノードのディスク使用率が使用率のしきい値を超えると、YARN NodeManager ヘルスサービスはノードを として報告しますUNHEALTHY。この状態になっていると、Amazon EMR はノードを拒否リストに追加し、YARN コンテナを割り当てません。ノードの異常が 45 分間続いた場合、Amazon EMR は関連付けられている Amazon EC2 インスタンスを終了するために `FAILED_BY_MASTER` のマークを付けます。コアノードに関連付けられているすべての Amazon EC2 インスタンスに終了のマークが付けられると、ジョブを実行するリソースがなくなるため、クラスターはステータス `NO_SLAVE_LEFT` で終了します。

1 つのコアノードのディスク使用率がしきい値を超えると、連鎖反応が起きる可能性があります。HDFS が原因で 1 つのノードのディスク使用率がしきい値を超えた場合、他のノードのディスク使用率もしきい値に近づいていると考えられます。最初のノードのディスク使用率がしきい値を超えると、Amazon EMR はそのノードを拒否リストに追加します。これにより、拒否リストに追加されたノードで失った HDFS データを残りのノードがそれぞれの間でレプリケートし始めるため、ノードのディスク使用率が上昇します。その後、各ノードも同じように `UNHEALTHY` の状態になり、最終的にはクラスターが終了します。

ベストプラクティスとレコメンデーション

クラスターハードウェアに十分なストレージを設定する

クラスターを作成するときに、十分な数のコアノードがあること、および各ノードに HDFS 用の十分なインスタンスストアと EBS ストレージボリュームがあることを確認します。詳細については、

「クラスターの必要な HDFS 容量の計算」を参照してください。手動で、または Auto Scaling を使用して既存のインスタンスグループにコアインスタンスを追加することもできます。新しいインスタンスのストレージ設定は、インスタンスグループ内の他のインスタンスと同じになります。詳細については、「クラスターのスケーリングを使用する」を参照してください。

終了保護を有効化する

削除保護を有効にします。このようにすると、コアノードが拒否リストに追加された場合に、SSH を使用して関連付けられている Amazon EC2 インスタンスに接続し、トラブルシューティングを行ってデータを復旧できます。終了保護を有効にすると、Amazon EMR が Amazon EC2 インスタンスを新しいインスタンスに置き換えなくなる点に注意してください。詳細については、「終了保護の使用」を参照してください。

MR UnhealthyNodes CloudWatch メトリクスのアラームを作成する

このメトリクスは、ステータスが UNHEALTHY のノードの数を報告します。これは、YARN メトリクス `mapred.resourcemanager.NoOfUnhealthyNodes` と同等です。このアラームの通知を設定すれば、45 分のタイムアウトに達する前に異常が発生したノードに関する警告を受け取ることができます。詳細については、「を使用した Amazon EMR メトリクスのモニタリング CloudWatch」を参照してください。

yarn-site を使用して設定を微調整する

以下の設定は、アプリケーションの要件に合わせて調整できます。たとえば、`yarn.nodemanager.disk-health-checker.max-disk-utilization-per-disk-percentage` の値を増やすことにより、ノードが UNHEALTHY と報告するディスク使用率のしきい値を上げることができます。

これらの値は、`yarn-site` 設定分類を使用してクラスターを作成するときに設定できます。詳細については、「Amazon EMR リリースガイドの」「アプリケーションの設定」を参照してください。SSH を使用してコアノードに関連付けられている Amazon EC2 インスタンスに接続し、テキストエディターを使用して `/etc/hadoop/conf.empty/yarn-site.xml` に値を追加することもできます。変更を加えたら、次 hadoop-yarn-nodemanager に示すように再起動する必要があります。

> **Important**
>
> NodeManager サービスを再起動すると、クラスターの作成時に `yarn-site`が設定分類`true`を使用して に設定されない限り`yarn.nodemanager.recovery.enabled`、アクティブな YARN コンテナは強制終了されます。そのた

め、yarn.nodemanager.recovery.dir プロパティを使用して、コンテナの状態を保存するディレクトリも指定する必要があります。

```
sudo /sbin/stop hadoop-yarn-nodemanager
sudo /sbin/start hadoop-yarn-nodemanager
```

最新の yarn-site プロパティとデフォルト値の詳細については、Apache Hadoop ドキュメントの「YARN のデフォルト設定」を参照してください。

プロパティ	デフォルト値	説明
yarn.nodemanager.disk-health-checker.interval-ms	120000	ディスクのヘルスチェッカーを実行する頻度 (秒)。
yarn.nodemanagerdisk-health-checker。min-healthy-disks	0.25	が新しいコンテナを起動 NodeManager するために正常でなければならないディスク数の最小割合。これは、yarn.nodemanager.local-dirs (デフォルトでは、Amazon EMR の /mnt/yarn) と yarn.nodemanager.log-dirs (デフォルトでは /var/log/hadoop-yarn/containers で、Amazon EMR の mnt/var/log/hadoop-yarn/containers に対してシンボリックリンクされています) の両方に対応しています。
yarn.nodemanager.disk-health-checker.max-disk-utilization-per-disk-percentage	90.0	ディスクが異常であるとマークされてから許容されるディス容量使用率の最大パーセンテージ。値の範囲は 0.0 ~ 100.0 です。値が 100 以

プロパティ	デフォルト値	説明
		上の場合、 はフルディスク NodeManager をチェックします。これは、yarn-nodemanager.local-dirs と yarn.nodemanager.log-dirs に適用されます。
yarn.nodemanager.disk-health-checker.min-free-space-per-disk-mb	0	使用するために最低限確保しておく必要があるディスク容量。これは、yarn-nodemanager.local-dirs と yarn.nodemanager.log-dirs に適用されます。

Cannot replicate block, only managed to replicate to zero nodes. (ブロックをレプリケートできません。ゼロノードにのみレプリケートできます。)

「Cannot replicate block, only managed to replicate to zero nodes.」というエラーは、通常、クラスターに十分な HDFS ストレージがない場合に発生します。このエラーは、HDFS の容量を超えるデータが含まれるクラスターを作成したときに発生します。このエラーはクラスターの実行中にのみ発生します。これは、ジョブが終了するとき、使用していた HDFS 容量を解放するためです。

クラスターが使用できる HDFS 容量は、コアノードとして使用される Amazon EC2 インスタンスの数とタイプによって異なります。タスクノードは HDFS ストレージに使用されません。アタッチされている EBS ストレージボリュームを含め、各 Amazon EC2 インスタンスのすべてのディスク容量を HDFS に使用できます。各 EC2 インスタンスタイプのローカルストレージの量の詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスタイプとファミリー」を参照してください。 Amazon EC2

」の「インスタンスのファミリーとタイプ」を参照してください。レプリケーション係数が、使用できる HDFS 容量に影響を及ぼすこともあります。レプリケーション係数は、冗長性を確保するために HDFS に格納された各データブロックのコピー数です。レプリケーション係数は、クラスター内のノード数に応じて大きくなります。これは、クラスターに 10 個以上のノードがある場合は、データブロックごとに 3 つのコピー、4 ~ 9 個のノードがある場合は各ブロックの 2 つのコピー、ノードが 3 つ以下のクラスターの場合は 1 つのコピー (冗長性なし) が格納されているからです。ジョブ

フローが使用できる容量を算出するには、使用できる HDFS 容量の合計をレプリケーション係数で割ります。ノードの数が 9 個から 10 個に増えた場合など、レプリケーション係数が増加することで、使用できる HDFS 容量が実質的には減ることがあります。

たとえば、10 個のコアノードを持つタイプ m1.large のクラスターでは、2833 GB の容量を HDFS で使用できます (10 ノード X ノードあたりの容量 850 GB) ÷レプリケーション係数 3) 。

HDFS で使用できる容量をクラスターが超えた場合は、コアノードをクラスターに追加するか、データを圧縮すると、HDFS 容量を生成できます。クラスターを停止して再起動できる場合は、さらに大きな Amazon EC2 インスタンスタイプのコアノードの使用を検討してください。また、レプリケーション係数を調整して対応することもできます。ただし、レプリケーション係数を小さくすると、HDFS データの冗長性と、紛失または破損した HDFS ブロックからクラスターを回復する機能が低下します。

EC2 QUOTA EXCEEDED

`EC2 QUOTA EXCEEDED` メッセージが表示される場合は、いくつかの原因が考えられます。設定によっては、1 つのクラスターが完全に終了して、割り当てられたリソースを解放するまでに 5 ~ 20 分かかることがあります。クラスターを起動しようとして `EC2 QUOTA EXCEEDED` エラーが発生する場合、そのエラーの原因として、最近終了したクラスターのリソースがまだ解放されていないことが考えられます。このメッセージは、インスタンスグループまたはインスタンスフリートを、そのアカウントの現在のインスタンスクォータよりも大きいターゲットサイズに変更した場合にも表示されることがあります。この変更は手動で、または Auto Scaling により自動的に行われる可能性があります。

この問題を解決するには、以下のオプションを検討してください。

- サービス制限の引き上げをリクエストするには、「Amazon Web Services 全般のリファレンス」の「AWS サービスクォータ」の手順に従ってください。一部の APIs では、制限を増やすよりも CloudWatch イベントを設定する方が適している場合があります。詳細については、「で EMR イベントをセットアップするタイミング CloudWatch」を参照してください。
- 実行中の 1 つ以上のクラスターの容量が不足している場合は、インスタンスグループのサイズを変更するか、実行中のクラスターのインスタンスフリートのターゲット容量を減らします。
- EC2 インスタンス数またはターゲット容量を減らしたクラスターを作成します。

Too many fetch-failures (フェッチの失敗が多すぎる)

ステップまたはタスク試行ログに [Too many fetch-failures] または [Error reading task output] というエラーメッセージが表示されている場合、実行中のタスクは別のタスクの出力に依存するということ

を示しています。これは多くの場合、リデューサータスクが実行のために列に並んでおり、1 つまたは複数のマップタスクの出力を必要としているとき、その出力がまだ利用できないときに発生します。

出力が利用できない場合、次のような理由が考えられます。

- 前提条件となるタスクが処理中です。これは多くの場合、マップタスクです。
- データが別のインスタンスに置かれている場合、ネットワークの接続状態が悪いためにデータが利用できないことがあります。
- 出力の取得に HDFS が利用される場合、HDFS の問題も考えられます。

このエラーの最も一般的な原因は、前のタスクが実行中であることです。これは特に、リデューサータスクが最初に実行を試みるときにエラーが発生する場合に該当します。これに該当するかどうかは、エラーを返したクラスターステップの syslog ログを調べることで確認できます。マップタスクとリデューサータスクの両方が進行していることが syslog から判明した場合、マップタスクが完了していないうちにリデューサーフェーズが開始されたことを示します。

ログで確認するべきことの 1 つは、100% に到達し、それから値を落とすマッパーの進捗パーセンテージです。マップのパーセンテージが 100% のとき、それはすべてのマップタスクが完了していることを意味しません。それが意味することは、Hadoop がすべてのマップタスクを実行していることだけです。この値が 100% より下に落ちる場合、それはマップタスクが失敗し、設定によっては、Hadoop がタスクの再スケジュールを試行する場合があることを意味します。マップの割合がログの 100% のままである場合は、 CloudWatch メトリクス、特に を調べてRunningMapTasks、マップタスクがまだ処理中かどうかを確認します。この情報は、マスターノードで Hadoop Web インターフェイスを利用し、確認することもできます。

この問題が発生する場合、次のような操作を試すことができます。

- リデューサーフェーズが開始までに待ち時間を長くします。この場合、Hadoop 設定の mapred.reduce.slowstart.completed.maps を変更し、時間を長くします。詳細については、「追加のソフトウェアをインストールするためのブートストラップアクションの作成」を参照してください。
- リデューサーのカウントをクラスターのリデューサー能力の合計に合わせます。この場合、ジョブの Hadoop 設定の mapred.reduce.tasks を調整します。
- コンバイナクラスコードを使用して、フェッチする必要がある出力の数を最小限に抑えます。
- Amazon EC2 サービスに、クラスターのネットワークパフォーマンスに影響を与えるような問題がないことを確認します。これは、サービスヘルスダッシュボードを利用して確認できます。

- クラスターのインスタンスの CPU とメモリリソースをチェックし、データ処理がノードのリソースに過度の負担を与えていないことを確認します。詳細については、「クラスターハードウェアとネットワークを設定する」を参照してください。
- Amazon EMR クラスターで使用する Amazon マシンイメージ (AMI) のバージョンを確認します。バージョンが 2.3.0 ~ 2.4.4 の場合は、より新しいバージョンに更新してください。このバージョンの範囲に該当する AMI で使用される Jetty のバージョンでは、マップフェーズからの出力の配信に失敗する可能性があります。リデューサーがマップフェーズからの出力を取得できない場合は、フェッチエラーが発生します。

  Jetty はオープンソースの HTTP サーバーで、Hadoop クラスター内でのマシン間の通信に使用されます。

File could only be replicated to 0 nodes instead of 1 (ファイルは 1 ノードではなく 0 ノードにのみレプリケートできる)

ファイルが HDFS に書き込まれるとき、複数のコアノードに複製されます。このエラーが表示される場合は、 NameNode デーモンに HDFS でデータを書き込むための使用可能な DataNodeインスタンスがないことを意味します。言い換えれば、ブロックレプリケーションが行われていません。このエラーは次のようなさまざまな問題から発生する場合があります。

- HDFS ファイルシステムに容量不足が発生している可能性があります。これが最も一般的な原因です。
- DataNode ジョブの実行時に インスタンスが使用できなかった可能性があります。
- DataNode インスタンスがマスターノードとの通信からブロックされている可能性があります。
- コアインスタンスグループのインスタンスが利用できない可能性があります。
- 権限が不足している可能性があります。例えば、 JobTracker デーモンにはジョブトラッカー情報を作成するアクセス許可がない場合があります。
- DataNode インスタンスの予約済みスペースの設定が不十分である可能性があります。これに該当するかどうかは、dfs.datanode.du.reserved 設定で確認します。

この問題の原因が HDFS のディスク容量不足にあるかどうかを確認するには、 の `HDFSUtilization`メトリクスを参照してください CloudWatch。この値が高すぎる場合、クラスターにコアノードを追加できます。HDFS ディスク容量が不足していると思われるクラスターがある場合は、 の値が特定のレベルを超えたときに警告 CloudWatch するアラームを `HDFSUtilization`

で設定できます。詳細については、「実行中のクラスターのサイズを手動で変更する」および「を使用した Amazon EMR メトリクスのモニタリング CloudWatch」を参照してください。

HDFS の容量不足が問題ではなかった場合は、 DataNode ログ、 NameNodeログ、ネットワーク接続をチェックして、HDFS がデータをレプリケートできなかった可能性のあるその他の問題がないか確認してください。詳細については、「 ログファイルを表示する」を参照してください。

拒否リストに登録されたノード

NodeManager デーモンは、コアノードとタスクノードでコンテナを起動および管理します。コンテナは、マスターノードで実行される NodeManager デーモンによって ResourceManager デーモンに割り当てられます。は、ハートビートを介して NodeManager ノードを ResourceManager モニタリングします。

ResourceManager デーモン拒否によって が一覧表示され NodeManager、タスクの処理に使用できるノードのプールから削除される状況がいくつかあります。

- NodeManager が過去 10 分間 (600,000 ミリ秒) に ResourceManager デーモンにハートビートを送信していない場合。この時間は `yarn.nm.liveness-monitor.expiry-interval-ms` 設定で設定できます。Yarn 設定の変更の詳細については「Amazon EMR リリースガイド」の「アプリケーションの設定」を参照してください。
- NodeManager は、 `yarn.nodemanager.local-dirs`と によって決定されるディスクの状態をチェックしますyarn.nodemanager.log-dirs。このチェックにはアクセス権限と空きディスク容量 (< 90%) が含まれます。ディスクがチェックに失敗した場合、 はその特定のディスクの使用を NodeManager 停止しますが、ノードのステータスは正常であると報告します。多数のディスクがチェックに失敗した場合、ノードは異常として に報告 ResourceManager され、新しいコンテナはノードに割り当てられません。

アプリケーションマスターは、障害が発生したタスクが 3 つ以上ある場合、 NodeManager ノードの拒否リストを表示することもできます。`mapreduce.job.maxtaskfailures.per.tracker` 設定パラメータを利用し、この値を増やすことができます。他には、タスクを失敗と判定するまでに試行する回数を制御する設定を変更できます。マップタスクの場合は `mapreduce.map.max.attempts` を、リデュースタスクの場合は `mapreduce.reduce.maxattempts` を変更します。設定の変更の詳細については「Amazon EMR リリースガイド」の「アプリケーションの設定」を参照してください。

スロットリングエラー

エラー「Throttled from *Amazon EC2* while launching cluster」と「Failed to provision instances due to throttling from *Amazon EC2*」は、別のサービスがアクティビティをスロットリングしたために、Amazon EMR がリクエストを完了できないときに発生します。スロットリングエラーの最も一般的な原因は Amazon EC2 ですが、他のサービスがスロットリングエラーの原因である可能性もあります。パフォーマンスを向上させるためにリージョンごとに AWS サービスの制限が適用されていて、スロットリングエラーは、そのリージョンでアカウントのサービス制限を超えたことを示します。

考えられる原因

Amazon EC2 スロットリングエラーの最も一般的な原因は、多数のクラスターインスタンスが起動されているために、EC2 インスタンスのサービス制限を超えたことです。クラスターインスタンスは以下の理由で作成される可能性があります。

- 新しいクラスターが作成された。
- クラスターのサイズが手動で変更された。詳細については、「実行中のクラスターのサイズを手動で変更する」を参照してください。
- クラスター内のインスタンスグループで Auto Scaling ルールの結果としてインスタンスが追加された (スケールアウトされた)。詳細については、「自動スケーリングルールについて」を参照してください。
- クラスター内のインスタンスフリートで増加したターゲット容量を満たすためにインスタンスが追加された。詳細については、「インスタンスフリートを設定する」を参照してください。

Amazon EC2 に対して発行された API リクエストの頻度またはタイプにより、スロットリングエラーが発生することもあります。Amazon EC2 が API リクエストをスロットリングする方法の詳細については、「Amazon EC2 API リファレンス」の「クエリ API リクエスト率」を参照してください。

解決方法

以下の解決方法を検討してください。

- サービス制限の引き上げをリクエストするには、「Amazon Web Services 全般のリファレンス」の「AWS サービスクォータ」の手順に従ってください。一部の APIs では、制限を増やすよりも CloudWatch イベントを設定する方が適している場合があります。詳細については、「で EMR イベントをセットアップするタイミング CloudWatch」を参照してください。

- 同じスケジュールで起動するクラスターがある場合 (特定時間の 0 分など)、開始時間をずらすことを検討してください。
- インスタンス容量のピーク需要に合わせたサイズのクラスターがあり、インスタンス容量の需要が定期的に変動する場合は、インスタンスをオンデマンドで追加および削除するように Auto Scaling を指定することを検討してください。この方法では、インスタンスがより効率的に使用され、需要プロファイルに応じて、アカウント全体で同時にリクエストされるインスタンスが少なくなります。詳細については、「カスタムポリシーによる自動スケーリングをインスタンスグループに使用する」を参照してください。

サポートされていないインスタンスタイプ

クラスターを作成し、「リクエストされたインスタンスタイプ*InstanceType*はリクエストされたアベイラビリティーゾーンでサポートされていません」というエラーメッセージが表示されて失敗した場合、クラスターを作成し、クラスターが作成されたリージョンとアベイラビリティーゾーンで Amazon EMR でサポートされていない 1 つ以上のインスタンスグループにインスタンスタイプを指定したことを意味します。Amazon EMR は、リージョン内のあるアベイラビリティーゾーンではインスタンスタイプをサポートし、別のアベイラビリティーゾーンではサポートしないことがあります。クラスターに選択したサブネットによって、リージョン内のアベイラビリティーゾーンが決まります。

ソリューション

を使用してアベイラビリティーゾーンで使用可能なインスタンスタイプを特定する AWS CLI

- `ec2 run-instances` コマンドで `--dry-run` オプションを使用します。以下の例で、*m5.xlarge* を使用するインスタンスタイプに置き換え、*ami-035be7bafff33b6b6* をそのインスタンスタイプに関連付けられている AMI に置き換え、*subnet-12ab3c45* をクエリ対象のアベイラビリティーゾーン内のサブネットに置き換えてください。

  ```
  aws ec2 run-instances --instance-type m5.xlarge --dry-run --image-
  id ami-035be7bafff33b6b6 --subnet-id subnet-12ab3c45
  ```

  AMI ID の検索手順については、「Linux AMI の検索」を参照してください。サブネット ID を見つけるには、describe-subnets コマンドを使用できます。

利用可能なインスタンスタイプを確認にする方法については、「Amazon EC2 インスタンスタイプの検索」を参照してください。

使用可能なインスタンスタイプを決定したら、以下のいずれかを実行できます。

- 同じリージョンと EC2 サブネットにクラスターを作成し、最初に選択したものと同様の機能を持つ別のインスタンスタイプを選択します。サポートされているインスタンスタイプについては、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。EC2 インスタンスタイプの機能を比較するには、「Amazon EC2 インスタンスタイプ」を参照してください。
- インスタンスタイプが使用可能かつ Amazon EMR によってサポートされているアベイラビリティーゾーン内のクラスターのサブネットを選択します。

EC2 が容量不足

「EC2 の容量が不足しています」という*InstanceType*エラーは、指定された EC2 インスタンスタイプがなくなったアベイラビリティーゾーンでクラスターを作成しようとしたり、クラスターにインスタンスを追加しようとしたりすると発生します。クラスターに選択したサブネットによって、アベイラビリティーゾーンが決まります。

クラスターを作成するには、以下のいずれかの操作を実行します。

- 類似の機能を持つ別のインスタンスタイプを指定する。
- クラスターを別のリージョンに作成する。
- 必要なインスタンスタイプが使用可能なアベイラビリティーゾーン内のサブネットを選択する。

実行中のクラスターにインスタンスを追加するには、次のいずれかを実行します。

- インスタンスグループ設定またはインスタンスフリート設定を変更して、類似の機能を持つ使用可能なインスタンスタイプを追加する。サポートされているインスタンスタイプについては、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。EC2 インスタンスタイプの機能を比較するには、「Amazon EC2 インスタンスタイプ」を参照してください。
- インスタンスタイプが使用可能なリージョンとアベイラビリティーゾーンでクラスターを終了して再作成する。

HDFS レプリケーション係数エラー

コアインスタンスグループまたはインスタンスフリート からコアノードを削除すると、Amazon EMR で HDFS レプリケーションエラーが発生する可能性があります。このエラーは、コアノードを削除し、コアノード数が Hadoop Distributed File System (HDFS) の設定済み dfs.replication 係

数を下回った場合に発生します。そのため、Amazon EMR は オペレーションを安全に実行できません。dfs.replication 設定のデフォルト値を決定するには、HDFS 設定 を使用します。

考えられる原因

HDFS レプリケーション係数エラーの考えられる原因については、以下を参照してください。

- コアインスタンスグループまたはインスタンスフリートのサイズを設定したdfs.replication係数より手動で変更した場合。
- マネージドスケーリングまたは自動スケーリングのポリシーにより、スケーリングによってコアノードの数が のしきい値を下回る場合がありますdfs.replication。
- このエラーは、クラスターに で定義されているコアノードの数が最小限である場合に、Amazon EMR が異常なコアノードを置き換えようとする場合にも発生する可能性がありますdfs.replication。

ソリューションとベストプラクティス

ソリューションとベストプラクティスについては、以下を参照してください。

- Amazon EMR クラスターのサイズを手動で変更する場合、Amazon EMR はサイズ変更を安全に完了できないdfs.replicationため、 より下にスケールダウンしないでください。
- マネージドスケーリングまたは自動スケーリングを使用する場合は、クラスターの最小容量がdfs.replication係数より小さくないことを確認してください。
- コアインスタンスの数は、少なくとも 1 dfs.replicationを足した数にする必要があります。これにより、異常なコア交換を有効にした場合、Amazon EMR は異常なコアノードを正常に置き換えることができます。

> **Important**
>
> 1 つのコアノードに障害が発生すると、 を 1 dfs.replicationに設定すると HDFS データが失われる可能性があります。クラスターに HDFS ストレージがある場合は、データ損失を回避し、少なくとも 2 のdfs.replication係数を設定するために、本番ワークロード用に少なくとも 4 つのコアノードでクラスターを設定することをお勧めします。

HDFS スペース不足エラー

Hadoop Distributed File System (HDFS) のスペース不足エラーは、コアノードを削除しようとしても、HDFS に不足しているスペースがあるため、Amazon EMR が安全にオペレーションを完了できない場合に発生する可能性があります。Amazon EMR がコアノードを削除する前に、ノード上のすべての HDFS データを他のコアノードに転送して、データの冗長性を確保する必要があります。ただし、他のコアノードにレプリケーション用の十分なスペースがない場合、Amazon EMR はノードを適切に廃止できません。

考えられる原因

HDFS スペース不足エラーの考えられる原因のリストについては、以下を参照してください。

- スケールダウン前に残りのノードにデータレプリケーション用の十分な HDFS スペースがない場合に、コアインスタンスグループまたはインスタンスフリートを手動でスケールダウンする場合。
- マネージドスケーリングまたは自動スケーリングは、データレプリケーションに十分な HDFS スペースがない場合に、コアインスタンスグループまたはインスタンスフリートをスケールダウンします。
- Amazon EMR は異常なコアノードの置き換えを試みますが、HDFS スペースが不足しているため、ノードを安全に置き換えることができません。

ソリューションとベストプラクティス

ソリューションとベストプラクティスについては、以下を参照してください。

- Amazon EMR クラスター内のコアノードの数をスケールアップします。マネージドスケーリングまたは自動スケーリングを使用する場合は、コアノードの最小容量を増やします。
- EMR クラスターを作成するときは、コアノードにより大きな EBS ボリュームを使用します。
- EMR クラスター内の不要な HDFS データを削除します。EMR クラスターの容量が少ないかどうかを確認するために、クラスター内の`HDFSUtilization`メトリクスをモニタリングする CloudWatch アラームを設定することをお勧めします。

入力エラーと出力エラー

次は、クラスターの入力および出力オペレーションに共通するエラーです。

トピック

- Amazon Simple Storage Service (Amazon S3) へのパスに少なくとも 3 つのスラッシュがありますか?
- 入力ディレクトリを再帰的に走査しようとしていませんか?
- 出力ディレクトリが存在していませんか?
- HTTP URL を使用してリソースを指定しようとしていませんか?
- Simple Storage Service (Amazon S3) バケットの参照に使用している名前のフォーマットが無効ではありませんか?
- Simple Storage Service (Amazon S3) との間でのデータの読み込みで問題が発生しますか?

Amazon Simple Storage Service (Amazon S3) へのパスに少なくとも 3 つのスラッシュがありますか?

Simple Storage Service (Amazon S3) バケットを指定するときは、終了スラッシュを URL の末尾に付ける必要があります。例えば、バケットを「s3n://DOC-EXAMPLE-BUCKET1」として参照するのではなく、「s3n://DOC-EXAMPLE-BUCKET1/」を使用する必要があります。スラッシュを付けないと、Hadoop では、ほとんどの場合、クラスターが失敗します。

入力ディレクトリを再帰的に走査しようとしていませんか?

Hadoop では、入力ディレクトリでファイルが再帰的に検索されることはありません。/corpus/01/01.txt、/corpus/01/02.txt、/corpus/02/01.txt のようなディレクトリ構造で、入力パラメータとして /corpus/ をクラスターに指定しても、Hadoop は入力ファイルを検出しません。/corpus/ ディレクトリが空で、Hadoop はサブディレクトリのコンテンツをチェックしないからです。同様に、Simple Storage Service (Amazon S3) バケットのサブディレクトリも Hadoop が再帰的に確認することはありません。

入力ファイルは、サブディレクトリではなく、指定する入力ディレクトリまたは Simple Storage Service (Amazon S3) バケットに直接配置されている必要があります。

出力ディレクトリが存在していませんか?

既に存在する出力パスを指定すると、Hadoop では、ほとんどの場合、クラスターが失敗します。つまり、クラスターを一度実行してから、まったく同じパラメータを指定して再度実行すると、おそらく最初は動作しますが、その後は動作しなくなります。最初の実行後、出力パスが存在するようになり、これにより以降の実行がすべて失敗するからです。

HTTP URL を使用してリソースを指定しようとしていませんか?

Hadoop では、http:// プレフィックスを使用して指定されたリソースの場所は認識されません。したがって、HTTP URL を使用したリソースを参照することはできません。たとえば、http://mysite/myjar.jar を JAR パラメータとして渡すと、クラスターは失敗します。

Simple Storage Service (Amazon S3) バケットの参照に使用している名前のフォーマットが無効ではありませんか?

Amazon EMR で「DOC-EXAMPLE-BUCKET1.1」のようなバケット名を使用しようとすると、クラスターが失敗します。Amazon EMR では、バケット名を有効な RFC 2396 ホスト名にする必要があるためです。名前の末尾に数値を使用することはできません。さらに、Hadoop の要件により、Amazon EMR で使用する Simple Storage Service (Amazon S3) バケット名に含めることができるのは、小文字、数値、ピリオド (.)、およびハイフン (-) に限られます。Simple Storage Service (Amazon S3) バケット名の形式を設定する方法の詳細については、「Amazon Simple Storage Service ユーザーガイド」の「バケットの制約と制限」を参照してください。

Simple Storage Service (Amazon S3) との間でのデータの読み込みで問題が発生しますか?

Simple Storage Service (Amazon S3) は、Amazon EMR の最も一般的な入力と出力のソースです。よくある間違いは、一般的なファイルシステムと同様に Simple Storage Service (Amazon S3) を扱うことです。Simple Storage Service (Amazon S3) とファイルシステムの間には、クラスターの実行時に考慮する必要がある差異があります。

- Simple Storage Service (Amazon S3) で内部エラーが発生した場合、アプリケーションでそのエラーを正しく処理し、オペレーションを再試行する必要があります。
- Simple Storage Service (Amazon S3) への呼び出しからの応答に時間がかかりすぎる場合、アプリケーションが Simple Storage Service (Amazon S3) を呼び出す頻度を減らすことが必要な場合があります。
- Simple Storage Service (Amazon S3) バケット内のすべてのオブジェクトの一覧表示は、負荷の高い呼び出しです。呼び出しを行う回数を最小限に抑えることがアプリケーションに要求されます。

クラスターと Simple Storage Service (Amazon S3) の対話を改善できる方法はいくつかあります。

- 最新のリリースバージョンの Amazon EMR を使用してクラスターを起動します。

- S3DistCp を使用して、Amazon S3 との間でオブジェクトを移動します。S3DistCp は、Amazon S3 の要件に合わせてエラー処理、再試行、バックオフを実装します。詳細については、S3DistCp を使用した分散コピー」を参照してください。
- 結果整合性を念頭に置いてアプリケーションを設計します。クラスターの実行中の中間データストレージには HDFS を使用し、初期データの入力と最終結果の出力にのみ Simple Storage Service (Amazon S3) を使用します。
- クラスターが 1 秒あたり 200 以上のトランザクションを Simple Storage Service (Amazon S3) にコミットする場合は、サポートに問い合わせて、1 秒あたりのトランザクション数の増大に対応するようにバケットを用意し、「Amazon S3 performance tips & tricks」で説明されているキーパーティション方式を使用することを検討してください。
- Hadoop 設定の io.file.buffer.size を 65536 に設定します。この設定により、Hadoop で Simple Storage Service (Amazon S3) オブジェクトの検索にかかる時間が短縮されます。
- クラスターで Simple Storage Service (Amazon S3) の並行処理問題が発生する場合、Hadoop の投機的実行機能の無効化を検討してください。この無効化は遅いクラスターのトラブルシューティングにも役立ちます。そのためには、`mapreduce.map.speculative` および `mapreduce.reduce.speculative` プロパティを `false` に設定します。クラスターを起動するときに、`mapred-env` 設定分類を使用してこれらの値を設定できます。詳細については、「Amazon EMR リリースガイド」の「アプリケーションの設定」を参照してください。
- Hive クラスターを実行している場合、「Simple Storage Service (Amazon S3) と Hive の間でのデータの読み込みで問題が発生しますか?」を参照してください。

詳細については、「Amazon Simple Storage Service ユーザーガイド」の「Amazon S3 のエラーに関するベストプラクティス」を参照してください。

権限エラー

次は権限または認証情報の利用時に共通するエラーです。

トピック

- 正しい認証情報を SSH に渡しましたか?
- IAM を使用している場合、適切な Amazon EC2 ポリシーを設定していますか?

正しい認証情報を SSH に渡しましたか?

SSH を使用してマスターノードに接続できない場合、それはセキュリティの認証情報の問題である可能性が高いです。

まず、SSH キーが含まれる .pem ファイルに適切な権限があることを確認します。次の例で示すように、chmod を使用して .pem ファイルの権限を変更できます。この例の mykey.pem は、ご自身の .pem ファイルに置き換えてください。

```
chmod og-rwx mykey.pem
```

原因として次に考えられるのは、クラスターの作成時に指定したキーペアを使用していないという状況です。複数のキーペアを作成した場合、これはよく起こります。Amazon EMR コンソールで (または CLI の `--describe` オプションを使用して) クラスターの詳細を調べて、クラスター作成時に指定したキーペアの名前を確認します。

正しいキーペアを使用していることと、.pem ファイルで権限が適切に設定されていることを確認したら、次のコマンドを使用して、マスターノードに SSH 接続できます。ここで、mykey.pem は使用する .pem ファイルの名前に、`hadoop@ec2-01-001-001-1.compute-1.amazonaws.com` はマスターノードのパブリック DNS 名に置き換えてください (CLI の `--describe`オプション、または Amazon EMR コンソール経由で入手できます)。

> **Important**
>
> Amazon EMR クラスターノードに接続するときは、ログイン名 hadoop を使用する必要があります。そうしないと、`Server refused our key` エラーなどのエラーが発生する場合があります。

```
ssh -i mykey.pem hadoop@ec2-01-001-001-1.compute-1.amazonaws.com
```

詳細については、「SSH を使用してプライマリノードに接続する」を参照してください。

IAM を使用している場合、適切な Amazon EC2 ポリシーを設定していますか?

Amazon EMR では EC2 インスタンスをノードとして使用するため、Amazon EMR がユーザーの代わりにそうしたインスタンスを管理できるようにするには、Amazon EMR ユーザーにも特定の Amazon EC2 ポリシーを設定する必要があります。必要な権限を設定していない場合、Amazon EMR から [account is not authorized to call EC2] というエラーが返ります。

Amazon EMR を実行するために IAM アカウントに設定する必要がある Amazon EC2 ポリシーに関する詳細については、「Amazon EMR で IAM が機能する仕組み」を参照してください。

Hive クラスターのエラー

Hive エラーの原因は、通常、syslog ファイルで確認できます。このファイルには、[Steps (ステップ)] ペインからアクセスします。このファイルで問題を特定できない場合は、Hadoop タスク試行のエラーメッセージを確認してください。ここには、[Task Attempts] ペインから接続します。

次は、Hive クラスターに共通するエラーです。

トピック

- 最新バージョンの Hive を使用していますか?
- Hive スクリプトに構文エラーがありましたか?
- インタラクティブに実行しているとき、ジョブに障害が発生しましたか?
- Simple Storage Service (Amazon S3) と Hive の間でのデータの読み込みで問題が発生しますか?

最新バージョンの Hive を使用していますか?

Hive の最新バージョンには現行のパッチとバグ修正プログラムがすべて含まれているので、問題を解決できる場合があります。

Hive スクリプトに構文エラーがありましたか?

ステップに障害が発生した場合、Hive スクリプトを実行したステップのログの stdout ファイルを確認してください。エラーがそこにない場合、障害が発生したタスク試行のタスク試行ログの syslog ファイルを確認してください。詳細については、「ログファイルを表示する」を参照してください。

インタラクティブに実行しているとき、ジョブに障害が発生しましたか?

マスターノードで Hive をインタラクティブに実行しているときにクラスターに障害が発生した場合、障害が発生したタスク試行のタスク試行ログの syslog エントリを確認してください。詳細については、「ログファイルを表示する」を参照してください。

Simple Storage Service (Amazon S3) と Hive の間でのデータの読み込みで問題が発生しますか?

Simple Storage Service (Amazon S3) のデータへのアクセスで問題が発生する場合、まず「Simple Storage Service (Amazon S3) との間でのデータの読み込みで問題が発生しますか?」に記載されている考えられる原因を確認します。そのいずれも原因として考えられない場合、Hive に固有の次の原因について考慮します。

- 最新バージョンの Hive を使用していることを確認してください。最新バージョンには、現行のすべてのパッチが含まれており、問題を解決できる可能性があります。詳細については、「Apache Hive」を参照してください。
- `INSERT OVERWRITE` を使用するには、Simple Storage Service (Amazon S3) バケットまたはフォルダーの内容をリストする必要があります。これはコストがかかるオペレーションです。可能であれば、Hive で既存のオブジェクトをリストアップし、削除する代わりに、パスを手動で取り除きます。
- 5.0 より前の Amazon EMR リリースバージョンを使用している場合は、HiveQL で次のコマンドを使用して、ローカルのクラスターに Simple Storage Service (Amazon S3) のリストオペレーションの結果を事前キャッシュできます。

```
set hive.optimize.s3.query=true;
```

- 可能な限り、静的パーティションを利用します。
- 一部のバージョンの Hive および Amazon EMR では、テーブルが Hive によって予期されているのとは異なる場所に保存されているために ALTER TABLES の使用が失敗する可能性があります。この問題を解決するには、`/home/hadoop/conf/core-site.xml` で を追加または更新します。

```
<property>
    <name>fs.s3n.endpoint</name>
    <value>s3.amazonaws.com</value>
</property>
```

VPC エラー

Amazon EMR の VPC 設定で一般的なエラーは次のとおりです。

トピック

- 無効なサブネット設定

- 不明な DHCP オプション設定
- 権限エラー
- START_FAILED になるエラー
- クラスターTerminated with errorsと の起動に NameNode 失敗する

無効なサブネット設定

[Cluster Details] ページの [Status] フィールドに、次のようなエラーが表示されます。

```
The subnet configuration was invalid: Cannot find route to InternetGateway
in main RouteTable rtb-id for vpc vpc-id.
```

この問題を解決するには、インターネットゲートウェイを作成し、VPC に接続する必要があります。詳細については、「インターネットゲートウェイを VPC に追加する」を参照してください。

または、[Enable DNS resolution] および [Enable DNS hostname support] を有効にした状態で、VPC を設定したことを検証します。詳細については、「Using DNS with Your VPC」を参照してください。

不明な DHCP オプション設定

クラスターのシステムログ (syslog) に次のようなエラーとともにステップの失敗が表示されます。

```
ERROR org.apache.hadoop.security.UserGroupInformation
(main): PriviledgedActionException as:hadoop (auth:SIMPLE)
cause:java.io.IOException:
org.apache.hadoop.yarn.exceptions.ApplicationNotFoundException: Application
with id 'application_id' doesn't exist in RM.
```

または

```
ERROR org.apache.hadoop.streaming.StreamJob (main): Error Launching job :
org.apache.hadoop.yarn.exceptions.ApplicationNotFoundException: Application
with id 'application_id' doesn't exist in RM.
```

この問題を解決するには、パラメータが以下の値に設定されている DHCP オプション設定を含む VPC を設定する必要があります。

> **Note**
>
> AWS GovCloud (米国西部) リージョンを使用する場合は、次の例で使用されている値**us-gov-west-1.compute.internal**の代わりに domain-name を に設定します。

- domain-name = **ec2.internal**

  リージョンが米国東部 (バージニア北部) の場合は、**ec2.internal** を使用します。その他のリージョンについては、*region-name*.**compute.internal** を使用します。たとえば、us-west-2 では、domain-name=**us-west-2.compute.internal** を使用します。
- domain-name-servers = **AmazonProvidedDNS**

詳細については、「DHCP オプションセット」を参照してください。

権限エラー

ステップの stderr ログ内のエラーは、Simple Storage Service (Amazon S3) リソースに適切な権限がないことを示します。これは 403 エラーで、エラーは以下のようになっています。

```
Exception in thread "main" com.amazonaws.services.s3.model.AmazonS3Exception: Access
 Denied (Service: Amazon S3; Status Code: 403; Error Code: AccessDenied; Request
 ID: REQUEST_ID
```

ActionOnFailure が に設定されている場合TERMINATE_JOB_FLOW、クラスターは 状態 で終了しますSHUTDOWN_COMPLETED_WITH_ERRORS。

この問題をトラブルシューティングする方法には、次のようなものがあります。

- VPC 内で Simple Storage Service (Amazon S3) バケットポリシーを使用している場合、すべてのバケットへのアクセスを許可する必要があります。このためには、VPC エンドポイントを作成し、その作成時に [ポリシー] オプションの下で [Allow all] (すべて許可) を選択します。
- S3 リソースに関連付けられたすべてのポリシーに、クラスターを起動する VPC が含まれていることを確認します。
- クラスターから次のコマンドを実行して、バケットにアクセスできることを確認します。

  ```
  hadoop fs -copyToLocal s3://path-to-bucket /tmp/
  ```

- より詳しいデバッグ情報を取得するには、log4j.logger.org.apache.http.wire ファイルの DEBUG パラメータを /home/hadoop/conf/log4j.properties に設定します。クラスターからバケットへのアクセスを試行してから、stderr ログファイルを確認します。ログファイルで詳細情報を確認できます。

```
Access denied for getting the prefix for bucket - us-west-2.elasticmapreduce with
 path samples/wordcount/input/
15/03/25 23:46:20 DEBUG http.wire: >> "GET /?prefix=samples%2Fwordcount%2Finput
%2F&delimiter=%2F&max-keys=1 HTTP/1.1[\r][\n]"
15/03/25 23:46:20 DEBUG http.wire: >> "Host: us-
west-2.elasticmapreduce.s3.amazonaws.com[\r][\n]"
```

START_FAILED になるエラー

AMI 3.7.0 より前のバージョンでは、ホスト名が指定された VPC の場合、Amazon EMR はサブネットの内部ホスト名を ip-*X.X.X.X.customdomain.com*.tld のようにカスタムドメインアドレスでマッピングします。例えば、ホスト名が ip-10.0.0.10 で、VPC のドメイン名オプションが customdomain.com に設定されている場合、Amazon EMR でマッピングされたホスト名は ip-10.0.1.0.customdomain.com になります。エントリは /etc/hosts に追加され、ホスト名を解決して 10.0.0.10 にします。この動作は AMI 3.7.0 で変更され、Amazon EMR は VPC の DHCP 設定を全面的に優先するようになりました。以前は、お客様は、ホスト名のマッピングを指定するブートストラップアクションも使用できました。

この動作を維持するには、DNS を提供し、カスタムドメインに必要な解決セットアップを転送する必要があります。

クラスター**Terminated with errors**と の起動に NameNode 失敗する

カスタム DNS ドメイン名を使用する VPC で EMR クラスターを起動すると、コンソールに次のエラーメッセージが表示されてクラスターで障害が発生することがあります。

```
Terminated with errors  On the master instance(instance-id), bootstrap action 1
 returned a  non-zero return code
```

失敗は、 が起動 NameNode できないことが原因です。これにより、Amazon S3 URI の形式が である NameNode ログに次のエラーが表示されますs3://*mybucket*/*logs*/*cluster-id*/daemons/*master instance-id*/hadoop-hadoop-namenode-*master node hostname*.log.gz。

```
2015-07-23 20:17:06,266 WARN
      org.apache.hadoop.hdfs.server.namenode.FSNamesystem (main): Encountered
 exception
      loading fsimage  java.io.IOException: NameNode is not formatted.
      at

 org.apache.hadoop.hdfs.server.namenode.FSImage.recoverTransitionRead(FSImage.java:212)
           at

 org.apache.hadoop.hdfs.server.namenode.FSNamesystem.loadFSImage(FSNamesystem.java:1020)
           at

 org.apache.hadoop.hdfs.server.namenode.FSNamesystem.loadFromDisk(FSNamesystem.java:739)
           at
      org.apache.hadoop.hdfs.server.namenode.NameNode.loadNamesystem(NameNode.java:537)
           at
      org.apache.hadoop.hdfs.server.namenode.NameNode.initialize(NameNode.java:596)

      at  org.apache.hadoop.hdfs.server.namenode.NameNode.<init>(NameNode.java:765)
           at
      org.apache.hadoop.hdfs.server.namenode.NameNode.<init>(NameNode.java:749)
      at

 org.apache.hadoop.hdfs.server.namenode.NameNode.createNameNode(NameNode.java:1441)
           at
      org.apache.hadoop.hdfs.server.namenode.NameNode.main(NameNode.java:1507)
```

これは、 AWSが提供する DNS サーバーとカスタムユーザーが提供する DNS サーバーの両方を利用する VPC で EMR クラスターを起動するときに、EC2 インスタンスに完全修飾ドメイン名のセットが複数存在する可能性があるという潜在的な問題が原因です。ユーザーが提供する DNS サーバーが、EMR クラスターでノードの指定に使用される A レコードのポインター (PTR) レコードを提供しない場合、この方法で設定するとクラスターが起動に失敗します。解決策として、EC2 インスタンスが VPC 内のいずれかのサブネットで起動されるときに作成される A レコードにごとに PTR レコードを 1 つ追加できます。

ストリーミングクラスターのエラー

ストリーミングエラーの原因は、通常、syslog ファイルで確認できます。ここには、[Steps] ペインで接続します。

ストリーミングクラスターの一般的なエラーを以下に示します。

トピック

データは間違った形式でマッパーに送信されていませんか?

これに該当するかを確認するには、タスク試行ログの失敗したタスク試行の syslog ファイルでエラーメッセージを検索します。詳細については、「ログファイルを表示する」を参照してください。

スクリプトがタイムアウトしていませんか?

マッパーまたはリデューサースクリプトに対するデフォルトのタイムアウトは 600 秒です。スクリプトにこれ以上時間がかかる場合、タスクの試行は失敗します。タスク試行ログで失敗したタスク試行の syslog ファイルを確認して、これに該当するかを確認できます。詳細については、「ログファイルを表示する」を参照してください。

mapred.task.timeout 設定に新しい値を設定して、制限時間を変更できます。この設定は、入力の読み取り、出力の書き込み、またはステータス文字列の更新がされなくなってから何ミリ秒後に、Amazon EMR がタスクを終了するかを指定します。追加のストリーミング引数 -jobconf mapred.task.timeout=800000 を渡して、この値を更新できます。

無効なストリーミング引数を渡していませんか?

Hadoop ストリーミングでは、次の引数のみがサポートされています。ここに示されていない引数を渡すと、クラスターは失敗します。

```
-blockAutoGenerateCacheFiles
-cacheArchive
-cacheFile
-cmdenv
-combiner
-debug
-input
-inputformat
-inputreader
```

```
-jobconf
-mapper
-numReduceTasks
-output
-outputformat
-partitioner
-reducer
-verbose
```

さらに、Hadoop ストリーミングでは、Java 構文を使って渡された引数、つまり先頭にハイフンが 1 つしか付いていない引数しか認識されません。先頭に 2 つのハイフンが付いている引数を渡すと、クラスターは失敗します。

スクリプトがエラーで終了しましたか?

マッパーまたはリデューサースクリプトがエラーで終了した場合、失敗したタスク試行のタスク試行ログの `stderr` ファイルでエラーを検索できます。詳細については、「ログファイルを表示する」を参照してください。

カスタム JAR クラスターのエラー

次は、カスタム JAR クラスターに共通するエラーです。

トピック

- ジョブを作成しようとすると JAR が例外をスローしますか?
- JAR はマップタスク内でエラーをスローしますか?

ジョブを作成しようとすると JAR が例外をスローしますか?

Hadoop ジョブの作成中にカスタム JAR のメインプログラムが例外をスローする場合は、ステップログの `syslog` ファイルを確認することをお勧めします。詳細については、「ログファイルを表示する」を参照してください。

JAR はマップタスク内でエラーをスローしますか?

入力データの処理中にカスタム JAR およびマッパーが例外をスローする場合は、タスク試行ログの `syslog` ファイルを確認することをお勧めします。詳細については、「ログファイルを表示する」を参照してください。

AWS GovCloud（米国西部) エラー

AWS GovCloud（米国西部) リージョンは、セキュリティ、設定、デフォルト設定の他のリージョンとは異なります。そのため、より一般的なトラブルシューティングの推奨事項を使用する前に、次のチェックリストを使用して、 AWS GovCloud（米国西部) リージョンに固有の Amazon EMR エラーをトラブルシューティングします。

- IAM ロールが正しく設定されていることを確認します。詳細については、「AWS のサービスおよびリソースへのアクセス許可を Amazon EMR に付与する IAM サービスロールの設定」を参照してください。
- VPC 設定により DNS 解決/ホスト名サポート、インターネットゲートウェイ、および DHCP オプション設定パラメータが正しく設定されていることを確認します。詳細については、「VPC エラー」を参照してください。

これらのステップで問題が解決されない場合は、一般的な Amazon EMR エラーをトラブルシューティングするステップに進みます。詳細については、「Amazon EMR の一般的なエラー」を参照してください。

見つからないクラスターを検索する

クラスターをコンソールリストまたは `ListClusters` API によって検索できない場合は、次の点を確認してください。

- クラスターの有効期間が、処理の完了時点から 2 か月未満であることを確認します。Amazon EMR では、完了したクラスターのメタデータ情報を 2 か月間無料で保存します。完了したクラスターをコンソールから削除することはできません。これらのクラスターは、Amazon EMR によって 2 か月後に自動削除されます。
- クラスターを表示するアクセス権限があることを確認します。
- クラスターが存在する AWS リージョン 場所と同じ を表示していることを確認します。

失敗したクラスターのトラブルシューティング

このセクションでは、失敗したクラスターをトラブルシューティングする手順を説明します。これは、クラスターがエラーコードで終了したことを意味します。

> **Note**
>
> EMR クラスターがエラーで終了すると、`DescribeCluster` と `ListClusters` の各 API からエラーコードとエラーメッセージが返ります。クラスターエラーによっては、`ErrorDetail` データ配列も障害のトラブルシューティングに役立つ場合があります。詳細については、「情報を含む ErrorDetailエラーコード」を参照してください。

クラスターを稼働させたにもかかわらず、結果が返るまでに時間がかかる場合は、「低速なクラスターのトラブルシューティング」参照してください。

トピック

ステップ 1: 問題に関するデータの収集

クラスターのトラブルシューティングの最初の手順は、不具合とクラスターの現在のステータスおよび設定に関する情報を収集することです。この情報は、問題の原因を確認または除外するために、以降の手順で使用されます。

問題の定義

まず、問題を明確に定義します。自問するいくつかの質問を次に示します。

- 何が起こると予想したか 代わりに何が起こったか
- この問題が最初に発生したのはいつか それ以来どのくらいの頻度で起こっているか
- クラスターの設定方法や実行方法に変化があったか

クラスターの詳細

問題を追跡するために、次のクラスターの詳細が役立ちます。この情報の収集方法の詳細については、「クラスターステータスと詳細の表示」を参照してください。

- クラスターの識別子。(ジョブフロー識別子とも呼ばれます)。
- AWS リージョン およびクラスターが起動されたアベイラビリティーゾーン。
- クラスターの状態 (最後の状態変更の詳細を含む)。
- マスター、コア、およびタスクの各ノードに指定されている EC2 インスタンスの種類と数。

ステップ 2: 環境の確認

Amazon EMR は、ウェブサービスのエコシステムおよびオープンソースソフトウェアの一部として動作します。これらの依存関係に影響する内容は、Amazon EMR のパフォーマンスに影響を及ぼす可能性があります。

トピック

サービスの停止の確認

Amazon EMR は、内部で複数のAmazon Web Services を使用します。Amazon EC2 で仮想サーバーを実行し、Amazon S3 にデータとスクリプトを保存し、メトリクスを にレポートします CloudWatch。これらのサービスを中断するイベントはまれですが、発生すると、Amazon EMR で問題が発生する恐れがあります。

次に進む前に、サービスヘルスダッシュボードを確認します。クラスターを起動したリージョンで、これらのサービスのいずれかに中断イベントがあるかどうかを確認します。

使用制限の確認

大規模なクラスターを起動している場合、多数のクラスターを同時に起動している場合、または を他のユーザー AWS アカウント と共有しているユーザーの場合、 AWS サービスの制限を超えたためクラスターが失敗した可能性があります。

Amazon EC2 は、1 つの AWS リージョンで実行されている仮想サーバーインスタンスの数を、オンデマンドインスタンスまたはリザーブドインスタンスの 20 個に制限します。20 を超えるノードを

持つクラスターを起動するか、 でアクティブな EC2 インスタンスの合計数が 20 AWS アカウントを超えるクラスターを起動すると、クラスターは必要なすべての EC2 インスタンスを起動できず、失敗する可能性があります。この場合、Amazon EMR は `EC2 QUOTA EXCEEDED` エラーを返します。Amazon EC2 インスタンス制限 AWS の引き上げリクエストアプリケーションを送信することで、アカウントで実行できる EC2 インスタンスの数を増やすようにリクエストできます。 Amazon EC2

使用制限を超えるもう 1 つの原因は、クラスターが終了してからすべてのリソースを解放するまでの遅延です。設定によっては、1 つのクラスターが完全に終了して、割り当てられたリソースを解放するまでに 5～20 分かかることがあります。クラスターを起動しようとして `EC2 QUOTA EXCEEDED` エラーが発生する場合、そのエラーの原因として、最近終了したクラスターのリソースがまだ解放されていないことが考えられます。この場合は、Amazon EC2 クォータ増加リクエストを送信するか、20 分待ってからクラスターを再起動してください。

Simple Storage Service (Amazon S3) では、アカウントで作成されるバケットの数を 100 までに制限しています。クラスターがこの制限を超える新しいバケットを作成すると、バケットの作成が失敗し、クラスターが失敗する恐れがあります。

リリースバージョンの確認

クラスターの起動に使用したリリースラベルと最新の Amazon EMR リリースを比較します。Amazon EMR の各リリースには、新しいアプリケーション、機能、パッチ、バグ修正などの改善内容が含まれています。クラスターに影響を及ぼしている問題が、最新のリリースバージョンでは修正されている可能性があります。可能な場合は、最新のバージョンを使用してクラスターをもう一度実行してください。

Amazon VPC サブネット設定の確認

クラスターが Amazon VPC サブネットで起動された場合は、「ネットワークを設定する」の説明に従ってサブネットを設定する必要があります。さらに、クラスターを起動するサブネットに存在する空きの Elastic IP アドレスが、クラスター内の各ノードに 1 つずつ割り当てるのに十分であることを確認します。

ステップ 3: 最終の状態変更の確認

最終状態変更は、最後にクラスターが状態を変更した内容に関する情報を示します。通常、ここにはクラスターが状態を `FAILED` に変更するときに発生した問題に関する情報があります。例えば、ストリーミングクラスターを起動して、Simple Storage Service (Amazon S3) で既存の出力場所を指定

した場合、クラスターは「Streaming output directory already exists」という最終の状態変更で失敗します。

`list-steps` または `describe-cluster` 引数を使用して CLI から、または `DescribeCluster` および `ListSteps` アクションを使用して API からクラスターの詳細ペインを表示することにより、コンソールから最終状態変更値を検索できます。詳細については、「クラスターステータスと詳細の表示」を参照してください。

ステップ 4: ログファイルの検証

次のステップは、ログファイルを調べて、クラスターで発生した問題のエラーコードまたはその他の兆候を見つけることです。使用可能なログファイル、ファイルの場所、および表示方法の詳細については、「ログファイルを表示する」を参照してください。

何が起こったのかを調べるには、しばらく調査作業をすることが必要な場合があります。Hadoop は、クラスター内のさまざまなノードでタスクを試行してジョブの作業を実行します。Amazon EMR は、完了しない他のタスク試行をまず終了して、投機的なタスク試行を開始する可能性があります。これは、発生時にコントローラー、stderr、および syslog ログファイルに記録される重大なアクティビティを生成します。さらに、複数のタスク試行が同時に実行されますが、ログファイルは結果を直線的にしか表示できません。

クラスターの起動中のブートストラップアクションログで、エラーや予期しない設定の変更をまず調べてください。次に、ステップログを調べて、エラーのあるステップの一部として起動された Hadoop ジョブを特定します。Hadoop のジョブログを調べて、失敗したタスク試行を特定します。タスク試行ログには、タスク試行が失敗した原因に関する詳細が含まれます。

さまざまなログファイルを使用してクラスターのエラーを識別する方法について以降のセクションで説明します。

ブートストラップアクションログの確認

ブートストラップアクションは、起動時にクラスター上でスクリプトを実行します。通常、クラスターへの追加ソフトウェアのインストールや、デフォルト値からの設定の変更に使用されます。これらのログを確認すると、クラスターのセットアップ中に発生したエラーや、パフォーマンスに影響する可能性のある設定の変更に関するインサイトが得られる場合があります。

ステップログの確認

ステップログには 4 種類あります。

- コントローラー——ステップ実行の試行中に発生したエラーから発生する Amazon EMR (Amazon EMR) によって生成されたファイルが含まれます。ロード中にステップが失敗した場合は、このログでスタックトレースを見つけることができます。アプリケーションの読み込みまたはアクセスのエラーは、しばしばここに記載されます。マッパーファイルがないエラーについても同様です。
- stderr—ステップの処理中に発生したエラーメッセージが含まれます。アプリケーションの読み込みエラーは、しばしばここに記載されます。このログには、スタックトレースが含まれている場合があります。
- stdout—マッパーおよびリデューサーの実行可能ファイルによって生成されたステータスが含まれます。アプリケーションの読み込みエラーは、しばしばここに記載されます。このログには、アプリケーションのエラーメッセージが含まれている場合があります。
- syslog—Apache や Hadoop などの、Amazon 以外のソフトウェアからのログが含まれます。ストリーミングエラーは、しばしばここに記載されます。

stderr で明らかなエラーを確認してください。stderr にエラーの短いリストが表示されている場合、ステップはエラーがスローされてからすぐに停止しています。これは、多くの場合、クラスターで実行されていたマッパーおよびリデューサーアプリケーションのエラーが発生の原因です。

コントローラーと syslog の最後の行を調べて、エラーまたは障害の通知を確認します。特に「Job Failed」と表示されている場合、失敗したタスクに関する通知に従ってください。

タスク試行ログの確認

前のステップログの分析で 1 つ以上の失敗したタスクが見つかった場合は、対応するタスク試行のログを調べて、詳細なエラー情報を入手してください。

ステップ 5: クラスターのステップバイステップテスト

エラーの原因を追跡しようとする場合、クラスターを再起動し、ステップを個別にクラスターに送信するのが有用です。これによって、次のステップを処理する前に各ステップの結果を確認することができ、失敗したステップを修正および再実行できます。これには、一度だけ入力データを読み込むという利点もあります。

手順を追ってクラスターをテストするには

1. キープアライブおよび有効な終了保護を有効にして、新しいクラスターを起動します。キープアライブでは、保留中のすべてのステップを処理した後、継続してクラスターを実行します。終了保護では、クラスターがエラーのイベントでシャットダウンできないようにします。詳細につい

ては、「ステップ実行後に継続または終了するようにクラスターを設定する」および「終了保護の使用」を参照してください。

2. ステップをクラスターに送信します。詳細については、「クラスターへの作業の送信」を参照してください。
3. ステップが処理を完了すると、ステップログファイルのエラーを確認します。詳細については、「ステップ 4: ログファイルの検証」を参照してください。これらのログファイルを検索する最も簡単な方法は、マスターノードに接続し、ログファイルを表示します。ステップログファイルは、ステップがしばらく実行されるまで、または終了あるいは失敗するまで表示されません。
4. ステップがエラーなしで成功した場合、次のステップを実行します。エラーが発生した場合、ログファイルでエラーを確認します。ユーザーのコードでエラーがあった場合、修正してステップを再実行します。すべてのステップがエラーなしで実行できるまで続けます。
5. クラスターのデバッグを完了し、これを終了する場合、手動で終了する必要があります。クラスターは終了保護を有効にして起動されているため、これが必要になります。詳細については、「終了保護の使用」を参照してください。

低速なクラスターのトラブルシューティング

このセクションでは、実行中のままですが、結果が返るのに時間がかかっているクラスターをトラブルシューティングする手順を説明します。クラスターがエラーコードで終了した場合の対処方法については、「失敗したクラスターのトラブルシューティング」を参照してください。

Amazon EMR では、クラスター内のインスタンスの数と種類を指定できます。これを指定することが、データ処理の実行速度に影響を与える主な手段です。クラスターの再実行を検討する可能性があります。今回は、より大きなリソースを持つ EC2 インスタンスを指定するか、より多い数のインスタンスをクラスターに指定します。詳細については、「クラスターハードウェアとネットワークを設定する」を参照してください。

次のトピックでは、クラスターを遅くさせる他の要因を特定する手順を説明します。

トピック

- ステップ 1: 問題に関するデータの収集
- ステップ 2: 環境の確認
- ステップ 3: ログファイルの検証
- ステップ 4: クラスターとインスタンスのヘルスの確認
- ステップ 5: 中断されたグループの確認

- ステップ 6: 設定のレビュー
- ステップ 7: 入力データの検証

ステップ 1: 問題に関するデータの収集

クラスターのトラブルシューティングの最初の手順は、不具合とクラスターの現在のステータスおよび設定に関する情報を収集することです。この情報は、問題の原因を確認または除外するために、以降の手順で使用されます。

問題の定義

まず、問題を明確に定義します。自問するいくつかの質問を次に示します。

- 何が起こると予想したか 代わりに何が起こったか
- この問題が最初に発生したのはいつか それ以来どのくらいの頻度で起こっているか
- クラスターの設定方法や実行方法に変化があったか

クラスターの詳細

問題を追跡するために、次のクラスターの詳細が役立ちます。この情報の収集方法の詳細については、「クラスターステータスと詳細の表示」を参照してください。

- クラスターの識別子。(ジョブフロー識別子とも呼ばれます)。
- AWS リージョン およびクラスターが起動されたアベイラビリティーゾーン。
- クラスターの状態 (最後の状態変更の詳細を含む)。
- マスター、コア、およびタスクの各ノードに指定されている EC2 インスタンスの種類と数。

ステップ 2: 環境の確認

トピック

- サービスの停止の確認
- 使用制限の確認
- Amazon VPC サブネット設定の確認
- クラスターの再起動

サービスの停止の確認

Amazon EMR は、内部で複数のAmazon Web Services を使用します。Amazon EC2 で仮想サーバーを実行し、Amazon S3 にデータとスクリプトを保存し、メトリクスを にレポートします CloudWatch。これらのサービスを中断するイベントはまれですが、発生すると、Amazon EMR で問題が発生する恐れがあります。

次に進む前に、サービスヘルスダッシュボードを確認します。クラスターを起動したリージョンで、これらのサービスのいずれかに中断イベントがあるかどうかを確認します。

使用制限の確認

大規模なクラスターを起動している場合、多数のクラスターを同時に起動している場合、または を他のユーザー AWS アカウント と共有しているユーザーの場合、 AWS サービスの制限を超えたためクラスターが失敗した可能性があります。

Amazon EC2 は、1 つの AWS リージョンで実行されている仮想サーバーインスタンスの数を、オンデマンドインスタンスまたはリザーブドインスタンスの 20 個に制限します。20 を超えるノードを持つクラスターを起動するか、 でアクティブな EC2 インスタンスの合計数が 20 AWS アカウントを超えるクラスターを起動すると、クラスターは必要なすべての EC2 インスタンスを起動できず、失敗する可能性があります。この場合、Amazon EMR は `EC2 QUOTA EXCEEDED` エラーを返します。Amazon EC2 インスタンス制限 AWS の引き上げリクエストアプリケーションを送信することで、アカウントで実行できる EC2 インスタンスの数を増やすようにリクエストできます。 Amazon EC2

使用制限を超えるもう 1 つの原因は、クラスターが終了してからすべてのリソースを解放するまでの遅延です。設定によっては、1 つのクラスターが完全に終了して、割り当てられたリソースを解放するまでに 5 ~ 20 分かかることがあります。クラスターを起動しようとして `EC2 QUOTA EXCEEDED` エラーが発生する場合、そのエラーの原因として、最近終了したクラスターのリソースがまだ解放されていないことが考えられます。この場合は、Amazon EC2 クォータ増加リクエストを送信するか、20 分待ってからクラスターを再起動してください。

Simple Storage Service (Amazon S3) では、アカウントで作成されるバケットの数を 100 までに制限しています。クラスターがこの制限を超える新しいバケットを作成すると、バケットの作成が失敗し、クラスターが失敗する恐れがあります。

Amazon VPC サブネット設定の確認

クラスターが Amazon VPC サブネットで起動された場合は、「ネットワークを設定する」の説明に従ってサブネットを設定する必要があります。さらに、クラスターを起動するサブネットに存在する

空きの Elastic IP アドレスが、クラスター内の各ノードに 1 つずつ割り当てるのに十分であることを確認します。

クラスターの再起動

処理速度の低下は、一時的な条件で発生している可能性があります。クラスターの終了および再起動を検討し、パフォーマンスが改善されるか確認します。

ステップ 3: ログファイルの検証

次のステップは、ログファイルを調べて、クラスターで発生した問題のエラーコードまたはその他の兆候を見つけることです。使用可能なログファイル、ファイルの場所、および表示方法の詳細については、「ログファイルを表示する」を参照してください。

何が起こったのかを調べるには、しばらく調査作業をすることが必要な場合があります。Hadoop は、クラスター内のさまざまなノードでタスクを試行してジョブの作業を実行します。Amazon EMR は、完了しない他のタスク試行をまず終了して、投機的なタスク試行を開始する可能性があります。これは、発生時にコントローラー、stderr、および syslog ログファイルに記録される重大なアクティビティを生成します。さらに、複数のタスク試行が同時に実行されますが、ログファイルは結果を直線的にしか表示できません。

クラスターの起動中のブートストラップアクションログで、エラーや予期しない設定の変更をまず調べてください。次に、ステップログを調べて、エラーのあるステップの一部として起動された Hadoop ジョブを特定します。Hadoop のジョブログを調べて、失敗したタスク試行を特定します。タスク試行ログには、タスク試行が失敗した原因に関する詳細が含まれます。

さまざまなログファイルを使用してクラスターのエラーを識別する方法について以降のセクションで説明します。

ブートストラップアクションログの確認

ブートストラップアクションは、起動時にクラスター上でスクリプトを実行します。通常、クラスターへの追加ソフトウェアのインストールや、デフォルト値からの設定の変更に使用されます。これらのログを確認すると、クラスターのセットアップ中に発生したエラーや、パフォーマンスに影響する可能性のある設定の変更に関するインサイトが得られる場合があります。

ステップログの確認

ステップログには 4 種類あります。

- コントローラー——ステップ実行の試行中に発生したエラーから発生する Amazon EMR (Amazon EMR) によって生成されたファイルが含まれます。ロード中にステップが失敗した場合は、このログでスタックトレースを見つけることができます。アプリケーションの読み込みまたはアクセスのエラーは、しばしばここに記載されます。マッパーファイルがないエラーについても同様です。
- stderr—ステップの処理中に発生したエラーメッセージが含まれます。アプリケーションの読み込みエラーは、しばしばここに記載されます。このログには、スタックトレースが含まれている場合があります。
- stdout—マッパーおよびリデューサーの実行可能ファイルによって生成されたステータスが含まれます。アプリケーションの読み込みエラーは、しばしばここに記載されます。このログには、アプリケーションのエラーメッセージが含まれている場合があります。
- syslog—Apache や Hadoop などの、Amazon 以外のソフトウェアからのログが含まれます。ストリーミングエラーは、しばしばここに記載されます。

stderr で明らかなエラーを確認してください。stderr にエラーの短いリストが表示されている場合、ステップはエラーがスローされてからすぐに停止しています。これは、多くの場合、クラスターで実行されていたマッパーおよびリデューサーアプリケーションのエラーが発生の原因です。

コントローラーと syslog の最後の行を調べて、エラーまたは障害の通知を確認します。特に「Job Failed」と表示されている場合、失敗したタスクに関する通知に従ってください。

タスク試行ログの確認

前のステップログの分析で 1 つ以上の失敗したタスクが見つかった場合は、対応するタスク試行のログを調べて、詳細なエラー情報を入手してください。

Hadoop デーモンログの確認

まれに、Hadoop 自体が失敗する場合があります。その場合に該当するかどうかを確認するには、Hadoop ログを確認する必要があります。ログは各ノードの `/var/log/hadoop/` にあります。

JobTracker ログを使用して、失敗したタスク試行を、実行されたノードにマッピングできます。タスク試行に関連付けられているノードが判明したら、そのノードをホストしている EC2 インスタンスのヘルスを確認して、CPU やメモリ不足などの問題がないかどうかを確認できます。

ステップ 4: クラスターとインスタンスのヘルスの確認

Amazon EMR クラスターは、Amazon EC2 インスタンスで実行されているノードで構成されています。これらのインスタンスがリソースにバインドされた場合 (CPU またはメモリの不足などによ

り)、ネットワーク接続の問題が発生したり、ネットワークが切断されたりして、クラスターの処理速度が低下します。

クラスターには最大 3 種類のノードがあります。

- マスターノード — クラスターを管理します。パフォーマンスに問題が発生した場合、クラスター全体に影響があります。
- コアノード — マップリデュースタスクを処理し、Hadoop Distributed Filesystem (HDFS) を保守します。これらのノードの 1 つにパフォーマンスの問題がある場合、HDFS の操作とマップリデュースの処理を行うスピードを下げる可能性があります。さらにコアノードをクラスターに追加し、パフォーマンスを改善できますが、コアノードを削除することはできません。詳細については、「実行中のクラスターのサイズを手動で変更する」を参照してください。
- タスクノード — マップリデュースタスクを処理します。専用の演算リソースがあり、データは保存しません。タスクノードをクラスターに追加してパフォーマンスの速度を上げるか、必要のないタスクノードを削除することができます。詳細については、「実行中のクラスターのサイズを手動で変更する」を参照してください。

クラスターの状態を確認する場合、クラスター全体のパフォーマンスと、個々の仮想サーバーインスタンスのパフォーマンスの両方を確認する必要があります。次のようなツールを使用できます。

でクラスターの状態を確認する CloudWatch

すべての Amazon EMR クラスターは、メトリクスを にレポートします CloudWatch。メトリクスでは、合計読み込み、HDFS 使用率、実行中のタスク、残りのタスク、および破損ブロックなどのクラスターに関するパフォーマンス情報の概要を提供します。 CloudWatch メトリクスを見ると、クラスターで何が起こっているかについての全体像が得られ、処理が遅くなっている原因についてのインサイトが得られます。 CloudWatch を使用して既存のパフォーマンスの問題を分析するだけでなく、将来のパフォーマンスの問題が発生した場合 CloudWatch に がアラートするアラームを設定することもできます。詳細については、「を使用した Amazon EMR メトリクスのモニタリング CloudWatch」を参照してください。

ジョブのステータスと HDFS のヘルスの確認

クラスター詳細ページの [Application user interfaces (アプリケーションユーザーインターフェイス)] タブを使用して、YARN アプリケーションの詳細を表示できます。特定のアプリケーションでは、さらに詳細やアクセスログに直接アクセスすることができます。これは、Spark アプリケーションで特に有益です。詳細については、「アプリケーションの履歴を表示する」を参照してください。

Hadoop では、情報の表示に使用できる Web インターフェイスのシリーズを提供します。Web インターフェイスへのアクセス方法については、「Amazon EMR クラスターでホストされているウェブインターフェイスを表示する」を参照してください。

- JobTracker — クラスターによって処理されるジョブの進行状況に関する情報を提供します。このインターフェイスを使用して、ジョブが停止したタイミングを特定できます。
- HDFS NameNode — 各ノードの HDFS 使用率と使用可能な領域に関する情報を提供します。このインターフェイスを使用して、HDFS がリソースにバインドされ、追加の容量を必要とするタイミングを特定できます。
- TaskTracker — クラスターによって処理されるジョブのタスクに関する情報を提供します。このインターフェイスを使用して、タスクが停止したタイミングを特定できます。

Amazon EC2 でのインスタンスヘルスの確認

別の方法でクラスターのインスタンスのステータスに関する情報を確認するには、Amazon EC2 コンソールを使用します。クラスターの各ノードは EC2 インスタンスで実行されるため、Amazon EC2 から提供されるツールを使用してステータスを確認できます。詳細については、「Amazon EC2 でクラスターインスタンスを表示する」を参照してください。

ステップ 5: 中断されたグループの確認

ノードの起動を試行中にエラーが多数発生したとき、インスタンスグループは中断されます。例えば、ブートストラップアクションの実行中に新しいノードが繰り返し失敗した場合、インスタンスグループは、しばらくすると、新しいノードのプロビジョニングを引き続き試みることなく、SUSPENDED 状態になります。

たとえば、次のような場合に、ノードが表示されないことがあります。

- Hadoop またはクラスターが何らかの理由で破損し、クラスターへの新しいノードを受け入れない
- ブートストラップアクションが新しいノードで失敗した
- ノードが適切に機能していないため、Hadoop でチェックインできない

インスタンスグループが SUSPENDED 状態で、クラスターが WAITING 状態の場合は、クラスターステップを追加して、必要な数のコアおよびタスクノードをリセットできます。ステップを追加することで、クラスターの処理が再開し、インスタンスグループが RUNNING 状態に戻ります。

中断状態のクラスターをリセットする方法については、「停止状態」を参照してください。

ステップ 6: 設定のレビュー

構成設定では、タスクを再試行する回数およびソートに利用できるメモリ量など、クラスターの実行方法に関する詳細を指定します。Amazon EMR を使用してクラスターを起動すると、標準の Hadoop 設定に加えて、Amazon EMR 固有の設定があります。構成設定はクラスターのマスターノードに保存されます。構成設定を確認し、効率的に実行するのに必要なリソースがクラスターにあるようにできます。

Amazon EMR は、クラスターの起動に使用されるデフォルトの Hadoop 設定を定義します。この値は、クラスターに指定した AMI およびインスタンスに基づいています。ブートストラップアクションを使用したデフォルトの値から、またはジョブ実行パラメータに新しい値を指定することによって、構成設定を変更できます。詳細については、「追加のソフトウェアをインストールするためのブートストラップアクションの作成」を参照してください。ブートストラップアクションで構成設定を変更したかどうかを判定するには、ブートストラップアクションログを確認します。

Amazon EMR は、各ジョブの実行に使用された Hadoop 設定をログに記録します。ログデータは、マスターノードの `/mnt/var/log/hadoop/history/` ディレクトリに `job_`*`job-id`*`_conf.xml` という名前のファイルで存されます。ここで *`job-id`* はジョブの識別子で置換されます。ログのアーカイブを有効にしている場合、このデータは Simple Storage Service (Amazon S3) の `logs/`*`date`*`/`*`jobflow-id`*`/jobs` フォルダーにコピーされます。ここで *`date`* はジョブが実行された日付、*`jobflow-id`* はクラスターの識別子です。

次の Hadoop ジョブ構成設定は、パフォーマンスの問題を調査するのに特に役立ちます。Hadoop の構成設定および Hadoop の動作にどのように影響するかについては、http://hadoop.apache.org/docs/ を参照してください。

> **Warning**
>
> 1. ノードが 4 つ未満のクラスターで `dfs.replication` を 1 に設定すると、単一ノードがダウンした場合に HDFS データが失われる可能性があります。本番環境のワークロードには、少なくとも 4 つのコアノードを持つクラスターを使用することをお勧めします。
> 2. Amazon EMR では、クラスターはコアノードを `dfs.replication` 未満にスケールすることはできません。例えば、`dfs.replication = 2` の場合、コアノードの最小数は 2 です。
> 3. マネージドスケーリングや自動スケーリングを使用する場合や、クラスターのサイズを手動で変更する場合は、`dfs.replication` を 2 以上に設定することをお勧めします。

構成設定	説明
dfs.replication	RAID のような環境を生成するために、単一ブロック（ハードドライブブロックなど）がコピーされる HDFS ノードの数。ブロックのコピーを含んでいる HDFS のノード数を決定します。
io.sort.mb	ソートに利用可能な合計メモリ。この値は io.sort.factor の 10 倍になります。この設定は、io.sort.mb に mapred.tasktracker.ap.tasks.maximum を掛けて計算して、タスクノードが使用する合計メモリを計算するためにも使用できます。
io.sort.spill.percent	割り当てられたソートメモリがいっぱいであるため、ディスクが使用を開始するポイントでソート中に使用されます。
mapred.child.java.opts	廃止済み。代わりに、mapred.map.child.java.opts および mapred.reduce.child.java.opts を使用します。Java オプションは、タスクを実行する JVM を起動するときに TaskTracker を使用します。共通パラメータは、最大メモリサイズの設定用の "-Xmx" です。
mapred.map.child.java.opts	Java オプションは、マップタスクを実行する JVM を起動するときに TaskTracker を使用します。共通パラメータは、最大メモリヒープサイズの設定用の "-Xmx" です。
mapred.map.tasks.speculative.execution	同じタスクのマップタスク試行を並行して起動できるかどうかを決定します。
mapred.reduce.tasks.speculative.execution	同じタスクのリデュースタスク試行を並行して起動できるかどうかを決定します。
mapred.map.max.attempts	マップタスクの最大試行回数。すべてが失敗した場合、マップタスクは失敗としてマークされます。
mapred.reduce.child.java.opts	Java オプションは、削減タスクを実行するために JVM を起動するときに TaskTracker を使用します。共通パラ

構成設定	説明
	メータは、最大メモリヒープサイズの設定用の "-Xmx" です。
mapred.reduce.max.attempts	Reduce タスクの最大試行回数。すべてが失敗した場合、マップタスクは失敗としてマークされます。
mapred.reduce.slowstart.completed.maps	Reduce タスクが試行される前に完了する必要のある Map タスクの量。待機時間が足りないと、試行中に「Too many fetch-failure」エラーが発生する場合があります。
mapred.reuse.jvm.num.tasks	タスクは単一の JVM 内で実行されます。同じ JVM を再利用できるタスク数を指定します。
mapred.tasktracker.map.tasks.maximum	マッピング中のタスクノードごとに並行して実行できる最大タスク数。
mapred.tasktracker.reduce.tasks.maximum	減らしている間のタスクノードごとに並行して実行できる最大タスク数。

クラスタータスクがメモリを大量に使用する場合、コアノードごとに使うタスクの数を減らしてジョブトラッカーのヒープサイズを減らすと、パフォーマンスを向上させることができます。

ステップ 7: 入力データの検証

入力データを確認します。キー値に均等に割り付けられていますか? データが 1 つまたは少数のキー値に偏っている場合、他のノードが待機中であるにもかかわらず、読み込み処理は少数のノードにマップされている可能性があります。この不均等な作業の割り付けは、処理時間を遅くさせる場合があります。

たとえば、不均等なデータセットでは、クラスターを実行して単語をアルファベット順にしていますが、所有しているデータセットには "a" の文字で始まる単語しかありません。作業を綿密に計画した場合、他の文字で始まる単語を処理するノードが待機中であっても、"a" で始まる値を処理しているノードに負担がかかることになります。

Lake Formation クラスターのトラブルシューティング

このセクションでは、 AWS Lake Formationで Amazon EMR を使用するときの一般的な問題をトラブルシューティングするプロセスについて説明します。

データレイクアクセスが許可されない

データレイク内のデータを分析して処理するには、Amazon EMR クラスターのデータフィルタリングを明示的にオプトインする必要があります。データアクセスに失敗すると、ノートブックエントリの出力に汎用的な `Access is not allowed` メッセージが表示されます。

Amazon EMR でオプトインしてデータフィルタリングを許可するには、「AWS Lake Formation デベロッパーガイド」の「Amazon EMR でのデータフィルタリングを許可する」の手順を参照してください。

セッションの期限切れ

EMR Notebooks と Zeppelin のセッションタイムアウトは、Lake Formation の `Maximum CLI/API session duration` 設定の IAM ロールによって制御されます。この設定のデフォルト値は 1 時間です。セッションタイムアウトが発生すると、Spark SQL コマンドを実行しようとしたときに、ノートブックエントリの出力に次のメッセージが表示されます。

```
Error 401    HTTP ERROR: 401 Problem accessing /sessions/2/statements.
Reason:  JWT token included in request failed validation.
Powered by Jetty:// 9.3.24.v20180605
 org.springframework.web.client.HttpClientErrorException: 401 JWT token included in
 request failed validation…
```

セッションを検証するには、ページを更新します。IdP を使用して再認証するよう求められ、ノートブックにリダイレクトされます。再認証後、クエリを実行することができます。

リクエストされたテーブルに対するユーザーのアクセス許可がない

アクセス許可のないテーブルにアクセスしようとすると、Spark SQL コマンドを実行しようとしたときに、ノートブックエントリの出力に次の例外が表示されます。

```
org.apache.spark.sql.AnalysisException:
 org.apache.hadoop.hive.ql.metadata.HiveException: Unable to fetch table table.
Resource does not exist or requester is not authorized to access requested
 permissions.
```

```
(Service: AWSGlue; Status Code: 400; Error Code: AccessDeniedException; Request ID: …
```

テーブルにアクセスするには、Lake Formation で、このテーブルに関連付けられているアクセス許可を更新して、ユーザーにアクセス許可を付与する必要があります。

アカウント間で共有する Lake Formation データをクエリする

Amazon EMR を使用して他のアカウントから共有されているデータにアクセスすると、一部の Spark ライブラリは `Glue:GetUserDefinedFunctions` API オペレーションの呼び出しを試みます。 AWS RAM 管理アクセス許可のバージョン 1 および 2 はこのアクションをサポートしていないため、次のエラーメッセージが表示されます。

```
"ERROR: User: arn:aws:sts::012345678901:assumed-role/my-
spark-role/i-06ab8c2b59299508a is not authorized to perform:
glue:GetUserDefinedFunctions on resource: arn:exampleCatalogResource
because no resource-based policy allows the glue:GetUserDefinedFunctions
action"
```

このエラーを解決するには、リソース共有を作成したデータレイク管理者が、リソース共有にアタッチされた AWS RAM 管理アクセス許可を更新する必要があります。 AWS RAM マネージドアクセス許可のバージョン 3 では、プリンシパルが `glue:GetUserDefinedFunctions` アクションを実行できます。

新しいリソース共有を作成すると、Lake Formation はデフォルトで最新バージョンの AWS RAM 管理アクセス許可を適用し、ユーザーによるアクションは必要ありません。既存のリソース共有のクロスアカウントデータアクセスを有効にするには、 AWS RAM マネージドアクセス許可をバージョン 3 に更新する必要があります。

で共有されているリソースに割り当てられた AWS RAM アクセス許可を表示できます AWS RAM。バージョン 3 には次のアクセス許可が含まれています。

```
Databases
  AWSRAMPermissionGlueDatabaseReadWriteForCatalog
  AWSRAMPermissionGlueDatabaseReadWrite

Tables
  AWSRAMPermissionGlueTableReadWriteForCatalog
  AWSRAMPermissionGlueTableReadWriteForDatabase

AllTables
```

```
  AWSRAMPermissionGlueAllTablesReadWriteForCatalog
  AWSRAMPermissionGlueAllTablesReadWriteForDatabase
```

既存のリソース共有の AWS RAM マネージドアクセス許可バージョンを更新するには

ユーザー (データレイク管理者) は、AWS RAM 「 ユーザーガイド」の手順に従ってAWS RAM 管理アクセス許可を新しいバージョンに更新することも、リソースタイプの既存のアクセス許可をすべて取り消して再付与することもできます。アクセス許可を取り消すと、 は AWS RAM リソースタイプに関連付けられたリソース共有 AWS RAM を削除します。アクセス許可を再付与すると、 AWS RAM は最新バージョンの AWS RAM マネージドアクセス許可をアタッチした新しいリソース共有を作成します。

テーブルで挿入、作成、変更の操作を行う

Lake Formation ポリシーによって保護されているデータベースのテーブルについて挿入、作成、または変更はサポートされていません。これらのオペレーションを実行すると、Spark SQL コマンドを実行しようとしたときに、ノートブックエントリの出力に次の例外が表示されます。

```
java.io.IOException:
 com.amazon.ws.emr.hadoop.fs.shaded.com.amazonaws.services.s3.model.AmazonS3Exception:
            Access Denied (Service: Amazon S3; Status Code: 403; Error Code:
 AccessDenied; Request ID: …
```

詳細については、「Amazon EMR と の統合の制限 AWS Lake Formation事項」を参照してください。

クラスターを起動し管理するアプリケーションの作成

トピック

いずれかの AWS SDKs でラッパー関数を呼び出すことで、Amazon EMR API が提供する機能にアクセスできます。 AWS SDKs、ウェブサービスの API をラップし、ウェブサービスへの接続を簡素化する言語固有の関数を提供し、接続の詳細の多くを処理します。いずれかの SDK を使用して Amazon EMR を呼び出すことの詳細については、「SDK を使用して Amazon EMR API を呼び出す」を参照してください。

> **Important**
>
> Amazon EMR の最大リクエストレートは、10 秒に 1 件です。

E nd-to-end Amazon EMR Java ソースコードサンプル

開発者は、カスタム Java コードを使用して Amazon EMR API を呼び出し、Amazon EMR コンソールまたは CLI で可能なことと同じことを実行できます。このセクションでは、 をインストール AWS Toolkit for Eclipse し、Amazon EMR クラスターに end-to-end ステップを追加する、完全に機能する Java ソースコードサンプルを実行するために必要なステップについて説明します。

> **Note**
>
> この例では Java に焦点を当てますが、Amazon EMR も Amazon EMR SDK のコレクションにより複数のプログラミング言語をサポートします。詳細については、「SDK を使用して Amazon EMR API を呼び出す」を参照してください。

この Java ソースコード例は、Amazon EMR API を使用して以下のタスクを実行する方法を示しています。

- AWS 認証情報を取得し、Amazon EMR に送信して API コールを行う
- 新しいカスタムステップと事前定義されたステップを設定する
- 既存の Amazon EMR クラスターに新しいステップを追加する
- 実行中のクラスターからクラスターステップ ID を取得する

> **Note**
>
> このサンプルは、既存のクラスターにステップを追加する方法を示すので、アカウントにアクティブなクラスターがある必要があります。

開始する前に、コンピューターのプラットフォームに適合するバージョンの Eclipse IDE for Java EE Developers をインストールします。詳細については、「Eclipse のダウンロード」を参照してください。

次に、Eclipse 用 Database Development プラグインをインストールします。

Database Development Eclipse プラグインをインストールする

1. Eclipse IDE を開きます。
2. [Help (ヘルプ)] を選択し、[Install New Software (新しいソフトウェアのインストール)] をクリックします。
3. [Work with: (使用場所:)] フィールドに「**`http://download.eclipse.org/releases/kepler`**」または Eclipse IDE のバージョン番号と一致するパスを入力します。
4. 項目リストで、[Database Development (データベースの開発)] を選択し、[Finish (完了)] をクリックします。
5. 指示が表示されたら、Eclipse を再起動します。

次に、Toolkit for Eclipse をインストールして、役立つように事前設定されたソースコードプロジェクトテンプレートを利用できるようにします。

Toolkit for Eclipse をインストールする

1. Eclipse IDE を開きます。
2. [Help (ヘルプ)] を選択し、[Install New Software (新しいソフトウェアのインストール)] をクリックします。

3. [Work with:] フィールドに「**https://aws.amazon.com/eclipse**」と入力します。
4. アイテムリストで、[AWS Toolkit for Eclipse] を選択し、[完了] を選択します。
5. 指示が表示されたら、Eclipse を再起動します。

次に、新しい AWS Java プロジェクトを作成し、サンプルの Java ソースコードを実行します。

新しい AWS Java プロジェクトを作成するには

1. Eclipse IDE を開きます。
2. [File (ファイル)]、[New (新規)]、[Other (その他)] の順に選択します。
3. [ウィザードの選択] ダイアログで、[AWS Java プロジェクト] を選択し、[次へ] をクリックします。
4. 「新しい AWS Java プロジェクト」ダイアログの **Project name:** フィールドに、新しいプロジェクトの名前を入力します。例えば、「」です**EMR-sample-code**。
5. AWS 「アカウントの設定」を選択し、パブリックアクセスキーとプライベートアクセスキーを入力し、「 の終了」を選択します。アクセスキーの作成の詳細については、「Amazon Web Services 全般のリファレンスガイド」の「セキュリティ認証情報の取得方法」を参照してください。

   > **Note**
   >
   > コードに直接アクセスキーを埋め込むことはできません。Amazon EMR SDK では、既知のロケーションにアクセスキーを配置できるため、コードで保持する必要はありません。

6. 新しい Java プロジェクトで、src フォルダを右クリックし、[New (新規)]、[Class (クラス)] の順にクリックします。
7. [Java Class (Java クラス)] ダイアログの [Name (名前)] フィールドに新しいクラスの名前を入力します (例: **main**)。
8. [Which method stubs would you like to create? (どのメソッドスタブを作成しますか?)] セクションで、[public static void main(String[]args)] を選択し、[Finish (完了)] をクリックします。
9. 新しいクラス内に Java ソースコードを入力し、サンプルのクラスと方法に適切な import ステートメントを追加します。参考までに、完全なソースコードのリストを下に示します。

> **Note**
>
> 次のサンプルコードで、サンプルクラスター ID (JobFlowId) 、、 を*j-xxxxxxxxxxxx*、 AWS Management Console または次の AWS CLI コマンドを使用して、 にあるアカウントの有効なクラスター ID に置き換えます。
>
> ```
> aws emr list-clusters --active | grep "Id"
> ```
>
> また、例の Amazon S3 パス (*s3://path/to/my/jarfolder*) を JAR への有効なパスで置き換えます。最後に、例のクラス名 (*com.my.Main1*) を JAR にあるクラスの正しい名前で置き換えます (該当する場合)。

```
import com.amazonaws.AmazonClientException;
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduce;
import com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduceClientBuilder;
import com.amazonaws.services.elasticmapreduce.model.*;
import com.amazonaws.services.elasticmapreduce.util.StepFactory;

public class Main {

 public static void main(String[] args) {
  AWSCredentials credentials_profile = null;
  try {
   credentials_profile = new
 ProfileCredentialsProvider("default").getCredentials();
  } catch (Exception e) {
   throw new AmazonClientException(
     "Cannot load credentials from .aws/credentials file. " +
       "Make sure that the credentials file exists and the profile name is
 specified within it.",
     e);
  }

  AmazonElasticMapReduce emr = AmazonElasticMapReduceClientBuilder.standard()
    .withCredentials(new AWSStaticCredentialsProvider(credentials_profile))
    .withRegion(Regions.US_WEST_1)
```

```
    .build();

  // Run a bash script using a predefined step in the StepFactory helper class
  StepFactory stepFactory = new StepFactory();
  StepConfig runBashScript = new StepConfig()
    .withName("Run a bash script")
    .withHadoopJarStep(stepFactory.newScriptRunnerStep("s3://jeffgoll/emr-scripts/
create_users.sh"))
    .withActionOnFailure("CONTINUE");

  // Run a custom jar file as a step
  HadoopJarStepConfig hadoopConfig1 = new HadoopJarStepConfig()
    .withJar("s3://path/to/my/jarfolder") // replace with the location of the jar
 to run as a step
    .withMainClass("com.my.Main1") // optional main class, this can be omitted if
 jar above has a manifest
    .withArgs("--verbose"); // optional list of arguments to pass to the jar
  StepConfig myCustomJarStep = new StepConfig("RunHadoopJar", hadoopConfig1);

  AddJobFlowStepsResult result = emr.addJobFlowSteps(new AddJobFlowStepsRequest()
    .withJobFlowId("j-xxxxxxxxxxxx") // replace with cluster id to run the steps
    .withSteps(runBashScript, myCustomJarStep));

  System.out.println(result.getStepIds());

 }
}
```

10. [Run (実行)]、[Run As (実行)]、[Java Application (Java アプリケーション)] の順にクリックします。
11. サンプルが正しく実行される場合、新しいステップの ID のリストが Eclipse IDE コンソールウィンドウに表示されます。正しい出力は次の例のようになります。

```
[s-39BLQZRJB2E5E, s-1L6A4ZU2SAURC]
```

API コールの一般的な考え方

トピック

- Amazon EMR におけるエンドポイント
- Amazon EMR でのクラスターパラメータの指定

- Amazon EMR のアベイラビリティーゾーン
- Amazon EMR クラスターで追加のファイルおよびライブラリを使用する方法

Amazon EMR API を呼び出すアプリケーションを作成する際、SDK のラッパー関数のいずれかを呼び出す場合に適用されるいくつかの概念があります。

Amazon EMR におけるエンドポイント

エンドポイントは、ウェブサービスのエントリポイントとなるURLです。ウェブサービスの各リクエストには、1 つずつエンドポイントが含まれている必要があります。エンドポイントは、クラスターを作成、説明、または終了する AWS リージョンを指定します。これは、`elasticmapreduce.regionname.amazonaws.com` という形式です。全般的なエンドポイント (`elasticmapreduce.amazonaws.com`) を指定すると、Amazon EMR はリクエストをデフォルトのリージョンのエンドポイントに送信します。2013 年 3 月 8 日以降に作成されたアカウントの場合、デフォルトのリージョンは us-west-2; です。それ以前のアカウントの場合、デフォルトのリージョンは us-east-1 です。

Amazon EMR のエンドポイントの詳細については、「Amazon Web Services 全般のリファレンス」の「リージョンとエンドポイント」を参照してください。

Amazon EMR でのクラスターパラメータの指定

`Instances` パラメータを使用すると、データを処理するために作成する EC2 インスタンスのタイプと数を設定できます。Hadoop は、データの処理を複数のクラスターノードに分配します。マスターノードには、コアノードおよびタスクノードが正常であるかどうかを追跡することと、ジョブの結果ステータスをノードに問い合わせるという責任があります。コアノードとタスクノードが実際のデータ処理を実行します。クラスターが単一のノードで処理される場合は、そのノードがマスターノードとコアノードの両方として機能します。

`KeepJobAlive` リクエストにおける `RunJobFlow` パラメータは、実行するクラスターステップがなくなったクラスターを終了するかどうかを決定します。クラスターが予定どおりに実行されていることが確かである場合は、この値を `False` に設定します。ジョブフローをトラブルシューティングしていて、クラスターの実行が中断されている間にステップを追加している場合は、この値を `True` に設定します。これにより、ステップを修正した後でクラスターを再開するために繰り返されるだけの処理である、Amazon Simple Storage Service (Amazon S3) への結果のアップロードにかかる時間やコストを減らすことができます。

KeepJobAlive が の場合true、クラスターが正常に作業を完了した後、TerminateJobFlowsリクエストを送信する必要があります。送信しないと、クラスターは引き続き実行され、 AWS 料金が生成されます。

に固有のパラメータの詳細については、RunJobFlow「」を参照してくださいRunJobFlow。リクエストの一般的なパラメータの詳細については、「リクエストの一般的なパラメータ」を参照してください。

Amazon EMR のアベイラビリティーゾーン

Amazon EMR は、クラスターを処理するノードとして EC2 インスタンスを使用します。この EC2 インスタンスには、アベイラビリティーゾーンとリージョンから構成される場所が含まれます。リージョンは、独立した地理的領域に分散して存在します。アベイラビリティーゾーンはリージョン内の特定の場所であり、他のアベイラビリティーゾーン内で障害が発生しても影響を受けません。各アベイラビリティーゾーンには、同一リージョン内の他のアベイラビリティーゾーンとの間に低コストでレイテンシーの少ないネットワーク接続が用意されています。Amazon EMR のリージョンとエンドポイントの一覧については、「Amazon Web Services 全般のリファレンス」の「リージョンとエンドポイント」を参照してください。

AvailabilityZone パラメータは、クラスターのおおよその場所を指定します。このパラメータはオプションであり、原則として使用しないことをお勧めします。Amazon EMR で AvailabilityZone を指定しないと、クラスターにとって最適な AvailabilityZone 値が自動的に選択されます。このパラメータは、インスタンスを既存の実行中のインスタンスやクラスターと同じ場所に置いて、それらのインスタンスのデータを読み書きする必要がある場合、役立つことがあります。詳細については、Amazon EC2 ユーザーガイド」を参照してください。

Amazon EMR クラスターで追加のファイルおよびライブラリを使用する方法

マッパーまたはリデューサーアプリケーションで、追加のファイルや独自のライブラリを使用したいことがあります。たとえば、PDF ファイルをプレーンテキストに変換するライブラリを使用する場合が考えられます。

Hadoop ストリーミングの使用中にマッパーまたはリデューサー向けのファイルをキャッシュするには

- JAR args フィールドで、以下の引数を追加します。

```
-cacheFile s3://bucket/path_to_executable#local_path
```

`local_path` ファイルは、ファイルを参照できるマッパーの作業ディレクトリにあります。

SDK を使用して Amazon EMR API を呼び出す

トピック

- AWS SDK for Java を使用して Amazon EMR クラスターを作成する

AWS SDKs は、API をラップし、署名の計算、リクエストの再試行処理、エラー処理など、接続の詳細の多くを処理する関数を提供します。SDKs には、 を呼び出すアプリケーションの記述を開始するのに役立つサンプルコード、チュートリアル、その他のリソースも含まれています AWS。SDK でラッパー関数を呼び出すと、 AWS アプリケーションの記述プロセスが大幅に簡素化されます。

AWS SDKs用ツール」のSDKs」を参照してください。

AWS SDK for Java を使用して Amazon EMR クラスターを作成する

AWS SDK for Java には、Amazon EMR 機能を備えた 3 つのパッケージが用意されています。

- com.amazonaws.services.elasticmapreduce
- com.amazonaws.services.elasticmapreduce.model
- com.amazonaws.services.elasticmapreduce.util

これらのパッケージの詳細については、「AWS SDK for Java API リファレンス」を参照してください。

次の例では、SDK で Amazon EMR を使用してプログラミングを簡素化する方法について説明します。このコード例では、一般的な Amazon EMR ステップタイプを作成するためのヘルパークラスである `StepFactory` オブジェクトを使用して、デバッグを有効にした状態でインタラクティブな Hive クラスターを作成します。

```
import com.amazonaws.AmazonClientException;
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduce;
import com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduceClientBuilder;
import com.amazonaws.services.elasticmapreduce.model.*;
```

```
import com.amazonaws.services.elasticmapreduce.util.StepFactory;

public class Main {

 public static void main(String[] args) {
  AWSCredentialsProvider profile = null;
  try {
   credentials_profile = new ProfileCredentialsProvider("default"); // specifies any
 named profile in
                                    // .aws/credentials as the credentials provider
  } catch (Exception e) {
   throw new AmazonClientException(
     "Cannot load credentials from .aws/credentials file. " +
       "Make sure that the credentials file exists and that the profile name is defined
 within it.",
     e);
  }

  // create an EMR client using the credentials and region specified in order to
  // create the cluster
  AmazonElasticMapReduce emr = AmazonElasticMapReduceClientBuilder.standard()
    .withCredentials(credentials_profile)
    .withRegion(Regions.US_WEST_1)
    .build();

  // create a step to enable debugging in the AWS Management Console
  StepFactory stepFactory = new StepFactory();
  StepConfig enabledebugging = new StepConfig()
    .withName("Enable debugging")
    .withActionOnFailure("TERMINATE_JOB_FLOW")
    .withHadoopJarStep(stepFactory.newEnableDebuggingStep());

  // specify applications to be installed and configured when EMR creates the
  // cluster
  Application hive = new Application().withName("Hive");
  Application spark = new Application().withName("Spark");
  Application ganglia = new Application().withName("Ganglia");
  Application zeppelin = new Application().withName("Zeppelin");

  // create the cluster
  RunJobFlowRequest request = new RunJobFlowRequest()
    .withName("MyClusterCreatedFromJava")
    .withReleaseLabel("emr-5.20.0") // specifies the EMR release version label, we
 recommend the latest release
```

```
    .withSteps(enabledebugging)
    .withApplications(hive, spark, ganglia, zeppelin)
    .withLogUri("s3://path/to/my/emr/logs") // a URI in S3 for log files is required
 when debugging is enabled
    .withServiceRole("EMR_DefaultRole") // replace the default with a custom IAM
 service role if one is used
    .withJobFlowRole("EMR_EC2_DefaultRole") // replace the default with a custom EMR
 role for the EC2 instance
                        // profile if one is used
    .withInstances(new JobFlowInstancesConfig()
      .withEc2SubnetId("subnet-12ab34c56")
      .withEc2KeyName("myEc2Key")
      .withInstanceCount(3)
      .withKeepJobFlowAliveWhenNoSteps(true)
      .withMasterInstanceType("m4.large")
      .withSlaveInstanceType("m4.large"));

  RunJobFlowResult result = emr.runJobFlow(request);
  System.out.println("The cluster ID is " + result.toString());

 }

}
```

少なくとも、DefaultRoleそれぞれ EMR_ と EMR_EC2_ に対応するサービスロールDefaultRole とジョブフローロールを渡す必要があります。これを行うには、同じアカウントに対してこの AWS CLI コマンドを呼び出します。最初に、ロールが既に存在するかどうかを確認します。

```
aws iam list-roles | grep EMR
```

インスタンスプロファイル (EMR_EC2_DefaultRole) とサービスロール (EMR_DefaultRole) の両方が存在する場合に表示されます。

```
"RoleName": "EMR_DefaultRole",
            "Arn": "arn:aws:iam::AccountID:role/EMR_DefaultRole"
            "RoleName": "EMR_EC2_DefaultRole",
            "Arn": "arn:aws:iam::AccountID:role/EMR_EC2_DefaultRole"
```

デフォルトのロールが存在しない場合は、次のコマンドを使用して作成できます。

```
aws emr create-default-roles
```

Amazon EMR Service Quotas を管理する

トピック

このセクションのトピックでは、EMR サービスクォータ (以前はサービス制限と呼ばれていました) 、 でそれらを管理する方法 AWS Management Console、クラスターのモニタリングとアクションのトリガーにサービスクォータの代わりに CloudWatch イベントを使用する方が有利な場合について説明します。

Amazon EMR Service Quotas とは

AWS アカウントには、各サービスの制限とも呼ばれるデフォルトの AWS サービスクォータがあります。EMR サービスには、次の 2 種類の制限があります。

- リソースの制限 - EMR を使用して EC2 リソースを作成できます。ただし、これらの EC2 リソースは Service Quotas の対象となります。このカテゴリのリソース制限は次のとおりです。
  - 同時に実行できるアクティブなクラスターの最大数。
  - インスタンスグループあたりのアクティブなインスタンスの最大数。
- API の制限 - EMR API を使用する場合、次の 2 種類の制限があります。
  - バースト制限 — これは、一度に実行できる API コールの最大数です。例えば、1 秒あたりに実行できる AddInstanceFleet API リクエストの最大数は、デフォルトとして 5 コール/秒に設定されています。これは、 AddInstanceFleet API のバースト制限が 5 コール/秒であるか、任意の時点で最大 5 回の AddInstanceFleet API コールを実行できることを意味します。ただし、バースト制限を使用した後で、それ以降の呼び出しはレート制限によって制限されます。
  - レート制限 — これは、API のバーストキャパシティの補充レートです。例えば、AddInstanceFleet 呼び出しの補充レートは、デフォルトとして 0.5 コール/秒に設定されています。つまり、バースト制限に達した後、API コールを行うには、少なくとも 2 秒 (0.5 コール/秒 X 2 秒 = 1 コール) 待つ必要があります。それよりも前にコールを行うと、EMR ウェブサービスによってスロットリングされます。どの時点でも、スロットリングなしでは、バーストキャパシティと同じ数のコールしか行うことができません。1 秒待つごとに、バースト制限である最大制限 5 に達するまで、バーストキャパシティは 0.5 コール増加します。

Amazon EMR Service Quotas の管理方法

Service Quotas は、、 AWS Management Console API、または CLI を使用して、Amazon EMR サービスクォータまたは制限を一元的な場所から表示および管理するために使用できる AWS 機能です。クォータの表示および引き上げのリクエストの詳細については、「Amazon Web Services 全般のリファレンス」の「AWS Service Quotas」を参照してください。

一部の APIs では、サービスクォータを増やすよりも CloudWatch イベントを設定する方が適している場合があります。 CloudWatch を使用してアラームを設定し、サービスクォータに達する前に増加リクエストを事前にトリガーすることで、時間を節約することもできます。詳細については、「で EMR イベントをセットアップするタイミング CloudWatch」を参照してください。

で EMR イベントをセットアップするタイミング CloudWatch

、 DescribeCluster、 などの一部のポーリング APIs では DescribeStep、 CloudWatch イベントを設定すると ListClusters、変更への応答時間が短縮され、サービスクォータが解放されます。例えば、ステップの完了時やクラスターの終了時など、クラスターの状態が変化したときに実行するように Lambda 関数を設定している場合、次のポーリングを待つ代わりに、そのトリガーを使用してワークフロー内の次のアクションを開始できます。そうではなく、専用の Amazon EC2 インスタンスまたは Lambda 関数が常に EMR API で変更をポーリングする場合は、コンピューティングリソースを無駄にするだけでなく、サービスクォータに達する可能性もあります。

以下に、イベント駆動型アーキテクチャに移行するとメリットが得られるケースをいくつか示します。

ケース 1: ステップ完了のための DescribeCluster API コールを使用して EMR をポーリングする

Example ステップ完了のための DescribeCluster API コールを使用した EMR のポーリング

一般的なパターンは、実行中のクラスターにステップを送信し、通常は DescribeCluster または DescribeStep APIs。このタスクは、Amazon EMR ステップ状態の変更イベントにフックすることで、最小限の遅延で実行することもできます。

このイベントには、ペイロードに次の情報が含まれます。

```
{
  "version": "0",
  "id": "999cccaa-eaaa-0000-1111-123456789012",
```

```
  "detail-type": "EMR Step Status Change",
  "source": "aws.emr",
  "account": "123456789012",
  "time": "2016-12-16T20:53:09Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
    "severity": "ERROR",
    "actionOnFailure": "CONTINUE",
    "stepId": "s-ZYXWVUTSRQPON",
    "name": "CustomJAR",
    "clusterId": "j-123456789ABCD",
    "state": "FAILED",
    "message": "Step s-ZYXWVUTSRQPON (CustomJAR) in Amazon EMR cluster j-123456789ABCD
 (Development Cluster) failed at 2016-12-16 20:53 UTC."
  }
}
```

詳細マップでは、Lambda 関数が「state」、「stepId」、または「clusterId」を解析して、関連する情報を検出できます。

ケース 2: EMR でワークフローを実行するために利用可能なクラスターをポーリングする

Example EMR でワークフローを実行するために利用可能なクラスターをポーリングする

複数のクラスターを実行するお客様のパターンは、クラスターが利用可能になったらすぐに、それらのクラスターでワークフローを実行することです。実行中のクラスターが多数あり、待機中のクラスターでワークフローを実行する必要がある場合、パターンとして、 DescribeCluster または ListClusters API コールを使用して EMR をポーリングして使用可能なクラスターを呼び出すことができます。クラスターでステップの準備が整ったことを認識するまでの遅延を減らす別の方法は、Amazon EMR クラスター状態の変更イベントを処理することになります。

このイベントには、ペイロードに次の情報が含まれます。

```
{
  "version": "0",
  "id": "999cccaa-eaaa-0000-1111-123456789012",
  "detail-type": "EMR Cluster State Change",
  "source": "aws.emr",
  "account": "123456789012",
```

```
  "time": "2016-12-16T20:43:05Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
    "severity": "INFO",
    "stateChangeReason": "{\"code\":\"\"}",
    "name": "Development Cluster",
    "clusterId": "j-123456789ABCD",
    "state": "WAITING",
    "message": "Amazon EMR cluster j-123456789ABCD ..."
  }
}
```

このイベントでは、ステータスが WAITING に変化したらすぐに待機中のワークフローをクラスターに送信するように Lambda 関数を設定できます。

ケース 3: EMR でクラスターの終了をポーリングする

Example EMR でクラスターの終了をポーリングする

多数の EMR クラスターを実行しているユーザーの一般的なパターンは、終了したクラスターに作業が送信されないように、Amazon EMR で終了したクラスターをポーリングすることです。このパターンは、 DescribeCluster および ListClusters API コール、または Amazon EMR クラスターの状態変更イベントを使用して実装できます。

クラスターの終了時に、生成されるイベントは次の例のようになります。

```
{
  "version": "0",
  "id": "1234abb0-f87e-1234-b7b6-000000123456",
  "detail-type": "EMR Cluster State Change",
  "source": "aws.emr",
  "account": "123456789012",
  "time": "2016-12-16T21:00:23Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
    "severity": "INFO",
    "stateChangeReason": "{\"code\":\"USER_REQUEST\",\"message\":\"Terminated by user
 request\"}",
    "name": "Development Cluster",
    "clusterId": "j-123456789ABCD",
    "state": "TERMINATED",
```

```
    "message": "Amazon EMR Cluster jj-123456789ABCD (Development Cluster) has
 terminated at 2016-12-16 21:00 UTC with a reason of USER_REQUEST."
  }
}
```

ペイロードの「detail」セクションには、処理できる clusterID と state が含まれます。

AWS 用語集

AWS の最新の用語については、「AWS の用語集リファレンス」の「AWS 用語集」を参照してください。

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。