AWS ParallelCluster ユーザーガイド (v3)

AWS ParallelCluster

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS ParallelCluster: AWS ParallelCluster ユーザーガイド (v3)

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon の商標およびトレードドレスはAmazon 以外の製品およびサービスに使用することはできません。また、お客様に誤解を与える可能性がある形式で、または Amazon の信用を損なう形式で使用することもできません。Amazon が所有していないその他のすべての商標は Amazon との提携、関連、支援関係の有無にかかわらず、それら該当する所有者の資産です。

Table of Contents

とは AWS ParallelCluster 1
の AWS ParallelCluster 仕組み 2
AWS ParallelCluster プロセス 2
`clustermgtd` 2
`clusterstatusmgtd` 3
`computemgtd` 3
AWS が使用する サービス AWS ParallelCluster 4
Amazon API Gateway 5
AWS Batch 5
AWS CloudFormation 5
Amazon CloudWatch 6
Amazon CloudWatch Events 6
Amazon CloudWatch Logs 6
AWS CodeBuild 6
Amazon DynamoDB 7
Amazon Elastic Block Store 7
Amazon Elastic Compute Cloud 7
Amazon Elastic Container Registry 8
Amazon EFS 8
Amazon FSx for Lustre 8
Amazon FSx for NetApp ONTAP 8
Amazon FSx for OpenZFS 9
AWS Identity and Access Management 9
AWS Lambda 9
Amazon RDS 9
Amazon Route 53 10
Amazon Simple Notification Service 10
Amazon Simple Storage Service 10
Amazon VPC 11
Elastic Fabric Adapter 11
EC2 イメージビルダー 11
Amazon DCV 11
AWS ParallelCluster 内部ディレクトリ 11
セットアップ AWS ParallelCluster 13

前提条件 13
のセットアップ AWS アカウント 13
キーペアを作成する 15
CLI AWS ParallelCluster のインストール 16
仮想環境に AWS ParallelCluster をインストールする (推奨) 16
pip を使用して AWS ParallelCluster 非仮想環境に をインストールする 18
スタンドアロンアプリケーション AWS ParallelCluster として をインストールする 20
インストール後に実行する手順 21
UI AWS ParallelCluster のインストール 22
PCUI をインストールする 23
スタックパラメータ 25
カスタムドメインを設定する 29
Amazon Cognito ユーザープールのオプション 31
AWS ParallelCluster および PCUI バージョンを特定する 33
PCUI のコスト 34
入門 34
CLI AWS ParallelCluster を使用してクラスターを設定および作成する 35
UI AWS ParallelCluster を使用してクラスターを設定および作成する 45
クラスターに接続する 47
クラスターへの複数のユーザーアクセス 47
Active Directory を作成する 49
AD ドメインを含むクラスターを作成します。 49
AD ドメインと統合されたクラスターにログインします。 52
MPI ジョブの実行 53
LDAP(S) クラスター設定 AWS Managed Microsoft AD の例 54
ベストプラクティス 58
ベストプラクティスヘッドノードインスタンスタイプの選択 58
ベストプラクティス: ネットワークパフォーマンス 58
ベストプラクティス: 予算アラート 60
ベストプラクティス: AWS ParallelCluster クラスターを新しいマイナーバージョンまたはパッチバージョンに移動する 60
AWS ParallelCluster 2.x から 3.x への移行 61
カスタムブートストラップアクション 62
AWS ParallelCluster 2.x と 3.x は異なる設定ファイルの構文を使用します 62
包括的言語 68
スケジューラのサポート 69

AWS ParallelCluster CLI ... 69
IMDS 設定の更新 ... 72
の使用 AWS ParallelCluster ... 73
AWS ParallelCluster UI ... 74
AWS Lambda での VPC 設定 AWS ParallelCluster ... 75
AWS Identity and Access Management での アクセス許可 AWS ParallelCluster ... 77
AWS ParallelCluster Amazon EC2 インスタンスロール ... 78
AWS ParallelCluster `pcluster` ユーザーポリシーの例 ... 78
AWS ParallelCluster IAM リソースを管理するためのユーザーポリシーの例 ... 93
AWS ParallelCluster IAM アクセス許可を管理するための設定パラメータ ... 99
ネットワークの設定 ... 114
AWS ParallelCluster 1 つのパブリックサブネット内の ... 115
AWS ParallelCluster 2 つのサブネットの使用 ... 117
AWS ParallelCluster を使用して接続された単一のプライベートサブネット内の AWS Direct Connect ... 119
AWS ParallelCluster スケジューラを使用する AWS Batch ... 120
AWS ParallelCluster インターネットアクセスのない単一のサブネット内 ... 122
によってプロビジョニングされたログインノード AWS ParallelCluster ... 129
ログインノードのセキュリティ ... 129
ログインノードのネットワーク ... 130
ログインノードのストレージ ... 130
ログインノードの Imds プロパティ ... 131
ログインノードのライフサイクル ... 131
ログインノードプールの実行に必要なアクセス許可 ... 132
カスタムブートストラップアクション ... 132
設定 ... 135
引数 ... 139
カスタムブートストラップアクションを使用したクラスターの例 ... 139
IMDSv2 のカスタムブートストラップスクリプトを更新する方法の例 ... 141
IMDSv1 の設定を更新する方法の例 ... 141
Amazon S3 での使用 ... 142
例 ... 142
スポットインスタンスの操作 ... 143
シナリオ 1: 実行中のジョブがないスポットインスタンスが中断される ... 144
シナリオ 2: 単一ノードジョブを実行しているスポットインスタンスが中断される ... 144
シナリオ 3: マルチノードジョブを実行しているスポットインスタンスが中断される ... 144

でサポートされているスケジューラ AWS ParallelCluster 145
Slurm Workload Manager 145
AWS Batch 210
共有ストレージ 218
Amazon EBS 221
Amazon EFS 222
FSx for Lustre 223
FSx for ONTAP、FSx for OpenZFS、およびファイルキャッシュ 224
共有ストレージの操作 224
クォータ 228
Tagging 229
タグの表示 229
AWS ParallelCluster と ログのモニタリング 232
Amazon CloudWatch Logs との統合 233
Amazon CloudWatch ダッシュボード 236
クラスターメトリクス用の Amazon CloudWatch アラーム 238
AWS ParallelCluster ログローテーションの設定 241
`pcluster` CLI ログ 242
Amazon EC2 コンソール出力ログ 243
PCUI と AWS ParallelCluster ランタイムログを取得する 244
ログの取得と保存 246
AWS CloudFormation カスタムリソース 249
によってホストされるプロバイダースタック AWS ParallelCluster 250
クラスターリソース 251
クラスターオペレーション 255
AWS ParallelCluster カスタムリソースを含むスタックのトラブルシューティング 255
Elastic Fabric Adapter 256
Intel MPI を有効にする 257
AWS ParallelCluster API 258
AWS ParallelCluster API ドキュメント 259
を使用して AWS ParallelCluster API をデプロイする AWS CLI 259
API の更新 262
AWS ParallelCluster API の呼び出し 263
API ログとメトリクスへのアクセス 265
AWS ParallelCluster Terraform の 266
Amazon DCV 経由でヘッドノードとログインノードに接続する 267

Amazon DCV の HTTPS 証明書 267
Amazon DCV のライセンス 267
`pcluster update-cluster` を使用する 268
ポリシー定義を更新します 268
`pcluster update-cluster` の例 273
AWS ParallelCluster AMI のカスタマイズ 276
AWS ParallelCluster AMI のカスタマイズに関する考慮事項 276
カスタムコンポーネントの検証テストを実行する 277
デバッグに役立つ `pcluster` コマンドを使用して Image Builder プロセスを監視する 277
その他の考慮事項 278
オンデマンドキャパシティ予約 (ODCR) を使用してインスタンスを起動する 279
での ODCR の使用 AWS ParallelCluster 279
キャパシティブロック (CB) を使用してインスタンスを起動する 288
での CB の使用 AWS ParallelCluster 288
AMI のパッチ適用と Amazon EC2 インスタンスの置換 290
ヘッドノードインスタンスの更新または交換 291
エフェメラルドライブからデータを保存する 291
クラスターのヘッドノードを停止して起動する 291
オペレーティングシステム 293
オペレーティングシステムに関する考慮事項 294
のリファレンス AWS ParallelCluster 296
AWS ParallelCluster バージョン 3 CLI コマンド 296
`pcluster` 297
`pcluster3-config-converter` 343
設定ファイル 344
クラスター設定ファイル 345
ビルドイメージの設定ファイル 488
AWS ParallelCluster API リファレンス 498
buildImage 499
createCluster 504
deleteCluster 510
deleteClusterInstances 513
deleteImage 515
describeCluster 518
describeClusterInstances 526
describeComputeFleet 529

describeImage ... 531
getClusterLogEvents ... 538
getClusterStackEvents ... 542
getImageLogEvents ... 546
getImageStackEvents ... 550
listClusters ... 554
listClusterLogStreams ... 558
listImageLogStreams ... 562
listImages ... 566
listOfficialImages ... 569
updateCluster ... 572
updateComputeFleet ... 579
AWS ParallelCluster Python ライブラリ API ... 581
AWS ParallelCluster Python ライブラリの認可 ... 581
Python AWS ParallelCluster ライブラリをインストールする ... 582
クラスター API オペレーション ... 582
コンピューティングフリートの API オペレーション ... 586
クラスターとスタックのログ操作 ... 588
イメージの API オペレーション ... 590
イメージとスタックのログ操作 ... 593
例 ... 596
AWS Lambda Python AWS ParallelCluster ライブラリ用の ... 597
チュートリアル ... 599
で最初のジョブを実行する AWS ParallelCluster ... 599
インストールを確認する ... 600
初めてクラスターを作成する ... 600
ヘッドノードにログインする ... 601
Slurm を使用して最初のジョブを実行する ... 602
カスタム AMI AWS ParallelCluster の構築 ... 603
AMI AWS ParallelCluster をカスタマイズする方法 ... 604
カスタム AMI AWS ParallelCluster を構築する ... 604
AMI AWS ParallelCluster の変更 ... 611
Active Directory の統合 ... 614
AD インフラストラクチャを作成する ... 616
(オプション) AD のユーザーおよびグループの管理 ... 632
クラスターを作成する ... 634

ユーザーとしてクラスターに接続する 640
クリーンアップ 640
AWS KMS キーを使用した共有ストレージ暗号化の設定 645
ポリシーの作成 646
クラスターの設定と作成 647
マルチキューモードのクラスターでジョブを実行する 649
クラスターを設定する 650
クラスターを作成する 651
ヘッドノードにログインします。 652
マルチキューモードでジョブを実行する 653
AWS ParallelCluster API の使用 657
Slurm アカウンティングによるクラスターの作成 671
ステップ 1: の VPC とサブネットを作成する AWS ParallelCluster 672
ステップ 2: データベーススタックを作成する 672
ステップ 3: Slurm アカウンティングを有効にしたクラスターを作成する 673
外部 Slurmdbd アカウンティングによるクラスターの作成 673
ステップ 1: Slurmdbd スタックを作成する 674
ステップ 2: 外部 Slurmdbd を有効にしたクラスターを作成する 676
以前の AWS Systems Manager ドキュメントバージョンに戻す 677
SSM ドキュメントを以前のバージョンに戻す 677
を使用したクラスターの作成 AWS CloudFormation 679
CloudFormation クイック作成スタックによるクラスター作成 680
コマンドラインインターフェイス (CLI) AWS CloudFormation を使用したクラスターの作成 682
CloudFormation クラスターの出力を表示する 684
クラスターへのアクセス 685
クリーンアップ 685
Terraform を使用して ParallelCluster API をデプロイする 686
Terraform プロジェクトを定義する 686
API をデプロイする 688
必要なアクセス許可 689
Terraform を使用したクラスターの作成 692
Terraform プロジェクトを定義する 692
クラスターをデプロイする 699
必要なアクセス許可 700
Terraform を使用したカスタム AMI の作成 701

Terraform プロジェクトを定義する 701
AMI をデプロイする 704
必要なアクセス許可 705
AWS ParallelCluster UI と Identity Center の統合 705
IAM Identity Center を有効にする 706
アプリケーションを IAM アイデンティティセンターに追加する 709
Pyxis でコンテナ化されたジョブを実行する 716
クラスターを作成する 717
ジョブを送信する 719
EFA 対応 FSx Lustre を使用したクラスターの作成 720
要件 720
セキュリティグループを作成する 720
ファイルシステムを作成する 722
クラスターを作成する 722
EFA で FSx が動作していることを検証する 724
AWS ParallelCluster トラブルシューティング 725
クラスターの作成を試行する 726
`failureCode` が `OnNodeConfiguredExecutionFailure` 726
`failureCode` が `OnNodeConfiguredDownloadFailure` 726
`failureCode` が `OnNodeConfiguredFailure` 727
`failureCode` が `OnNodeStartExecutionFailure` 727
`failureCode` が `OnNodeStartDownloadFailure` 728
`failureCode` が `OnNodeStartFailure` 728
`failureCode` が `EbsMountFailure` 728
`failureCode` が `EfsMountFailure` 729
`failureCode` が `FsxMountFailure` 729
`failureCode` が `RaidMountFailure` 729
`failureCode` が `AmiVersionMismatch` 730
`failureCode` が `InvalidAmi` 730
`failureCode` が `HeadNodeBootstrapFailure` と `failureReason` で、ヘッドノードの設定に失敗した。 730
`failureCode` は `HeadNodeBootstrapFailure` で、`failureReason` クラスター作成がタイムアウトした。 731
`failureCode` は `HeadNodeBootstrapFailure` で、`failureReason` はヘッドノードのブートストラップに失敗した。 732
`failureCode` が `ResourceCreationFailure` 732

failureCode が ClusterCreationFailure 733
CloudFormation スタックの WaitCondition timed out... が表示されている 733
CloudFormation スタックの Resource creation cancelled が表示されている 733
AWS CloudFormation スタック内の表示Failed to run cfn-init...またはその他のエラー 733
INFO: Waiting for static fleet capacity provisioning の最後に chef-client.log が表示されている 733
Failed to run preinstall or postinstall in cfn-init.log が表示されている 734
CloudFormation スタックの This AMI was created with xxx, but is trying to be used with xxx... が表示されている 734
CloudFormation スタックの This AMI was not baked by AWS ParallelCluster... が表示されている 734
pcluster create-cluster コマンドがローカルで実行できないことが表示されている 734
追加のサポート 734
ジョブの実行を試行する 734
srun のインタラクティブなジョブがエラー srun: error: fwd_tree_thread: can't find address for <host>, check slurm.conf で失敗する 735
ジョブが squeue コマンドで、CF 状態でスタックしている 735
大規模なジョブを実行し、nfsd: too many open connections, consider increasing the number of threads in /var/log/messages が表示されている 736
MPI ジョブを実行する 736
クラスターの更新を試行する 737
pcluster update-cluster コマンドのローカル実行に失敗する 737
pcluster describe-cluster コマンドで clusterStatus が UPDATE_FAILED であることが表示されている 737
クラスターの更新がタイムアウトになる 738
ストレージへのアクセスを試行する 738
外部の Amazon FSx for Lustre ファイルシステムを使用する 738
外部の Amazon Elastic File System ファイルシステムを使用する 738
クラスターの削除を試行する 738
pcluster delete-cluster コマンドのローカル実行に失敗する 738
クラスタースタックが削除に失敗する 739
AWS ParallelCluster API スタックのアップグレードを試行する 739

コンピューティンティングノードの初期化のエラーが表示されている 739
clustermgtd.log に Node bootstrap error が表示されている 740
オンデマンドキャパシティ予約 (ODCR) またはゾーンレベルのリザーブドインスタンスを設定しました。 740
ジョブの実行に失敗したとき slurm_resume.log で、またはクラスターの作成に失敗したとき clustermgtd.log で An error occurred (VcpuLimitExceeded) が表示されている 741
ジョブの実行に失敗したとき slurm_resume.log で、またはクラスターの作成に失敗したとき clustermgtd.log で An error occurred (InsufficientInstanceCapacity) が表示されている 742
ノードが Reason (Code:InsufficientInstanceCapacity)... と共に DOWN ステータスで表示されている 742
slurm_resume.log に cannot change locale (en_US.utf-8) because it has an invalid name が表示されている 742
前のシナリオはどれも私の状況には当てはまりません。 743
クラスターヘルスメトリクスのトラブルシューティング 743
インスタンスプロビジョニングエラーグラフが表示されている 743
「異常なインスタンスエラー」グラフが表示されている 745
コンピューティングフリートのアイドル時間グラフが表示されている 747
クラスターデプロイの問題のトラブルシューティング 748
で AWS CloudFormation イベントを表示する CREATE_FAILED 749
CLI を使用してログストリームを表示する 751
rollback-on-failure を使用して失敗したクラスターを再作成する 753
Terraform を使用したクラスターのデプロイのトラブルシューティング 754
ParallelCluster API が見つからない 754
ParallelCluster API を呼び出す権限がない 755
スケーリング問題のトラブルシューティング 755
デバッグ用のキーログ 756
ジョブの実行に失敗したとき slurm_resume.log で、またはクラスターの作成に失敗したとき clustermgtd.log で InsufficientInstanceCapacity エラーが表示されている 758
ノードの初期化に関する問題のトラブルシューティング 759
予期しないノードの置換や終了のトラブルシューティング 762
問題のあるインスタンスやノードの置換、終了、電源オフ 763
キュー (パーティション) の Inactive ステータス 764
その他の既知のノードやジョブの問題のトラブルシューティング 764

プレイスメントグループとインスタンスの起動に関する問題 764
ディレクトリの置換 765
Amazon DCV の問題のトラブルシューティング 765
Amazon DCV のログ 765
Ubuntu Amazon DCV の問題 765
AWS Batch 統合によるクラスターの問題のトラブルシューティング 766
ヘッドノードの問題 766
コンピュートの問題 767
ジョブの失敗 767
エンドポイント URL の接続タイムアウトエラー 767
Active Directory を使用したマルチユーザー統合のトラブルシューティング 767
Active Directory 固有のトラブルシューティング 768
デバッグモードを有効にする 769
LDAPS から LDAP に移行する方法 769
LDAPS サーバー証明書の検証を無効にする方法 770
パスワードではなく SSH キーを使用してログインする方法 770
ユーザーパスワードと失効しているパスワードをリセットする方法 770
参加しているドメインを確認する方法 771
証明書に関する問題をトラブルシューティングする方法 771
Active Directory との統合が動作していることを確認する方法 773
コンピューティングノードへのログインのトラブルシューティング方法 774
マルチユーザー環境での SimCenter StarCCM+ ジョブに関する既知の問題 775
ユーザー名解決に関する既知の問題 775
ホームディレクトリ作成の問題の解決方法 775
カスタム AMI の問題のトラブルシューティング 776
`cfn-hup` が実行していない場合のクラスター更新タイムアウトのトラブルシューティング ... 777
ネットワークのトラブルシューティング 778
単一のパブリックサブネットのクラスターに関する問題 778
クラスターの更新が `onNodeUpdated` カスタムアクションで失敗した 779
カスタム Slurm 設定でエラーが表示されている 779
クラスターのアラーム 779
AWS ParallelCluster サポートポリシー 781
セキュリティ 783
が使用する サービスのセキュリティ情報 AWS ParallelCluster 783
データ保護 784
データの暗号化 785

関連情報 786
Identity and Access Management 787
コンプライアンス検証 787
TLS 1.2 の適用 788
現在サポートされているプロトコルの確認 789
OpenSSL と Python のコンパイル 790
サポートされている AWS リージョン 792
リリースノートとドキュメント履歴 794
..... cmxxi

とは AWS ParallelCluster

AWS ParallelCluster は、 でのハイパフォーマンスコンピューティング (HPC) クラスターのデプロイと管理に役立つ、 AWS サポートされているオープンソースのクラスター管理ツールです AWS クラウド。必要なコンピューティングリソース、スケジューラ、共有ファイルシステムが自動的に設定されます。 AWS Batch および スSlurmケジューラ AWS ParallelCluster で を使用できます。

を使用すると AWS ParallelCluster、概念実証環境と本番稼働用 HPC コンピューティング環境をすばやく構築してデプロイできます。また、DNA シーケンスワークフロー全体を自動化するゲノミクスポータルなど AWS ParallelCluster、 上に高レベルのワークフローを構築してデプロイすることもできます。

次の方法 AWS ParallelCluster を使用して にアクセスできます。

- AWS ParallelCluster コマンドラインインターフェイス (CLI)
- AWS ParallelCluster API
- PCUI (リリース 3.5.0 で追加)
- AWS ParallelCluster Python ライブラリ API (リリース 3.5.0 で追加)
- AWS CloudFormation カスタムリソース として (リリース 3.6.0 で追加)

料金表

AWS ParallelCluster コマンドラインインターフェイス (CLI) または API を使用する場合、 AWS ParallelCluster イメージとクラスターを作成または更新したときに作成された AWS リソースに対してのみ料金が発生します。詳細については、「AWS が使用する サービス AWS ParallelCluster」を参照してください。

の AWS ParallelCluster 仕組み

AWS ParallelCluster は、クラスターを管理する方法としてだけでなく、 AWS サービスを使用して HPC 環境を構築する方法のリファレンスとして構築されました。以下のトピックでは、 AWS ParallelCluster プロセス、 AWS のサービス が AWS ParallelCluster 使用する と方法、および内部ディレクトリについて説明します。

トピック

- AWS ParallelCluster プロセス
- AWS が使用する サービス AWS ParallelCluster
- AWS ParallelCluster 内部ディレクトリ

AWS ParallelCluster プロセス

このセクションは、Slurm でデプロイされたクラスターに適用されます。このスケジューラで使用すると、 は基盤となるジョブスケジューラと AWS ParallelCluster やり取りして、コンピューティングノードのプロビジョニングと削除を管理します。

に基づく HPC クラスターの場合 AWS Batch、 はコンピューティングノードを管理する AWS Batch ために が提供する機能 AWS ParallelCluster に依存します。

`clustermgtd`

クラスター管理デーモン (`clustermgtd`) は、次のタスクを実行します。

- 非アクティブなパーティションをクリーンアップする
- キャパシティブロックに関連付けられたSlurm予約とノードを管理する (次のセクションを参照)
- 静的容量を管理して、常に稼働し、正常であることを確認する
- スケジューラを Amazon EC2 と同期します。
- 孤立したインスタンスをクリーンアップする
- 停止ワークフローの外部で発生した Amazon EC2 終了時にスケジューラノードのステータスを復元する
- 異常な Amazon EC2 インスタンス (Amazon EC2 ヘルスチェックに合格しないインスタンス) を管理する

- スケジュールされたメンテナンスイベントを管理する
- 異常なスケジューラノード (スケジューラのヘルスチェックに失敗したノード) を管理する

キャパシティブロックに関連付けられたSlurm予約とノードの管理

ParallelCluster は、オンデマンドキャパシティ予約 (ODCR) と機械学習用キャパシティブロック (CB) をサポートしています。ODCR とは異なり、CB は将来の開始時間を持つ場合があり、期限があります。

`clustermgtd` はループ内の異常なノードを検索し、ダウンしている Amazon EC2 インスタンスをすべて終了し、静的ノードの場合は新しいインスタンスに置き換えます。

AWS ParallelCluster はキャパシティブロックに関連付けられた静的ノードを異なる方法で管理します。CB がまだアクティブでなくてもクラスターを作成し、CB がアクティブになるとインスタンスを自動的に起動します。

まだアクティブでない CBs に関連付けられたコンピューティングリソースに対応するSlurmノードは、CB 開始時刻に達するまでメンテナンス状態のままになります。これらのSlurmノードは、Slurm管理者ユーザーに関連付けられた予約/メンテナンス状態のままです。つまり、ジョブを受け入れることができますが、Slurm予約が削除されるまでジョブは保留中のままです。

`clustermgtd` は自動的にSlurm予約を作成または削除します。これにより、関連する CB ノードが CB 状態に基づいてメンテナンス状態になります。CB がアクティブになると、Slurm予約が削除され、ノードは開始され、保留中のジョブまたは新しいジョブ送信で使用可能になります。

CB 終了時刻に達すると、ノードは予約/メンテナンス状態に戻ります。CB がアクティブでなくなり、インスタンスが終了したときに、ジョブを新しいキュー/コンピューティングリソースに再送信/再クエリするかどうかは、ユーザー次第です。

clusterstatusmgtd

クラスターステータス管理デーモン (`clusterstatusmgtd`) は、コンピューティングフリートのステータス更新を管理します。DynamoDB テーブルに保存されているフリートのステータスを毎分取得し、すべての STOP/START リクエストを管理します。

computemgtd

コンピューティング管理デーモン (`computemgtd`) プロセスは、各クラスターコンピューティングノードで実行されます。5 分ごとに、コンピューティング管理デーモンはヘッドノードに到達できる

こと、および正常であることを確認します。5 分が経過し、ヘッドノードに到達できない、または正常でない場合、コンピューティングノードはシャットダウンされます。

AWS が使用する サービス AWS ParallelCluster

次の Amazon Web Services (AWS) サービスが によって使用されます AWS ParallelCluster。

トピック

- Amazon API Gateway
- AWS Batch
- AWS CloudFormation
- Amazon CloudWatch
- Amazon CloudWatch Events
- Amazon CloudWatch Logs
- AWS CodeBuild
- Amazon DynamoDB
- Amazon Elastic Block Store
- Amazon Elastic Compute Cloud
- Amazon Elastic Container Registry
- Amazon EFS
- Amazon FSx for Lustre
- Amazon FSx for NetApp ONTAP
- Amazon FSx for OpenZFS
- AWS Identity and Access Management
- AWS Lambda
- Amazon RDS
- Amazon Route 53
- Amazon Simple Notification Service
- Amazon Simple Storage Service
- Amazon VPC
- Elastic Fabric Adapter
- EC2 イメージビルダー

- Amazon DCV

Amazon API Gateway

Amazon API Gateway は、REST、HTTP、WebSocket APIsで作成、公開、保守、モニタリング、保護できるようにする AWS サービスです。

AWS ParallelCluster は API Gateway を使用して AWS ParallelCluster API をホストします。

Amazon API Gateway の詳細については、https://aws.amazon.com/api-gateway/ および https://docs.aws.amazon.com/apigateway/ を参照してください。

AWS Batch

AWS Batch はマネージド AWS ジョブスケジューラサービスです。 AWS Batch クラスター内のコンピューティングリソース (CPU やメモリ最適化インスタンスなど) の最適な量とタイプを動的にプロビジョニングします。これらのリソースは、ボリューム要件など、バッチジョブの特定の要件に基づいてプロビジョニングされます。を使用すると AWS Batch、ジョブを効果的に実行するために、追加のバッチコンピューティングソフトウェアやサーバークラスターをインストールまたは管理する必要はありません。

AWS Batch は AWS Batch クラスターでのみ使用されます。

詳細については AWS Batch、 および https://aws.amazon.com/batch/ を参照してくださいhttps://docs.aws.amazon.com/batch/。

AWS CloudFormation

AWS CloudFormation は、クラウド環境で AWS およびサードパーティーのアプリケーションリソースをモデル化およびプロビジョニングするための共通言語を提供するinfrastructure-as-codeサービスです。これは、 が使用するメインサービスです AWS ParallelCluster。の各クラスター AWS ParallelCluster はスタックとして表され、各クラスターに必要なすべてのリソースは AWS ParallelCluster CloudFormation テンプレート内で定義されます。ほとんどの場合、 AWS ParallelCluster CLI コマンドは、作成、更新、削除などの AWS CloudFormation スタックコマンドに直接対応します。クラスター内で起動されたインスタンスは、クラスターが起動 AWS リージョンされた の AWS CloudFormation エンドポイントに対して HTTPS 呼び出しを行います。

詳細については AWS CloudFormation、 および https://aws.amazon.com/cloudformation/ を参照してくださいhttps://docs.aws.amazon.com/cloudformation/。

Amazon CloudWatch

Amazon CloudWatch (CloudWatch) は、データと実用的なインサイトを提供するモニタリングおよびオブザーバビリティサービスです。これらのインサイトは、アプリケーションのモニタリング、パフォーマンスの変化やサービスの例外への対応、リソースの使用状況の最適化に利用できます。では AWS ParallelCluster、CloudWatch はダッシュボードに使用され、Docker イメージのビルドステップと AWS Batch ジョブの出力をモニタリングしてログに記録します。

AWS ParallelCluster バージョン 2.10.0 以前は、CloudWatch は AWS Batch クラスターでのみ使用されていました。

CloudWatch の詳細については、https://aws.amazon.com/cloudwatch/ および https://docs.aws.amazon.com/cloudwatch/ を参照してください。

Amazon CloudWatch Events

Amazon CloudWatch Events (CloudWatch Events) では、Amazon Web Services (AWS) リソースの変更を記述した、システムイベントのほぼリアルタイムのストリーミングを配信します。すぐに設定できる簡単なルールを使用して、ルールに一致したイベントを 1 つ以上のターゲット関数またはストリームに振り分けることができます。では AWS ParallelCluster、CloudWatch Events が AWS Batch ジョブに使用されます。

CloudWatch Events の詳細については、https://docs.aws.amazon.com//eventbridge/latest/userguide/eb-cwe-now-eb を参照してください。

Amazon CloudWatch Logs

Amazon CloudWatch Logs (CloudWatch Logs) は、Amazon CloudWatch のコア機能のひとつです。AWS ParallelClusterが使用している多くのコンポーネントのログファイルをモニタリング、保存、閲覧、検索するために使用できます。

AWS ParallelCluster バージョン 2.6.0 以前は、CloudWatch Logs は AWS Batch クラスターでのみ使用されていました。

詳細については、「Amazon CloudWatch Logs との統合」を参照してください。

AWS CodeBuild

AWS CodeBuild (CodeBuild) は、ソースコードをコンパイルし、テストを実行し、デプロイ可能なソフトウェアパッケージを生成する AWS マネージド型の継続的統合サービスです。では AWS

ParallelCluster、CodeBuild はクラスターの作成時に Docker イメージを自動的かつ透過的に構築するために使用されます。

CodeBuild は AWS Batch クラスターでのみ使用されます。

CodeBuild の詳細については、https://aws.amazon.com/codebuild/ および https://docs.aws.amazon.com/codebuild/ を参照してください。

Amazon DynamoDB

Amazon DynamoDB (DynamoDB) は、高速で柔軟な NoSQL データベースサービスです。クラスターの最小限の状態情報を保存するために使用されます。ヘッドノードは、プロビジョニングされたインスタンスを DynamoDB テーブルで追跡します。

DynamoDB は AWS Batch クラスターでは使用されません。

DynamoDB の詳細については、https://aws.amazon.com/dynamodb/ および https://docs.aws.amazon.com/dynamodb/ を参照してください。

Amazon Elastic Block Store

Amazon Elastic Block Store (Amazon EBS) は、共有ボリュームの永続的ストレージを提供する高性能ブロックストレージサービスです。すべての Amazon EBS の設定を構成することができます。Amazon EBS ボリュームは、空白で初期化することも、既存の Amazon EBS スナップショットから初期化することもできます。

Amazon EBS の詳細については、https://aws.amazon.com/ebs/ および https://docs.aws.amazon.com/ebs/ を参照してください。

Amazon Elastic Compute Cloud

Amazon Elastic Compute Cloud (Amazon EC2) は、コンピューティング性能を提供します AWS ParallelCluster。ヘッドノードとコンピューティングノードは、Amazon EC2 インスタンスです。ハードウェア仮想マシン (HVM) をサポートする任意のインスタンスタイプを選択できます。ヘッドノードとコンピューティングノードは、異なるインスタンスタイプにすることができます。また、複数のキューを使用する場合は、一部または全部のコンピューティングノードをスポットインスタンスとして起動することも可能です。インスタンスで見つかったインスタンスストアボリュームは、ストライプの Logical Volume Manager (LVM) ボリュームとしてマウントされます。

Amazon EC2 の詳細については、https://aws.amazon.com/ec2/ および https://docs.aws.amazon.com/ec2/ を参照してください。

Amazon Elastic Container Registry

Amazon Elastic Container Registry (Amazon ECR) は、フルマネージドされた Docker コンテナレジストリで、Docker コンテナイメージの保存、管理、デプロイを容易に行うことができます。Amazon ECR は AWS ParallelCluster、クラスターの作成時に構築された Docker イメージを に保存します。Docker イメージは、送信されたジョブのコンテナを実行する AWS Batch ために によって使用されます。

Amazon ECR は AWS Batch クラスターでのみ使用されます。

詳細については、https://aws.amazon.com/ecr/ および https://docs.aws.amazon.com/ecr/ を参照してください。

Amazon EFS

Amazon Elastic File System (Amazon EFS)は、 AWS クラウド サービスやオンプレミスのリソースで使用できる、シンプルでスケーラブルな、フルマネージドされた伸縮自在な NFS ファイルシステムを提供します。Amazon EFS は、`EfsSettings` が指定されている場合に使用されます。Amazon EFS のサポートが AWS ParallelCluster バージョン 2.1.0 で追加されました。

Amazon EFS の詳細については、https://aws.amazon.com/efs/ および https://docs.aws.amazon.com/efs/ を参照してください。

Amazon FSx for Lustre

FSx for Lustre は、オープンソースの Lustre ファイルシステムを使用した高性能なファイルシステムを提供します。`FsxLustreSettings のプロパティ` が指定されている場合、FSx for Lustre が使用されます。FSx for Lustre のサポートが AWS ParallelCluster バージョン 2.2.1 で追加されました。

FSx for Lustre の詳細については、https://aws.amazon.com/fsx/lustre/ および https://docs.aws.amazon.com/fsx/ を参照してください。

Amazon FSx for NetApp ONTAP

FSx for ONTAP は、NetApp の一般的な ONTAP ファイルシステム上に構築されたフルマネージド型の共有ストレージシステムを提供します。`FsxOntapSettings のプロパティ` が指定されている場合、FSx for ONTAP が使用されます。FSx for ONTAP のサポートが AWS ParallelCluster バージョン 3.2.0 で追加されました。

FSx for ONTAP の詳細については、https://aws.amazon.com/fsx/netapp-ontap/ および https://docs.aws.amazon.com/fsx/ を参照してください。

Amazon FSx for OpenZFS

FSx for OpenZFS は、一般的な OpenZFS ファイルシステム上に構築されたフルマネージド型の共有ストレージシステムを提供します。`FsxOpenZfsSettings のプロパティ` が指定されている場合、FSx for OpenZFS が使用されます。FSx for OpenZFS のサポートが AWS ParallelCluster バージョン 3.2.0 に追加されました。

FSx for OpenZFS の詳細については、https://aws.amazon.com/fsx/openzfs/ および https://docs.aws.amazon.com/fsx/ を参照してください。

AWS Identity and Access Management

AWS Identity and Access Management (IAM) は、 内で使用され AWS ParallelCluster 、個々のクラスターに固有のインスタンスに対して Amazon EC2 の最小特権の IAM ロールを提供します。 AWS ParallelCluster インスタンスには、クラスターのデプロイと管理に必要な特定の API コールにのみアクセス権が付与されます。

AWS Batch クラスターでは、クラスターの作成時に Docker イメージ構築プロセスに関連するコンポーネントに対しても IAM ロールが作成されます。これらのコンポーネントには、Amazon ECR リポジトリとの間で Docker イメージを追加および削除することが許可されている Lambda 関数が含まれます。また、クラスターと CodeBuild プロジェクト用に作成された Amazon S3 バケットを削除するための機能も含まれています。 AWS Batch リソース、インスタンス、ジョブのロールもあります。

IAM の詳細については、https://aws.amazon.com/iam/ および https://docs.aws.amazon.com/iam/ を参照してください。

AWS Lambda

AWS Lambda (Lambda) は、Docker イメージの作成を調整する関数を実行します。また、Lambda は Amazon ECR リポジトリや Amazon S3 に保存された Docker イメージなど、カスタムクラスターソースのクリーンアップを管理します。

Lambda の詳細については、https://aws.amazon.com/lambda/ および https://docs.aws.amazon.com/lambda/ を参照してください。

Amazon RDS

Amazon Relational Database Service (Amazon RDS) は、 AWS クラウドでのリレーショナルデータベースのセットアップ、運用、スケーリングを容易にするウェブサービスです。

AWS ParallelCluster は、 AWS Batch と に Amazon RDS を使用しますSlurm。

Amazon RDS の詳細については、https://aws.amazon.com/rds/ および https://docs.aws.amazon.com/rds/ を参照してください。

Amazon Route 53

Amazon Route 53 (Route 53) は、各コンピューティングノードのホスト名と完全に適正のドメイン名を持つホストゾーンを作成するために使用します。

Route 53 の詳細については、https://aws.amazon.com/route53/ および https://docs.aws.amazon.com/route53/ を参照してください。

Amazon Simple Notification Service

(Amazon SNS) は、パブリッシャーからサブスクライバー (生産者および消費者とも呼ばれます) へのメッセージ配信を提供するマネージドサービスです。

AWS ParallelCluster は API ホスティングに Amazon SNS を使用します。

Amazon SNS の詳細については、https://aws.amazon.com/sns/ および https://docs.aws.amazon.com/sns/ を参照してください。

Amazon Simple Storage Service

Amazon Simple Storage Service (Amazon S3) は、各 にある AWS ParallelCluster テンプレートを保存します AWS リージョン。 は AWS ParallelCluster 、CLI/SDK ツールが Amazon S3 を使用できるように設定できます。

AWS ParallelCluster また、 は に Amazon S3 バケットを作成し AWS アカウント 、クラスター設定ファイルなど、クラスターで使用されるリソースを保存します。 は、クラスターを作成する各 AWS リージョン に 1 つの Amazon S3 バケット AWS ParallelCluster を維持します。

AWS Batch クラスターを使用すると、アカウントの Amazon S3 バケットが関連データの保存に使用されます。例えば、バケットには、投入されたジョブから Docker イメージやスクリプトが作成された際に作成されたアーティファクトが保存されます。

詳細については、https://aws.amazon.com/s3/ および https://docs.aws.amazon.com/s3/ を参照してください。

Amazon VPC

Amazon Virtual Private Cloud (VPC) は、クラスター内のノードで使用されるネットワークを定義します。

Amazon VPC の詳細については、https://aws.amazon.com/vpc/ および https://docs.aws.amazon.com/vpc/ を参照してください。

Elastic Fabric Adapter

Elastic Fabric Adapter (EFA) は、大規模な高レベルのノード間通信を必要とするアプリケーションを実行するために使用できるインスタンス用のネットワークインターフェイスです AWS。

Elastic Fabric Adapter の詳細については、https://aws.amazon.com/hpc/efa/ を参照してください。

EC2 イメージビルダー

EC2 Image Builder は、カスタマイズされ、安全でup-to-dateサーバーイメージの作成、管理、デプロイを自動化するフルマネージド AWS サービスです。

AWS ParallelCluster は Image Builder を使用して AWS ParallelCluster イメージを作成および管理します。

EC2 Image Builder の詳細については、https://aws.amazon.com/image-builder/ と https://docs.aws.amazon.com/imagebuilder/ を参照してください。

Amazon DCV

Amazon DCV は、高性能なリモートディスプレイプロトコルで、さまざまなネットワーク条件下で、あらゆるデバイスにリモートデスクトップやアプリケーションストリーミングを安全に配信する方法を提供します。Amazon DCV は、`HeadNode` セクション/`Dcv` 設定が指定されている場合に使用されます。Amazon DCV のサポートが AWS ParallelCluster バージョン 2.5.0 で追加されました。

Amazon DCV の詳細については、https://aws.amazon.com/hpc/dcv/ および https://docs.aws.amazon.com/dcv/ を参照してください。

AWS ParallelCluster 内部ディレクトリ

クラスター内でデータを共有するために が AWS ParallelCluster 使用する内部ディレクトリがいくつかあります。以下のディレクトリは、ヘッドノード、コンピューティングノード、ログインノード間で共有されます。

- `/opt/slurm`
- `/opt/intel`
- `/opt/parallelcluster/shared (only with compute nodes)`
- `/opt/parallelcluster/shared_login_nodes (only with login nodes)`
- `/home (unless specified in SharedStorage)`

> **Note**
>
> デフォルトでは、これらのディレクトリはヘッドノードの EBS ボリューム上に作成され、コンピューティングノードとログインノードへの NFS エクスポートとして共有されます。AWS ParallelCluster 3.8 以降では、SharedStorageType パラメータ AWS ParallelCluster を efs に設定することで、 がこれらのディレクトリをホストおよび共有するための Amazon EFS ファイルシステムを作成および管理できます。
> クラスターがスケールアウトすると、EBS ボリュームを介した NFS エクスポートがパフォーマンスのボトルネックをもたらす可能性があります。EFS を使用すると、クラスターのスケールアウトに伴う NFS エクスポートを回避し、それらに関連するパフォーマンスのボトルネックを回避できます。

セットアップ AWS ParallelCluster

以下のトピックでは、 を設定する方法について説明します AWS ParallelCluster。必要なツールをインストールおよび使用する方法、クラスターへの複数のユーザーアクセスを実装および管理する方法、さらにベストプラクティスについて説明します。

トピック

- 前提条件
- AWS ParallelCluster コマンドラインインターフェイスのインストール (CLI)
- インストール後に実行する手順
- PCUI のインストール
- の開始方法 AWS ParallelCluster
- クラスターへの複数のユーザーアクセス
- ベストプラクティス
- AWS ParallelCluster 2.x から 3.x への移行

前提条件

のセットアップと使用を開始する前に AWS ParallelCluster、次の前提条件を満たしていることを確認してください。

のセットアップ AWS アカウント

使用する AWS アカウントを設定します AWS ParallelCluster。

にサインアップする AWS アカウント

がない場合は AWS アカウント、次の手順を実行して作成します。

にサインアップするには AWS アカウント

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

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

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

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

管理アクセスを持つユーザーを作成する

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

を保護する AWS アカウントのルートユーザー

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

   ルートユーザーを使用してサインインする方法については、AWS サインイン ユーザーガイドのルートユーザーとしてサインインするを参照してください。

2. ルートユーザーの多要素認証 (MFA) を有効にします。

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

管理アクセスを持つユーザーを作成する

1. IAM アイデンティティセンターを有効にします。

   手順については、「AWS IAM 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 アイデンティティセンターで、ユーザーに管理アクセスを付与します。

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

管理アクセス権を持つユーザーとしてサインインする

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

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

追加のユーザーにアクセス権を割り当てる

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

   手順については、「AWS IAM Identity Center ユーザーガイド」の「権限設定を作成する」を参照してください。

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

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

キーペアを作成する

クラスターをデプロイするために、 は Amazon EC2 インスタンス AWS ParallelCluster を起動してクラスターヘッドノードとコンピューティングノードを作成します。ジョブの実行やモニタリング、ユーザーの管理などのクラスタータスクを実行するには、クラスターヘッドノードにアクセスできる必要があります。SSH を使用してヘッドノードインスタンスにアクセスできることを確認するには、Amazon EC2 キーペアを使用する必要があります。キーペアの作成については、Linux インスタンス用の「Amazon Elastic Compute Cloud ユーザーガイド」の「キーペアを作成する」を参照してください。

AWS ParallelCluster コマンドラインインターフェイスのインストール (CLI)

AWS ParallelCluster は Python パッケージとして配布され、Python pipパッケージマネージャーを使用してインストールされます。Python パッケージのインストールの詳細については、「Python Packaging User Guide」の「Installing packages」を参照してください。

のインストール方法 AWS ParallelCluster：

- 仮想環境に AWS ParallelCluster をインストールする (推奨)
- pip を使用して AWS ParallelCluster 非仮想環境に をインストールする
- スタンドアロンアプリケーション AWS ParallelCluster として をインストールする

最新の CLI のバージョン番号は、GitHub のリリースページで確認できます。このガイドのコマンド例では、Python バージョン 3.6 以降がインストールされていることを前提としています。pip コマンド例は pip3 バージョンを使用します。

AWS ParallelCluster 2 と AWS ParallelCluster 3 の両方を管理する

AWS ParallelCluster 2 と AWS ParallelCluster 3 の両方を使用し、両方のパッケージの CLIsを管理する場合は、異なる仮想環境に AWS ParallelCluster 2 と AWS ParallelCluster 3 をインストールすることをお勧めします。これにより、各バージョンの AWS ParallelCluster と関連するクラスターリソースを継続して使用することができます。

仮想環境に AWS ParallelCluster をインストールする (推奨)

要件バージョン AWS ParallelCluster が他のpipパッケージと競合しないように、仮想環境に をインストールすることをお勧めします。

前提条件

- AWS ParallelCluster には Python 3.7 以降が必要です。まだインストールされていない場合は、python.org からお使いのプラットフォームに対応したバージョンをダウンロードしてください。

AWS ParallelCluster 仮想環境に をインストールするには

1. `virtualenv` がインストールされていない場合は、`pip3` を使用して `virtualenv` をインストールします。`python3 -m virtualenv help` がヘルプ情報を表示する場合は、ステップ 2 に進みます。

```
$ python3 -m pip install --upgrade pip
$ python3 -m pip install --user --upgrade virtualenv
```

   `exit` を実行して現在のターミナルウィンドウを終了し、新しいターミナルウィンドウを開いて環境への変更を取得します。

2. 仮想環境を作成して名前を付けます。

```
$ python3 -m virtualenv ~/apc-ve
```

   または、`-p` オプションを使用して Python の特定のバージョンを指定することもできます。

```
$ python3  -m virtualenv -p $(which python3) ~/apc-ve
```

3. 新しい仮想環境をアクティブ化します。

```
$ source ~/apc-ve/bin/activate
```

4. AWS ParallelCluster 仮想環境に をインストールします。

```
(apc-ve)~$ python3 -m pip install --upgrade "aws-parallelcluster"
```

5. Node Version Manager と最新の長期サポート (LTS) Node.js バージョンをインストールします。 テンプレートの生成には CloudFormation に Node.js AWS Cloud Development Kit (AWS CDK) が必要です。

> **Note**
>
> Node.js のインストールがご使用のプラットフォームで動作しない場合は、最新の LTS バージョンより前の LTS バージョンをインストールできます。詳細については、「Node.js release schedule」および「AWS CDK prerequisites」を参照してください。
> Node.js インストールコマンドの例:

```
$  nvm install --lts=Hydrogen
```

```
$ curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.38.0/install.sh | bash
$ chmod ug+x ~/.nvm/nvm.sh
$ source ~/.nvm/nvm.sh
$ nvm install --lts
$ node --version
```

6. が正しくインストール AWS ParallelCluster されていることを確認します。

```
$ pcluster version
{
  "version": "3.13.2"
}
```

`deactivate` コマンドを使用して、仮想環境を終了できます。新しいセッションを開始するたびに、環境を再度アクティブ化する必要があります。

の最新バージョンにアップグレードするには AWS ParallelCluster、インストールコマンドを再度実行します。

```
(apc-ve)~$ python3 -m pip install --upgrade "aws-parallelcluster"
```

pip を使用して AWS ParallelCluster 非仮想環境に をインストールする

Python パッケージ AWS ParallelCluster のパッケージマネージャーである pip を使用して、非仮想環境に をインストールすることもできます。

前提条件

- AWS ParallelCluster には Python 3.7 以降が必要です。まだインストールされていない場合は、python.org からお使いのプラットフォームに対応したバージョンをダウンロードしてください。

のインストール AWS ParallelCluster

1. を使用して をインストールpipします AWS ParallelCluster。

```
$ python3 -m pip install "aws-parallelcluster" --upgrade --user
```

   `--user` スイッチを使用すると、 は AWS ParallelCluster にpipインストールします`~/.local/bin`。

2. Node Version Manager と最新の長期サポート (LTS) Node.js バージョンをインストールします。 テンプレートの生成には CloudFormation に Node.js AWS Cloud Development Kit (AWS CDK) が必要です。

> **Note**
>
> Node.js のインストールがご使用のプラットフォームで動作しない場合は、最新の LTS バージョンより前の LTS バージョンをインストールできます。詳細については、「Node.js release schedule」および「AWS CDK prerequisites」を参照してください。
>
> ```
> $  nvm install --lts=Hydrogen
> ```

```
$ curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.38.0/install.sh | bash
$ chmod ug+x ~/.nvm/nvm.sh
$ source ~/.nvm/nvm.sh
$ nvm install --lts
$ node --version
```

3. が正しく AWS ParallelCluster インストールされていることを確認します。

```
$ pcluster version
{
  "version": "3.13.2"
}
```

4. 最新バージョンにアップグレードするには、インストールコマンドを再び実行します。

```
$ python3 -m pip install "aws-parallelcluster" --upgrade --user
```

スタンドアロンアプリケーション AWS ParallelCluster として をインストールする

をスタンドアロンアプリケーション AWS ParallelCluster として環境にインストールします。次のセクションの使用可能な OS AWS ParallelCluster に をインストールする手順に従います。

前提条件

- 使用可能なバージョンのインストーラと互換性のあるオペレーティングシステムを備えた環境。

> **Note**
>
> AWS ParallelCluster には NodeJS が必要です。 AWS ParallelCluster Installer には NodeJS (v18) のバンドルバージョンが含まれており、まだ存在しない場合はインストールされます。システムが NodeJS v18 と互換性がない場合は、インストールする前に NodeJS をインストールする必要があります AWS ParallelCluster。

リナックス

Linux x86 (64-bit)

を環境にインストール AWS ParallelCluster します。

1. 最新の pcluster インストーラ (チェックサム) をダウンロードします。
2. インストーラバンドルを解凍し、次のコマンド AWS ParallelCluster を使用して をインストールします。

```
$ unzip pcluster-installer-bundle-<VERSION>-Linux_x86_64-signed.zip -d pcluster-installer-bundle
$ cd pcluster-installer-bundle
$ chmod +x install_pcluster.sh
```

3. 次のインストールスクリプトを実行します。

```
$ bash install_pcluster.sh
```

4. が正しくインストール AWS ParallelCluster されていることを確認します。

```
$ pcluster version
```

```
{
    "version": "3.13.2"
}
```

pcluster インストールエラーのトラブルシューティング

- ステップ 4 で AWS ParallelCluster バージョンが返されない場合は、ターミナルまたは `source` を再起動`bash_profile`して、次の例に示すように、`PATH`変数を更新して新しいバイナリディレクトリを含めます。

```
$ source ~/.bash_profile
```

- `pcluster` インストールを使用して S3 URI ではなく HTTPS リソースとして `CustomActions` を指定してクラスターを作成すると、これらのリソースは検証されていない可能性があることを示す `WARNING` メッセージが表示されることがあります ([`SSL: CERTIFICATE_VERIFY_FAILED`])。これは既知の問題によるもので、指定したリソースの信頼性が確認できる場合は、この警告は無視して構いません。

以前のインストーラバンドルのバージョン

- なし

インストール後に実行する手順

このセクションでは、 がインストールされていることを前提としています AWS ParallelCluster。が正しく AWS ParallelCluster インストールされていることを確認する方法、最新バージョンの に更新する方法 AWS ParallelCluster、およびアンインストールする方法について説明します。

を実行して、 が正しくインストール AWS ParallelCluster されたことを確認できます`pcluster version`。

```
$ pcluster version
{
"version": "3.13.2"
}
```

AWS ParallelCluster は定期的に更新されます。の最新バージョンに更新するには AWS ParallelCluster、インストールコマンドを再度実行します。の最新バージョンの詳細については AWS ParallelCluster、AWS ParallelCluster リリースノートを参照してください。

```
$ pip3 install aws-parallelcluster --upgrade --user
```

をアンインストールするには AWS ParallelCluster、 を使用しますpip3 uninstall。

```
$ pip3 uninstall aws-parallelcluster
```

Python と pip3 がインストールされていない場合は、使用している環境に応じた手順に従ってください。

PCUI のインストール

AWS ParallelCluster UI (PCUI) は、コンソールのようなエクスペリエンスを提供しながら CLI をミラー AWS ParallelCluster pclusterリングするウェブベースのユーザーインターフェイスです。に PCUI をインストールしてアクセスします AWS アカウント。これを実行すると、PCUI は の Amazon AWS ParallelCluster API Gateway でホストされている API のインスタンスにアクセスします AWS アカウント。 Amazon API Gateway PCUI の詳細については、「AWS ParallelCluster UI」を参照してください。

前提条件:

- が必要です AWS アカウント
- へのアクセス権が必要です AWS Management Console

詳細については、「のセットアップ AWS アカウント」を参照してください。

トピック

- PCUI をインストールする
- スタックパラメータ
- カスタムドメインを設定する
- Amazon Cognito ユーザープールのオプション
- AWS ParallelCluster および PCUI バージョンを特定する
- PCUI のコスト

PCUI をインストールする

UI (PCUI) AWS ParallelCluster のインスタンスをインストールするには、クラスターを作成する AWS リージョン の AWS CloudFormation クイック作成リンクを選択します。クイック作成 URL から [スタック作成ウィザード] に移動し、クイック作成スタックテンプレートの入力を行い、スタックをデプロイします。CloudFormation クイック作成スタックの詳細については、「AWS CloudFormation ユーザーガイド」の「スタックのクイック作成リンクを作成する」を参照してください。

> **Note**
>
> PCUI のインストールに使用するのと同じ AWS ParallelCluster バージョンでのみ、クラスターを作成および編集したり、イメージを構築したりできます。

AWS CloudFormation クイック作成リンクを使用して、ネストされた Amazon Cognito、API Gateway、Amazon EC2 Systems Manager スタックを持つ PCUI スタックをデプロイします。

1. AWS Management Consoleにサインインします。
2. リストから AWS リージョン クイック作成リンクを選択して、PCUI をデプロイします。これにより、コンソールの CloudFormation [スタック作成ウィザード] が表示されます。

   - us-east-1
   - us-east-2
   - us-west-1
   - us-west-2
   - eu-west-1
   - eu-west-2
   - eu-west-3
   - eu-central-1
   - eu-north-1
   - me-south-1
   - sa-east-1
   - ca-central-1
   - ap-northeast-1

- ap-northeast-2
- ap-south-1
- ap-southeast-1
- ap-southeast-2
- us-gov-west-1

3. 管理者の E メールと ParallelCluster バージョンの有効な E メールアドレスを入力します。ParallelCluster

   デプロイが正常に完了すると、PCUI はこの E メールアドレスに一時的なパスワードを送信し、これを使用して PCUI にアクセスできます。一時パスワードを保存または使用する前に E メールを削除した場合は、スタックを削除して PCUI を再インストールする必要があります。

4. フォームの残りの部分は空白のままにするか、(オプションの) パラメータの値を入力して PCUI ビルドをカスタマイズします。
5. 後のステップで使用するために、スタックの名前をメモします。
6. [機能] に移動します。CloudFormation の機能に同意します。
7. [作成] を選択します。 AWS ParallelCluster API と PCUI のデプロイが完了するまでに約 15 分かかります。
8. スタックの作成時にスタックの詳細を表示します。
9. デプロイが完了したら、入力したアドレスに送信され、一時パスワードを含む管理者 E メールを開きます。これを使用して PCUI にアクセスします。(PCUI にログインする前に E メールを完全に削除した場合は、作成した PCUI スタックを削除して PCUI を再インストールする必要があります。
10. スタックの AWS CloudFormation コンソールリストで、前のステップでメモしたスタック名へのリンクを選択します。
11. スタックの詳細で、出力を選択し、***Stackname*** URL という名前のキーのリンクを選択して PCUI を開きます (***Stackname*** は前のステップでメモした名前です) 。
12. 一時パスワードを入力します。手順に従って独自のパスワードを作成し、再度ログインします。
13. これで、 AWS リージョン 選択した の PCUI のホームページが表示されます。
14. PCUI の使用を開始するには、「PCUI によるクラスターの設定と作成」を参照してください。

> **Note**
>
> PCUI セッションのデフォルトの継続時間は 5 分です。これは、PCUI 2023.12.0 の時点で Cognito が提供する最小値です。したがって、Cognito ユーザープールから削除されたユーザーは、セッションの有効期限が切れるまでシステムにアクセスできることが予想されます。

スタックパラメータ

AdditionalPoliciesPCAPI:

説明: (オプション) ParallelCluster Lambda 関数のデフォルトの実行ロールにアタッチする追加の IAM ポリシーの ARN。指定できるポリシーは 1 つだけです。

タイプ: 文字列

デフォルト: ''

AllowedPattern: "^(arn:.*:iam::.*:policy\\/([a-zA-Z0-9_-]+))|()$"

AdminUserEmail:

説明: デフォルトで設定する管理ユーザーの E メールアドレス (新しい Cognito インスタンスでのみ) 。

タイプ: 文字列

デフォルト: ''

CognitoCustomDomain:

説明: (オプション) Cognito のカスタムドメイン名。省略すると、デフォルトの Cognito ドメイン名が使用されます。

タイプ: 文字列

デフォルト: ''

CognitoCustomDomainCertificateArn:

説明: (オプション) Cognito カスタムドメイン用に発行された ACM 証明書の ARN。これは、`CognitoCustomDomain`が指定されている場合にのみ必要です。

タイプ: 文字列

デフォルト: ''

CustomDomain:

説明: (オプション) カスタムドメイン名。省略すると、デフォルトのドメイン名が使用されます。

タイプ: 文字列

デフォルト: ''

CustomDomainCertificateArn:

説明: (オプション) カスタムドメイン用に発行された ACM 証明書の ARN。これは、`CustomDomain`が指定されている場合にのみ必要です。

タイプ: 文字列

デフォルト: ''

IAMRoleAndPolicyPrefix:

説明: すべての IAM ロールとポリシーの名前に適用されるプレフィックス (最大長: 10)。[ParallelCluster >= 3.8.0]

タイプ: 文字列

デフォルト: ''

MaxLength: 10

ImageBuilderSubnetId:

説明: (オプション) コンテナイメージの構築に使用するサブネットを選択します。サブネットはパブリックで、パブリック IPs を自動割り当てする必要があります。選択しない場合、デフォルトのサブネットが使用されます。

タイプ: 文字列

デフォルト: ''

ImageBuilderVpcId:

説明: (オプション) コンテナイメージの構築に使用する VPC を選択します。選択しない場合、デフォルトの VPC が使用されます。

タイプ: 文字列

デフォルト: ''

InfrastructureBucket:

説明: (オプション) CloudFormation ファイルが保存されている S3 バケット。このパラメータは、インフラストラクチャ自体に加えられた変更をテストする場合にのみ変更します。

タイプ: 文字列

デフォルト: ''

LambdaSecurityGroupIds:

説明: PCUI Lambda 関数に関連付けるセキュリティグループのカンマ区切りリスト。

タイプ: CommaDelimitedList

デフォルト: ''

LambdaSubnetIds:

説明: PCUI Lambda 関数に関連付けるサブネット IDs のカンマ区切りリスト。これらのサブネットはプライベートで、VPC エンドポイントに関連付けられている必要があります。

タイプ: CommaDelimitedList

デフォルト: ''

PermissionsBoundaryPolicy:

説明: ParallelCluster UI インフラストラクチャによって作成されたすべての IAM ロールのアクセス許可の境界として使用する IAM ポリシーの ARN。

タイプ: 文字列

デフォルト: ''

AllowedPattern: "^(arn:.*:iam::.*:policy\\/([a-zA-Z0-9_-]+))|()$"

PermissionsBoundaryPolicyPCAPI:

説明: ParallelCluster API インフラストラクチャによって作成されたすべての IAM ロールのアクセス許可の境界として使用する IAM ポリシーの ARN。[ParallelCluster >= 3.8.0]

タイプ: 文字列

デフォルト: ''

AllowedPattern: "^(arn:.*:iam::.*:policy\\/([a-zA-Z0-9_-]+))|()$"

PublicEcrImageUri:

説明: 指定した場合、ParallelCluster UI コンテナの Lambda の Docker イメージの URI。

タイプ: 文字列

デフォルト: public.ecr.aws/pcm/parallelcluster-ui:2024.11.0

SNSRole:

説明: 以前にデプロイされた PCUI Cognito スタックの SNSRole ARN。新しいものを作成するには、空白のままにします。

タイプ: 文字列

デフォルト: ''

UserPoolAuthDomain:

説明: 以前にデプロイされた PCUI Cognito ユーザープールの UserPoolAuthDomain。新しいものを作成するには、空白のままにします。

タイプ: 文字列

デフォルト: ''

UserPoolId:

説明: 以前にデプロイされた PCUI Cognito ユーザープールの UserPoolId。新しいものを作成するには、空白のままにします。

タイプ: 文字列

デフォルト: ''

バージョン：

説明: デプロイする AWS ParallelCluster のバージョン。

タイプ: 文字列

デフォルト: 3.11.1

VpcEndpointId:

説明: execute-api サービスのタイプインターフェイスを持つ VPC エンドポイントを入力して、プライベート PCUI 実装を有効にします。有効にすると、API は指定された VPC 内からのリクエストのみを受け入れます。

タイプ: 文字列

デフォルト: "

カスタムドメインを設定する

UI (PCUI) AWS ParallelCluster のカスタムドメインを設定する方法について説明します。UI は AWS アカウントの Amazon API Gateway でホストされています。カスタムドメインは、API Gateway コンソールで設定できます。

前提条件:

- があります AWS アカウント。
- ドメインを所有している。
- ドメインルートは解決可能である。たとえば、カスタムドメイン`xyz.example.com`として を使用する場合、ルートドメインは解決可能である`example.com`必要があります。つまり、IP `example.com`を指すには、少なくとも A レコードが必要です。このようなレコードがない場合は、0.0.0.0 を指すダミーレコードを追加します。
- AWS Certificate Manager us-east-1 で Cognito のカスタムドメイン証明書を作成またはインポートした。
- ドメインネームシステム (DNS) 設定を変更するための必要なアクセス許可がある。

トピック

- PCUI をデプロイする
- DNS を設定する

PCUI をデプロイする

カスタムドメインを使用するように UI (PCUI) AWS ParallelCluster デプロイを作成または更新するには、次の手順を実行します。

この例では、カスタムドメイン `xyz.example.com` を使用して PCUI をデプロイし、カスタムドメイン `auth-xyz.example.com` を使用して Amazon Cognito インターフェイスをデプロイすることを前提としています。

> **Note**
>
> Amazon Cognito ドメインのカスタマイズは必須ではなく、デフォルト値のままにしておくこともできます。このカスタマイズは、完全性のために含まれています。

> **Note**
>
> Amazon Cognito のカスタムドメインは、 AWS GovCloud (US) Regionsではサポートされていません。

これを行うには、次のパラメータを使用して PCUI スタックをデプロイします。

- CustomDomain: *`xyz.example.com`*。
- CustomDomainCertificateArn: *`xyz.example.com`* の ACM 証明書 Amazon リソースネーム (ARN)。
- CognitoCustomDomain: *`auth-xyz.example.com`*。
- CognitoCustomDomainCertificateArn を *`xyz.example.com`* の ACM 証明書 Amazon リソースネーム (ARN) に送信します。

DNS を設定する

次の手順を実行して、UI (PCUI) AWS ParallelCluster と Amazon Cognito が目的のカスタムドメインに応答するようにドメイン名サービス (DNS) を設定します。

この例では、カスタムドメイン `xyz.example.com` を使用して PCUI をデプロイし、カスタムドメイン `auth-xyz.example.com` を使用して Amazon Cognito インターフェイスをデプロイすることを前提としています。

1. 次のパラメータを使用して PCUI スタックをデプロイします。

   - CustomDomainEndpoint: 出力で指定されたエイリアスを指す A レコードを *`xyz.example.com`* の DNS に作成します。

- CognitoCustomDomainEndpoint: 出力で指定されたエイリアスを指す A レコードを *auth-xyz.example.com* の DNS に作成します。

2. DNS の変更が配布されるまで約 10 分待ちます。
3. DNS の変更が配布されると、PCUI は *xyz.example.com/pcui* で応答し、Amazon Cognito 認証ページは *auth-xyz.example.com* で応答します。

Amazon Cognito ユーザープールのオプション

以下のセクションでは、CloudFormation クイック作成リンクまたはクイック作成 URL について説明します。クイック作成 URL を使用すると、スタック作成ウィザードに移動し、クイック作成スタックテンプレートの入力を指定してスタックをデプロイできます。CloudFormation クイック作成スタックの詳細については、「AWS CloudFormation ユーザーガイド」の「スタックのクイック作成リンクを作成する」を参照してください。

複数の UI (PCUI) AWS ParallelCluster インスタンスで使用できる Amazon Cognito ユーザープールを維持するには、次のオプションを検討してください。

- ネストされた CloudFormation スタックから作成された Amazon Cognito ユーザープールにリンクする既存の PCUI インスタンスを使用します。これは、クイック作成リンクを使用して PCUI をデプロイするときに作成され、すべての Amazon Cognito パラメータを空白のままにします。
- まず、スタンドアロンの Amazon Cognito ユーザープールをデプロイします。次に、デプロイしたスタンドアロンの Amazon Cognito ユーザープールにリンクされた新しい PCUI インスタンスをデプロイします。この方法では、Amazon Cognito のデプロイを PCUI のデプロイから分離します。ネストされていない PCUI CloudFormation スタックは更新が容易であることに注意してください。

既存の Amazon Cognito ユーザープールを新しい PCUI インスタンスで使用する

既存の Amazon Cognito ユーザープールを新しい PCUI インスタンスで使用するには、次の手順を実行します。

既存の Amazon Cognito ユーザープールを新しい PCUI インスタンスで使用する

1. CloudFormation コンソールで、複数の PCUI インスタンスで使用する Amazon Cognito ユーザープールが含まれている PCUI スタックを選択します。
2. Amazon Cognito ユーザープールを作成したネストされたスタックに移動します。

3. [出力] タブを選択します。
4. 以下のパラメータの値をコピーします。
   - `UserPoolId`
   - `UserPoolAuthDomain`
   - `SNSRole`
5. クイック作成リンクを使用して新しい PCUI インスタンスをデプロイし、コピーした出力をすべての `External PCUI Amazon Cognito` パラメータに入力します。これにより、新しい PCUI スタックは、新しいプールを作成しなくなり、ネストされたスタックから作成した既存の Amazon Cognito ユーザープールにリンクされます。同じパラメータ値を持つ後続の新しい PCUI インスタンスをデプロイし、Amazon Cognito ユーザープールにリンクできます。

スタンドアロンの Amazon Cognito ユーザープールを作成する

スタンドアロン Amazon Cognito ユーザープールを作成するには

1. 次のリストから PCUI インスタンスをデプロイする同じ AWS リージョン というラベルの付いたクイック作成リンクを選択して、Amazon Cognito 専用スタックを起動します。
   - us-east-1
   - us-east-2
   - us-west-1
   - us-west-2
   - eu-west-1
   - eu-west-2
   - eu-west-3
   - eu-central-1
   - eu-north-1
   - me-south-1
   - sa-east-1
   - ca-central-1
   - ap-northeast-1
   - ap-northeast-2
   - ap-south-1

- ap-southeast-1
- ap-southeast-2
- us-gov-west-1

2. スタックの作成が完了したら、[出力] タブを選択し、以下のパラメータの値をコピーします。
   - `UserPoolId`
   - `UserPoolAuthDomain`
   - `SNSRole`
3. PCUI クイックスタートリンクを選択し、コピーした値をすべての `External PCUI Amazon Cognito` パラメータに入力して、新しい PCUI インスタンスをデプロイします。新しい PCUI インスタンスは、スタンドアロンの Amazon Cognito ユーザープールにリンクされ、ネストされたスタックや新しいユーザープールは作成されません。同じパラメータ値を持つ後続の新しい PCUI インスタンスをデプロイし、スタンドアロンの Amazon Cognito ユーザープールにリンクできます。

AWS ParallelCluster および PCUI バージョンを特定する

AWS ParallelCluster および PCUI バージョンを識別するには：

1. CloudFormation コンソールで、PCUI スタックを選択します。
2. [パラメータ] タブを選択します。
3. AWS ParallelCluster バージョンは、 パラメータ Version の値です。
4. PCUI バージョンは PublicEcrImageUri 値の末尾にあります。例えば、値が `public.ecr.aws/pcui/parallelcluster-ui-awslambda:2023.02` の場合、バージョンは `2023.02` です。

> **Note**
>
> PCUI を AWS ParallelCluster 最新バージョンに更新するには、クイック作成リンクを選択して新しいスタックを起動します。

PCUI のコスト

PCUI はサーバーレスアーキテクチャ上に構築されており、ほとんどの場合、 AWS 無料利用枠カテゴリ内で使用できます。次の表に、PCUI が依存 AWS のサービス する とその無料利用枠の制限を示します。一般的な使用量では、1 か月あたりのコストは 1 USD 未満と推定されます。

サービス	AWS 無料利用枠
Amazon Cognito	50,000 人の月間アクティブユーザー
Amazon API Gateway	100 万回の REST API コール
AWS Lambda	毎月 100 万件の無料リクエスト、毎月 40 万 GB 秒のコンピューティング時間
EC2 イメージビルダー	EC2 以外は無料
Amazon Elastic Compute Cloud	15 分の 1 回限りのコンテナイメージビルド
AWS CloudFormation	5 GB のデータ (取り込み、アーカイブストレージ、および Logs Insights クエリでスキャンされたデータ)

の開始方法 AWS ParallelCluster

コマンドラインインターフェイス (CLI) またはウェブベースのユーザーインターフェイス (UI) を使用して AWS ParallelCluster クラスター AWS ParallelCluster を設定および作成するには。PCUI は、リリース 3.5.0 で追加されました。

トピック

- 「Configure and create a cluster with the AWS ParallelCluster command line interface」
- PCUI によるクラスターの設定と作成
- クラスターに接続する

「Configure and create a cluster with the AWS ParallelCluster command line interface」

をインストールしたら AWS ParallelCluster、以下の設定ステップを完了します。

1. AWS アカウントに CLI `pcluster` の実行に必要なアクセス許可を含むロールがあることを確認します。詳細については、「AWS ParallelCluster `pcluster` ユーザーポリシーの例」を参照してください。
2. AWS 認証情報を設定します。詳細については、「AWS CLI ユーザーガイド」の「AWS CLIを設定する」を参照してください。

   ```
   $ aws configure
   AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
   AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
   Default region name [us-east-1]: us-east-1
   Default output format [None]:
   ```

3. クラスターを起動 AWS リージョン する には、少なくとも 1 つの Amazon EC2 キーペアが必要です。詳細については、「Linux インスタンス用 Amazon Elastic Compute Cloud ユーザーガイド」の「Amazon Elastic Compute Cloud のキーペア」を参照してください。

AWS ParallelCluster コマンドラインインターフェイス (CLI) を使用する場合、 AWS ParallelCluster イメージとクラスターを作成または更新したときに作成された AWS リソースに対してのみ料金が発生します。詳細については、「AWS が使用する サービス AWS ParallelCluster」を参照してください。

最初のクラスターを設定して作成する

最初のクラスターを作成するには、 `pcluster configure` CLI コマンドを使用して、クラスターの設定と作成に必要なすべての情報の入力を求めるウィザードを開始します。スケジューラ AWS Batch として を使用する場合と を使用する場合では、シーケンスの詳細が異なりますSlurm。

スラム

```
$  pcluster configure --config cluster-config.yaml
```

有効な AWS リージョン 識別子のリストから、クラスターを実行する AWS リージョン を選択します。

> **Note**
>
> AWS リージョン 表示される のリストは、アカウントのパーティションに基づいており、アカウントで有効 AWS リージョン になっている のみが含まれます。アカウント AWS リージョン で を有効にする方法の詳細については、 の「 の管理 AWS リージョン」を参照してくださいAWS 全般のリファレンス。示されている例は、 AWS グローバルパーティションからのものです。アカウントが AWS GovCloud (US) パーティションにある場合、そのパーティション AWS リージョン の のみが表示されます (`gov-us-east-1` および `gov-us-west-1`) 。同様に、アカウントが AWS 中国パーティションにある場合、 `cn-northwest-1` `cn-north-1`と のみが表示されます。でサポートされている の完全なリスト AWS リージョン については AWS ParallelCluster、「」を参照してくださいAWS リージョン でサポートされる AWS ParallelCluster。

```
Allowed values for AWS ##### ID:
1. af-south-1
2. ap-east-1
3. ap-northeast-1
4. ap-northeast-2
5. ap-south-1
6. ap-southeast-1
7. ap-southeast-2
8. ap-southeast-3
9. ap-southeast-5
10. ap-southeast-7
11. ca-central-1
12. eu-central-1
13. eu-north-1
14. eu-south-1
15. eu-west-1
16. eu-west-2
17. eu-west-3
18. il-central-1
19. me-south-1
20. sa-east-1
21. us-east-1
22. us-east-2
23. us-west-1
24. us-west-2
AWS ##### ID [ap-northeast-1]:
```

選択した で Amazon Elastic Compute Cloud に登録されているキーペアからキーペアを選択します AWS リージョン。

```
Allowed values for Amazon EC2 Key Pair Name:
1. your-key-1
2. your-key-2
Amazon EC2 Key Pair Name [your-key-1]:
```

クラスターで使用するスケジューラを選択します。

```
Allowed values for Scheduler:
1. slurm
2. awsbatch
Scheduler [slurm]:
```

オペレーティングシステムを選択します。

```
Allowed values for Operating System:
1. alinux2
2. alinux2023
3. ubuntu2404
4. ubuntu2204
5. ubuntu2004
6. rhel8
7. rhel9
Operating System [alinux2]:
```

ヘッドノードインスタンスタイプを選択します。

```
Head node instance type [t2.micro]:
```

キューの構成を選択します。注意: 同一キュー内の複数のコンピューティングリソースに対してインスタンスタイプを指定することはできません。

```
Number of queues [1]:
Name of queue 1 [queue1]:
Number of compute resources for queue1 [1]: 2
Compute instance type for compute resource 1 in queue1 [t2.micro]:
Maximum instance count [10]:
```

EFA を有効にして、追加料金なしで高レベルのインスタンス間通信を必要とするアプリケーションを大規模 AWS に実行できます。

- Elastic Fabric Adapter (EFA) をサポートするインスタンスタイプを選択します。
- EFA を有効にします。
- 既存のプレイスメントグループ名を指定します。空白のままにすると、 によって AWS ParallelCluster 自動的に作成されます。

```
Compute instance type for compute resource 2 in queue1 [t2.micro]: c5n.18xlarge
Enable EFA on c5n.18xlarge (y/n) [y]: y
Maximum instance count [10]:
Placement Group name []:
```

前のステップが完了したら、既存の VPC を使用するか、 が VPC AWS ParallelCluster を作成するかを決定します。適切に設定された VPC がない場合は、 で新しい VPC AWS ParallelCluster を作成できます。ヘッダーノードとコンピューティングノードの両方を同じパブリックサブネット内に配置するか、ヘッドノードのみをパブリックサブネット内に配置し、すべてのコンピューティングノードをプライベートサブネット内に配置します。 AWS ParallelCluster が VPC を作成できるようにする場合は、すべてのノードをパブリックサブネットに配置するかどうかを決定する必要があります。詳細については、「ネットワークの設定」を参照してください。

複数のネットワークインターフェイスまたはネットワークカードを備えたインスタンスタイプを使用するようにクラスターを設定する場合、追加のネットワーク要件については「ネットワークの設定」を参照してください。

AWS リージョンで許可されている VPC 数のクォータに達する可能性があります。デフォルトのクォータは、 AWS リージョンあたり 5 つの VPC です。このクォータと引き上げをリクエストする方法の詳細については、「Amazon VPC ユーザーガイド」の「VPC とサブネット」を参照してください。

> **Important**
>
> によって作成された VPCs、デフォルトでは VPC フローログを有効に AWS ParallelCluster しません。VPC フローログは、VPC のネットワークインターフェイスとの間で行き来する IP トラフィックに関する情報をキャプチャすることができます。詳細については、「Amazon VPC User Guide」(Amazon VPC ユーザーガイド) の「VPC Flow Logs」(VPC フローログ) を参照してください。

VPC AWS ParallelCluster を作成できるようにする場合は、すべてのノードをパブリックサブネットに配置するかどうかを決定してください。

> **Note**
>
> を選択した場合`1. Head node in a public subnet and compute fleet in a private subnet`、 は、無料利用枠リソースを指定しても、追加コストが発生する NAT ゲートウェイ AWS ParallelCluster を作成します。

```
Automate VPC creation? (y/n) [n]: y
Allowed values for Availability Zone:
1. us-east-1a
2. us-east-1b
3. us-east-1c
4. us-east-1d
5. us-east-1e
6. us-east-1f
Availability Zone [us-east-1a]:
Allowed values for Network Configuration:
1. Head node in a public subnet and compute fleet in a private subnet
2. Head node and compute fleet in the same public subnet
Network Configuration [Head node in a public subnet and compute fleet in a private
 subnet]: 1
Beginning VPC creation. Please do not leave the terminal until the creation is
 finalized
```

新しい VPC を作成しない場合、既存の VPC を選択する必要があります。

で VPC AWS ParallelCluster を作成する場合は、VPC ID を書き留めて、 を使用して後で AWS CLI 削除できるようにします。

```
Automate VPC creation? (y/n) [n]: n
Allowed values for VPC ID:
  #  id                     name                                number_of_subnets
---  ---------------------  ----------------------------------  -------------------
  1  vpc-0b4ad9c4678d3c7ad  ParallelClusterVPC-20200118031893                    2
  2  vpc-0e87c753286f37eef  ParallelClusterVPC-20191118233938                    5
VPC ID [vpc-0b4ad9c4678d3c7ad]: 1
```

VPC を選択したら、既存のサブネットを使用するか、新しいサブネットを作成するかを決定します。

```
Automate Subnet creation? (y/n) [y]: y
```

```
Creating CloudFormation stack...
Do not leave the terminal until the process has finished
```

AWS Batch

```
$  pcluster configure --config cluster-config.yaml
```

有効な AWS リージョン 識別子のリストから、クラスターを実行する AWS リージョン を選択します。

> **Note**
>
> AWS リージョン 表示される のリストは、アカウントのパーティションに基づいています。これには AWS リージョン 、アカウントで有効になっている のみが含まれます。アカウント AWS リージョン で を有効にする方法の詳細については、 の「 の管理 AWS リージョン」を参照してくださいAWS 全般のリファレンス。示されている例は、 AWS グローバルパーティションからのものです。アカウントが AWS GovCloud (US) パーティションにある場合、そのパーティション AWS リージョン 内の のみが表示されます (`gov-us-east-1` および `gov-us-west-1`) 。同様に、アカウントが AWS 中国パーティションにある場合、 `cn-northwest-1` `cn-north-1`と のみが表示されます。で AWS リージョン サポートされている の完全なリストについては AWS ParallelCluster、「」を参照してくださいAWS リージョン でサポートされる AWS ParallelCluster。

```
Allowed values for AWS ##### ID:
1. af-south-1
2. ap-east-1
3. ap-northeast-1
4. ap-northeast-2
5. ap-south-1
6. ap-southeast-1
7. ap-southeast-2
8. ap-southeast-3
```

```
9. ap-southeast-5
10. ap-southeast-7
11. ca-central-1
12. eu-central-1
13. eu-north-1
14. eu-south-1
15. eu-west-1
16. eu-west-2
17. eu-west-3
18. il-central-1
19. me-south-1
20. sa-east-1
21. us-east-1
22. us-east-2
23. us-west-1
24. us-west-2
AWS ##### ID [us-east-1]:
```

キーペアは、選択した AWS リージョンに Amazon EC2 で登録されているキーペアから選択されます。キーペアを選択します。

```
Allowed values for Amazon EC2 Key Pair Name:
1. your-key-1
2. your-key-2
Amazon EC2 Key Pair Name [your-key-1]:
```

クラスターで使用するスケジューラを選択します。

```
Allowed values for Scheduler:
1. slurm
2. awsbatch
Scheduler [slurm]: 2
```

`awsbatch` がスケジューラとして選択されている場合、`alinux2` がオペレーティングシステムとして使用されます。ヘッドノードのインスタンスタイプが入力されます。

```
Head node instance type [t2.micro]:
```

キューの構成を選択します。ス AWS Batch ケジューラには 1 つのキューのみが含まれます。コンピューティングノードのクラスターの最大サイズを入力します。これは vCPU 単位で測定されます。

```
Number of queues [1]:
Name of queue 1 [queue1]:
Maximum vCPU [10]:
```

既存の VPCs を使用するか、 が VPCs AWS ParallelCluster を作成するかを決定します。適切に設定された VPC がない場合は、 AWS ParallelCluster で新しい VPC を作成できます。同じパブリックサブネット内のヘッダーノードとコンピューティングノードの両方を使用するか、プライベートサブネット内のすべてのノードを持つパブリックサブネット内のヘッダーノードのみを使用します。リージョン内で許可されている VPC 数のクォータに達する可能性があります。VPC のデフォルト数は 5 です。このクォータと引き上げをリクエストする方法の詳細については、「Amazon VPC ユーザーガイド」の「VPC とサブネット」を参照してください。

> **Important**
>
> によって作成された VPCs、デフォルトでは VPC フローログを有効に AWS ParallelCluster しません。VPC フローログは、VPC のネットワークインターフェイスとの間で行き来する IP トラフィックに関する情報をキャプチャすることができます。詳細については、「Amazon VPC User Guide」(Amazon VPC ユーザーガイド) の「VPC Flow Logs」(VPC フローログ) を参照してください。

VPC AWS ParallelCluster を作成できるようにする場合は、すべてのノードをパブリックサブネットに配置するかどうかを決定してください。

> **Note**
>
> `1. Head node in a public subnet and compute fleet in a private subnet` を選択すると、 AWS ParallelCluster は NAT ゲートウェイを作成するため、無料利用枠のリソースを指定しても追加コストが発生します。

```
Automate VPC creation? (y/n) [n]: y
Allowed values for Availability Zone:
1. us-east-1a
2. us-east-1b
3. us-east-1c
4. us-east-1d
5. us-east-1e
```

```
6. us-east-1f
Availability Zone [us-east-1a]:
Allowed values for Network Configuration:
1. Head node in a public subnet and compute fleet in a private subnet
2. Head node and compute fleet in the same public subnet
Network Configuration [Head node in a public subnet and compute fleet in a private
 subnet]: *1*
Beginning VPC creation. Please do not leave the terminal until the creation is
 finalized
```

新しい VPC を作成しない場合、既存の VPC を選択する必要があります。

で VPC AWS ParallelCluster を作成する場合は、VPC ID を書き留めて、 AWS CLI または を使用して後で AWS Management Console 削除できるようにします。

```
Automate VPC creation? (y/n) [n]: n
Allowed values for VPC ID:
  #  id                     name                                 number_of_subnets
---  ---------------------  ---------------------------------  -------------------
  1  vpc-0b4ad9c4678d3c7ad  ParallelClusterVPC-20200118031893                    2
  2  vpc-0e87c753286f37eef  ParallelClusterVPC-20191118233938                    5
VPC ID [vpc-0b4ad9c4678d3c7ad]: 1
```

VPC を選択したら、既存のサブネットを使用するか、新しいサブネットを作成するかを決定する必要があります。

```
Automate Subnet creation? (y/n) [y]: y
```

```
Creating CloudFormation stack...
Do not leave the terminal until the process has finished
```

前述の手順が完了すると、VPC にシンプルなクラスターが起動します。VPC では、パブリック IP アドレスをサポートする既存のサブネットを使用しています。サブネットのルートテーブルは、`0.0.0.0/0 => igw-xxxxxx` です。以下の条件をご確認ください。

- VPC には `DNS Resolution = yes` と `DNS Hostnames = yes` が必要です。
- VPC は、 AWS リージョンに対して正しい `domain-name` がある DHCP オプションも必要です。デフォルトの DHCP オプションセットでは、必要な AmazonProvidedDNS がすでに指定されています。複数のドメインネームサーバーを指定する場合は、「Amazon VPC User Guide」(Amazon

VPC ユーザーガイド) の「DHCP options sets」(DHCP オプションセット) を参照してください。プライベートサブネットを使用する場合は、NAT ゲートウェイまたは内部プロキシを使用して、コンピューティングノードへのウェブアクセスを有効にします。詳細については、「ネットワークの設定」を参照してください。

すべての設定に有効な値が入力されたら、create コマンドを実行して、クラスターを起動することができます。

```
$ pcluster create-cluster --cluster-name test-cluster --cluster-configuration cluster-
config.yaml
{
  "cluster": {
    "clusterName": "test-cluster",
    "cloudformationStackStatus": "CREATE_IN_PROGRESS",
    "cloudformationStackArn": "arn:aws:cloudformation:eu-west-1:xxx:stack/test-cluster/
abcdef0-f678-890a-5abc-021345abcdef",
    "region": "eu-west-1",
    "version": "3.13.2",
    "clusterStatus": "CREATE_IN_PROGRESS"
  },
  "validationMessages": []
}
```

クラスターの進行状況に従います。

```
$ pcluster describe-cluster --cluster-name test-cluster
```

or

```
$ pcluster list-clusters --query 'clusters[?clusterName==`test-cluster`]'
```

クラスターが `"clusterStatus": "CREATE_COMPLETE"` ステータスになったら、通常の SSH クライアント設定を使用して接続できます。Amazon EC2 インスタンスへの接続の詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EC2 の使用を開始する」を参照してください。または、次のようにクラスターに接続することもできます。

```
$ pcluster ssh --cluster-name test-cluster -i ~/path/to/keyfile.pem
```

次のコマンドを実行して、クラスターを削除します。

```
$ pcluster delete-cluster --region us-east-1 --cluster-name test-cluster
```

クラスターを削除した後、CloudFormation ネットワークスタックを削除することで、VPC 内のネットワークリソースを削除できます。スタックの名前は「parallelclusternetworking-」で始まり、「YYYYMMDDHHMMSS」のフォーマットで作成時刻が含まれます。list-stacks コマンドでスタックを一覧表示できます。

```
$ aws --region us-east-1 cloudformation list-stacks \
   --stack-status-filter "CREATE_COMPLETE" \
   --query "StackSummaries[].StackName" | \
   grep -e "parallelclusternetworking-"
   "parallelclusternetworking-pubpriv-20191029205804"
```

スタックは、delete-stack コマンドで削除することができます。

```
$ aws --region us-east-1 cloudformation delete-stack \
   --stack-name parallelclusternetworking-pubpriv-20191029205804
```

pcluster configure が作成する VPC は、CloudFormation ネットワークスタックでは作成されません。この VPC は、コンソールで手動で削除することも、 AWS CLIを使用して削除することもできます。

```
$ aws --region us-east-1 Amazon EC2 delete-vpc --vpc-id vpc-0b4ad9c4678d3c7ad
```

PCUI によるクラスターの設定と作成

PCUI は、コンソールのようなエクスペリエンスを提供しながら CLI `pcluster` をミラーリング AWS ParallelCluster するウェブベースのユーザーインターフェイスです。に PCUI をインストールしてアクセスします AWS アカウント。これを実行すると、PCUI は の Amazon AWS ParallelCluster API Gateway でホストされている API のインスタンスにアクセスします AWS アカウント。 Amazon API Gateway

> **Note**
>
> PCUI ウィザードには、サポートされている最新バージョンのすべての機能の UI オプションがない場合があります AWS ParallelCluster 。必要に応じて設定ファイルを手動で編集することも、 CLI AWS ParallelCluster を使用することもできます。

このセクションでは、PCUI を使用してクラスターを設定および作成する方法について説明します。

前提条件:

- 実行中の PCUI インスタンスへのアクセス権。詳細については、「PCUI のインストール」を参照してください。

クラスターの設定と作成

1. PCUI の [クラスター] ビューで、[クラスターを作成]、[ステップバイステップ] の順に選択します。
2. [クラスター] で、[名前] にクラスター名を入力します。
3. クラスターのパブリックサブネットを持つ VPC を選択し、次へを選択します。
4. Head ノードで、「SSM セッションの追加」を選択し、「次へ」を選択します。
5. [キュー] の [コンピューティングリソース] で、[スタティックノード] に [1] を選択します。
6. インスタンスタイプで、選択したデフォルトのインスタンスタイプを削除し、t2.micro を選択し、次へを選択します。
7. [ストレージ] で [次へ] を選択します。
8. [クラスター設定] で、クラスター設定 YAML を確認し、[ドライラン] を選択して検証します。
9. [作成] を選択し、検証済みの構成に基づいてクラスターを作成します。
10. 数秒後、PCUI は自動的に [クラスター] に戻り、クラスターの作成ステータスとスタックイベントをモニタリングできるようになります。
11. [詳細] を選択して、バージョンやステータスなどのクラスターの詳細を確認します。
12. [インスタンス] を選択して、Amazon EC2 インスタンスおよびステータスのリストを確認します。
13. [スタックイベント] を選択して、クラスターのスタックイベントと、クラスターを作成する CloudFormation スタックへの AWS Management Console のリンクを確認します。
14. [詳細] で、クラスターの作成が完了したら、[YAML を表示] を選択してクラスター設定 YAML ファイルを表示またはダウンロードします。
15. クラスターの作成が完了したら、[シェル] を選択してクラスターヘッドノードにアクセスします。

> **Note**
>
> シェルを選択すると、 は Amazon EC2 Systems Manager セッション AWS ParallelCluster を開き、 `ssm-user`を に追加します`/etc/sudoers`。詳細については、「Amazon EC2 Systems Manager ユーザーガイド」の「`ssm-user` アカウントの管理アクセス許可を有効または無効にする」を参照してください。

16. クリーンアップするには、[クラスター] ビューでクラスターを選択し、[アクション]、[クラスターの削除] を選択します。

クラスターに接続する

を使用すると AWS ParallelCluster、クラスターヘッドノードに接続して、ジョブの実行、結果の表示、ユーザーの管理、クラスターとジョブのステータスのモニタリングを行うことができます。クラスターヘッドノードインスタンスに接続するには、次のいずれかの方法を使用します。

- キーペア`ssh`で を使用してログインできます。クラスター設定の `HeadNode`/`KeyName` にプライベートキーを指定します。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「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。
- `pcluster ssh` コマンドラインインターフェイス (CLI) コマンドを使用してログインできます。クラスター設定 `HeadNode`/`KeyName` でプライベートキーを指定します。詳細については、「`pcluster ssh`」を参照してください。
- SSM セッションを使用して、クラスターヘッドノードに接続できます。SSM セッションを使用して接続するには、`AmazonSSMManagedInstanceCore` 管理ポリシーをクラスター設定の `HeadNode`/`AdditionalIamPolicies` に追加する必要があります。詳細については、「SSM ユーザーガイド」の「SSM セッションマネージャー」を参照してください。
- Amazon DCV を使用してクラスターヘッドノードに接続できます。詳細については、「Amazon DCV 経由でヘッドノードとログインノードに接続する」を参照してください。
- PCUI を使用する場合、UI が提供する Amazon EC2 Connect コマンドを使用してクラスターヘッドノードに接続することもできます。

クラスターへの複数のユーザーアクセス

1 つのクラスターへの複数のユーザーアクセスを実装して管理する方法について説明します。

このトピックでは、 AWS ParallelCluster ユーザーはコンピューティングインスタンスのシステムユーザーを指します。例として `ec2-user` は Amazon EC2 インスタンス用です。

AWS ParallelCluster マルチユーザーアクセスのサポートは、 AWS リージョン AWS ParallelCluster が現在利用可能なすべての で利用できます。Amazon FSx for Lustre や Amazon Amazon Elastic File Systemなど AWS のサービス、他の と連携します。

AWS Directory Service for Microsoft Active Directory または Simple AD を使用してクラスターアクセスを管理できます。これらのサービスが利用できるAWS リージョンを確認してください。クラスターを設定するには、AWS ParallelCluster DirectoryService 設定を指定します。 AWS Directory Service ディレクトリは複数のクラスターに接続できます。これにより、複数の環境にわたる ID を一元管理し、統一されたログイン操作が可能になります。

を AWS ParallelCluster 複数のユーザーアクセス AWS Directory Service に使用すると、 ディレクトリで定義されているユーザー認証情報を使用してクラスターにログインできます。これらの認証情報は、ユーザー名とパスワードで構成されます。クラスターに初めてログインすると、ユーザー SSH キーが自動的に生成されます。これを使用して、パスワードなしでログインできます。

ディレクトリサービスのデプロイ後に、クラスターのユーザーまたはグループを作成、削除、変更できます。では AWS Directory Service、 で、 AWS Management Console または Active Directory ユーザーとコンピュータツールを使用してこれを行うことができます。このツールには、Active Directory に参加しているすべての Amazon EC2 インスタンスからアクセスできます。詳細については、「Active Directory 管理ツールのインストール」を参照してください。

インターネットアクセスのない単一のサブネット AWS ParallelCluster で を使用する予定がある場合は、「」を参照して追加の要件AWS ParallelCluster インターネットアクセスのない単一のサブネット内を確認してください。

トピック

- Active Directory を作成する
- AD ドメインを含むクラスターを作成します。
- AD ドメインと統合されたクラスターにログインします。
- MPI ジョブの実行
- LDAP(S) クラスター設定 AWS Managed Microsoft AD の例

Active Directory を作成する

クラスターを作成する前に、Active Directory (AD) を作成します。クラスターの Active Directory のタイプを選択する方法については、「AWS Directory Service 管理ガイド」の「オプションの選択」を参照してください。

ディレクトリが空の場合は、ユーザー名とパスワードを使用してユーザーを追加します。詳細については、「AWS Directory Service for Microsoft Active Directory」または「Simple AD」に固有のドキュメントを参照してください。

> **Note**
>
> AWS ParallelCluster では、すべての Active Directory ユーザーディレクトリが `/home/$user` ディレクトリに存在する必要があります。

AD ドメインを含むクラスターを作成します。

> **Warning**
>
> この入門セクションでは、Lightweight Directory Access Protocol (LDAP) 経由で Managed Active Directory (AD) サーバー AWS ParallelCluster で を設定する方法について説明します。LDAP は安全でないプロトコルです。実稼働システムでは、以下の LDAP(S) クラスター設定 AWS Managed Microsoft AD の例 セクションで説明するように TLS 証明書 (LDAPS) の使用を強く推奨します。

クラスター設定ファイルの `DirectoryService` セクションに関連情報を指定して、クラスターをディレクトリと統合するように設定します。詳細については、`DirectoryService` 設定セクションを参照してください。

次の例を使用して、Lightweight Directory Access Protocol (LDAP) AWS Managed Microsoft AD 経由でクラスターを と統合できます。

LDAP AWS Managed Microsoft AD 経由の設定に必要な特定の定義：

- `DirectoryService`/`AdditionalSssdConfigs` で `ldap_auth_disable_tls_never_use_in_production` パラメータを `True` に設定する必要があります。

- DirectoryService/DomainAddr にはコントローラーのホスト名または IP アドレスのいずれかを指定できます。
- DirectoryService/DomainReadOnlyUser 構文は次のようである必要があります。

```
cn=ReadOnly,ou=Users,ou=CORP,dc=corp,dc=example,dc=com
```

AWS Managed Microsoft AD 設定データを取得します。

```
$ aws ds describe-directories --directory-id "d-abcdef01234567890"
```

```
{
    "DirectoryDescriptions": [
        {
            "DirectoryId": "d-abcdef01234567890",
            "Name": "corp.example.com",
            "DnsIpAddrs": [
                "203.0.113.225",
                "192.0.2.254"
            ],
            "VpcSettings": {
                "VpcId": "vpc-021345abcdef6789",
                "SubnetIds": [
                    "subnet-1234567890abcdef0",
                    "subnet-abcdef01234567890"
                ],
                "AvailabilityZones": [
                    "region-idb",
                    "region-idd"
                ]
            }
        }
    ]
}
```

AWS Managed Microsoft ADのクラスター設定。

```
Region: region-id
Image:
  Os: alinux2
HeadNode:
```

```
  InstanceType: t2.micro
  Networking:
    SubnetId: subnet-1234567890abcdef0
  Ssh:
    KeyName: pcluster
Scheduling:
  Scheduler: slurm
  SlurmQueues:
    - Name: queue1
      ComputeResources:
        - Name: t2micro
          InstanceType: t2.micro
          MinCount: 1
          MaxCount: 10
      Networking:
        SubnetIds:
          - subnet-abcdef01234567890
DirectoryService:
  DomainName: dc=corp,dc=example,dc=com
  DomainAddr: ldap://203.0.113.225,ldap://192.0.2.254
  PasswordSecretArn: arn:aws:secretsmanager:region-
id:123456789012:secret:MicrosoftAD.Admin.Password-1234
  DomainReadOnlyUser: cn=ReadOnly,ou=Users,ou=CORP,dc=corp,dc=example,dc=com
  AdditionalSssdConfigs:
    ldap_auth_disable_tls_never_use_in_production: True
```

Simple AD にこの設定を使用するには、**DirectoryService** セクションの **DomainReadOnlyUser** プロパティ値を変更します。

```
DirectoryService:
  DomainName: dc=corp,dc=example,dc=com
  DomainAddr: ldap://203.0.113.225,ldap://192.0.2.254
  PasswordSecretArn: arn:aws:secretsmanager:region-
id:123456789012:secret:SimpleAD.Admin.Password-1234
  DomainReadOnlyUser: cn=ReadOnlyUser,cn=Users,dc=corp,dc=example,dc=com
  AdditionalSssdConfigs:
    ldap_auth_disable_tls_never_use_in_production: True
```

考慮事項:

- LDAP だけを使用するのではなく、TLS/SSL (または LDAPS) を介した LDAP を使用することをお勧めします。TLS/SSL は接続を確実に暗号化します。

- DirectoryService/DomainAddr プロパティの値は、describe-directories 出力の DnsIpAddrs リスト内のエントリと一致します。
- クラスターには、DirectoryService/DomainAddr が指しているのと同じアベイラビリティーゾーンにあるサブネットを使用することをお勧めします。ディレクトリ VPC に推奨されるカスタム Dynamic Host Configuration Protocol (DHCP) 設定を使用していて、サブネットが DirectoryService/DomainAddr アベイラビリティーゾーンにない場合、アベイラビリティーゾーン間のクロストラフィックが発生する可能性があります。マルチユーザー AD 統合機能を使用するために、カスタム DHCP 設定を使用する必要はありません。
- DirectoryService/DomainReadOnlyUser プロパティ値は、ディレクトリに作成する必要があるユーザーを指定します。このユーザーはデフォルトでは作成されません。このユーザーにはディレクトリデータを変更するアクセス許可を与えないことをお勧めします。
- DirectoryService / PasswordSecretArnプロパティ値は、/ DirectoryService DomainReadOnlyUserプロパティに指定したユーザーのパスワードを含む AWS Secrets Manager シークレットを指します。このユーザーのパスワードが変更された場合は、シークレット値を更新してクラスターを更新してください。新しいシークレット値に合わせてクラスターを更新するには、pcluster update-compute-fleet コマンドを使用してコンピューティングフリートを停止する必要があります。LoginNodes を使用するようにクラスターを設定した場合は、LoginNodes/Pools を停止し、LoginNodes/Pools/Count を 0 に設定した後でクラスターを更新します。クラスターヘッドノード内から、次のコマンドを実行します。

```
 sudo /opt/parallelcluster/scripts/directory_service/
update_directory_service_password.sh
```

別の例については、「Active Directory の統合」を参照してください。

AD ドメインと統合されたクラスターにログインします。

Active Delivery (AD) ドメイン統合機能を有効にすると、パスワードによる認証がクラスターヘッドノードで有効になります。AD ユーザーのホームディレクトリは、ユーザーがヘッドノードに初めてログインしたとき、または sudo-user がヘッドノードで AD ユーザーに初めて切り替わったときに作成されます。

クラスターコンピューティングノードではパスワード認証は有効になっていません。AD ユーザーは SSH キーを使用してコンピューティングノードにログインする必要があります。

デフォルトでは、SSH キーはヘッドノードへの最初の SSH ログイン時に AD ユーザー /${HOME}/.ssh ディレクトリに設定されます。この動作は、クラスター設定で

`DirectoryService`/`GenerateSshKeysForUsers` ブール値プロパティを `false` に設定することで無効にできます。デフォルトで、`DirectoryService`/`GenerateSshKeysForUsers` は `true` に設定されています。

AWS ParallelCluster アプリケーションがクラスターノード間でパスワードレス SSH を必要とする場合は、SSH キーがユーザーのホームディレクトリに正しく設定されていることを確認してください。

AWS Managed Microsoft AD パスワードは 42 日後に期限切れになります。詳細については、「AWS Directory Service 管理ガイド」の「AWS Managed Microsoft ADのパスワードポリシーを管理する」を参照してください。パスワードの有効期限が切れた場合、クラスターアクセスを回復するにはパスワードをリセットする必要があります。詳細については、「ユーザーパスワードと失効しているパスワードをリセットする方法」を参照してください。

> **Note**
>
> AD 統合機能が期待どおりに機能しない場合、SSSD ログから問題のトラブルシューティングに役立つ診断情報が得られます。これらのログはクラスターノード上の `/var/log/sssd` ディレクトリにあります。デフォルトでは、これらはクラスターの Amazon CloudWatch ロンググループにも保存されます。
> 詳細については、「Active Directory を使用したマルチユーザー統合のトラブルシューティング」を参照してください。

MPI ジョブの実行

SchedMD で提案されているように、MPI ブートストラップ方法としてSlurm を使用して MPI ジョブをブートストラップします。詳細については、公式の Slurm ドキュメントまたは MPI ライブラリの公式ドキュメントを参照してください。

例えば、「IntelMPI 公式ドキュメント」では、StarCCM ジョブを実行する場合、環境変数 `I_MPI_HYDRA_BOOTSTRAP=slurm` をエクスポートしてプロセスオーケストレーターとして Slurm を設定する必要があることを説明しています。

> **Note**
>
> [既知の問題]

MPI アプリケーションが MPI ジョブを生成するメカニズムとして SSH に依存している場合、Slurm の既知のバグが原因で、ディレクトリのユーザー名が「nobody」に誤って解決される可能性があります。
MPI ブートストラップ方法として Slurm を使用するようにアプリケーションを設定するか、トラブルシューティングセクションの「ユーザー名解決に関する既知の問題」を参照して、詳細と考えられる回避策を確認してください。

LDAP(S) クラスター設定 AWS Managed Microsoft AD の例

AWS ParallelCluster は、Lightweight Directory Access Protocol (LDAP) AWS Directory Service 経由の または TLS/SSL (LDAPS) 経由の LDAP と統合することで、複数のユーザーアクセスをサポートします。

以下の例は、LDAP を介して AWS Managed Microsoft AD と統合するクラスター設定を作成する方法を示しています。

AWS Managed Microsoft AD 証明書検証による LDAPS 経由

この例を使用して、証明書の検証を使用して、クラスターを LDAPS AWS Managed Microsoft AD 経由の と統合できます。

証明書設定を使用した LDAPS AWS Managed Microsoft AD 経由の の特定の定義：

- 証明書検証付きの LDAPS では、`DirectoryService`/`LdapTlsReqCert` を `hard` (デフォルト) に設定する必要があります。
- `DirectoryService`/`LdapTlsCaCert` には認証局 (CA) 証明書のパスを指定する必要があります。

  CA 証明書は、AD ドメインコントローラーの証明書を発行した CA チェーン全体の証明書を含む証明書バンドルです。

  CA 証明書と証明書はクラスターノードにインストールする必要があります。
- コントローラーのホスト名は IP アドレスではなく `DirectoryService`/`DomainAddr` に指定する必要があります。
- `DirectoryService`/`DomainReadOnlyUser` 構文は次のようである必要があります。

```
cn=ReadOnly,ou=Users,ou=CORP,dc=corp,dc=example,dc=com
```

LDAPS を介した AD を使用する場合のクラスター設定ファイルの例。

```
Region: region-id
Image:
  Os: alinux2
HeadNode:
  InstanceType: t2.micro
  Networking:
    SubnetId: subnet-1234567890abcdef0
  Ssh:
    KeyName: pcluster
  Iam:
    AdditionalIamPolicies:
      - Policy: arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess
  CustomActions:
    OnNodeConfigured:
      Script: s3://&example-s3-bucket;/scripts/pcluster-dub-msad-ldaps.post.sh
Scheduling:
  Scheduler: slurm
  SlurmQueues:
    - Name: queue1
      ComputeResources:
        - Name: t2micro
          InstanceType: t2.micro
          MinCount: 1
          MaxCount: 10
      Networking:
        SubnetIds:
          - subnet-abcdef01234567890
      Iam:
        AdditionalIamPolicies:
          - Policy: arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess
      CustomActions:
        OnNodeConfigured:
          Script: s3://&example-s3-bucket;/scripts/pcluster-dub-msad-ldaps.post.sh
DirectoryService:
  DomainName: dc=corp,dc=example,dc=com
  DomainAddr: ldaps://win-abcdef01234567890.corp.example.com,ldaps://win-
abcdef01234567890.corp.example.com
  PasswordSecretArn: arn:aws:secretsmanager:region-
id:123456789012:secret:MicrosoftAD.Admin.Password-1234
  DomainReadOnlyUser: cn=ReadOnly,ou=Users,ou=CORP,dc=corp,dc=example,dc=com
  LdapTlsCaCert: /etc/openldap/cacerts/corp.example.com.bundleca.cer
```

```
  LdapTlsReqCert: hard
```

インストール後のスクリプトで証明書を追加し、ドメインコントローラーを設定します。

```
*#!/bin/bash*
set -e

AD_CERTIFICATE_S3_URI="s3://amzn-s3-demo-bucket/bundle/corp.example.com.bundleca.cer"
AD_CERTIFICATE_LOCAL="/etc/openldap/cacerts/corp.example.com.bundleca.cer"

AD_HOSTNAME_1="win-abcdef01234567890.corp.example.com"
AD_IP_1="192.0.2.254"

AD_HOSTNAME_2="win-abcdef01234567890.corp.example.com"
AD_IP_2="203.0.113.225"

# Download CA certificate
mkdir -p $(dirname "${AD_CERTIFICATE_LOCAL}")
aws s3 cp "${AD_CERTIFICATE_S3_URI}" "${AD_CERTIFICATE_LOCAL}"
chmod 644 "${AD_CERTIFICATE_LOCAL}"

# Configure domain controllers reachability
echo "${AD_IP_1} ${AD_HOSTNAME_1}" >> /etc/hosts
echo "${AD_IP_2} ${AD_HOSTNAME_2}" >> /etc/hosts
```

以下の例のように、ドメインに参加しているインスタンスからドメインコントローラーのホスト名を取得できます。

Windows インスタンスから

```
$ nslookup 192.0.2.254
```

```
Server:  corp.example.com
Address:  192.0.2.254

Name:    win-abcdef01234567890.corp.example.com
Address:  192.0.2.254
```

Linux インスタンスから

```
$ nslookup 192.0.2.254
```

```
192.0.2.254.in-addr.arpa   name = corp.example.com
192.0.2.254.in-addr.arpa   name = win-abcdef01234567890.corp.example.com
```

AWS Managed Microsoft AD 証明書の検証なしの LDAPS 経由

この例を使用して、証明書の検証なしで、クラスターを LDAPS AWS Managed Microsoft AD 経由でと統合できます。

証明書検証設定のない LDAPS AWS Managed Microsoft AD 経由の の特定の定義：

- DirectoryService/LdapTlsReqCert を never に設定する必要があります。
- DirectoryService/DomainAddr にはコントローラーのホスト名または IP アドレスのいずれかを指定できます。
- DirectoryService/DomainReadOnlyUser 構文は次のようである必要があります。

```
cn=ReadOnly,ou=Users,ou=CORP,dc=corp,dc=example,dc=com
```

証明書の検証なしで LDAPS AWS Managed Microsoft AD 経由で を使用するためのクラスター設定ファイルの例：

```
Region: region-id
Image:
  Os: alinux2
HeadNode:
  InstanceType: t2.micro
  Networking:
    SubnetId: subnet-1234567890abcdef0
  Ssh:
    KeyName: pcluster
Scheduling:
  Scheduler: slurm
  SlurmQueues:
    - Name: queue1
      ComputeResources:
        - Name: t2micro
          InstanceType: t2.micro
          MinCount: 1
          MaxCount: 10
      Networking:
        SubnetIds:
```

```
          - subnet-abcdef01234567890
DirectoryService:
  DomainName: dc=corp,dc=example,dc=com
  DomainAddr: ldaps://203.0.113.225,ldaps://192.0.2.254
  PasswordSecretArn: arn:aws:secretsmanager:region-
id:123456789012:secret:MicrosoftAD.Admin.Password-1234
  DomainReadOnlyUser: cn=ReadOnly,ou=Users,ou=CORP,dc=corp,dc=example,dc=com
  LdapTlsReqCert: never
```

ベストプラクティス

以下のセクションでは、 を使用するためのベストプラクティスについて説明します。これには AWS ParallelCluster、ネットワークパフォーマンスと予算アラートが含まれます。これらのベストプラクティスに従っていても問題が発生した場合は、考えられる解決策AWS ParallelCluster トラブルシューティングについて「」を参照してください。

ベストプラクティスヘッドノードインスタンスタイプの選択

ヘッドノードはジョブを実行しませんが、その機能とサイジングは、クラスターの全体的なパフォーマンスにとって重要です。ヘッドノードに使用するインスタンスタイプを選択するときは、次の特性を考慮してください。

クラスターサイズ: ヘッドノードは、クラスターのスケーリングロジックをオーケストレーションし、新しいノードをスケジューラに追加する責任を負います。多数のノードを持つクラスターをスケールアップまたはスケールダウンするには、ヘッドノードの処理能力を向上させることを検討してください。

共有ファイルシステム: 共有ファイルシステムを使用する場合は、ワークフローを処理するのに十分なネットワーク帯域幅と Amazon EBS の帯域幅を持つインスタンスタイプを選択してください。ヘッドノードがクラスターに十分な NFS サーバーディレクトリを公開できることと、コンピューティングノードとヘッドノード間で共有する必要のあるアーティファクトを処理できることをそれぞれ確認してください。

ベストプラクティス: ネットワークパフォーマンス

ハイパフォーマンスコンピューティング (HPC) アプリケーションには、ネットワークのパフォーマンスが重要です。信頼できるネットワークパフォーマンスがなければ、これらのアプリケーションは期待どおりに動作しません。ネットワークのパフォーマンスを最適化するには、次のベストプラクティスを検討してください。

- プレイスメントグループ: Slurm を使用している場合には、クラスタープレイスメントグループを使用するように各 Slurm キューを設定することを検討してください。クラスタープレイスメントグループは、単一のアベイラビリティーゾーン内のインスタンスを論理的にグループ化したものです。詳細については、「Amazon EC2 ユーザーガイド」の「プレイスメントグループ」を参照してください。キューの `Networking` セクションで `PlacementGroup` を指定できます。各コンピューティングリソースはキューのプレイスメントグループに割り当てられます。コンピュートリソースの `Networking` セクションで `PlacementGroup` を指定すると、その特定のコンピューティングリソースがそのプレイスメントグループに割り当てられます。コンピューティングリソースのプレイスメントグループの仕様は、コンピューティングリソースのキュー仕様よりも優先されます。詳細については、「`SlurmQueues`/`Networking`/`PlacementGroup`」および「`SlurmQueues`/`ComputeResources`/`Networking`/`PlacementGroup`」を参照してください。

```
Networking:
  PlacementGroup:
    Enabled: true
    Id: your-placement-group-name
```

  または、 でプレイスメントグループ AWS ParallelCluster を作成することもできます。

```
Networking:
  PlacementGroup:
    Enabled: true
```

  AWS ParallelCluster バージョン 3.3.0 以降、プレイスメントグループの作成と管理が変更されました。キューに name または `Id` を付けずに有効にするプレイスメントグループを指定すると、キュー全体に 1 つの管理グループを割り当てるのではなく、各コンピューティングリソースに独自のマネージドプレイスメントグループが割り当てられます。これにより、容量不足によるエラーを減らすことができます。キュー全体に 1 つのプレイスメントグループが必要な場合は、名前付きのプレイスメントグループを使用できます。

  `SlurmQueues`/`Networking`/`PlacementGroup`/`Name` が優先的な代替として `SlurmQueues`/`Networking`/`PlacementGroup`/`Id` に追加されました。

  詳細については、「`Networking`」を参照してください。

- 拡張ネットワーク: 拡張ネットワークをサポートするインスタンスタイプの選択を検討してください。この推奨事項は、すべての現行世代のインスタンスに適用されます。詳細については、「Amazon EC2 ユーザーガイド」の「Linux での拡張ネットワーキング」を参照してください。

- Elastic Fabric Adapter: 高レベルのスケーラブルなインスタンス間通信をサポートするには、ネットワークに EFA ネットワークインターフェイスを選択することを検討してください。EFA のカスタムビルドのオペレーティングシステム (OS) バイパスハードウェアは、 AWS クラウドのオンデマンドの伸縮性と柔軟性により、インスタンス間の通信を強化します。`Efa` を使用するための Slurm キュー `ComputeResource` はそれぞれ設定できます。での EFA の使用の詳細については AWS ParallelCluster、「」を参照してくださいElastic Fabric Adapter。

```
ComputeResources:
  - Name: your-compute-resource-name
    Efa:
      Enabled: true
```

  EFA 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「Elastic Fabric Adapter」を参照してください。

- インスタンス帯域幅: 帯域幅はインスタンスのサイズに合わせて調整されます。さまざまなインスタンスタイプについては、「Amazon EC2 ユーザーガイド」の「Amazon EBS 最適化インスタンス」と「Amazon EBS ボリュームタイプ」を参照してください。

ベストプラクティス: 予算アラート

でリソースコストを管理するには AWS ParallelCluster、 AWS Budgets アクションを使用して予算を作成することをお勧めします。選択した AWS リソースに対して定義された予算しきい値アラートを作成することもできます。詳細については、「AWS Budgets ユーザーガイド」の「予算アクションの設定」を参照してください。同様に、Amazon CloudWatch を使用して請求アラームを作成することもできます。詳細については、「AWS 予想料金をモニタリングする請求アラームの作成」を参照してください。

ベストプラクティス: AWS ParallelCluster クラスターを新しいマイナーバージョンまたはパッチバージョンに移動する

現在、各 AWS ParallelCluster マイナーバージョンは CLI `pcluster` とともに自己完結型です。クラスタを新しいマイナーバージョンまたはパッチバージョンに移行するには、新しいバージョンの CLI を使用してクラスターを再作成する必要があります。

クラスターを新しいマイナーバージョンまたはパッチバージョンに移行するプロセスを最適化するには、次のことをお勧めします。

- Amazon EFS や FSx for Lustre など、クラスターの外部で作成された外部ボリュームに個人データを保存します。これにより、将来、あるクラスターから別のクラスターにデータを簡単に移動できます。
- 以下のタイプを使用して共有ストレージシステムを作成します。これらのシステムは、 AWS CLI または を使用して作成できます AWS Management Console。
  - `SharedStorage` / `EbsSettings` / `VolumeId`
  - `SharedStorage` / `EfsSettings` / `FileSystemId`
  - `SharedStorage` / `FsxLustreSettings` / `FileSystemId`

  クラスター設定内のファイルシステムまたはボリュームを既存のファイルシステムまたはボリュームとして定義します。これにより、クラスターを削除しても保持され、新しいクラスターにアタッチできます。

  Amazon EFS または FSx for Lustre をファイルシステムに使用することをお勧めします。これらのシステムは両方とも、同時に複数のクラスターに接続できます。さらに、既存のクラスターを削除する前に、これらのシステムのいずれかを新しいクラスターに接続できます。
- カスタム AMI を使用するのではなく、カスタムブートストラップアクションを使用してインスタンスをカスタマイズします。代わりにカスタム AMI を使用する場合は、新しいバージョンがリリースされるたびにその AMI を削除して再作成する必要があります。
- 上記の推奨事項を次の順序で適用することをお勧めします。
  1. 既存のファイルシステム定義を使用するようにクラスター設定を更新します。
  2. `pcluster` バージョンを確認し、必要に応じて更新します。
  3. 新しいクラスターを作成してテストします。新しいクラスターをテストするときは、以下をチェックしてください。
     - データが新しいクラスターで利用可能であることを確認します。
     - アプリケーションが新しいクラスターで機能することを確認します。
  4. 新しいクラスターが完全にテストされて動作し、既存のクラスターが不要になったら、そのクラスターを削除します。

AWS ParallelCluster 2.x から 3.x への移行

以下のセクションでは、あるバージョンから別のバージョンへの変更を含め、 AWS ParallelCluster 2.x から 3.x に移行するとどうなるかについて説明します。

カスタムブートストラップアクション

AWS ParallelCluster 3 では、 および Scheduling/ SlurmQueuesセクションの OnNodeStart (pre_install AWS ParallelCluster バージョン 2) および OnNodeConfigured (post_install AWS ParallelCluster バージョン 2) パラメータを使用して、ヘッドノードHeadNodeとコンピューティングノードに異なるカスタムブートストラップアクションスクリプトを指定できます。詳細については、「カスタムブートストラップアクション」を参照してください。

for AWS ParallelCluster 2 用に開発されたカスタムブートストラップアクションスクリプトは、AWS ParallelCluster 3 で使用するように調整する必要があります。

- ヘッドノードとコンピューティングノードを区別するために /etc/parallelcluster/cfnconfig と cfn_node_type を使用することは推奨されません。代わりに、HeadNode と Scheduling/SlurmQueues の 2 種類のスクリプトを指定することをお勧めします。
- ブートストラップアクションスクリプトで使用するために /etc/parallelcluster/cfnconfig をロードを続けたい場合は、cfn_node_type の値が「MasterServer」から「HeadNode」に変更されていることに注意してください (参照: 包括的言語)。
- On AWS ParallelCluster 2 では、アクションスクリプトをブートストラップする最初の入力引数は、スクリプトへの S3 URL であり、予約されました。 AWS ParallelCluster 3 では、構成で設定された引数のみがスクリプトに渡されます。

> **Warning**
>
> /etc/parallelcluster/cfnconfig ファイルで提供される内部変数の使用は公式にはサポートされていません。このファイルは将来のリリースの一部として削除される可能性があります。

AWS ParallelCluster 2.x と 3.x は異なる設定ファイルの構文を使用します

AWS ParallelCluster 3.x 設定では YAML 構文を使用します。リファレンス全文は、設定ファイル にあります。

AWS ParallelCluster 3.x では、YAML ファイル形式の必須化に加えて、多くの設定セクション、設定、パラメータ値が更新されました。このセクションでは、 AWS ParallelCluster 設定の主な変更点と、 の各バージョンにおけるこれらの違いを示すside-by-sideの例を書き留めます AWS ParallelCluster。

ハイパースレッディングを有効または無効にした場合の複数スケジューラキュー構成例

AWS ParallelCluster 2:

```
[cluster default]
queue_settings = ht-enabled, ht-disabled
...

[queue ht-enabled]
compute_resource_settings = ht-enabled-i1
disable_hyperthreading = false

[queue ht-disabled]
compute_resource_settings = ht-disabled-i1
disable_hyperthreading = true

[compute_resource ht-enabled-i1]
instance_type = c5n.18xlarge
[compute_resource ht-disabled-i1]
instance_type = c5.xlarge
```

AWS ParallelCluster 3:

```
...
Scheduling:
  Scheduler: slurm
  SlurmQueues:
    - Name: ht-enabled
      Networking:
        SubnetIds:
          - compute_subnet_id
      ComputeResources:
        - Name: ht-enabled-i1
          DisableSimultaneousMultithreading: true
          InstanceType: c5n.18xlarge
    - Name: ht-disabled
      Networking:
        SubnetIds:
          - compute_subnet_id
      ComputeResources:
        - Name: ht-disabled-i1
          DisableSimultaneousMultithreading: false
          InstanceType: c5.xlarge
```

新しい FSx for Lustre ファイルシステムの構成例

AWS ParallelCluster 2:

```
[cluster default]
fsx_settings = fsx
...

[fsx fsx]
shared_dir = /shared-fsx
storage_capacity = 1200
imported_file_chunk_size = 1024
import_path = s3://amzn-s3-demo-bucket
export_path = s3://amzn-s3-demo-bucket/export_dir
weekly_maintenance_start_time = 3:02:30
deployment_type = PERSISTENT_1
data_compression_type = LZ4
```

AWS ParallelCluster 3:

```
...
SharedStorage:
  - Name: fsx
    MountDir: /shared-fsx
    StorageType: FsxLustre
    FsxLustreSettings:
      StorageCapacity: 1200
      ImportedFileChunkSize: 1024
      ImportPath: s3://amzn-s3-demo-bucket
      ExportPath: s3://amzn-s3-demo-bucket/export_dir
      WeeklyMaintenanceStartTime: "3:02:30"
      DeploymentType: PERSISTENT_1
      DataCompressionType: LZ4
```

既存の FSx for Lustre ファイルシステムをマウントしたクラスター構成例

AWS ParallelCluster 2:

```
[cluster default]
fsx_settings = fsx
...
```

```
[fsx fsx]
shared_dir = /shared-fsx
fsx_fs_id = fsx_fs_id
```

AWS ParallelCluster 3:

```
...
SharedStorage:
  - Name: fsx
    MountDir: /shared-fsx
    StorageType: FsxLustre
    FsxLustreSettings:
      FileSystemId: fsx_fs_id
```

インテル HPC プラットフォーム仕様ソフトウェアスタックを搭載したクラスターの例

AWS ParallelCluster 2:

```
[cluster default]
enable_intel_hpc_platform = true
...
```

AWS ParallelCluster 3:

```
...
AdditionalPackages:
  IntelSoftware:
    IntelHpcPlatform: true
```

注記:

- Intel HPC プラットフォーム仕様ソフトウェアのインストールには、該当するIntel エンドユーザーライセンス契約の条件が適用されます。

インスタンスプロファイル、インスタンスロール、インスタンスの追加ポリシー、クラスターに関連する Lambda 関数のロールなどのカスタム IAM 構成の例

AWS ParallelCluster 2:

```
[cluster default]
```

```
additional_iam_policies = arn:aws:iam::aws:policy/
AmazonS3ReadOnlyAccess,arn:aws:iam::aws:policy/AmazonDynamoDBReadOnlyAccess
ec2_iam_role = ec2_iam_role
iam_lambda_role = lambda_iam_role
...
```

AWS ParallelCluster 3:

```
...
Iam:
  Roles:
    CustomLambdaResources: lambda_iam_role
HeadNode:
  ...
  Iam:
    InstanceRole: ec2_iam_role
Scheduling:
  Scheduler: slurm
  SlurmQueues:
    - Name: queue1
      ...
      Iam:
        InstanceProfile: iam_instance_profile
    - Name: queue2
      ...
      Iam:
        AdditionalIamPolicies:
          - Policy: arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess
          - Policy: arn:aws:iam::aws:policy/AmazonDynamoDBReadOnlyAccess
```

注記:

- AWS ParallelCluster 2 `additional_iam_policies`の場合、IAM 設定はクラスターのすべてのインスタンスに適用され、 と組み合わせて使用することはできません`ec2_iam_role`。
- For AWS ParallelCluster 3 では、ヘッドノードとコンピューティングノードに異なる IAM 設定を指定し、コンピューティングキューごとに異なる IAM 設定を指定することもできます。
- For AWS ParallelCluster 3 では、IAM ロールの代わりに IAM インスタンスプロファイルを使用できます。 `InstanceProfile``InstanceRole`または を一緒に設定`AdditionalIamPolicies`することはできません。

カスタムブートストラップアクションの例

AWS ParallelCluster 2:

```
[cluster default]
s3_read_resource = arn:aws:s3:::amzn-s3-demo-bucket/*
pre_install = s3://amzn-s3-demo-bucket/scripts/pre_install.sh
pre_install_args = 'R curl wget'
post_install = s3://amzn-s3-demo-bucket/scripts/post_install.sh
post_install_args = "R curl wget"
...
```

AWS ParallelCluster 3:

```
...
HeadNode:
  ...
  CustomActions:
    OnNodeStart:
      Script: s3://amzn-s3-demo-bucket/scripts/pre_install.sh
      Args:
        - R
        - curl
        - wget
    OnNodeConfigured:
      Script: s3://amzn-s3-demo-bucket/scripts/post_install.sh
      Args: ['R', 'curl', 'wget']
  Iam:
    S3Access:
      - BucketName: amzn-s3-demo-bucket
Scheduling:
  Scheduler: slurm
  SlurmQueues:
    - Name: queue1
      ...
      CustomActions:
        OnNodeStart:
          Script: s3://amzn-s3-demo-bucket/scripts/pre_install.sh
          Args: ['R', 'curl', 'wget']
        OnNodeConfigured:
          Script: s3://amzn-s3-demo-bucket/scripts/post_install.sh
          Args: ['R', 'curl', 'wget']
      Iam:
        S3Access:
          - BucketName: amzn-s3-demo-bucket
```

S3 バケットのリソースに読み取りと書き込みのアクセスがあるクラスターの例

AWS ParallelCluster 2:

```
[cluster default]
s3_read_resource = arn:aws:s3:::amzn-s3-demo-bucket/read_only/*
s3_read_write_resource = arn:aws:s3:::amzn-s3-demo-bucket/read_and_write/*
...
```

AWS ParallelCluster 3:

```
...
HeadNode:
  ...
  Iam:
    S3Access:
      - BucketName: amzn-s3-demo-bucket
        KeyName: read_only/
        EnableWriteAccess: False
      - BucketName: amzn-s3-demo-bucket
        KeyName: read_and_write/
        EnableWriteAccess: True
Scheduling:
  Scheduler: slurm
  SlurmQueues:
    - Name: queue1
      ...
      Iam:
        S3Access:
          - BucketName: amzn-s3-demo-bucket
            KeyName: read_only/
            EnableWriteAccess: False
          - BucketName: amzn-s3-demo-bucket
            KeyName: read_and_write/
            EnableWriteAccess: True
```

包括的言語

AWS ParallelCluster 3 は、「マスター」が AWS ParallelCluster 2 で使用された場所で「ヘッドノード」という単語を使用します。これには以下が含まれます。

- AWS Batch ジョブ環境でエクスポートされた変数が から `MASTER_IP`に変更されましたPCLUSTER_HEAD_NODE_IP。
- すべての AWS CloudFormation 出力が から `Master*`に変更されましたHeadNode*。
- すべての NodeType とタグが `Master` から `HeadNode` に変更されました。

スケジューラのサポート

AWS ParallelCluster 3.x は、Son of Grid Engine (SGE) および Torque スケジューラをサポートしていません。

AWS Batch コマンド `awsbhosts`、`awsbkill`、`awsbout`、`awsbstat`、、 `awsbsub`は、個別の `aws-parallelcluster-awsbatch-cli` PyPI `awsbqueues`パッケージとして配布されます。このパッケージは、ヘッドノードの AWS ParallelCluster によってインストールされます。クラスターのヘッドノードからこれらの AWS Batch コマンドを引き続き使用できます。ただし、ヘッドノード以外の場所から AWS Batch コマンドを使用したい場合は、まず `aws-parallelcluster-awsbatch-cli` PyPI パッケージをインストールする必要があります。

AWS ParallelCluster CLI

AWS ParallelCluster コマンドラインインターフェイス (CLI) が変更されました。新しい構文については、AWS ParallelCluster CLI コマンド で説明しています。CLI の出力形式は JSON 文字列です。

新しいクラスターを構成する

`pcluster configure` コマンドには、 AWS ParallelCluster 2 とは異なるパラメータが in AWS ParallelCluster 3 に含まれています。詳細については、「`pcluster configure`」を参照してください。

また、設定ファイルの構文が AWS ParallelCluster 2 から変更されていることに注意してください。クラスター構成設定の完全なリファレンスについては、「クラスター設定ファイル」を参照してください。

新しいクラスターを作成する

AWS ParallelCluster 2 の `pcluster create` コマンドは `pcluster create-cluster` コマンドに置き換えられました。

AWS ParallelCluster 2.x のデフォルトの動作では、 `-nw`オプションを指定せずにクラスター作成イベントを待機し、 AWS ParallelCluster 3.x コマンドはすぐに を返すことに注意してください。クラスター作成の進捗状況は、`pcluster describe-cluster` でモニタリングできます

An AWS ParallelCluster 3 設定ファイルには 1 つのクラスター定義が含まれているため、 `-t`パラメータは必要ありません。

設定ファイルの例を以下に示します。

```
# AWS ParallelCluster v2
$ pcluster create \
    -r REGION \
    -c V2_CONFIG_FILE \
    -nw \
    -t CLUSTER_TEMPLATE \
    CLUSTER_NAME

# AWS ParallelCluster v3
$ pcluster create-cluster \
    --region REGION \
    --cluster-configuration V3_CONFIG_FILE \
    --cluster-name CLUSTER_NAME
```

クラスターの一覧表示

`pcluster list` AWS ParallelCluster 2.x コマンドは `pcluster list-clusters` コマンドに置き換える必要があります。

注: 2 AWS ParallelCluster .x バージョンの で作成されたクラスターを一覧表示するには、v2 CLI が必要です AWS ParallelCluster。仮想環境を利用して複数のバージョンの AWS ParallelCluster をインストールする方法については、「仮想環境に AWS ParallelCluster をインストールする (推奨)」を参照してください。

```
# AWS ParallelCluster v2
$ pcluster list -r REGION

# AWS ParallelCluster v3
$ pcluster list-clusters --region REGION
```

クラスターの起動と停止

`pcluster start` および `pcluster stop` AWS ParallelCluster 2.x コマンドは `pcluster update-compute-fleet` コマンドに置き換える必要があります。

コンピューティングフリートの開始:

```
# AWS ParallelCluster v2
$ pcluster start \
    -r REGION \
    CLUSTER_NAME

# AWS ParallelCluster v3 - Slurm fleets
$ pcluster update-compute-fleet \
    --region REGION \
    --cluster-name CLUSTER_NAME \
    --status START_REQUESTED

# AWS ParallelCluster v3 - AWS Batch fleets
$ pcluster update-compute-fleet \
    --region REGION \
    --cluster-name CLUSTER_NAME \
    --status ENABLED
```

コンピューティングフリートの停止:

```
# AWS ParallelCluster v2
$ pcluster stop \
    -r REGION \
    CLUSTER_NAME

# AWS ParallelCluster v3 - Slurm fleets
$ pcluster update-compute-fleet \
    --region REGION \
    --cluster-name CLUSTER_NAME \
    --status STOP_REQUESTED

# AWS ParallelCluster v3 - AWS Batch fleets
$ pcluster update-compute-fleet \
    --region REGION \
    --cluster-name CLUSTER_NAME \
    --status DISABLED
```

クラスターへの接続

`pcluster ssh` AWS ParallelCluster 2.x コマンドには、 AWS ParallelCluster 3.x で異なるパラメータ名があります。「pcluster ssh」を参照してください。

クラスターへの接続:

```
# AWS ParallelCluster v2
$ pcluster ssh \
    -r REGION \
    CLUSTER_NAME \
    -i ~/.ssh/id_rsa

# AWS ParallelCluster v3
$ pcluster ssh \
    --region REGION \
    --cluster-name CLUSTER_NAME \
    -i ~/.ssh/id_rsa
```

IMDS 設定の更新

バージョン 3.0.0 以降、デフォルトでヘッドノードの IMDS (およびインスタンスプロファイル認証情報) へのアクセスをスーパーユーザーのサブセットに制限するサポート AWS ParallelCluster が導入されました。詳細については、「Imds のプロパティ」を参照してください。

の使用 AWS ParallelCluster

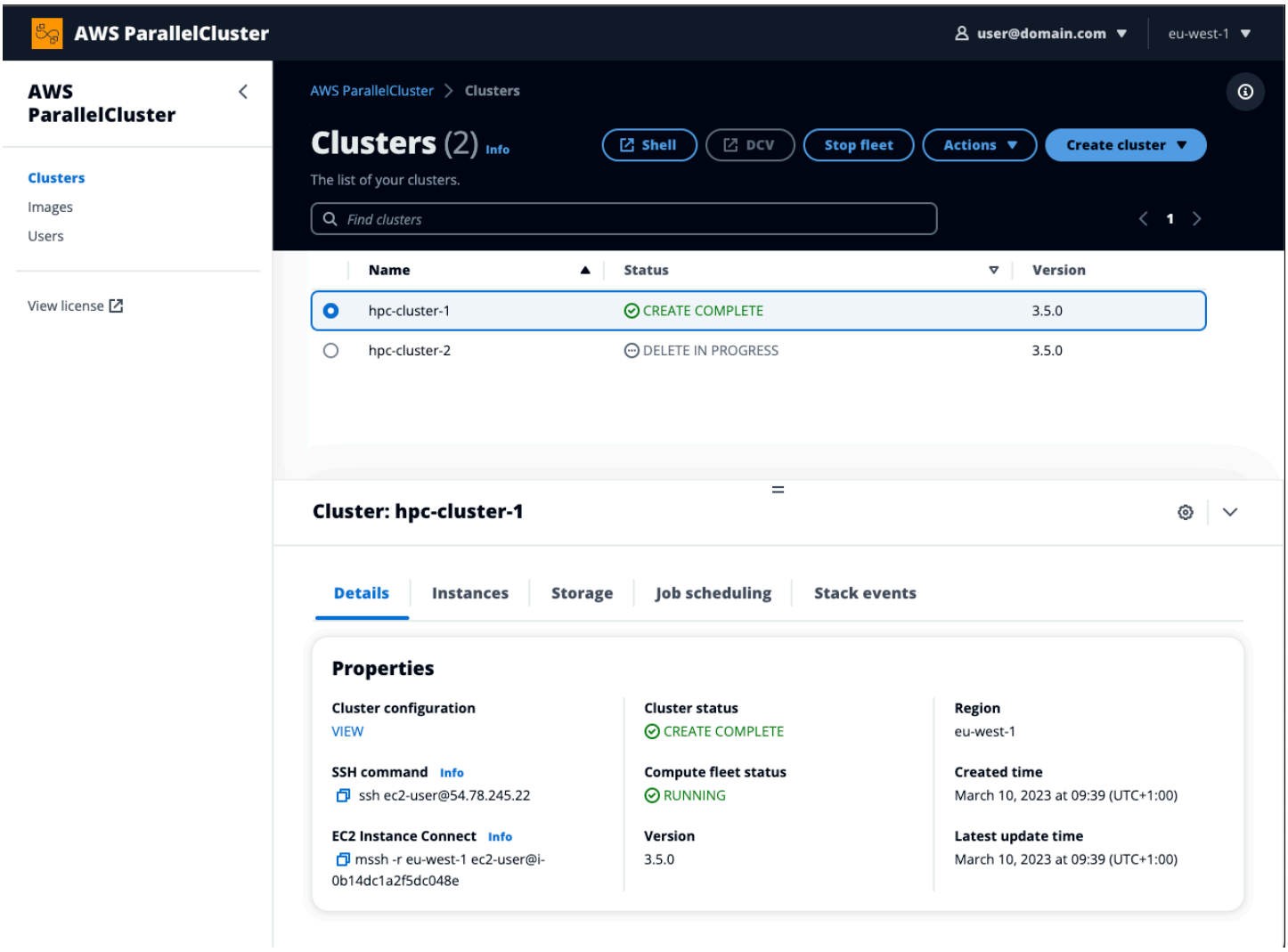
トピック

- AWS ParallelCluster UI
- AWS Lambda での VPC 設定 AWS ParallelCluster
- AWS Identity and Access Management での アクセス許可 AWS ParallelCluster
- ネットワークの設定
- によってプロビジョニングされたログインノード AWS ParallelCluster
- カスタムブートストラップアクション
- Amazon S3 での使用
- スポットインスタンスの操作
- でサポートされているスケジューラ AWS ParallelCluster
- 共有ストレージ
- AWS ParallelCluster リソースとタグ付け
- AWS ParallelCluster と ログのモニタリング
- AWS CloudFormation カスタムリソース
- Elastic Fabric Adapter
- Intel MPI を有効にする
- AWS ParallelCluster API
- AWS ParallelCluster Terraform の
- Amazon DCV 経由でヘッドノードとログインノードに接続する
- pcluster update-cluster を使用する
- AWS ParallelCluster AMI のカスタマイズ
- オンデマンドキャパシティ予約 (ODCR) を使用してインスタンスを起動する
- キャパシティブロック (CB) を使用してインスタンスを起動する
- AMI のパッチ適用と Amazon EC2 インスタンスの置換
- オペレーティングシステム

AWS ParallelCluster UI

AWS ParallelCluster UI (PCUI) は、クラスターを作成、モニタリング、管理するためのダッシュボードとして機能するウェブベースのユーザーインターフェイスです。に PCUI をインストールしてアクセスします AWS アカウント。PCUI は AWS ParallelCluster バージョン 3.5.0 で追加されました。

PCUI をインストールして開始するには、「PCUI のインストール」と「PCUI によるクラスターの設定と作成」を参照してください。



PCUI は以下の機能をサポートしています。

- 以下を表示します。
  - AWS アカウント で作成したクラスターのリスト AWS ParallelCluster。
  - リストされたクラスターの使用可能なステータスと詳細。
  - モニタリングに使用できる CloudFormation スタックイベントと AWS ParallelCluster ログ。

  - クラスターで実行中のジョブのステータス。
  - クラスターをビルドする際に使用できる、カスタムイメージのリスト。
  - UI がクラスターの作成に使用する公式イメージのリスト。
  - PCUI にアクセスできるユーザーのリスト。ユーザーの追加や削除ができます。
- クラスターを作成、編集 (更新) し、サポートされているクラスター機能を選択して追加、編集、削除する方法を段階的に説明します。編集中のクラスター構成では、アクセスできない入力フィールドは変更できません。クラスターをデプロイする前に、クラスター構成のドライラン検証を実行することもできます。
- [クラスター] ビューのヘッドノードにアクセスするための直接シェルリンクを備えています。ステップバイステップのガイダンスで [SSM セッションの追加] を選択し、ヘッドノードにダイレクトシェルアクセスと SSM マネージドインスタンスコアポリシーを追加します。

PCUI を使用してクラスターを作成および管理する際は、以下の点を考慮します。

- PCUI の作成に使用されたのと同じ AWS ParallelCluster バージョンでのみ、クラスターを作成および編集したり、イメージを構築したりできます。それ以前のバージョンのクラスターまたはイメージでは表示のみ可能です。複数のバージョンのクラスターやイメージを管理する場合は、各バージョンをサポートする PCUI インスタンスを作成することをお勧めします。
- PCUI は `pcluster` CLI 機能を反映するように設計されています。ただし、相違点がいくつかあります。ステップバイステップのガイダンスに従えば、サポートされているすべての機能を使用していることになります。デプロイする前に、クラスターまたはイメージの設定を手動で編集するオプションがあります。その場合は、[ドライラン] を選択して設定を検証し、編集が完全にサポートされていることを確認することをお勧めします。

> **Note**
>
> PCUI は AWS Batchをサポートしていません。

AWS Lambda での VPC 設定 AWS ParallelCluster

AWS ParallelCluster は AWS Lambda を使用して、クラスターのライフサイクル中にオペレーションを実行します。Lambda サービスが所有する VPC 内でAWS Lambda は常に実行されます。この Lambda 関数を 仮想プライベートクラウド (VPC) のプライベートサブネットに接続して、プライベートリソースにアクセスすることもできます。

> **Note**
>
> Lambda 関数は、ハードウェア専有インスタンスのテナンシーを使用して VPC に直接接続することはできません。専有 VPC のリソースに接続するには、専有 VPC に接続できるデフォルトのテナンシーを使用し、専有 VPC を 2 番目の VPC にピア接続します。
> 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「ハードウェア専有インスタンス」、および「AWS ナレッジセンター」の「「Lambda 関数を専有 VPC に接続するにはどうしたらよいですか?」を参照してください。

によって作成された Lambda 関数は、プライベート VPC に接続 AWS ParallelCluster できます。これらの Lambda 関数は AWS のサービスにアクセスする必要があります。次の方法を使用して、インターネットまたは VPC エンドポイント経由でアクセスを提供できます。

- インターネットアクセス

  インターネットと にアクセスするには AWS のサービス、Lambda 関数にネットワークアドレス変換 (NAT) が必要です。プライベートサブネットからのアウトバウンドトラフィックをパブリックサブネットの NAT ゲートウェイにルーティングします。

- VPC エンドポイント

  いくつかの AWS サービスは VPC エンドポイントを提供します。VPC エンドポイントを使用して、インターネットアクセスのない VPC AWS のサービス から に接続できます。 AWS ParallelCluster VPC エンドポイントのリストを表示するには、「ネットワーク」を参照してください。

> **Note**
>
> サブネットとセキュリティグループのすべての組み合わせは、これらの方法のいずれか AWS のサービス を使用して へのアクセスを提供する必要があります。すべてのサブネットとセキュリティグループは、同じ VPC 内にある必要があります。

詳細については、「Amazon Virtual Private Cloud ユーザーガイド」の「VPC エンドポイント」、および「AWS Lambda 開発者ガイド」の「VPC に接続した関数のインターネットアクセスとサービスアクセス」を参照してください。

Lambda 関数と VPC の使用を設定するには、クラスターの場合は `DeploymentSettings`/`LambdaFunctionsVpcConfig` を、イメージの場合は `DeploymentSettings`/`LambdaFunctionsVpcConfig` を参照してください。

AWS Identity and Access Management での アクセス許可 AWS ParallelCluster

AWS ParallelCluster は IAM アクセス許可を使用して、クラスターの作成および管理時にリソースへのアクセスを制御します。

AWS アカウントでクラスターを作成および管理するには、次の 2 つのレベルでアクセス許可 AWS ParallelCluster が必要です。

- クラスターを作成および管理するための `pcluster` CLI コマンドを呼び出すために `pcluster` ユーザーが必要とするアクセス許可。
- クラスターリソースがクラスターアクションを実行するために必要なアクセス許可。

AWS ParallelCluster は Amazon EC2 インスタンスプロファイルとロールを使用して、クラスターリソースのアクセス許可を提供します。クラスターリソースのアクセス許可を管理するには、AWS ParallelCluster にも IAM リソースへのアクセス許可が必要です。詳細については、「AWS ParallelCluster IAM リソースを管理するためのユーザーポリシーの例」を参照してください。

`pcluster` ユーザーが `pcluster` CLI を使用してクラスターとそのリソースを作成および管理するには、IAM アクセス許可が必要です。これらのアクセス許可は IAM ポリシーに含まれており、ユーザーまたはロールに追加できます。IAM ロールの詳細については、「AWS Identity and Access Management ユーザーガイド」の「ユーザーロールの作成」を参照してください。

AWS ParallelCluster IAM アクセス許可を管理するための設定パラメータ を使用することもできます。

以下のセクションでは、必要なアクセス許可について、例を挙げて説明します。

ポリシーの例を使用するには、*`<REGION>`*、*`<AWS ACCOUNT ID>`* などの文字列を適切な値に置き換えます。

次のポリシーの例では、リソースの Amazon リソースネーム (ARN) が含まれています。 AWS GovCloud (US) または AWS 中国パーティションで作業している場合は、ARNsを変更する必要があります。具体的には、 AWS GovCloud (US) パーティションの場合は「arn:aws」から「arn:aws-us-gov」、 AWS 中国パーティションの場合は「arn:aws-cn」に変更する必要があります。詳細

については、「 ユーザーガイド」の「 リージョンの AWS GovCloud (US) Amazon リソースネーム (ARNs)」および「中国での AWS サービスの開始方法」のARNs」を参照してください。 AWS GovCloud (US)　AWS

ポリシー例への変更は、AWS ParallelCluster documentation on GitHub で追跡できます。

トピック

- AWS ParallelCluster Amazon EC2 インスタンスロール
- AWS ParallelCluster pcluster ユーザーポリシーの例
- AWS ParallelCluster IAM リソースを管理するためのユーザーポリシーの例
- AWS ParallelCluster IAM アクセス許可を管理するための設定パラメータ

AWS ParallelCluster Amazon EC2 インスタンスロール

デフォルト設定でクラスターを作成すると、 は Amazon EC2 インスタンスプロファイル AWS ParallelCluster を使用して、クラスターとそのリソースの作成と管理に必要なアクセス許可を提供するデフォルトのクラスター Amazon EC2 インスタンスロールを自動的に作成します。

デフォルトの AWS ParallelCluster インスタンスロールを使用する代替方法

デフォルトの AWS ParallelCluster インスタンスロールの代わりに、`InstanceRole`クラスター設定を使用して EC2 の既存の IAM ロールを指定できます。詳細については、「AWS ParallelCluster IAM アクセス許可を管理するための設定パラメータ」を参照してください。通常、EC2 に付与されるアクセス許可を完全に制御するために、既存の IAM ロールを指定します。

デフォルトのインスタンスロールにポリシーを追加する場合は、`InstanceProfile` または `InstanceRole` 設定の代わりに `AdditionalIamPolicies` 構成設定を使用して追加の IAM ポリシーを渡すことをお勧めします。クラスターを更新するときに `AdditionalIamPolicies` を更新することはできますが、`InstanceRole` を更新することはできません。

AWS ParallelCluster **`pcluster`** ユーザーポリシーの例

次の例は、 `pcluster` CLI を使用して AWS ParallelCluster とそのリソースを作成および管理するために必要なユーザーポリシーを示しています。ポリシーはユーザーまたはロールにアタッチできます。

トピック

- AWS ParallelCluster pcluster 基本ユーザーポリシー

- AWS Batch スケジューラを使用する際の AWS ParallelCluster pcluster ユーザーポリシーの追加
- Amazon FSx for Lustre を使用する場合の追加の AWS ParallelCluster pcluster ユーザーポリシー
- AWS ParallelCluster イメージビルドpclusterユーザーポリシー

AWS ParallelCluster **pcluster** 基本ユーザーポリシー

次のポリシーは、コマンドの実行 AWS ParallelCluster `pcluster`に必要なアクセス許可を示しています。

ポリシーにリストされている最後のアクションは、クラスターの設定で指定されているシークレットを検証するためのものです。たとえば、 AWS Secrets Manager シークレットを使用して`DirectoryService`統合を設定します。この場合、クラスターは `PasswordSecretArn` に有効なシークレットが存在する場合にのみ作成されます。このアクションを省略すると、シークレットの検証はスキップされます。セキュリティ体制を強化するために、クラスターの設定で指定されているシークレットのみを追加して、このポリシーステートメントの範囲を絞り込むことをお勧めします。

> **Note**
>
> 既存の Amazon EFS ファイルシステムだけがクラスターで使用されるファイルシステムの場合、Amazon EFS ポリシーステートメントの例をクラスター設定ファイルの `SharedStorage` セクション で参照されている特定のファイルシステムに絞り込むことができます。

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Action": [
              "ec2:Describe*"
          ],
          "Resource": "*",
          "Effect": "Allow",
          "Sid": "EC2Read"
      },
      {
          "Action": [
              "ec2:AllocateAddress",
              "ec2:AssociateAddress",
```

```
                "ec2:AttachNetworkInterface",
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:CreateFleet",
                "ec2:CreateLaunchTemplate",
                "ec2:CreateLaunchTemplateVersion",
                "ec2:CreateNetworkInterface",
                "ec2:CreatePlacementGroup",
                "ec2:CreateSecurityGroup",
                "ec2:CreateSnapshot",
                "ec2:CreateTags",
                "ec2:DeleteTags",
                "ec2:CreateVolume",
                "ec2:DeleteLaunchTemplate",
                "ec2:DeleteNetworkInterface",
                "ec2:DeletePlacementGroup",
                "ec2:DeleteSecurityGroup",
                "ec2:DeleteVolume",
                "ec2:DisassociateAddress",
                "ec2:ModifyLaunchTemplate",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:ModifyVolume",
                "ec2:ModifyVolumeAttribute",
                "ec2:ReleaseAddress",
                "ec2:RevokeSecurityGroupEgress",
                "ec2:RevokeSecurityGroupIngress",
                "ec2:RunInstances",
                "ec2:TerminateInstances"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "EC2Write"
        },
        {
            "Action": [
                "dynamodb:DescribeTable",
                "dynamodb:ListTagsOfResource",
                "dynamodb:CreateTable",
                "dynamodb:DeleteTable",
                "dynamodb:GetItem",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem",
                "dynamodb:Query",
                "dynamodb:TagResource"
```

```
            ],
            "Resource": "arn:aws:dynamodb:*:<AWS ACCOUNT ID>:table/parallelcluster-*",
            "Effect": "Allow",
            "Sid": "DynamoDB"
        },
        {
            "Action": [
                "route53:ChangeResourceRecordSets",
                "route53:ChangeTagsForResource",
                "route53:CreateHostedZone",
                "route53:DeleteHostedZone",
                "route53:GetChange",
                "route53:GetHostedZone",
                "route53:ListResourceRecordSets",
                "route53:ListQueryLoggingConfigs"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "Route53HostedZones"
        },
        {
            "Action": [
                "cloudformation:*"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "CloudFormation"
        },
        {
            "Action": [
                "cloudwatch:PutDashboard",
                "cloudwatch:ListDashboards",
                "cloudwatch:DeleteDashboards",
                "cloudwatch:GetDashboard",
                "cloudwatch:PutMetricAlarm",
                "cloudwatch:DeleteAlarms",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:PutCompositeAlarm"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "CloudWatch"
        },
        {
```

```
            "Action": [
                "iam:GetRole",
                "iam:GetRolePolicy",
                "iam:GetPolicy",
                "iam:SimulatePrincipalPolicy",
                "iam:GetInstanceProfile"
            ],
            "Resource": [
                "arn:aws:iam::<AWS ACCOUNT ID>:role/*",
                "arn:aws:iam::<AWS ACCOUNT ID>:policy/*",
                "arn:aws:iam::aws:policy/*",
                "arn:aws:iam::<AWS ACCOUNT ID>:instance-profile/*"
            ],
            "Effect": "Allow",
            "Sid": "IamRead"
        },
        {
            "Action": [
                "iam:CreateInstanceProfile",
                "iam:DeleteInstanceProfile",
                "iam:AddRoleToInstanceProfile",
                "iam:RemoveRoleFromInstanceProfile"
            ],
            "Resource": [
                "arn:aws:iam::<AWS ACCOUNT ID>:instance-profile/parallelcluster/*"
            ],
            "Effect": "Allow",
            "Sid": "IamInstanceProfile"
        },
        {
            "Condition": {
                "StringEqualsIfExists": {
                    "iam:PassedToService": [
                        "lambda.amazonaws.com",
                        "ec2.amazonaws.com",
                        "spotfleet.amazonaws.com"
                    ]
                }
            },
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::<AWS ACCOUNT ID>:role/parallelcluster/*"
```

```
            ],
            "Effect": "Allow",
            "Sid": "IamPassRole"
        },
        {
            "Action": [
                "lambda:CreateFunction",
                "lambda:DeleteFunction",
                "lambda:GetFunctionConfiguration",
                "lambda:GetFunction",
                "lambda:InvokeFunction",
                "lambda:AddPermission",
                "lambda:RemovePermission",
                "lambda:UpdateFunctionConfiguration",
                "lambda:TagResource",
                "lambda:ListTags",
                "lambda:UntagResource"
            ],
            "Resource": [
                "arn:aws:lambda:*:<AWS ACCOUNT ID>:function:parallelcluster-*",
                "arn:aws:lambda:*:<AWS ACCOUNT ID>:function:pcluster-*"
            ],
            "Effect": "Allow",
            "Sid": "Lambda"
        },
        {
            "Action": [
                "s3:*"
            ],
            "Resource": [
                "arn:aws:s3:::parallelcluster-*",
                "arn:aws:s3:::aws-parallelcluster-*"
            ],
            "Effect": "Allow",
            "Sid": "S3ResourcesBucket"
        },
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": "arn:aws:s3:::*-aws-parallelcluster*",
            "Effect": "Allow",
            "Sid": "S3ParallelClusterReadOnly"
```

```
        },
        {
            "Action": [
                "elasticfilesystem:*"
            ],
            "Resource": [
                "arn:aws:elasticfilesystem:*:<AWS ACCOUNT ID>:*"
            ],
            "Effect": "Allow",
            "Sid": "EFS"
        },
        {
            "Action": [
                "logs:DeleteLogGroup",
                "logs:PutRetentionPolicy",
                "logs:DescribeLogGroups",
                "logs:CreateLogGroup",
                "logs:TagResource",
                "logs:UntagResource",
                "logs:FilterLogEvents",
                "logs:GetLogEvents",
                "logs:CreateExportTask",
                "logs:DescribeLogStreams",
                "logs:DescribeExportTasks",
                "logs:DescribeMetricFilters",
                "logs:PutMetricFilter",
                "logs:DeleteMetricFilter"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "CloudWatchLogs"
        },
        {
            "Action": [
                "resource-groups:ListGroupResources"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "ResourceGroupRead"
        },
        {
            "Sid": "AllowDescribingFileCache",
            "Effect": "Allow",
            "Action": [
```

```
                "fsx:DescribeFileCaches"
            ],
            "Resource": "*"
        },
        {
            "Action": "secretsmanager:DescribeSecret",
            "Resource": "arn:aws:secretsmanager:<REGION>:<AWS ACCOUNT ID>:secret:<SECRET
 NAME>",
            "Effect": "Allow"
        }
    ]
}
```

AWS Batch スケジューラを使用する際の AWS ParallelCluster **pcluster** ユーザーポリシーの追加

ス AWS Batch ケジューラを使用してクラスターを作成および管理する必要がある場合は、次の追加ポリシーが必要です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Condition": {
                "StringEqualsIfExists": {
                    "iam:PassedToService": [
                        "ecs-tasks.amazonaws.com",
                        "batch.amazonaws.com",
                        "codebuild.amazonaws.com"
                    ]
                }
            },
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::<AWS ACCOUNT ID>:role/parallelcluster/*"
            ],
            "Effect": "Allow",
            "Sid": "IamPassRole"
        },
        {
            "Condition": {
```

```
                "StringEquals": {
                    "iam:AWSServiceName": [
                        "batch.amazonaws.com"
                    ]
                }
            },
            "Action": [
                "iam:CreateServiceLinkedRole",
                "iam:DeleteServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::<AWS ACCOUNT ID>:role/aws-service-role/
batch.amazonaws.com/*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "codebuild:*"
            ],
            "Resource": "arn:aws:codebuild:*:<AWS ACCOUNT ID>:project/pcluster-*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "ecr:*"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "ECR"
        },
        {
            "Action": [
                "batch:*"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "Batch"
        },
        {
            "Action": [
                "events:*"
            ],
            "Resource": "*",
```

```
            "Effect": "Allow",
            "Sid": "AmazonCloudWatchEvents"
        },
        {
            "Action": [
                "ecs:DescribeContainerInstances",
                "ecs:ListContainerInstances"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "ECS"
        }
    ]
}
```

Amazon FSx for Lustre を使用する場合の追加の AWS ParallelCluster `pcluster` ユーザーポリシー

Amazon FSx for Lustre を使用してクラスターを作成および管理する必要がある場合は、以下の追加ポリシーが必要です。

> **Note**
>
> 既存の Amazon FSx ファイルシステムだけがクラスターで使用されるファイルシステムの場合、Amazon FSx ポリシーステートメントの例をクラスター設定ファイルの `SharedStorage` セクション で参照されている特定のファイルシステムに絞り込むことができます。

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Condition": {
              "StringEquals": {
                  "iam:AWSServiceName": [
                      "fsx.amazonaws.com",
                      "s3.data-source.lustre.fsx.amazonaws.com"
                  ]
              }
          },
```

```
            "Action": [
                "iam:CreateServiceLinkedRole",
                "iam:DeleteServiceLinkedRole"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "fsx:*"
            ],
            "Resource": [
                "arn:aws:fsx:*:<AWS ACCOUNT ID>:*"
            ],
            "Effect": "Allow",
            "Sid": "FSx"
        },
        {
            "Action": [
                "iam:CreateServiceLinkedRole",
                "iam:AttachRolePolicy",
                "iam:PutRolePolicy"
            ],
            "Resource": "arn:aws:iam::<AWS ACCOUNT ID>:role/aws-service-role/s3.data-
source.lustre.fsx.amazonaws.com/*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:Get*",
                "s3:List*",
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::<S3 NAME>",
            "Effect": "Allow"
        }
    ]
}
```

AWS ParallelCluster イメージビルド**pcluster**ユーザーポリシー

でカスタム Amazon EC2 イメージを作成するユーザーには、次のアクセス許可のセット AWS ParallelCluster が必要です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeImages",
                "ec2:DescribeInstanceTypeOfferings",
                "ec2:DescribeInstanceTypes",
                "ec2:DeregisterImage",
                "ec2:DeleteSnapshot"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "EC2"
        },
        {
            "Action": [
                "iam:CreateInstanceProfile",
                "iam:AddRoleToInstanceProfile",
                "iam:GetRole",
                "iam:GetRolePolicy",
                "iam:GetInstanceProfile",
                "iam:RemoveRoleFromInstanceProfile"
            ],
            "Resource": [
                "arn:aws:iam::<AWS ACCOUNT ID>:instance-profile/parallelcluster/*",
                "arn:aws:iam::<AWS ACCOUNT ID>:instance-profile/ParallelClusterImage*",
                "arn:aws:iam::<AWS ACCOUNT ID>:role/parallelcluster/*"
            ],
            "Effect": "Allow",
            "Sid": "IAM"
        },
        {
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "lambda.amazonaws.com",
                        "ec2.amazonaws.com"
                    ]
                }
            },
            "Action": [
```

```
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::<AWS ACCOUNT ID>:instance-profile/parallelcluster/*",
                "arn:aws:iam::<AWS ACCOUNT ID>:role/parallelcluster/*"
            ],
            "Effect": "Allow",
            "Sid": "IAMPassRole"
        },
        {
            "Action": [
                "logs:GetLogEvents",
                "logs:CreateLogGroup",
                "logs:TagResource",
                "logs:UntagResource",
                "logs:DeleteLogGroup"
            ],
            "Resource": [
                "arn:aws:logs:*:<AWS ACCOUNT ID>:log-group:/aws/imagebuilder/
ParallelClusterImage-*",
                "arn:aws:logs:*:<AWS ACCOUNT ID>:log-group:/aws/lambda/
ParallelClusterImage-*"
            ],
            "Effect": "Allow",
            "Sid": "CloudWatch"
        },
        {
            "Action": [
                "cloudformation:DescribeStacks",
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack"
            ],
            "Resource": [
                "arn:aws:cloudformation:*:<AWS ACCOUNT ID>:stack/*"
            ],
            "Effect": "Allow",
            "Sid": "CloudFormation"
        },
        {
            "Action": [
                "lambda:CreateFunction",
                "lambda:GetFunction",
                "lambda:AddPermission",
                "lambda:RemovePermission",
```

```
                "lambda:DeleteFunction",
                "lambda:TagResource",
                "lambda:ListTags",
                "lambda:UntagResource"
            ],
            "Resource": [
                "arn:aws:lambda:*:<AWS ACCOUNT ID>:function:ParallelClusterImage-*"
            ],
            "Effect": "Allow",
            "Sid": "Lambda"
        },
        {
            "Action": [
                "imagebuilder:Get*"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "ImageBuilderGet"
        },
        {
            "Action": [
                "imagebuilder:CreateImage",
                "imagebuilder:TagResource",
                "imagebuilder:CreateImageRecipe",
                "imagebuilder:CreateComponent",
                "imagebuilder:CreateDistributionConfiguration",
                "imagebuilder:CreateInfrastructureConfiguration",
                "imagebuilder:DeleteImage",
                "imagebuilder:DeleteComponent",
                "imagebuilder:DeleteImageRecipe",
                "imagebuilder:DeleteInfrastructureConfiguration",
                "imagebuilder:DeleteDistributionConfiguration"
            ],
            "Resource": [
                "arn:aws:imagebuilder:*:<AWS ACCOUNT ID>:image/parallelclusterimage-*",
                "arn:aws:imagebuilder:*:<AWS ACCOUNT ID>:image-recipe/
parallelclusterimage-*",
                "arn:aws:imagebuilder:*:<AWS ACCOUNT ID>:component/
parallelclusterimage-*",
                "arn:aws:imagebuilder:*:<AWS ACCOUNT ID>:distribution-configuration/
parallelclusterimage-*",
                "arn:aws:imagebuilder:*:<AWS ACCOUNT ID>:infrastructure-configuration/
parallelclusterimage-*"
            ],
```

```
            "Effect": "Allow",
            "Sid": "ImageBuilder"
        },
        {
            "Action": [
                "s3:CreateBucket",
                "s3:ListBucket",
                "s3:ListBucketVersions"
            ],
            "Resource": [
                "arn:aws:s3:::parallelcluster-*"
            ],
            "Effect": "Allow",
            "Sid": "S3Bucket"
        },
        {
            "Action": [
                "sns:GetTopicAttributes",
                "sns:TagResource",
                "sns:CreateTopic",
                "sns:Subscribe",
                "sns:Publish",
                "SNS:DeleteTopic",
                "SNS:Unsubscribe"
            ],
            "Resource": [
                "arn:aws:sns:*:<AWS ACCOUNT ID>:ParallelClusterImage-*"
            ],
            "Effect": "Allow",
            "Sid": "SNS"
        },
        {
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:DeleteObject",
                "s3:DeleteObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::parallelcluster-*/*"
            ],
            "Effect": "Allow",
            "Sid": "S3Objects"
```

```
        },
        {
            "Action": "iam:CreateServiceLinkedRole",
            "Effect": "Allow",
            "Resource": "arn:aws:iam::*:role/aws-service-role/
imagebuilder.amazonaws.com/AWSServiceRoleForImageBuilder",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "imagebuilder.amazonaws.com"
                }
            }
        }
    ]
}
```

AWS ParallelCluster IAM リソースを管理するためのユーザーポリシーの例

AWS ParallelCluster を使用してクラスターまたはカスタム AMIs を作成する場合、コンポーネントに必要な一連のアクセス許可を付与するアクセス許可を含む IAM ポリシーを提供する必要があります AWS ParallelCluster 。これらの IAM リソースは、 によって自動的に作成することも、クラスター AWS ParallelCluster またはカスタムイメージの作成時に入力として提供することもできます。

次のモードを使用して、 設定で追加の IAM ポリシーを使用して、IAM リソースにアクセスするために必要なアクセス許可を AWS ParallelCluster ユーザーに付与できます。

トピック

- 特権 IAM アクセスモード
- 制限付き IAM アクセスモード
- PermissionsBoundary モード

特権 IAM アクセスモード

このモードでは、 は必要なすべての IAM リソース AWS ParallelCluster を自動的に作成します。これらの IAM ポリシーは、クラスターリソースへのアクセスのみを可能にするように範囲が制限されています。

特権 IAM アクセスモードを有効にするには、ユーザーロールに以下のポリシーを追加します。

> **Note**
>
> / HeadNode Iam/ AdditionalPoliciesまたは Scheduling / SlurmQueues / Iam AdditionalPoliciesパラメータを設定する場合は、次のポリシーに示すように、追加のポリシーごとにロールポリシーをアタッチおよびデタッチするアクセス許可を AWS ParallelCluster ユーザーに付与する必要があります。ロールポリシーをアタッチおよびデタッチするための条件に、追加のポリシー ARN を追加します。

> **Warning**
>
> このモードでは、ユーザーは で IAM 管理者権限を持つことができます。 AWS アカウント

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
           "Action": [
               "iam:CreateServiceLinkedRole",
               "iam:DeleteRole",
               "iam:TagRole"
           ],
           "Resource": [
               "arn:aws:iam::<AWS ACCOUNT ID>:role/parallelcluster/*"
           ],
           "Effect": "Allow",
           "Sid": "IamRole"
       },
       {
           "Action": [
               "iam:CreateRole"
           ],
           "Resource": [
               "arn:aws:iam::<AWS ACCOUNT ID>:role/parallelcluster/*"
           ],
           "Effect": "Allow",
           "Sid": "IamCreateRole"
       },
       {
           "Action": [
```

```
                "iam:PutRolePolicy",
                "iam:DeleteRolePolicy"
            ],
            "Resource": "arn:aws:iam::<AWS ACCOUNT ID>:role/parallelcluster/*",
            "Effect": "Allow",
            "Sid": "IamInlinePolicy"
        },
        {
            "Condition": {
                "ArnLike": {
                    "iam:PolicyARN": [
                        "arn:aws:iam::<AWS ACCOUNT ID>:policy/parallelcluster*",
                        "arn:aws:iam::<AWS ACCOUNT ID>:policy/parallelcluster/*",
                        "arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy",
                        "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
                        "arn:aws:iam::aws:policy/AWSBatchFullAccess",
                        "arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess",
                        "arn:aws:iam::aws:policy/service-role/AWSBatchServiceRole",
                        "arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role",
                        "arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy",
                        "arn:aws:iam::aws:policy/service-role/
AmazonEC2SpotFleetTaggingRole",
                        "arn:aws:iam::aws:policy/EC2InstanceProfileForImageBuilder",
                        "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole"
                    ]
                }
            },
            "Action": [
                "iam:AttachRolePolicy",
                "iam:DetachRolePolicy"
            ],
            "Resource": "arn:aws:iam::<AWS ACCOUNT ID>:role/parallelcluster/*",
            "Effect": "Allow",
            "Sid": "IamPolicy"
        }
    ]
}
```

制限付き IAM アクセスモード

ユーザーに追加の IAM ポリシーが付与されていない場合、クラスターやカスタムイメージの構築に必要な IAM ロールは、管理者が手動で作成し、クラスター構成の一部として渡す必要があります。

クラスターを作成する際には、以下のパラメータが必要です。

- Iam / Roles / LambdaFunctionsRole
- HeadNode / Iam / InstanceRole | InstanceProfile
- Scheduling / SlurmQueues / Iam / InstanceRole | InstanceProfile

カスタムイメージを構築する際には、以下のパラメータが必要です。

- Build / Iam / InstanceRole | InstanceProfile
- Build / Iam / CleanupLambdaRole

上記のパラメータの一部として渡された IAM ロールは、`/parallelcluster/` パスのプレフィックスに作成する必要があります。これができない場合は、ユーザーポリシーを更新して、特定のカスタムロールに `iam:PassRole` アクセス許可を付与する必要があります (以下の例を参照)。

```
{
   "Condition": {
       "StringEqualsIfExists": {
           "iam:PassedToService": [
               "ecs-tasks.amazonaws.com",
               "lambda.amazonaws.com",
               "ec2.amazonaws.com",
               "spotfleet.amazonaws.com",
               "batch.amazonaws.com",
               "codebuild.amazonaws.com"
           ]
       }
   },
   "Action": [
       "iam:PassRole"
   ],
   "Resource": [
       <list all custom IAM roles>
   ],
   "Effect": "Allow",
```

```
    "Sid": "IamPassRole"
}
```

> **Warning**
>
> 現在、すべての IAM ロールを AWS Batch クラスター設定で渡すことができるわけではないため、このモードではクラスターの管理が許可されません。

PermissionsBoundary モード

このモードは AWS ParallelCluster 、設定された IAM アクセス許可の境界にバインドされた IAM ロールの作成を委任します。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境界」を参照してください。

以下のポリシーを ユーザーロールに追加する必要があります。

ポリシーでは、*<permissions-boundary-arn>* を許可の境界として強制する IAM ポリシーの ARN に置き換えます。

> **Warning**
>
> HeadNode/Iam/AdditionalPolicies または Scheduling/SlurmQueues/Iam/AdditionalPolicies パラメータを設定する場合、以下のポリシーに示すように、追加ポリシーごとにロールポリシーをアタッチおよびデタッチする権限をユーザーに付与する必要があります。ロールポリシーをアタッチおよびデタッチするための条件に、追加のポリシー ARN を追加します。

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
           "Action": [
               "iam:CreateServiceLinkedRole",
               "iam:DeleteRole",
               "iam:TagRole"
           ],
           "Resource": [
               "arn:aws:iam::<AWS ACCOUNT ID>:role/parallelcluster/*"
```

```
            ],
            "Effect": "Allow",
            "Sid": "IamRole"
        },
        {
            "Condition": {
                "StringEquals": {
                    "iam:PermissionsBoundary": [
                        <permissions-boundary-arn>
                    ]
                }
            },
            "Action": [
                "iam:CreateRole"
            ],
            "Resource": [
                "arn:aws:iam::<AWS ACCOUNT ID>:role/parallelcluster/*"
            ],
            "Effect": "Allow",
            "Sid": "IamCreateRole"
        },
        {
            "Condition": {
                "StringEquals": {
                    "iam:PermissionsBoundary": [
                        <permissions-boundary-arn>
                    ]
                }
            },
            "Action": [
                "iam:PutRolePolicy",
                "iam:DeleteRolePolicy"
            ],
            "Resource": "arn:aws:iam::<AWS ACCOUNT ID>:role/parallelcluster/*",
            "Effect": "Allow",
            "Sid": "IamInlinePolicy"
        },
        {
            "Condition": {
                "StringEquals": {
                    "iam:PermissionsBoundary": [
                        <permissions-boundary-arn>
                    ]
                },
```

```
                "ArnLike": {
                    "iam:PolicyARN": [
                        "arn:aws:iam::<AWS ACCOUNT ID>:policy/parallelcluster*",
                        "arn:aws:iam::<AWS ACCOUNT ID>:policy/parallelcluster/*",
                        "arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy",
                        "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
                        "arn:aws:iam::aws:policy/AWSBatchFullAccess",
                        "arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess",
                        "arn:aws:iam::aws:policy/service-role/AWSBatchServiceRole",
                        "arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role",
                        "arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy",
                        "arn:aws:iam::aws:policy/service-role/
AmazonEC2SpotFleetTaggingRole",
                        "arn:aws:iam::aws:policy/EC2InstanceProfileForImageBuilder",
                        "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole"
                    ]
                }
            },
            "Action": [
                "iam:AttachRolePolicy",
                "iam:DetachRolePolicy"
            ],
            "Resource": "arn:aws:iam::<AWS ACCOUNT ID>:role/parallelcluster/*",
            "Effect": "Allow",
            "Sid": "IamPolicy"
        }
    ]
}
```

このモードを有効にすると、クラスターの作成または更新の際には `Iam/PermissionsBoundary` パラメータで、またカスタムイメージの構築の際には`Build/Iam/PermissionBoundary` パラメータで、アクセス許可の境界 ARN を指定する必要があります。

AWS ParallelCluster IAM アクセス許可を管理するための設定パラメータ

AWS ParallelCluster は、クラスターまたはカスタム AMI 作成プロセスで使用される IAM アクセス許可とロールをカスタマイズおよび管理するための一連の設定オプションを公開します。

トピック

- クラスターの設定
- カスタムイメージ構成

クラスターの設定

トピック

- ヘッドノードの IAM ロール
- Amazon S3 アクセス
- 追加の IAM ポリシー:
- AWS Lambda 関数ロール
- コンピューティングノードの IAM ロール
- アクセス許可の境界

ヘッドノードの IAM ロール

`HeadNode` / `Iam` / `InstanceRole` | `InstanceProfile`

このオプションでは、クラスターのヘッドノードに割り当てられているデフォルトの IAM ロールを上書きすることができます。詳細については、「`InstanceProfile` リファレンス」を参照してください。

スケジューラが Slurm の場合、このロールの一部として使用されるポリシーの最小セットを以下に示します。

- `arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy` マネージド IAM ポリシー 詳細については、「Amazon CloudWatch ユーザーガイド」の「CloudWatch エージェントで使用する IAM ロールとユーザーを作成する」を参照してください。
- `arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore` マネージド IAM ポリシー 詳細については、「AWS Systems Manager ユーザーガイド」の「AWS 用の管理ポリシー AWS Systems Manager」を参照してください。
- 追加の IAM ポリシー:

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

```
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::<REGION>-aws-parallelcluster/*",
                "arn:aws:s3:::dcv-license.<REGION>/*",
                "arn:aws:s3:::parallelcluster-*-v1-do-not-delete/*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "dynamodb:GetItem",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem",
                "dynamodb:BatchWriteItem",
                "dynamodb:BatchGetItem"
            ],
            "Resource": "arn:aws:dynamodb:<REGION>:<AWS ACCOUNT ID>:table/
parallelcluster-*",
            "Effect": "Allow"
        },
        {
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/parallelcluster:node-type": "Compute"
                }
            },
            "Action": "ec2:TerminateInstances",
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "ec2:RunInstances",
                "ec2:CreateFleet"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
```

```
                        "ec2.amazonaws.com"
                    ]
                }
            },
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::<AWS ACCOUNT ID>:role/parallelcluster/*",
                "arn:aws:iam::<AWS ACCOUNT ID>:instance-profile/parallelcluster/*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "ec2:DescribeInstances",
                "ec2:DescribeInstanceStatus",
                "ec2:DescribeVolumes",
                "ec2:DescribeInstanceAttribute",
                "ec2:DescribeCapacityReservations"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "ec2:CreateTags",
                "ec2:AttachVolume"
            ],
            "Resource": [
                "arn:aws:ec2:<REGION>:<AWS ACCOUNT ID>:instance/*",
                "arn:aws:ec2:<REGION>:<AWS ACCOUNT ID>:volume/*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "cloudformation:DescribeStacks",
                "cloudformation:DescribeStackResource",
                "cloudformation:SignalResource"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
```

```
        {
            "Action": [
                "route53:ChangeResourceRecordSets"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "arn:aws:secretsmanager:<REGION>:<AWS ACCOUNT
 ID>:secret:<SECRET_ID>",
            "Effect": "Allow"
        }
    ]
}
```

なお、Scheduling/SlurmQueues/Iam/InstanceRole がコンピューティング IAM ロールをオーバーライドするために使用されている場合、上記のヘッドノードポリシーでは、iam:PassRole アクセス許可の Resource セクションにそのようなロールを含める必要があります。

ここでは、スケジューラが AWS Batchである場合に、このロールの一部として使用されるポリシーの最小セットを示します。

- arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy マネージド IAM ポリシー 詳細については、「Amazon CloudWatch ユーザーガイド」の「CloudWatch エージェントで使用する IAM ロールとユーザーを作成する」を参照してください。
- arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore マネージド IAM ポリシー 詳細については、「AWS Systems Manager ユーザーガイド」の「AWS 用の管理ポリシー AWS Systems Manager」を参照してください。
- 追加の IAM ポリシー:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
```

```
                "arn:aws:s3:::parallelcluster-*-v1-do-not-delete/*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": "s3:GetObject",
            "Resource": [
                "arn:aws:s3:::dcv-license.<REGION>/*",
                "arn:aws:s3:::<REGION>-aws-parallelcluster/*"
            ],
            "Effect": "Allow"
        },
        {
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "batch.amazonaws.com"
                    ]
                }
            },
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::<AWS ACCOUNT ID>:role/parallelcluster/*",
                "arn:aws:iam::<AWS ACCOUNT ID>:instance-profile/parallelcluster/*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "batch:DescribeJobQueues",
                "batch:DescribeJobs",
                "batch:ListJobs",
                "batch:DescribeComputeEnvironments"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "batch:SubmitJob",
                "batch:TerminateJob",
                "logs:GetLogEvents",
```

```
                "ecs:ListContainerInstances",
                "ecs:DescribeContainerInstances"
            ],
            "Resource": [
                "arn:aws:logs:<REGION>:<AWS ACCOUNT ID>:log-group:/aws/batch/job:log-
stream:PclusterJobDefinition*",
                "arn:aws:ecs:<REGION>:<AWS ACCOUNT ID>:container-instance/AWSBatch-
PclusterComputeEnviron*",
                "arn:aws:ecs:<REGION>:<AWS ACCOUNT ID>:cluster/AWSBatch-Pcluster*",
                "arn:aws:batch:<REGION>:<AWS ACCOUNT ID>:job-queue/
PclusterJobQueue*",
                "arn:aws:batch:<REGION>:<AWS ACCOUNT ID>:job-definition/
PclusterJobDefinition*:*",
                "arn:aws:batch:<REGION>:<AWS ACCOUNT ID>:job/*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "ec2:DescribeInstances",
                "ec2:DescribeInstanceStatus",
                "ec2:DescribeVolumes",
                "ec2:DescribeInstanceAttribute"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "ec2:CreateTags",
                "ec2:AttachVolume"
            ],
            "Resource": [
                "arn:aws:ec2:<REGION>:<AWS ACCOUNT ID>:instance/*",
                "arn:aws:ec2:<REGION>:<AWS ACCOUNT ID>:volume/*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "cloudformation:DescribeStackResource",
                "cloudformation:DescribeStacks",
                "cloudformation:SignalResource"
            ],
```

```
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "arn:aws:secretsmanager:<REGION>:<AWS ACCOUNT
 ID>:secret:<SECRET_ID>",
            "Effect": "Allow"
        }
    ]
}
```

Amazon S3 アクセス

HeadNode/Iam/S3Access または Scheduling/SlurmQueues/S3Access

これらの設定セクションでは、クラスターのヘッドノードまたはコンピューティングノードに関連する IAM ロールが AWS ParallelClusterによって作成された場合、そのロールに追加の Amazon S3 ポリシーを付与することで、Amazon S3 アクセスをカスタマイズすることができます。詳細については、各設定パラメータのリファレンスドキュメントを参照してください。

このパラメータは、ユーザーが 特権 IAM アクセスモード または PermissionsBoundary モードに設定されている場合にのみ使用できます。

追加の IAM ポリシー:

HeadNode/Iam/AdditionalIamPolicies または SlurmQueues/Iam/AdditionalIamPolicies

このオプションを使用して、クラスターのヘッドノードまたはコンピューティングノードに関連付けられた IAM ロールが によって作成されたときに、追加のマネージド IAM ポリシーをアタッチします AWS ParallelCluster。

> **Warning**
>
> このオプションを使用するには、AWS ParallelCluster ユーザーに、アタッチする必要のある IAM ポリシーの `iam:AttachRolePolicy` および `iam:DetachRolePolicy` の権限が付与されていることを確認してください。

AWS Lambda 関数ロール

Iam / Roles / LambdaFunctionsRole

このオプションは、クラスター作成プロセス中に使用されるすべての AWS Lambda 関数にアタッチされたロールを上書きします。 は、ロールを引き受けることができるプリンシパルとして設定 AWS Lambda する必要があります。

> **Note**
>
> DeploymentSettings/LambdaFunctionsVpcConfig が設定されている場合、LambdaFunctionsRole には VPC 設定を設定するための AWS Lambda ロールのアクセス許可を含める必要があります。

ここでは、このロールの一部として使用されるポリシーの最小セットを示します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "route53:ListResourceRecordSets",
        "route53:ChangeResourceRecordSets"
      ],
      "Resource": "arn:aws:route53:::hostedzone/*",
      "Effect": "Allow"
    },
    {
      "Action": ["logs:CreateLogStream", "logs:PutLogEvents"],
      "Effect": "Allow",
      "Resource": "arn:aws:logs:<REGION>:<AWS ACCOUNT ID>:log-group:/aws/lambda/
pcluster-*"
    },
    {
      "Action": "ec2:DescribeInstances",
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Action": "ec2:TerminateInstances",
      "Condition": {
```

```
        "StringEquals": {
          "ec2:ResourceTag/parallelcluster:node-type": "Compute"
        }
      },
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Action": [
        "s3:DeleteObject",
        "s3:DeleteObjectVersion",
        "s3:ListBucket",
        "s3:ListBucketVersions"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3:::parallelcluster-*-v1-do-not-delete",
        "arn:aws:s3:::parallelcluster-*-v1-do-not-delete/*"
      ]
    }
  ]
}
```

コンピューティングノードの IAM ロール

Scheduling / SlurmQueues / Iam / _InstanceRole | InstanceProfile

このオプションでは、クラスターのコンピューティングノードに割り当てられている IAM ロールを上書きすることができます。詳細については、「InstanceProfile」を参照してください。

ここでは、このロールの一部として使用されるポリシーの最小セットを示します。

- arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy マネージド IAM ポリシー 詳細については、「Amazon CloudWatch ユーザーガイド」の「CloudWatch エージェントで使用する IAM ロールとユーザーを作成する」を参照してください。
- arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore マネージド IAM ポリシー 詳細については、「AWS Systems Manager ユーザーガイド」の「AWS 用の管理ポリシー AWS Systems Manager」を参照してください。
- 追加の IAM ポリシー:

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Action": [
                "dynamodb:Query",
                "dynamodb:UpdateItem",
                "dynamodb:PutItem",
                "dynamodb:GetItem"
            ],
            "Resource": "arn:aws:dynamodb:<REGION>:<AWS ACCOUNT ID>:table/
parallelcluster-*",
            "Effect": "Allow"
        },
        {
            "Action": "s3:GetObject",
            "Resource": [
                "arn:aws:s3:::<REGION>-aws-parallelcluster/*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": "ec2:DescribeInstanceAttribute",
            "Resource": "*",
            "Effect": "Allow"
        },

        {
            "Action": "cloudformation:DescribeStackResource",
            "Resource":
                [ "arn:aws:cloudformation:<REGION>:<AWS ACCOUNT ID>:stack/*/*" ],
            "Effect" "Allow"
        }
    ]
}
```

アクセス許可の境界

Iam / PermissionsBoundary

このパラメータ AWS ParallelCluster は、クラスターデプロイの一部として作成されたすべての IAM ロールPermissionsBoundaryに、指定された IAM ポリシーを として強制的にアタッチします。

この設定を定義する際にユーザーが必要とするポリシーのリストについては、PermissionsBoundary モード を参照してください。

カスタムイメージ構成

トピック

- EC2 Image Builder のインスタンスロール
- AWS Lambda クリーンアップロール
- 追加の IAM ポリシー
- アクセス許可の境界

EC2 Image Builder のインスタンスロール

`Build` / `Iam` / `InstanceRole` | `InstanceProfile`

このオプションでは、EC2 Image Builder で起動した Amazon EC2 インスタンスに割り当てられている IAM ロールを上書きして、カスタム AMI を作成できます。

ここでは、このロールの一部として使用されるポリシーの最小セットを示します。

- `arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore` マネージド IAM ポリシー 詳細については、「AWS Systems Manager ユーザーガイド」の「AWS 用の管理ポリシー AWS Systems Manager」を参照してください。
- `arn:aws:iam::aws:policy/EC2InstanceProfileForImageBuilder` マネージド IAM ポリシー 詳細については、「Image Builder ユーザーガイド」の「`EC2InstanceProfileForImageBuilder` policy」を参照してください。
- 追加の IAM ポリシー:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ec2:CreateTags",
                "ec2:ModifyImageAttribute"
            ],
            "Resource": "arn:aws:ec2:<REGION>::image/*",
            "Effect": "Allow"
        }
    ]
}
```

AWS Lambda クリーンアップロール

Build / Iam / CleanupLambdaRole

このオプションは、カスタムイメージビルドプロセス中に使用されるすべての AWS Lambda 関数にアタッチされたロールを上書きします。 は、ロールを引き受けることができるプリンシパルとして設定 AWS Lambda する必要があります。

> **Note**
>
> DeploymentSettings/LambdaFunctionsVpcConfig が設定されている場合、CleanupLambdaRole には VPC 設定を設定するための AWS Lambda ロールのアクセス許可を含める必要があります。

ここでは、このロールの一部として使用されるポリシーの最小セットを示します。

- arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole マネージド IAM ポリシー 詳細については、「AWS Lambda 開発者ガイド」の「Lambda 機能のAWS マネージドポリシー」を参照してください。
- 追加の IAM ポリシー:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "iam:DetachRolePolicy",
                "iam:DeleteRole",
                "iam:DeleteRolePolicy"
            ],
            "Resource": "arn:aws:iam::<AWS ACCOUNT ID>:role/parallelcluster/*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "iam:DeleteInstanceProfile",
                "iam:RemoveRoleFromInstanceProfile"
            ],
            "Resource": "arn:aws:iam::<AWS ACCOUNT ID>:instance-profile/
parallelcluster/*",
            "Effect": "Allow"
```

```
        },
        {
            "Action": "imagebuilder:DeleteInfrastructureConfiguration",
            "Resource": "arn:aws:imagebuilder:<REGION>:<AWS ACCOUNT
 ID>:infrastructure-configuration/parallelclusterimage-*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "imagebuilder:DeleteComponent"
            ],
            "Resource": [
                "arn:aws:imagebuilder:<REGION>:<AWS ACCOUNT ID>:component/
parallelclusterimage-*/*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": "imagebuilder:DeleteImageRecipe",
            "Resource": "arn:aws:imagebuilder:<REGION>:<AWS ACCOUNT ID>:image-recipe/
parallelclusterimage-*/*",
            "Effect": "Allow"
        },
        {
            "Action": "imagebuilder:DeleteDistributionConfiguration",
            "Resource": "arn:aws:imagebuilder:<REGION>:<AWS ACCOUNT ID>:distribution-
configuration/parallelclusterimage-*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "imagebuilder:DeleteImage",
                "imagebuilder:GetImage",
                "imagebuilder:CancelImageCreation"
            ],
            "Resource": "arn:aws:imagebuilder:<REGION>:<AWS ACCOUNT ID>:image/
parallelclusterimage-*/*",
            "Effect": "Allow"
        },
        {
            "Action": "cloudformation:DeleteStack",
            "Resource": "arn:aws:cloudformation:<REGION>:<AWS ACCOUNT ID>:stack/*/*",
            "Effect": "Allow"
        },
```

```
        {
            "Action": "ec2:CreateTags",
            "Resource": "arn:aws:ec2:<REGION>::image/*",
            "Effect": "Allow"
        },
        {
            "Action": "tag:TagResources",
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "lambda:DeleteFunction",
                "lambda:RemovePermission"
            ],
            "Resource": "arn:aws:lambda:<REGION>:<AWS ACCOUNT
 ID>:function:ParallelClusterImage-*",
            "Effect": "Allow"
        },
        {
            "Action": "logs:DeleteLogGroup",
            "Resource": "arn:aws:logs:<REGION>:<AWS ACCOUNT ID>:log-group:/aws/
lambda/ParallelClusterImage-*:*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "SNS:GetTopicAttributes",
                "SNS:DeleteTopic",
                "SNS:GetSubscriptionAttributes",
                "SNS:Unsubscribe"
            ],
            "Resource": "arn:aws:sns:<REGION>:<AWS ACCOUNT ID>:ParallelClusterImage-
*",
            "Effect": "Allow"
        }
    ]
}
```

追加の IAM ポリシー

Build / Iam / AdditionalIamPolicies

このオプションを使用して、EC2 Image Builder がカスタム AMI を生成するために使用する Amazon EC2 インスタンスに関連付けられたロールに、追加のマネージド IAM ポリシーをアタッチします。

> **Warning**
>
> このオプションを使用するには、AWS ParallelClusterユーザーに、アタッチする必要のある IAM ポリシーの `iam:AttachRolePolicy` および `iam:DetachRolePolicy` の権限が付与されていることを確認してください。

アクセス許可の境界

`Build` / `Iam` / `PermissionsBoundary`

このパラメータ AWS ParallelCluster は、カスタム AMI ビルドの一部として作成されたすべての IAM ロール`PermissionsBoundary`に、指定された IAM ポリシーを として強制的にアタッチします。

このような機能を使用するために必要となるポリシーのリストについては、「`PermissionsBoundary` モード」を参照してください。

ネットワークの設定

AWS ParallelCluster は、ネットワークに Amazon Virtual Private Cloud (VPC) を使用します。VPC は、クラスターをデプロイすることができる柔軟で設定可能なネットワーキングプラットフォームを提供します。

VPC には`DNS Resolution = yes`、リージョンの正しいドメイン名を持つ `DNS Hostnames = yes`、、および DHCP オプションが必要です。デフォルトの DHCP オプションセットでは、必要な AmazonProvidedDNS がすでに指定されています。複数のドメインネームサーバーを指定する場合は、「Amazon VPC User Guide」(Amazon VPC ユーザーガイド) の「DHCP options sets」(DHCP オプションセット) を参照してください。

AWS ParallelCluster は、次の高レベル設定をサポートしています。

- ヘッドノードとコンピューティングノードの両方に 1 つのサブネット。
- 1 つのパブリックサブネットにヘッドノードを持ち、プライベートサブネットにコンピューティングノードを持つ 2 つのサブネット。サブネットは新規でも既存でもかまいません。

これらの設定はすべて、パブリック IP アドレス指定の有無にかかわらず動作します。 は、すべての AWS リクエストに HTTP プロキシを使用するようにデプロイ AWS ParallelCluster することもでき

ます。これらの設定を組み合わせることで、さまざまなデプロイシナリオにつながります。例えば、インターネットですべてのアクセスが可能な単一のパブリックサブネットを設定することができます。または、すべてのトラフィックに対して AWS Direct Connect と HTTP プロキシを使用して、完全プライベートネットワークを設定できます。

AWS ParallelCluster 3.0.0 以降では、キューごとに異なる 、`SecurityGroups`、`AdditionalSecurityGroups` `PlacementGroup`を設定できます。詳細については、「`HeadNode`/`Networking`」、「`SlurmQueues`/`Networking`」および「`AwsBatchQueues`/`Networking`」を参照してください。

ネットワークのシナリオのいくつかについては、以下のアーキテクチャ図を参照してください。

トピック

- AWS ParallelCluster 1 つのパブリックサブネット内の
- AWS ParallelCluster 2 つのサブネットの使用
- AWS ParallelCluster を使用して接続された単一のプライベートサブネット内の AWS Direct Connect
- AWS ParallelCluster スケジューラを使用する AWS Batch
- AWS ParallelCluster インターネットアクセスのない単一のサブネット内

AWS ParallelCluster 1 つのパブリックサブネット内の

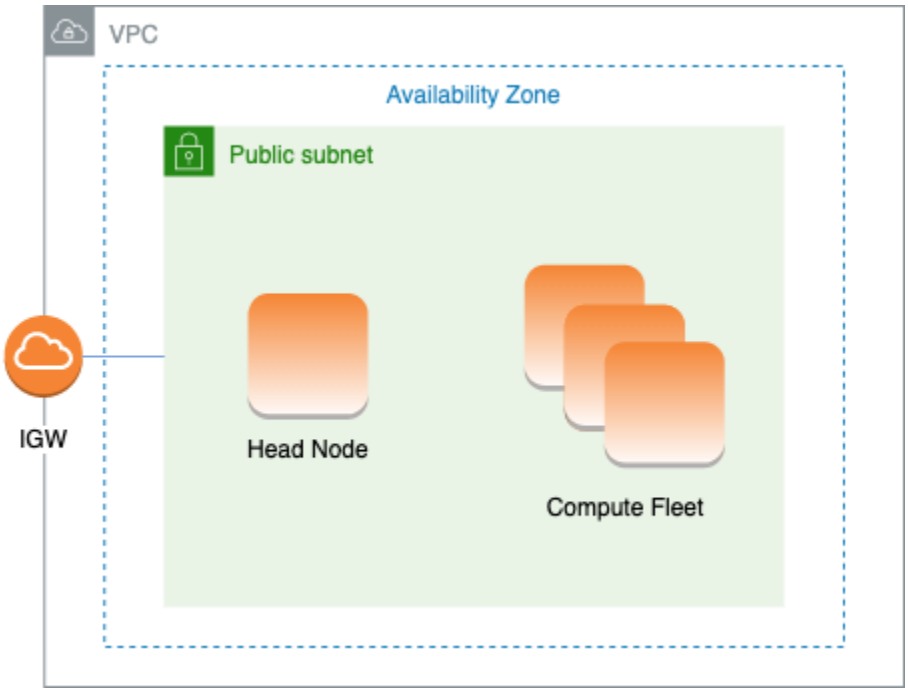
この構成では、インターネットにアクセスするために、クラスターのすべてのインスタンスにパブリック IP を割り当てる必要があります。これを達成するには、次の操作を行います。

- `HeadNode`/`Networking`/`SubnetId` で使用するサブネットの「パブリック IPv4 アドレスの自動割り当てを有効にする」設定をオンにするか、`HeadNode`/`Networking`/`ElasticIp` で Elastic IP を割り当てて、ヘッドノードにパブリック IP が割り当てられていることを確認してください。
- `Scheduling`/`SlurmQueues`/`Networking`/`SubnetIds` で使用しているサブネットの「パブリック IPv4 アドレスの自動割り当てを有効にする」設定をオンにするか、`Scheduling`/`SlurmQueues`/`Networking` で `AssignPublicIp` を true に設定することにより、コンピューティングノードにパブリック IP が割り当てられていることを確認してください。
- p4d インスタンスタイプ、またはヘッドノードに複数のネットワークインターフェイスまたはネットワークインターフェイスカードを持つ別のインスタンスタイプを定義する場合、パブリックアクセスを提供する`true`には`Networking`、`HeadNode`// `ElasticIp` を に設定する必要があります。AWS パブリック IPsは、単一のネットワークインターフェイスで起動されたインスタンスにのみ

割り当てることができます。この場合、NAT ゲートウェイを使用してクラスターコンピューティングノードへのパブリックアクセスを提供することをお勧めします。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「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。

- AWS パブリック IPs は 1 つのネットワークインターフェイスで起動されたインスタンスにのみ割り当てることができるため、 p4dまたは hp6id インスタンスタイプ、または複数のネットワークインターフェイスまたはネットワークインターフェイスカードを持つ別のインスタンスタイプを定義してノードを計算することはできません。詳細については、「Linux インスタンス用 Amazon EC2 ユーザーガイド」の「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。

詳細については、「Amazon VPC User Guide」(インターネットアクセスを有効にする) を参照してください。」の「Enabling internet access」(インターネットアクセスを有効にする)を参照してください。



このアーキテクチャの設定には、次の設定が必要です。

```
# Note that all values are only provided as examples
HeadNode:
  ...
  Networking:
    SubnetId: subnet-12345678 # subnet with internet gateway
```

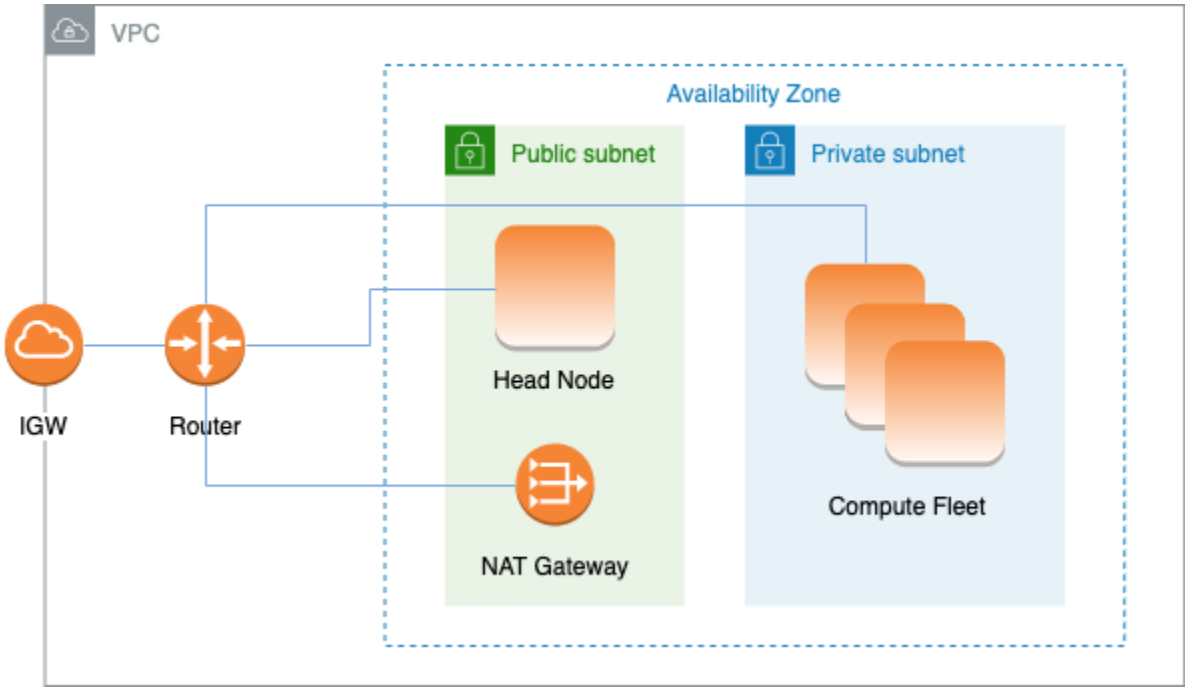
```
    #ElasticIp: true | false | eip-12345678
Scheduling:
  Scheduler: slurm
  SlurmQueues:
    - ...
      Networking:
        SubnetIds:
          - subnet-12345678 # subnet with internet gateway
        #AssignPublicIp: true
```

AWS ParallelCluster 2 つのサブネットの使用

この構成では、クラスターのヘッドノードにのみ、パブリック IP を割り当てる必要があります。HeadNode/Networking/SubnetId で使用するサブネットの [パブリック IPv4 アドレスの自動割り当てを有効にする] 設定をオンにするか、HeadNode/Networking/ElasticIp で Elastic IP を割り当てることで実現できます。

p4d インスタンスタイプ、またはヘッドノードに複数のネットワークインターフェイスまたはネットワークインターフェイスカードを持つ別のインスタンスタイプを定義する場合、パブリックアクセスを提供するtrueには Networking HeadNode// ElasticIpを に設定する必要があります。 AWS パブリック IPsは、単一のネットワークインターフェイスで起動されたインスタンスにのみ割り当てることができます。IP アドレスの詳細については、「Linux インスタンス用 Amazon EC2 ユーザーガイド」の「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。

この構成では、キューで使用するサブネットに NAT ゲートウェイまたは内部プロキシを設置し、コンピューティングインスタンスにインターネットアクセスを与える必要があります。

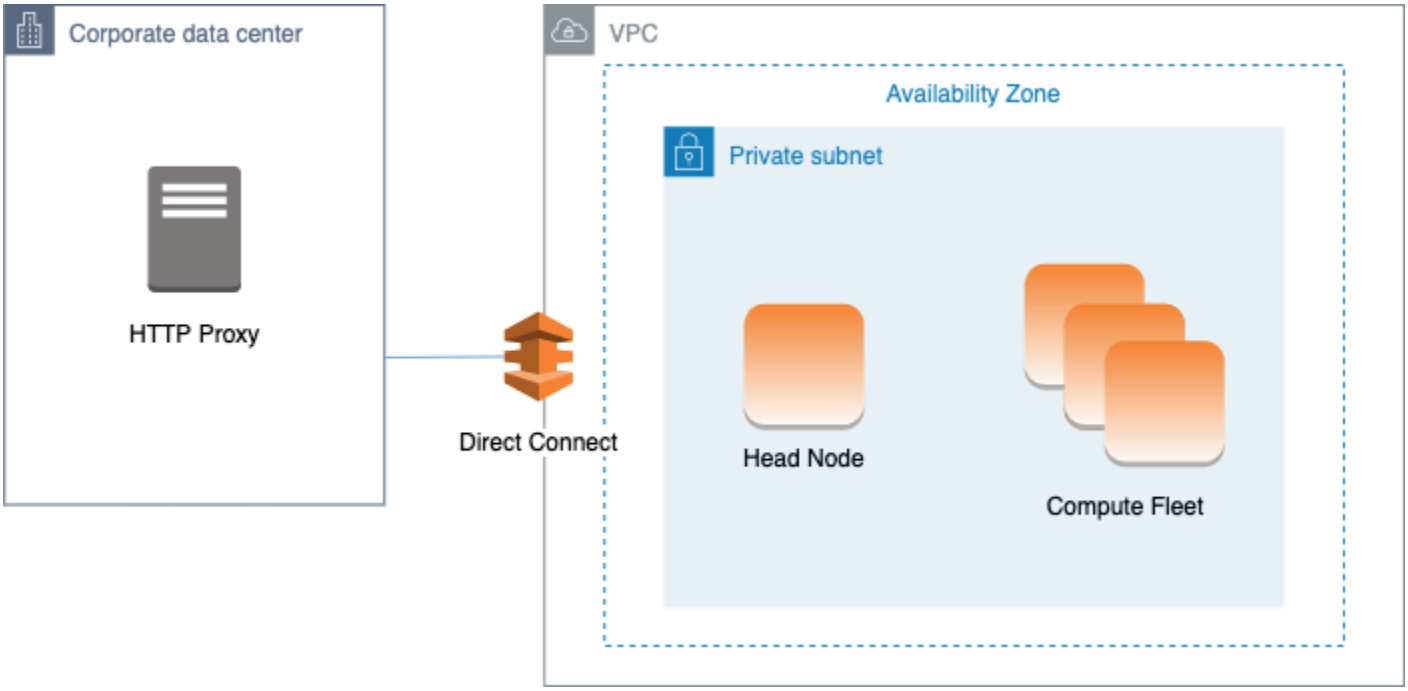


コンピューティングインスタンス用に既存のプライベートサブネットを使用するための構成には、次の設定が必要です。

```
# Note that all values are only provided as examples
HeadNode:
  ...
  Networking:
    SubnetId: subnet-12345678 # subnet with internet gateway
    #ElasticIp: true | false | eip-12345678
Scheduling:
  Scheduler: slurm
  SlurmQueues:
    - ...
      Networking:
        SubnetIds:
          - subnet-23456789 # subnet with NAT gateway
        #AssignPublicIp: false
```

AWS ParallelCluster を使用して接続された単一のプライベートサブネット内の AWS Direct Connect

Scheduling/SlurmQueues/Networking/AssignPublicIp が false に設定されている場合は、すべてのトラフィックにプロキシを使用するようにサブネットを正しく設定する必要があります。ウェブアクセスはヘッドノードとコンピューティングノードの両方に必要です。



このアーキテクチャの設定には、次の設定が必要です。

```
# Note that all values are only provided as examples
HeadNode:
  ...
  Networking:
    SubnetId: subnet-34567890 # subnet with proxy
    Proxy:
      HttpProxyAddress: http://proxy-address:port
  Ssh:
    KeyName: ec2-key-name
Scheduling:
  Scheduler: slurm
  SlurmQueues:
    - ...
      Networking:
        SubnetIds:
```

```
          - subnet-34567890 # subnet with proxy
        AssignPublicIp: false
        Proxy:
          HttpProxyAddress: http://proxy-address:port
```

AWS ParallelCluster スケジューラを使用する AWS Batch

をスケジューラタイプ`awsbatch`として使用すると、 は AWS Batch マネージドコンピューティング環境 AWS ParallelCluster を作成します。 AWS Batch 環境は、Amazon Elastic Container Service (Amazon ECS) コンテナインスタンスを管理します。これらのインスタンスは、`AwsBatchQueues`/`Networking`/`SubnetIds` パラメータで設定したサブネットで起動します。が正しく機能 AWS Batch するには、Amazon ECS コンテナインスタンスが Amazon ECS サービスエンドポイントと通信するための外部ネットワークアクセスが必要です。これは以下のシナリオに変換されます。

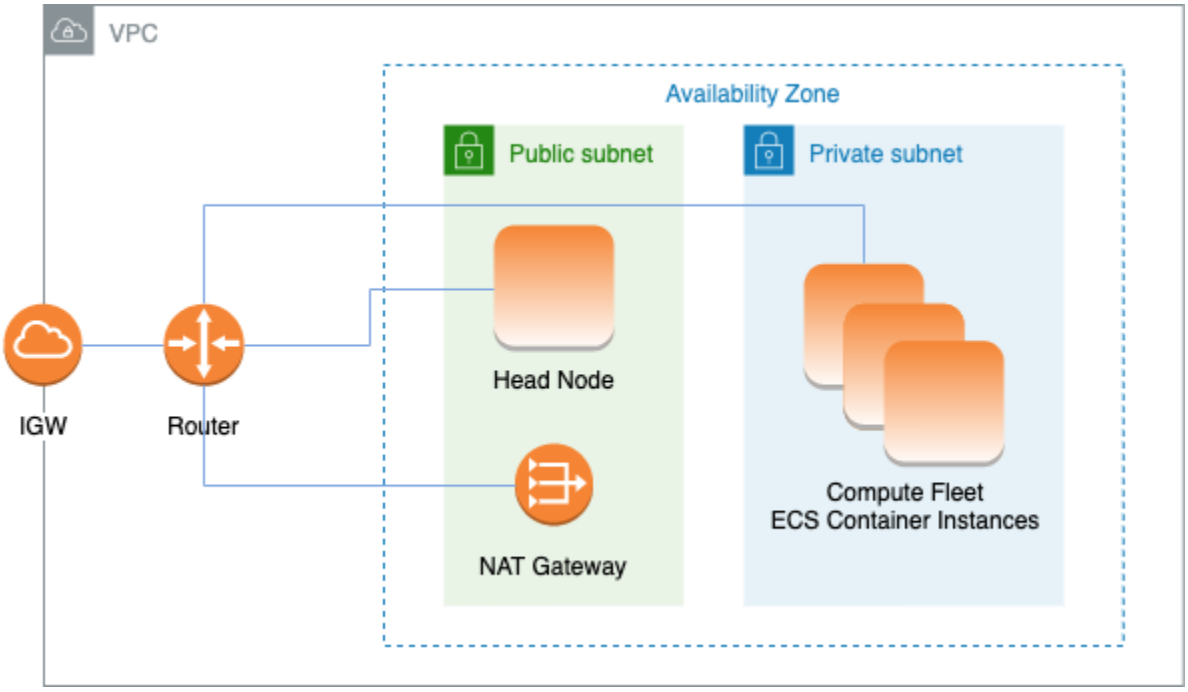
- キューに指定されているサブネット ID は、インターネットのアクセスに NAT ゲートウェイを使用しています。この手法をお勧めします。
- キューサブネットで起動したインスタンスは、パブリック IP アドレスを持ち、インターネットゲートウェイを介してインターネットに接続できます。

さらに、マルチノード並列ジョブ (AWS Batch ドキュメントより) に興味がある場合は以下を参照してください。

AWS Batch マルチノード並列ジョブは Amazon ECS `awsvpc`ネットワークモードを使用します。これにより、マルチノード並列ジョブコンテナは、Amazon EC2 インスタンスと同じネットワークプロパティを取得します。各マルチノード並列ジョブコンテナは、独自の Elastic Network Interface、プライマリプライベート IP アドレス、および内部の DNS ホスト名を取得します。ネットワークインターフェイスは、ホストコンピューティングリソースと同じ VPC サブネットで作成されます。コンピューティングリソースに適用されるすべてのセキュリティグループも同じく適用されます。

Amazon ECS タスクネットワークを使用している場合、`awsvpc` ネットワークモードは、Amazon EC2 起動タイプを使用するタスクにはパブリック IP アドレスを使用する Elastic Network Interface を提供しません。Amazon EC2 起動タイプを使用するタスクでインターネットにアクセスするには、NAT ゲートウェイを使用するよう設定されたプライベートサブネットでタスクを起動する必要があります。

クラスターがマルチノード並列ジョブを実行できるようにするには、NAT ゲートウェイを設定する必要があります。



前述のすべての設定と考慮事項 AWS Batchも有効です。 AWS Batch ネットワーク設定の例を次に示します。

```
# Note that all values are only provided as examples
HeadNode:
  ...
  Networking:
    SubnetId: subnet-12345678 # subnet with internet gateway, NAT gateway or proxy
    #ElasticIp: true | false | eip-12345678
    #Proxy:
      #HttpProxyAddress: http://proxy-address:port
  Ssh:
    KeyName: ec2-key-name
Scheduling:
  Scheduler: awsbatch
  AwsBatchQueues:
    - ...
      Networking:
        SubnetIds:
          - subnet-23456789 # subnet with internet gateway, NAT gateway or proxy
        #AssignPublicIp: true | false
```

`Scheduling`/`AwsBatchQueues`/`Networking` セクションでは `SubnetIds` はリストタイプですが、現在サポートされているのは 1 つのサブネットのみです。

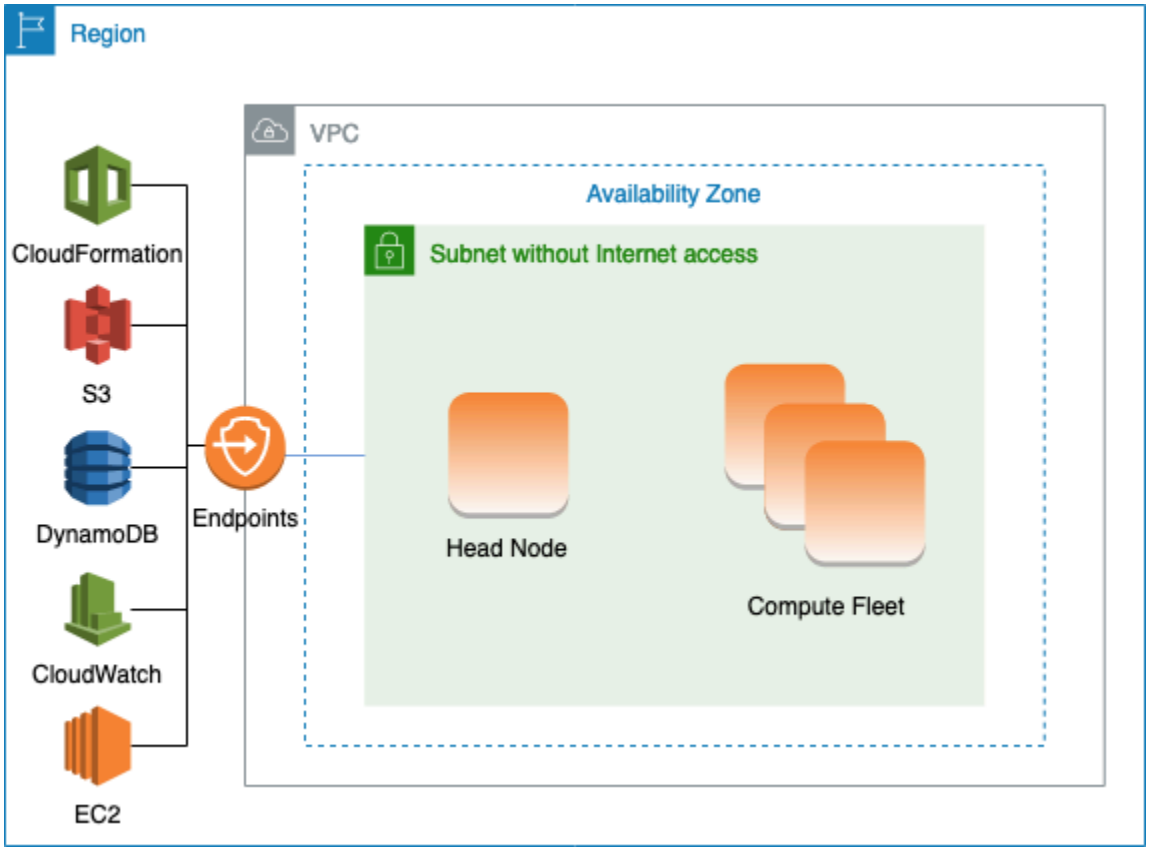
詳細については、以下の各トピックを参照してください。

- AWS Batch マネージドコンピューティング環境
- AWS Batch マルチノード並列ジョブ
- awsvpc ネットワークモードによる Amazon ECS タスクネットワーキング

AWS ParallelCluster インターネットアクセスのない単一のサブネット内

インターネットアクセスのないサブネットでは、インターネットへのインバウンド接続またはアウトバウンド接続は許可されていません。 AWS ParallelCluster この設定は、セキュリティに関心のあるお客様が AWS ParallelCluster リソースのセキュリティをさらに強化するのに役立ちます。 AWS ParallelCluster ノードは、インターネットアクセスなしでクラスターを実行するために必要なすべてのソフトウェアを含む AWS ParallelCluster AMIs から構築されます。これにより、 はインターネットアクセスのないノードを持つクラスターを作成および管理 AWS ParallelCluster できます。

このセクションでは、クラスターの設定方法について説明します。また、インターネットにアクセスせずにクラスターを実行する場合の制限についても説明します。



VPC エンドポイントの設定

クラスターが適切に機能するためには、クラスターノードが多数の AWS サービスとやり取りできる必要があります。

クラスターノードがインターネットアクセスなしで AWS サービスとやり取りできるように、次の VPC エンドポイントを作成して設定します。

Commercial and AWS GovCloud (US) partitions

サービス	サービス名	タイプ
Amazon CloudWatch	com.amazonaws.*region-id*.logs	インターフェイス
AWS CloudFormation	com.amazonaws.*region-id*.cloudformation	インターフェイス
Amazon EC2	com.amazonaws.*region-id*.ec2	インターフェイス
Amazon S3	com.amazonaws.*region-id*.s3	ゲートウェイ
Amazon DynamoDB	com.amazonaws.*region-id*.dynamodb	ゲートウェイ
AWS Secrets Manager**	com.amazonaws.*region-id*.secretsmanager	インターフェイス
AWS Elastic Load Balancing ***	com.amazonaws.*region-id*.elasticloadbalancing	インターフェイス
AWS Auto Scaling***	com.amazonaws.*region-id*.autoscaling	インターフェイス

China partition

サービス	サービス名	タイプ
Amazon CloudWatch	com.amazonaws.*region-id* .logs	インターフェイス
AWS CloudFormation	cn.com.amazonaws.*region-id* .cloudformation	インターフェイス
Amazon EC2	cn.com.amazonaws.*region-id* .ec2	インターフェイス
Amazon S3	com.amazonaws.*region-id* .s3	ゲートウェイ
Amazon DynamoDB	com.amazonaws.*region-id* .dynamodb	ゲートウェイ
AWS Secrets Manager**	com.amazonaws.*region-id* .secretsmanager	インターフェイス
AWS Elastic Load Balancing ***	com.amazonaws.*region-id* .elasticloadbalancing	インターフェイス
AWS Auto Scaling***	cn.com.amazonaws.*region-id* .autoscaling	インターフェイス

** このエンドポイントは `DirectoryService` が有効な場合にのみ必要で、それ以外の場合はオプションです。

*** これらのエンドポイントは LoginNodes が有効になっている場合にのみ必要です。それ以外の場合はオプションです。

VPC 内のすべてのインスタンスには、エンドポイントと通信するための適切なセキュリティグループが必要です。これを行うには、セキュリティグループを `HeadNode` の下にある `AdditionalSecurityGroups` と `SlurmQueues` 設定の下にある `AdditionalSecurityGroups`

に追加します。たとえば、セキュリティグループを明示的に指定せずに VPC エンドポイントを作成すると、デフォルトのセキュリティグループがエンドポイントに関連付けられます。デフォルトのセキュリティグループを `AdditionalSecurityGroups` に追加することで、クラスターとエンドポイント間の通信が可能になります。

> **Note**
>
> IAM ポリシーを使用して VPC エンドポイントへのアクセスを制限する場合、Amazon S3 VPC エンドポイントに以下を追加する必要があります。

```
PolicyDocument:
  Version: 2012-10-17
  Statement:
    - Effect: Allow
      Principal: "*"
      Action:
        - "s3:PutObject"
      Resource:
        - !Sub "arn:${AWS::Partition}:s3:::cloudformation-waitcondition-
${AWS::Region}/*"
```

Route 53 を無効にし、Amazon EC2 ホスト名を使用する

Slurm クラスターを作成すると、 は などのカスタムコンピューティングノードのホスト名を解決するために使用されるプライベート Route 53 ホストゾーン AWS ParallelCluster を作成します`{queue_name}-{st|dy}-{compute_resource}-{N}`。Route 53 は VPC エンドポイントをサポートしていないため、この機能を無効にする必要があります。さらに、 は、 などのデフォルトの Amazon EC2 ホスト名を使用するように設定 AWS ParallelCluster する必要があります`ip-1-2-3-4`。以下の設定をクラスター設定に適用します。

```
...
Scheduling:
  ...
  SlurmSettings:
    Dns:
      DisableManagedDns: true
      UseEc2Hostnames: true
```

> **Warning**
>
> SlurmSettings/Dns/DisableManagedDns および UseEc2Hostnames を true に設定して作成したクラスターの場合、Slurm NodeName は DNS によって解決されません。代わりに、Slurm NodeHostName を使用してください。

> **Note**
>
> このメモは、 AWS ParallelCluster バージョン 3.3.0 以降は関係ありません。
> 3.3.0 より前の AWS ParallelCluster サポートされているバージョンの場合：
> UseEc2Hostnames が に設定されている場合true、Slurm設定ファイルは AWS ParallelCluster prologおよび epilogスクリプトで設定されます。
>
> - prolog が実行されると、各ジョブが割り当てられるときにコンピューティングノードの /etc/hosts にノード情報が追加されます。
> - epilog が実行されると、prolog によって書き込まれた内容がクリーンアップされます。
>
> カスタム prolog または epilog スクリプトを追加するには、それぞれ /opt/slurm/etc/pcluster/prolog.d/ または /opt/slurm/etc/pcluster/epilog.d/ フォルダに追加します。

クラスターの設定

インターネットにアクセスしていないサブネットで実行するようにクラスターを構成する方法について説明します。

このアーキテクチャの設定には、次の設定が必要です。

```
# Note that all values are only provided as examples
...
HeadNode:
  ...
  Networking:
    SubnetId: subnet-1234567890abcdef0 # the VPC of the subnet needs to have VPC
 endpoints
```

```
    AdditionalSecurityGroups:
      - sg-abcdef01234567890 # optional, the security group that enables the
 communication between the cluster and the VPC endpoints
LoginNodes: # optional, if enabled, requires creation and configuration of VPC
 endpoints for AWS Elastic Load Balancing (ELB) and Auto Scaling services
  Pools:
    - ...
      Networking:
        SubnetIds:
          - subnet-1234567890abcdef0 # the VPC of the subnet needs to have VPC
 endpoints attached
        AdditionalSecurityGroups:
          - sg-1abcdef01234567890 # optional, the security group that enables the
 communication between the cluster and the VPC endpoints
Scheduling:
  Scheduler: Slurm # Cluster in a subnet without internet access is supported only when
 the scheduler is Slurm.
  SlurmSettings:
    Dns:
      DisableManagedDns: true
      UseEc2Hostnames: true
  SlurmQueues:
    - ...
      Networking:
        SubnetIds:
          - subnet-1234567890abcdef0 # the VPC of the subnet needs to have VPC
 endpoints attached
        AdditionalSecurityGroups:
          - sg-1abcdef01234567890 # optional, the security group that enables the
 communication between the cluster and the VPC endpoints
```

- SubnetId(s): インターネットアクセスのないサブネット

  AWS ParallelCluster と AWS サービス間の通信を有効にするには、サブネットの VPC に VPC エンドポイントがアタッチされている必要があります。クラスターを作成する前に、サブネットでパブリック IPv4 アドレスの自動割り当てが無効になっていることを確認して、pcluster コマンドがクラスターにアクセスできることを確認します。

- AdditionalSecurityGroups: クラスターと VPC エンドポイント間の通信を有効にするセキュリティグループ

  オプション:

- セキュリティグループを明示的に指定せずに VPC エンドポイントを作成すると、VPC のデフォルトのセキュリティグループが関連付けられます。そのため、デフォルトのセキュリティグループを `AdditionalSecurityGroups` に指定してください。
- クラスターや VPC エンドポイントの作成の際にカスタムセキュリティグループを使用する場合は、カスタムセキュリティグループがクラスターと VPC エンドポイント間の通信を有効にしている限り `AdditionalSecurityGroups` は不要です。

- `Scheduler`: クラスタースケジューラー

  唯一の有効な値は `slurm` です。インターネットアクセスのないサブネット内のクラスターをサポートするのは Slurm スケジューラーだけです。

- `SlurmSettings`: Slurm 設定。

  前のセクション「Route 53 を無効にし、Amazon EC2 ホスト名を使用する」を参照してください。

制約事項

- SSH または Amazon DCV 経由のヘッドノードへの接続: クラスターに接続するときは、接続のクライアントがプライベート IP アドレスを使用してクラスターのヘッドノードにアクセスできることを確認してください。クライアントがヘッドノードと同じ VPC にない場合は、VPC のパブリックサブネットにあるプロキシインスタンスを使用してください。この要件は SSH 接続と DCV 接続の両方に適用されます。サブネットにインターネットアクセスがない場合、ヘッドノードのパブリック IP にはアクセスできません。`pcluster ssh` および `dcv-connect` コマンドは、パブリック IP (存在する場合) またはプライベート IP を使用します。クラスターを作成する前に、サブネットでパブリック IPv4 アドレスの自動割り当てが無効になっていることを確認して、`pcluster` コマンドがクラスターにアクセスできることを確認します。

  次の例は、クラスターのヘッドノードで実行されている DCV セッションに接続する方法を示しています。プロキシ Amazon EC2 インスタンス経由で接続します。インスタンスは、PC の Amazon DCV サーバーとして、またプライベートサブネットのヘッドノードのクライアントとして機能します。

  パブリックサブネットのプロキシインスタンスを介して DCV 経由で接続します。

  1. クラスターのサブネットと同じ VPC にあるパブリックサブネットに Amazon EC2 インスタンスを作成します。

2. Amazon DCV クライアントおよびサーバーが Amazon EC2 インスタンスにインストールされていることを確認します。
3. ユーザー AWS ParallelCluster ポリシーをプロキシ Amazon EC2 インスタンスにアタッチします。詳細については、「AWS ParallelCluster `pcluster` ユーザーポリシーの例」を参照してください。
4. プロキシ Amazon EC2 インスタンス AWS ParallelCluster に をインストールします。
5. DCV 経由でプロキシ Amazon EC2 インスタンスに接続します。
6. プロキシインスタンスの `pcluster dcv-connect` コマンドを使用して、インターネットにアクセスせずにサブネット内のクラスターに接続します。

- 他の AWS サービスとやり取りする： が厳密に必要とするサービスのみを上記 AWS ParallelCluster に示します。クラスターが他のサービスとやり取りする必要がある場合は、対応する VPC エンドポイントを作成します。

によってプロビジョニングされたログインノード AWS ParallelCluster

バージョン 3.7.0 以降、 AWS ParallelCluster クラスター管理者は、クラスターヘッドノードに直接アクセスするのではなく、ジョブを実行するためのアクセスをユーザーに提供するために使用できるログインノードをプロビジョニングできます。 適切なアクセス許可を持つクラスターユーザーは、Active Directory または SSH 認証情報を使用して、ジョブのログイン、送信、管理を行うことができます。その結果、クラスター管理が改善され、Slurm でクラスター管理に必要なヘッドノードのリソースを使い果たす可能性を最小限に抑えることができます。ログインしたユーザーは、ログインノードにマウントされているクラスターのすべての共有ストレージにもアクセスできます。ログインノードを停止する必要がある場合、ログインしているユーザーは、使用しているアクティブなシェルセッションを通じて事前に通知されます。

ログインノードはプールとして指定され、プールは同じリソース設定を持つログインノードのグループを定義します。プール内のすべてのログインノードは、ラウンドロビン方式でログインノード全体にセッションを分散できる Network Load Balancer の一部となるように設定されます。現在の実装では、複数のログインノードプールを指定することをユーザーに許可します。

ログインノードのセキュリティ

ログインノードは、ヘッドノードから `AllowedIPs` 設定 `AllowedIps` を継承します。ただし、ログインノードプールに AllowedIps が指定されている場合を除きます。このようにして、クラスター

管理者は、ソース CIDR またはプレフィックスリストを指定することで、クラスターのセキュリティ体制を制限できます。これにより、ヘッドノードまたはログインノードのプールで SSH 接続が許可されます。

現在の実装では、ログインノードを有効にしてもヘッドノードへのアクセスは自動的に制限されません。必要に応じて、クラスター管理者は標準の Linux コマンドを使用してヘッドノードの SSH 設定を更新することで、このアクセスを制限できます。これは、ParallelCluster YAML ファイルのヘッドノードセクションの `AdditionalSecurityGroups` 設定を使用してヘッドノードにカスタムセキュリティグループを指定し、権限のないユーザーからの接続を拒否することによっても実現できます。

ログインノードのネットワーク

ログインノードには、ログインノードのプール用に設定された Network Load Balancer への単一の接続アドレスがプロビジョニングされます。アドレスの接続設定は、ログインノードのプール設定で指定されたサブネットのタイプに基づいています。

- サブネットがプライベートの場合、アドレスはプライベートになります。ログインノードへのアクセスを許可するには、クラスター管理者が踏み台ホストをプロビジョニングする必要があります。
- サブネットがパブリックの場合、アドレスはパブリックになります。

すべての接続リクエストは、ラウンドロビンルーティングを使用して Network Load Balancer によって管理されます。

ログインノードのストレージ

マネージドストレージを含め、ParallelCluster を使用してクラスターに設定されたすべての共有ストレージは、すべてのログインノードにマウントされます。

ログインノード情報の取得

ログインノードにアクセスするためにプロビジョニングされた単一接続のアドレスを取得するために、クラスター管理者は `describe-cluster` コマンドを実行できます。このコマンドは、ログインノードのステータスに関する詳細情報も提供します。

ログインノードは ParallelCluster でサポートされる新しいノードタイプで、特定のノードタイプのステータスを問い合わせる際にコマンド `describe-cluster-instances` で指定できます。

ログインノードプールへの単一接続アドレスが使用可能であっても、特定のログインノードへの直接アクセスが妨げられることはありません。ただし、SSH クライアントからの警告を避けるために

直接接続を使用することはお勧めしません。SSH クライアントは、ターゲットアドレスごとにホスト識別子をローカルに保存します。ホスト識別子はプールごとに固有であるため、異なる IP アドレスや単一接続アドレスを使用すると、同じホスト識別子が異なるターゲットアドレスに関連付けられることがあります。そのため、同じホスト 識別子が複数のターゲットに関連付けられているので、SSH クライアントから警告が表示されることがあります。

ログインノードの Imds プロパティ

ログインノードの IMDS (およびインスタンスプロファイル認証情報) へのアクセスは、ルートユーザー、クラスター管理ユーザー (デフォルトで `pc-cluster-admin`)、およびオペレーティングシステム固有のデフォルトユーザー (Amazon Linux 2 および RedHat の `ec2-user`、Ubuntu 18.04 の `ubuntu`) に制限されます。

IMDS アクセスを制限するには、 がチェーン AWS ParallelCluster を管理します`iptables`。

> **Note**
>
> `iptables` または `ip6tables` ルールをカスタマイズすると、ログインノードの IMDS アクセスを制限するメカニズムを妨げる可能性があります。「`Imds property setting`」も参照してください。

ログインノードのライフサイクル

現在、プール内のログインノードを停止および起動するための専用コマンドはありません。プール内のログインノードを停止するには、クラスター管理者がログインノードの数をゼロ (`Count: 0`) に指定してクラスター設定を更新し、`pcluster.update-cluster-v3` コマンドを実行する必要があります。

> **Note**
>
> ログインしているユーザーには、特定のインスタンスの終了およびそれに関連する猶予期間が通知されます。猶予期間中は、クラスターのデフォルトユーザーからの接続を除き、新しい接続は許可されません。表示されるメッセージは、クラスター管理者がヘッドノードまたはログインノードからファイル `/opt/parallelcluster/shared_login_nodes/loginmgtd_config.json` を編集してカスタマイズできます。この終了メッセージは、AWS Systems Manager Session ManagerSession Manager を使用して接続している場合は表示されません。

ログインノードプールを起動するために、クラスター管理者はクラスター設定内の以前の Count 値を復元してから update-cluster コマンドを実行する必要があります。

ログインノードプールの実行に必要なアクセス許可

ログインノードプールを管理するために、クラスター管理者は以下の追加のアクセス許可を持っている必要があります。

```
        - Action:
          - iam:CreateServiceLinkedRole
          - autoscaling:DeleteAutoScalingGroup
          - autoscaling:DeleteLifecycleHook
          - autoscaling:Describe*
          - autoscaling:PutLifecycleHook
          - autoscaling:UpdateAutoScalingGroup
          - elasticloadbalancing:CreateListener
          - elasticloadbalancing:CreateTargetGroup
          - elasticloadbalancing:DeleteListener
          - elasticloadbalancing:DeleteLoadBalancer
          - elasticloadbalancing:DeleteTargetGroup
          - elasticloadbalancing:Describe*
          - elasticloadbalancing:ModifyLoadBalancerAttributes
        Resource: '*'
        Condition:
          ForAllValues:StringEquals:
            aws:TagKeys: [ "parallelcluster:cluster-name" ]
        - Action:
          - autoscaling:CreateAutoScalingGroup
          - elasticloadbalancing:AddTags
          - elasticloadbalancing:CreateLoadBalancer
        Resource: '*'
        Effect: Allow
```

カスタムブートストラップアクション

/HeadNode/ CustomActionsOnNodeStart設定を定義すると、 はノードの起動直後に任意のコード AWS ParallelCluster を実行します。/HeadNode/ CustomActionsOnNodeConfigured設定を定義すると、 はノード設定が正しく完了した後にコード AWS ParallelCluster を実行します。

AWS ParallelCluster バージョン 3.4.0 以降では、// HeadNode CustomActionsOnNodeUpdated設定を定義すると、ヘッドノードの更新後にコードを実行できます。

通常、このコードは Amazon Simple Storage Service (Amazon S3) に保存され、HTTPS 接続でアクセスされます。コードは root として実行され、クラスターのオペレーティングシステムでサポートされている任意のスクリプト言語で実行できます。多くの場合、コードは Bash か Python で書かれています。

> **Note**
>
> AWS ParallelCluster バージョン 3.7.0 以降、クラスターImds/ImdsSupportデフォルト設定は ですv2.0。
> 新しいクラスターを作成してバージョン 3.7.0 以降のバージョンにアップグレードする場合は、カスタムブートストラップアクションスクリプトを IMDSv2 と互換性があるように更新するか、クラスター設定ファイルで Imds/ImdsSupport を v1.0 に設定してください。

> **Warning**
>
> 責任共有モデルで説明されているように、カスタムスクリプトと引数を設定するのはお客様の責任です。カスタムブートストラップスクリプトと引数が、クラスターノードへのフルアクセス権があると信頼できるソースからのものであることを確認してください。

> **Warning**
>
> AWS ParallelCluster は、 /etc/parallelcluster/cfnconfig ファイルを通じて提供される内部変数の使用をサポートしていません。このファイルは将来のリリースの一部として削除される可能性があります。

OnNodeStart アクションは、NAT、Amazon Elastic Block Store (Amazon EBS)、スケジューラの設定など、ノードデプロイのブートストラップアクションが開始される前に呼び出されます。OnNodeStart のブートストラップアクションには、ストレージの変更、その他のユーザーの追加、パッケージの追加などがあります。

> **Note**
>
> クラスターに DirectoryServiceと HeadNode // CustomActions OnNodeStartスクリプトを設定すると、 はOnNodeStartスクリプトを実行する前に sssd AWS ParallelClusterを設定DirectoryServiceして再起動します。

OnNodeConfigured アクションは、ノードのブートストラップ・プロセスが完了した後に呼び出されます。OnNodeConfigured アクションは、インスタンスが完全に構成され、完了したとみなされる前の最後のアクションです。一般的な OnNodeConfigured のアクションには、スケジューラ設定の変更、ストレージやパッケージの変更などがあります。設定時に指定することで、スクリプトに引数を渡すことができます。

ヘッドノードの更新が完了し、スケジューラーと共有ストレージが最新のクラスター設定変更に対応するようになった後、OnNodeUpdated アクションが呼び出されます。

OnNodeStart または OnNodeConfigured カスタムアクションが成功すると、終了コード 0 で成功が示されます。それ以外の終了コードは、インスタンスのブートストラップが失敗したことを示します。

OnNodeUpdated カスタムアクションが成功すると、終了コード 0 で成功が通知されます。それ以外の終了コードは、失敗したことを示します。

> **Note**
>
> OnNodeUpdated を設定した場合、更新が失敗したときに OnNodeUpdated アクションを以前の状態に手動で復元する必要があります。
> OnNodeUpdated カスタムアクションが失敗した場合、更新は以前の状態にロールバックします。ただし、OnNodeUpdated アクションは更新時にのみ実行され、スタックのロールバック時には実行されません。

/ および i / HeadNode / CustomActions設定セクションでは、ヘッドノードCustomActionsとキューごとに異なるスクリプトを指定できます。 SlurmQueues は Scheduling HeadNodeセクションOnNodeUpdatedでのみ設定できます。

> **Note**
>
> AWS ParallelCluster バージョン 3.0 以前は、ヘッドノードとコンピューティングノードに異なるスクリプトを指定できませんでした。「AWS ParallelCluster 2.x から 3.x への移行」を参照してください。

トピック

- アクションと引数を定義するための構成設定
- 引数
- カスタムブートストラップアクションを使用したクラスターの例
- IMDSv2 のカスタムブートストラップスクリプトを更新する方法の例
- IMDSv1 の設定を更新する方法の例

アクションと引数を定義するための構成設定

次の設定は、HeadNode/CustomActions/OnNodeStart & OnNodeConfigured & OnNodeUpdated および Scheduling/CustomActions/OnNodeStart & OnNodeConfigured アクションと引数を定義するために使用されます。

```
HeadNode:
  [...]
  CustomActions:
    OnNodeStart:
      # Script URL. This is run before any of the bootstrap scripts are run
      Script: s3://amzn-s3-demo-bucket/on-node-start.sh
      Args:
        - arg1
    OnNodeConfigured:
      # Script URL. This is run after all the bootstrap scripts are run
      Script: s3://amzn-s3-demo-bucket/on-node-configured.sh
      Args:
        - arg1
    OnNodeUpdated:
      # Script URL. This is run after the head node update is completed.
      Script: s3://amzn-s3-demo-bucket/on-node-updated.sh
      Args:
        - arg1
  # Bucket permissions
```

```
  Iam:
    S3Access:
      - BucketName: bucket_name
        EnableWriteAccess: false
Scheduling:
  Scheduler: slurm
   [...]
  SlurmQueues:
    - Name: queue1
      [...]
      CustomActions:
        OnNodeStart:
          Script: s3://amzn-s3-demo-bucket/on-node-start.sh
          Args:
            - arg1
        OnNodeConfigured:
          Script: s3://amzn-s3-demo-bucket/on-node-configured.sh
          Args:
            - arg1
      Iam:
        S3Access:
          - BucketName: bucket_name
            EnableWriteAccess: false
```

Sequence 設定の使用 (AWS ParallelCluster バージョン 3.6.0 で追加):

```
HeadNode:
  [...]
  CustomActions:
    OnNodeStart:
      # Script URLs. The scripts are run in the same order as listed in the
 configuration, before any of the bootstrap scripts are run.
      Sequence:
        - Script: s3://amzn-s3-demo-bucket/on-node-start1.sh
          Args:
            - arg1
        - Script: s3://amzn-s3-demo-bucket/on-node-start2.sh
          Args:
            - arg1
        [...]
    OnNodeConfigured:
      # Script URLs. The scripts are run in the same order as listed in the
 configuration, after all the bootstrap scripts are run.
```

```
      Sequence:
        - Script: s3://amzn-s3-demo-bucket/on-node-configured1.sh
          Args:
            - arg1
        - Script: s3://amzn-s3-demo-bucket/on-node-configured2.sh
          Args:
            - arg1
        [...]
    OnNodeUpdated:
      # Script URLs. The scripts are run in the same order as listed in the
 configuration, after the head node update is completed.
      Sequence:
        - Script: s3://amzn-s3-demo-bucket/on-node-updated1.sh
          Args:
            - arg1
        - Script: s3://amzn-s3-demo-bucket/on-node-updated2.sh
          Args:
            - arg1
        [...]
  # Bucket permissions
  Iam:
    S3Access:
      - BucketName: bucket_name
        EnableWriteAccess: false
Scheduling:
  Scheduler: slurm
   [...]
  SlurmQueues:
    - Name: queue1
      [...]
      CustomActions:
        OnNodeStart:
          # Script URLs. The scripts are run in the same order as listed in the
 configuration, before any of the bootstrap scripts are run
          Sequence:
            - Script: s3://amzn-s3-demo-bucket/on-node-start1.sh
              Args:
                - arg1
            - Script: s3://amzn-s3-demo-bucket/on-node-start2.sh
              Args:
                - arg1
            [...]
        OnNodeConfigured:
```

```
          # Script URLs. The scripts are run in the same order as listed in the
 configuration, after all the bootstrap scripts are run
          Sequence:
            - Script: s3://amzn-s3-demo-bucket/on-node-configured1.sh
              Args:
                - arg1
            - Script: s3://amzn-s3-demo-bucket/on-node-configured2.sh
              Args:
                - arg1
            [...]
      Iam:
        S3Access:
          - BucketName: bucket_name
            EnableWriteAccess: false
```

Sequence 設定は AWS ParallelCluster バージョン 3.6.0 から追加されます。Sequence を指定すると、カスタムアクション用の複数のスクリプトを一覧表示できます。Sequence を追加しなくても、 AWS ParallelCluster は 1 つのスクリプトによるカスタムアクションの設定を引き続きサポートします。

AWS ParallelCluster は、同じカスタムアクションSequenceに 1 つのスクリプトと の両方を含めることをサポートしていません。たとえば、次の設定を指定すると、 は AWS ParallelCluster 失敗します。

```
[...]
  CustomActions:
    OnNodeStart:
      # Script URL. This is run before any of the bootstrap scripts are run
      Script: s3://amzn-s3-demo-bucket/on-node-start.sh
          Args:
            - arg1
      # Script URLs. The scripts are run in the same order as listed in the
 configuration, before any of the bootstrap scripts are run.
      Sequence:
        - Script: s3://amzn-s3-demo-bucket/on-node-start1.sh
          Args:
            - arg1
        - Script: s3://amzn-s3-demo-bucket/on-node-start2.sh
          Args:
            - arg1
[...]
```

引数

In AWS ParallelCluster 2.x 引$1数は、カスタムスクリプトの URL を保存するために予約された引数でした。for AWS ParallelCluster 2.x with AWS ParallelCluster 3.x で作成されたカスタムブートストラップスクリプトを再利用する場合は、引数のシフトを考慮して調整する必要があります。「AWS ParallelCluster 2.x から 3.x への移行」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を使用したクラスターの例

次の手順では、ノードの設定後に実行される簡単なスクリプトを作成し、クラスターのノードに `R`, `curl` および `wget` パッケージをインストールします。

1. [Create a script].(スクリプトを作成します)。

```
#!/bin/bash
  echo "The script has $# arguments"
  for arg in "$@"
  do
      echo "arg: ${arg}"
  done
  yum -y install "${@:1}"
```

2. Amazon S3 に正しいアクセス許可でスクリプトをアップロードしてください。パブリック読み取りアクセス許可が適切でない場合は、`HeadNode`/`Iam`/`S3Access` および `Scheduling`/`SlurmQueues` の設定セクションを使用してください。詳細については、「Amazon S3 での使用」を参照してください。

```
$ aws s3 cp --acl public-read /path/to/myscript.sh s3://amzn-s3-demo-
bucket/myscript.sh
```

> **Important**
> スクリプトが Windows で編集された場合、スクリプトを Amazon S3 にアップロードする前に、行末を CRLF から LF に変更する必要があります。

3. 新しい`OnNodeConfigured`アクションを含めるように AWS ParallelCluster 設定を更新します。

```
CustomActions:
  OnNodeConfigured:
    Script: https://<amzn-s3-demo-bucket>.s3.<region>.amazonaws.com/myscript.sh
```

```
    Args:
      - "R"
      - "curl"
      - "wget"
```

バケットにパブリック読み取りのアクセス許可がない場合は、URL プロトコルとして s3 を使用します。

```
CustomActions:
  OnNodeConfigured:
    Script: s3://amzn-s3-demo-bucket/myscript.sh
    Args:
      - "R"
      - "curl"
      - "wget"
```

4. クラスターを起動します。

```
$ pcluster create-cluster --cluster-name mycluster \
  --region <region> --cluster-configuration config-file.yaml
```

5. 出力の検証

- HeadNode 設定にカスタムアクションを追加した場合は、ヘッドノードにログインし、次のコマンド/var/log/cfn-init.logを実行して にある cfn-init.log ファイルを確認します。

```
$ less /var/log/cfn-init.log
  2021-09-03 10:43:54,588 [DEBUG] Command run
  postinstall output: The script has 3 arguments
  arg: R
  arg: curl
  arg: wget
  Loaded plugins: dkms-build-requires, priorities, update-motd, upgrade-helper
  Package R-3.4.1-1.52.amzn1.x86_64 already installed and latest version
  Package curl-7.61.1-7.91.amzn1.x86_64 already installed and latest version
  Package wget-1.18-4.29.amzn1.x86_64 already installed and latest version
  Nothing to do
```

- SlurmQueues 設定にカスタムアクションを追加した場合は、コンピューティングノードの /var/log/cloud-init.log にある cloud-init.log を確認します。CloudWatch を使用してこれらのログを表示します。

Amazon CloudWatch コンソールでこれら両方のログを表示できます。詳細については、「Amazon CloudWatch Logs との統合」を参照してください。

IMDSv2 のカスタムブートストラップスクリプトを更新する方法の例

次の例では、IMDSv1 で使用されていたカスタムブートストラップアクションスクリプトを IMDSv2 で使用できるように更新します。IMDSv1 スクリプトは Amazon EC2 インスタンスの AMI ID メタデータを取得します。

```
#!/bin/bash
AMI_ID=$(curl http://169.254.169.254/latest/meta-data/ami-id)
echo $AMI_ID >> /home/ami_id.txt
```

以下は、IMDSv2 と互換性があるように変更されたカスタムブートストラップアクションスクリプトを示しています。

```
#!/bin/bash
AMI_ID=$(TOKEN=`curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-
metadata-token-ttl-seconds: 21600"` \
         && curl -H "X-aws-ec2-metadata-token: $TOKEN" -v http://169.254.169.254/
latest/meta-data/ami-id)
echo $AMI_ID >> /home/ami_id.txt
```

詳細については、Linux インスタンス用 Amazon EC2 ユーザーガイドのインスタンスメタデータの取得を参照してください。 Amazon EC2

IMDSv1 の設定を更新する方法の例

以下は、 AWS ParallelCluster バージョン 3.7.0 以前を使用している場合に IMDSv1 をサポートするクラスター設定の例です。

```
Region: us-east-1
Imds:
  ImdsSupport: v1.0
Image:
  Os: alinux2
HeadNode:
  InstanceType: t2.micro
  Networking:
```

```
    SubnetId: subnet-abcdef01234567890
  Ssh:
    KeyName: key-name
  CustomActions:
    OnNodeConfigured:
      Script: Script-path
Scheduling:
  Scheduler: slurm
  SlurmQueues:
  - Name: queue1
    CustomActions:
      OnNodeConfigured:
        Script: Script-path
    ComputeResources:
    - Name: t2micro
      Instances:
      - InstanceType: t2.micro
      MinCount: 11
    Networking:
      SubnetIds:
      - subnet-abcdef01234567890
```

Amazon S3 での使用

設定 AWS ParallelClusterの / / S3Accessおよび HeadNode / Iam / Scheduling / SlurmQueues / S3Accessパラメータを使用して、Amazon S3 - Name へのアクセスを設定できます AWS ParallelCluster 。 Iam

例

次の例では、*firstbucket/read_only/* のすべてのオブジェクトへの読み取り専用アクセスと、 のすべてのオブジェクトへの読み取り/書き込みアクセスを設定します*secondbucket/read_and_write/*。

```
...
HeadNode:
  ...
  Iam:
    S3Access:
      - BucketName: firstbucket
        KeyName: read_only/*
```

```
        EnableWriteAccess: false
      - BucketName: secondbucket
        KeyName: read_and_write/*
        EnableWriteAccess: true
...
```

次の例では、アカウント内の任意のバケット (*) にあるフォルダ *read_only/* 内のすべてのオブジェクトへの読み取り専用アクセスを設定します。

```
...
HeadNode:
  ...
  Iam:
    S3Access:
      - BucketName: *
        KeyName: read_only/*
        EnableWriteAccess: false
...
```

最後の例では、アカウント内のすべてのバケットとオブジェクトへの読み取り専用アクセスを設定します。

```
...
HeadNode:
  ...
  Iam:
    S3Access:
      - BucketName: *
...
```

スポットインスタンスの操作

AWS ParallelCluster クラスター設定ファイルSPOTで `SlurmQueues` / `CapacityType`または / `AwsBatchQueues` `CapacityType`を に設定している場合、 はスポットインスタンスを使用します。スポットインスタンスはオンデマンドインスタンスよりも費用対効果が高いが、中断される可能性があります。スポットインスタンスの中断通知を利用すると、Amazon EC2 がスポットインスタンスを停止または終了するまで 2 分間の警告が表示されます。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の中断」を参照してください。`AwsBatchQueues` とスポットインスタンスとの連携方法については、「AWS Batch ユーザーガイド」の「コンピューティングリソース」を参照してください。

AWS ParallelCluster 設定されたスケジューラは、オンデマンドインスタンスを持つキュー内のコンピューティングリソースにジョブを割り当てるのと同じ方法で、スポットインスタンスを持つキュー内のコンピューティングリソースにジョブを割り当てます。

スポットインスタンスを使用する場合、アカウントに AWSServiceRoleForEC2Spot サービスにリンクされたロールが存在する必要があります。を使用してアカウントにこのロールを作成するには AWS CLI、次のコマンドを実行します。

```
$ aws iam create-service-linked-role --aws-service-name spot.amazonaws.com
```

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リクエスト向けのサービスにリンクされたロール」を参照してください。

以下のセクションでは、`SlurmQueues` の使用時にスポットインスタンスを中断できる 3 つのシナリオについて説明します。

シナリオ 1: 実行中のジョブがないスポットインスタンスが中断される

この中断が発生すると、スケジューラキューに追加のインスタンスを必要とする保留中のジョブがある場合、またはアクティブなインスタンスの数が `SlurmQueues` /`ComputeResources`/ よりも少ない場合、 はインスタンスの置き換え AWS ParallelCluster を試みます`MinCount`。 AWS ParallelCluster が新しいインスタンスをプロビジョニングできない場合、新しいインスタンスのリクエストが定期的に繰り返されます。

シナリオ 2: 単一ノードジョブを実行しているスポットインスタンスが中断される

ステートコード `NODE_FAIL` で失敗し、ジョブは再キューされます (ジョブ投入時に `--no-requeue` が指定されていない場合)。静的ノードの場合は、置換されます。動的ノードの場合は、ノードは終了してリセットされます。`sbatch` の詳細 (`--no-requeue` パラメータを含む) については、Slurm のドキュメントの「sbatch」を参照してください。

シナリオ 3: マルチノードジョブを実行しているスポットインスタンスが中断される

ステートコード `NODE_FAIL` で失敗し、ジョブは再キューされます (ジョブ投入時に `--no-requeue` が指定されていない場合)。静的ノードの場合は、置換されます。動的ノードの場合は、ノードは終了してリセットされます。終了したジョブを実行していた他のノードは、他の保留中の

ジョブに割り当てられたり、設定された SlurmSettings/ScaledownIdletime 時間が経過した後にスケールダウンされたりする場合があります。

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。

でサポートされているスケジューラ AWS ParallelCluster

AWS ParallelCluster は、 Scheduler 設定を使用して設定された Slurmおよび ス AWS Batch ケジューラをサポートします。以下のトピックでは、各スケジューラとその使用方法について説明します。

トピック

- Slurm Workload Manager (slurm)
- での AWS Batch （ awsbatch) スケジューラの使用 AWS ParallelCluster

Slurm Workload Manager (**slurm**)

クラスター容量のサイズと更新

クラスターの容量は、クラスターがスケールできるコンピューティングノードの数によって定義されます。コンピューティングノードは、 AWS ParallelCluster 設定 のコンピューティングリソース内で定義された Amazon EC2 インスタンスによってバックアップされ(Scheduling/SlurmQueues/ComputeResources)、Slurmパーティションに 1:1 をマッピング(Scheduling/SlurmQueues)するキューに編成されます。

コンピューティングリソース内では、クラスターで常に実行する必要があるコンピューティングノード (インスタンス) の最小数 (MinCount) 、およびコンピューティングリソースがスケールできるインスタンスの最大数 (MaxCount3) を設定できます。

クラスターの作成時、またはクラスターの更新時に、 はクラスターで定義された各コンピューティングリソース (Scheduling/SlurmQueues/ ComputeResources) MinCount に対して で設定された数だけ Amazon EC2 インスタンス AWS ParallelCluster を起動します。クラスター内のコンピューティングリソースの最小ノード数をカバーするために起動されたインスタンスは、静的ノードと呼ばれます。静的ノードは、起動されるとクラスター内で永続的となり、特定のイベントや条件が発生しない限り、システムによって終了されることはありません。このようなイベントには、Slurm や Amazon EC2 ヘルスチェックの失敗、Slurm ノードステータスの DRAIN や DOWN への変更などがあります。

クラスターの負荷の増加に対応するために、1 から ‘MaxCount - MinCount’ (MaxCount マイナス MinCount) までの範囲で、オンデマンドで起動される Amazon EC2 インスタンスは、動的ノードと呼ばれます。動的ノードはエフェメラルであり、保留中のジョブを処理するために起動され、クラスター設定の Scheduling/SlurmSettings/ScaledownIdletime で定義した期間 (デフォルトは 10 分) にわたってアイドル状態が続くと終了されます。

静的ノードと動的ノードは、次の命名スキーマに準拠します。

- 静的ノード <Queue/Name>-st-<ComputeResource/Name>-<num> (この場合、<num> = 1..ComputeResource/MinCount となります)
- 動的ノード <Queue/Name>-dy-<ComputeResource/Name>-<num> (この場合、<num> = 1..(ComputeResource/MaxCount - ComputeResource/MinCount) となります)

たとえば、次の AWS ParallelCluster 設定があるとします。

```
Scheduling:
    Scheduler: Slurm
    SlurmQueues:
        - Name: queue1
            ComputeResources:
                - Name: c5xlarge
                    Instances:
                        - InstanceType: c5.xlarge
                        MinCount: 100
                        MaxCount: 150
```

Slurm では、以下のノードが定義されます。

```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
queue1*      up   infinite     50  idle~ queue1-dy-c5xlarge-[1-50]
queue1*      up   infinite    100   idle queue1-st-c5xlarge-[1-100]
```

コンピューティングリソースに MinCount == MaxCount がある場合、すべての対応するコンピューティングノードは静的になり、すべてのインスタンスはクラスターの作成/更新時に起動されて、稼働状態が維持されます。次に例を示します。

```
Scheduling:
  Scheduler: slurm
  SlurmQueues:
    - Name: queue1
      ComputeResources:
        - Name: c5xlarge
          Instances:
            - InstanceType: c5.xlarge
          MinCount: 100
          MaxCount: 100
```

```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
queue1*      up   infinite    100   idle queue1-st-c5xlarge-[1-100]
```

クラスター容量の更新

クラスター容量の更新には、キューやコンピューティングリソースの追加または削除、コンピューティングリソースの `MinCount/MaxCount` の変更などが含まれます。 AWS ParallelCluster バージョン 3.9.0 以降では、キューのサイズを小さくするには、クラスターの更新が行われる前にコンピューティングフリートを停止するか、QueueUpdateStrategy を TERMINATE に設定する必要があります。以下の場合は、コンピューティングフリートを停止したり、QueueUpdateStrategy を TERMINATE に設定したりする必要はありません。

- 新しいキューをスケジューリング/`SlurmQueues` に追加する
- 新しいコンピューティングリソースを `Scheduling/SlurmQueues/ComputeResources` をキューに追加する
- コンピューティングリソースの `MaxCount` を増やす
- コンピューティングリソースの MinCount を増やし、同じコンピューティングリソースの MaxCount を同等以上に増やす

考慮事項と制限事項

このセクションでは、クラスター容量のサイズを変更する際に考慮すべき重要な要因、制約、または制限について概説します。

- Scheduling/SlurmQueues からキューを削除すると、名前が <Queue/Name>-* であるすべてのコンピューティングノード (静的と動的の両方) が Slurm 設定から削除され、対応する Amazon EC2 インスタンスが終了します。
- キューからコンピューティングリソース Scheduling/SlurmQueues/ComputeResources を削除すると、名前が <Queue/Name>-*-<ComputeResource/Name>-* であるすべてのコンピューティングノード (静的と動的の両方) が Slurm 設定から削除され、対応する Amazon EC2 インスタンスが終了します。

コンピューティングリソースの MinCount パラメータを変更するときは、MaxCount が MinCount と等しい場合 (静的容量のみ) と MaxCount が MinCount より大きい場合 (静的容量と動的容量の混合) の 2 つの異なるシナリオを区別できます。

静的ノードのみの容量変更

- MinCount == MaxCount の場合、MinCount (および MaxCount) を増やすには、静的ノードの数を MinCount の新しい値まで拡張するようにクラスターを設定します (<Queue/Name>-st-<ComputeResource/Name>-<new_MinCount>)。システムは新しい静的容量を満たすまで必要な数の Amazon EC2 インスタンスを起動し続けます。
- MinCount == MaxCount の場合、MinCount (および MaxCount) を量 N だけ減らすには、最後の N 個の静的ノードを削除するようにクラスターを設定します (<Queue/Name>-st-<ComputeResource/Name>-<old_MinCount - N>...<old_MinCount>])。システムは対応する数の Amazon EC2 インスタンスを終了します。
  - 初期状態: MinCount = MaxCount = 100
  -
    ```
    $ sinfo
    PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
    queue1*      up   infinite    100   idle queue1-st-c5xlarge-[1-100]
    ```
  - 更新 (-30) を適用した MinCount および MaxCount: MinCount = MaxCount = 70
  -

```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
queue1*      up   infinite     70   idle queue1-st-c5xlarge-[1-70]
```

混合ノードの容量変更

MinCount < MaxCount の場合、MinCount を量 N だけ増やすには (MaxCount は変更しないと仮定)、静的ノードの数を MinCount の新しい値 (old_MinCount + N) まで拡張するようにクラスターを設定します (<Queue/Name>-st-<ComputeResource/Name>-<old_MinCount + N>)。システムは新しい静的容量を満たすまで必要な数の Amazon EC2 インスタンスを起動し続けます。さらに、コンピューティングリソースの MaxCount 容量を保持するために、最後の N 個の動的ノードを削除するようにクラスター設定を更新します (<Queue/Name>-dy-<ComputeResource/Name>-[<MaxCount - old_MinCount - N>...<MaxCount - old_MinCount>])。システムは対応する数の Amazon EC2 インスタンスを終了します。

- 初期状態: MinCount = 100; MaxCount = 150
-
```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
queue1*      up   infinite     50  idle~ queue1-dy-c5xlarge-[1-50]
queue1*      up   infinite    100   idle queue1-st-c5xlarge-[1-100]
```
- 更新 (+30) を適用した MinCount : MinCount = 130 (MaxCount = 150)
-
```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
queue1*      up   infinite     20  idle~ queue1-dy-c5xlarge-[1-20]
queue1*      up   infinite    130   idle queue1-st-c5xlarge-[1-130]
```

MinCount < MaxCount の場合、MinCount と MaxCount を同じ量 N だけ増やすには、静的ノードの数を MinCount の新しい値 (old_MinCount + N) まで拡張するようにクラスターを設定します (<Queue/Name>-st-<ComputeResource/Name>-<old_MinCount + N>)。システムは新し

い静的容量を満たすまで必要な数の Amazon EC2 インスタンスを起動し続けます。さらに、動的ノードの数は変更せずに、新しい

MaxCount の値を保持します。

- 初期状態: MinCount = 100; MaxCount = 150
-
```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
queue1*      up   infinite     50  idle~ queue1-dy-c5xlarge-[1-50]
queue1*      up   infinite    100   idle queue1-st-c5xlarge-[1-100]
```
- 更新 (+30) を適用した MinCount : MinCount = 130 (MaxCount = 180)
-
```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
queue1*      up   infinite     20  idle~ queue1-dy-c5xlarge-[1-50]
queue1*      up   infinite    130   idle queue1-st-c5xlarge-[1-130]
```

MinCount < MaxCount の場合、MinCount を量 N だけ減らすには (MaxCount は変更しないと仮定)、最後の N 個の静的ノードを削除するようにクラスターを設定します (<Queue/Name>-st-<ComputeResource/Name>-[<old_MinCount - N>...<old_MinCount>)。システムは対応する数の Amazon EC2 インスタンスを終了します。さらに、コンピューティングリソースの MaxCount 容量を保持するために、動的ノードの数を拡張してギャップを埋めるようにクラスター設定を更新します (MaxCount - new_MinCount: <Queue/Name>-dy-<ComputeResource/Name>-[1..<MazCount - new_MinCount>])。この場合、これらは動的ノードであるため、新しいノードでスケジューラに保留中のジョブがない限り、新しい Amazon EC2 インスタンスは起動されません。

- 初期状態: MinCount = 100; MaxCount = 150
-
```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
queue1*      up   infinite     50  idle~ queue1-dy-c5xlarge-[1-50]
```

```
queue1*      up   infinite    100   idle queue1-st-c5xlarge-[1-100]
```

- 更新 (-30) を適用した MinCount : MinCount = 70 (MaxCount = 120)
-

```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
queue1*      up   infinite     80  idle~ queue1-dy-c5xlarge-[1-80]
queue1*      up   infinite     70   idle queue1-st-c5xlarge-[1-70]
```

MinCount < MaxCount の場合、MinCount と MaxCount を同じ量 N だけ減らすには、最後の N 個の静的ノードを削除するようにクラスターを設定します (<Queue/Name>-st-<ComputeResource/Name>-<old_MinCount - N>...<oldMinCount>])。システムは対応する数の Amazon EC2 インスタンスを終了します。

さらに、動的ノードの数は変更せずに、新しい MaxCount の値を保持します。

- 初期状態: MinCount = 100; MaxCount = 150
-

```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
queue1*      up   infinite     50  idle~ queue1-dy-c5xlarge-[1-50]
queue1*      up   infinite    100   idle queue1-st-c5xlarge-[1-100]
```

- 更新 (-30) を適用した MinCount : MinCount = 70 (MaxCount = 120)
-

```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
queue1*      up   infinite     80  idle~ queue1-dy-c5xlarge-[1-50]
queue1*      up   infinite     70   idle queue1-st-c5xlarge-[1-70]
```

MinCount < MaxCount の場合、MaxCount を量 N だけ減らすには (MinCount は変更しないと仮定)、最後の N 個の動的ノードを削除するようにクラスターを設定します (<Queue/Name>-dy-<ComputeResource/Name>-<old_MaxCount - N...<oldMaxCount>])。システムは対応する数の Amazon EC2 インスタンス (実行している場合) を終了します。静的ノードには影響がないと想定されます。

- 初期状態: MinCount = 100; MaxCount = 150
-
```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
queue1*      up   infinite     50  idle~ queue1-dy-c5xlarge-[1-50]
queue1*      up   infinite    100   idle queue1-st-c5xlarge-[1-100]
```
- 更新 (-30) を適用した MaxCount : MinCount = 100 (MaxCount = 120)
-
```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
queue1*      up   infinite     20  idle~ queue1-dy-c5xlarge-[1-20]
queue1*      up   infinite    100   idle queue1-st-c5xlarge-[1-100]
```

ジョブへの影響

ノードを削除して Amazon EC2 インスタンスを終了すると、削除したノードで実行していた sbatch ジョブは常に再びキューに入れられます。ただし、ジョブ要件を満たす他のノードがない場合を除きます。この場合、ジョブはステータス NODE_FAIL で失敗し、キューから消えるため、手動で再送信する必要があります。

クラスターのサイズ変更の更新を実行する予定がある場合は、予定した更新中に削除対象のノードでジョブが実行されないようにすることができます。これを行うには、ノードをメンテナンス中に削除するように設定します。メンテナンス中にノードを設定しても、ノードで既に実行しているジョブには最終的に影響しないことに注意してください。

予定したクラスターのサイズ変更の更新で、ノード qeueu-st-computeresource-[9-10] を削除するとします。次のコマンドを使用して Slurm 予約を作成できます。

```
sudo -i scontrol create reservation ReservationName=maint_for_update user=root
 starttime=now duration=infinite flags=maint,ignore_jobs nodes=qeueu-st-
computeresource-[9-10]
```

これにより、Slurm 予約が `maint_for_update` という名前でノード `qeueu-st-computeresource-[9-10]` に作成されます。予約を作成した時点から、ノード `qeueu-st-computeresource-[9-10]` では以降のジョブを実行できなくなります。予約では、ノード `qeueu-st-computeresource-[9-10]` へのジョブの割り当てを最終的には阻止できないことに注意してください。

Slurm 予約を設定しておいたノードのみがサイズ変更の更新中に削除された場合、クラスターのサイズ変更の更新後に、メンテナンス予約は自動的に削除されます。一方、Slurm 予約を作成しておいたノードがクラスターのサイズ変更の更新後も存続する場合は、次のコマンドを使用して、サイズ変更の更新の実行後にノードのメンテナンス予約を削除できます。

```
sudo -i scontrol delete ReservationName=maint_for_update
```

Slurm 予約の詳細については、こちらで公式の SchedMD ドキュメントを参照してください。

容量変更時のクラスター更新プロセス

スケジューラ設定を変更すると、クラスターの更新プロセス中に次の手順が実行されます。

- 停止 AWS ParallelCluster `clustermgtd` (`supervisorctl stop clustermgtd`)
- 更新された Slurm パーティション設定を AWS ParallelCluster 設定から生成する
- `slurmctld` を再起動する (Chef サービスレシピを使用して実行)
- `slurmctld` のステータスを確認する (`systemctl is-active --quiet slurmctld.service`)
- Slurm 設定を再ロードする (`scontrol reconfigure`)
- 起動する: `clustermgtd` (`supervisorctl start clustermgtd`)

Slurm の詳細については、「https://slurm.schedmd.com」を参照してください。ダウンロードについては、「https://github.com/SchedMD/slurm/tags」を参照してください。ソースコードについては、「https://github.com/SchedMD/slurm」を参照してください。

サポートされているクラスターと SLURM バージョン

次の表に、 がサポートする AWS AWS ParallelCluster および Slurmのバージョンを示します。

AWS ParallelCluster バージョン (複数可)	サポートされる Slurm のバージョン
3.13.0	24.05.07
3.12.0	23.11.10
3.11.0	23.11.10
3.9.2、3.9.3、3.10.0	23.11.7
3.9.0、3.9.1	23.11.4
3.8.0	23.02.7
3.7.2	23.02.6
3.7.1	23.02.5
3.7.0	23.02.4
3.6.0、3.6.1	23.02.2
3.5.0、3.5.1	22.05.8
3.4.0、3.4.1	22.05.7
3.3.0、3.3.1	22.05.5
3.1.4、3.1.5、3.2.0、3.2.1	21.08.8-2
3.1.2、3.1.3	21.08.6
3.1.1	21.08.5
3.0.0	20.11.8

トピック

- 複数のキューの設定
- マルチキューモードの Slurm ガイド
- Slurm クラスター保護モード

- Slurm クラスタ高速容量不足フェイルオーバー
- Slurm メモリベースのスケジューリング
- Slurm による複数のインスタンスタイプの割り当て
- 動的ノードのクラスタースケーリング
- Slurm による アカウンティング AWS ParallelCluster
- Slurm 設定のカスタマイズ
- Slurmprolog と epilog
- クラスター容量のサイズと更新

複数のキューの設定

AWS ParallelCluster バージョン 3 では、 `Scheduler`を に設定`slurm`し、設定ファイル`SlurmQueues`で に複数のキューを指定することで、複数のキューを設定できます。このモードでは、設定ファイルの `ComputeResources` セクションで指定されているコンピューティングノードに異なるインスタンスタイプが共存します。異なるインスタンスタイプの `ComputeResources` は、`SlurmQueues` の必要に応じてスケールアップまたはスケールダウンされます。

ワークロードが同じ基盤となるインフラストラクチャとリソース (共有ストレージ、ネットワーク、ログインノードなど) を共有する場合、通常、1 つのクラスター内の複数のキューが複数のクラスターよりも優先されます。ワークロードのコンピューティング、ストレージ、ネットワークのニーズが類似している場合、1 つのクラスター内で複数のキューを使用すると、リソース共有が可能になり、不要な重複を回避できるため、効率が向上します。このアプローチは、効率的なジョブスケジューリングとリソース割り当てを可能にしながら、管理を簡素化し、オーバーヘッドを削減します。一方、ワークロード間に強力なセキュリティ、データ、または運用分離要件がある場合は、複数のクラスターを使用する必要があります。たとえば、異なるスケジュール、更新サイクル、またはアクセスポリシーでワークロードを個別に管理および運用する必要がある場合は、複数のクラスターが適しています。

クラスターキューとコンピューティングリソースのクォータ

リソース	クォータ
`Slurm queues`	クラスターあたり 50 キュー
`Compute resources`	1 キューあたり 50 のコンピューティングリソース

リソース	クォータ
	1 クラスターあたり 50 のコンピューティングリソース

ノード数

キュー内の ComputeResources の各コンピューティングリソースには、固有の Name、InstanceType、MinCount、および MaxCount が必要です。MinCount および MaxCount は、キュー内の ComputeResources でコンピューティングリソースのインスタンス範囲を定義するデフォルト値があります。MinCount および MaxCount には独自の値を指定することもできます。ComputeResources 内の各コンピューティングリソースは、1 からMinCount の値までの番号が付けられた静的ノードと、MinCount の値から MaxCount の値までの番号が付けられた動的ノードで構成されます。

サンプルの構成

クラスター設定ファイルの Scheduling セクションの例を次に示します。この設定では、queue1 および queue2 という 2 つのキューがあり、それぞれのキューには MaxCount を指定した ComputeResources があります。

```
Scheduling:
  Scheduler: slurm
  SlurmQueues:
  - Name: queue1
    ComputeResources:
    - InstanceType: c5.xlarge
      MaxCount: 5
      Name: c5xlarge
    - InstanceType: c4.xlarge
      MaxCount: 5
      Name: c4xlarge
  - Name: queue2
    ComputeResources:
    - InstanceType: c5.xlarge
      MaxCount: 5
      Name: c5xlarge
```

HostNames

コンピューティングフリートに起動されるインスタンスは、動的に割り当てられます。各ノードにはホスト名が生成されます。デフォルトでは AWS ParallelCluster 、 はホスト名 の次の形式を使用します。

```
$HOSTNAME=$QUEUE-$STATDYN-$COMPUTE_RESOURCE-$NODENUM
```

- $QUEUE はキューの名前です。例えば、SlurmQueues セクションに Name を「queue-name」に設定したエントリーがある場合、「$QUEUE」は「queue-name」になります。
- $STATDYN は、静的ノードの場合は st、動的ノードの場合は dy です。
- $COMPUTE_RESOURCE は、このノードに対応する ComputeResources コンピューティングリソースの Name です。
- $NODENUM はノードの番号です。$NODENUM は、静的ノードの場合は 1 から MinCount までの値、動的ノードの場合は 1 から MaxCount - MinCount までの値です。

上記の設定ファイルの例では、queue1 のノードと c5xlarge のコンピューティングリソースは、ホスト名が queue1-dy-c5xlarge-1 となります。

ホスト名と完全修飾ドメイン名 (FQDN) の両方は、Amazon Route 53 のホストゾーンを使用して作成されます。FQDN は $HOSTNAME.$CLUSTERNAME.pcluster で、$CLUSTERNAME はクラスターの名前です。

Slurm ノード名にも同じ形式が使用されることに注意してください。

ユーザーは、使用するデフォルトのホスト名形式ではなく、コンピューティングノードを使用するインスタンスのデフォルトの Amazon EC2 ホスト名を使用できます AWS ParallelCluster。これを行うには、UseEc2Hostnames パラメータを true に設定します。ただし、Slurmノード名は引き続きデフォルトの AWS ParallelCluster 形式を使用します。

マルチキューモードの Slurm ガイド

ここでは、キュー (パーティション) ノード AWS ParallelCluster Slurmの管理方法と、キューとノードの状態をモニタリングする方法について説明します。

概要

スケーリングアーキテクチャは、Slurm の「Cloud Scheduling Guide」(クラウドスケジュールガイド) と省電力プラグインに基づいています。省電力プラグインの詳細については、「Slurm Power Saving Guide」を参照してください。アーキテクチャでは、クラスターで利用できる可能性のあるリソースは、通常、クラウドノードとして Slurm 設定にあらかじめ定義されています。

クラウドノードのライフサイクル

クラウドノードはそのライフサイクルを通じて、すべてではないが以下のような状態になります。POWER_SAVING、POWER_UP (pow_up)、ALLOCATED (alloc)、および POWER_DOWN (pow_dn)。場合によっては、クラウドノードが OFFLINE 状態になることがあります。次のリストは、クラウドノードのライフサイクルにおけるこれらの状態のいくつかの側面を詳細に示しています。

- **POWER_SAVING** 状態のノードは、sinfo では ~ サフィックス (例: idle~) で表示されます。この状態では、ノードをバックアップする EC2 インスタンスは存在しません。ただし、Slurm は引き続きノードにジョブを割り当てることができます。
- **POWER_UP** 状態に遷移したノードは、sinfo では # サフィックス (例: idle#) で表示されます。Slurm が POWER_SAVING 状態のノードにジョブを割り当てると、ノードは自動的に POWER_UP 状態に遷移します。

  su ルートユーザーとして次のコマンドを使用し、手動でノードを POWER_UP 状態に遷移させることもできます。

  ```
  $ scontrol update nodename=nodename state=power_up
  ```

  この段階で ResumeProgram が呼び出され、EC2 インスタンスが起動して構成され、ノードが POWER_UP 状態に遷移します。
- 現在使用可能なノードは、sinfo にサフィックス (例: idle) なしで表示されます。ノードのセットアップが完了し、クラスターに参加した後、ジョブの実行が可能になります。この段階で、ノードは適切に設定され、使用可能な状態になります。

  原則として、Amazon EC2 インスタンス数は利用可能なノード数と同じにすることを推奨します。通常、静的ノードはクラスター作成後に利用可能です。
- **POWER_DOWN** 状態に遷移しているノードは、sinfo に % サフィックス (例: idle%) で表示されます。動的ノードは ScaledownIdletime の後、自動的に POWER_DOWN 状態になります。対照的に、ほとんどの場合、静的ノードの電源はオフになりません。ただし、su ルートユーザーとして次のコマンドを使用し、手動でノードを POWER_DOWN 状態に配置できます。

  ```
  $ scontrol update nodename=nodename state=down reason="manual draining"
  ```

  この状態では、ノードに関連するインスタンスは終了し、ノードは ScaledownIdletime 後に使用可能になるように POWER_SAVING 状態に戻されます。

ScaledownIdletime の設定は、Slurm の設定の SuspendTimeout 設定に保存されます。

- オフラインのノードは、sinfo に * サフィックス (例: down*) で表示されます。Slurm コントローラーがノードにコンタクトできない場合、または静的ノードが無効化され、バッキングインスタンスが終了した場合、ノードはオフラインになります。

次の sinfo の例で示されるノードの状態を考えてみます。

```
$ sinfo
  PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
  efa          up   infinite      4  idle~ efa-dy-efacompute1-[1-4]
  efa          up   infinite      1   idle efa-st-efacompute1-1
  gpu          up   infinite      1  idle% gpu-dy-gpucompute1-1
  gpu          up   infinite      9  idle~ gpu-dy-gpucompute1-[2-10]
  ondemand     up   infinite      2   mix# ondemand-dy-ondemandcompute1-[1-2]
  ondemand     up   infinite     18  idle~ ondemand-dy-ondemandcompute1-
[3-10],ondemand-dy-ondemandcompute2-[1-10]
  spot*        up   infinite     13  idle~ spot-dy-spotcompute1-[1-10],spot-dy-
spotcompute2-[1-3]
  spot*        up   infinite      2   idle spot-st-spotcompute2-[1-2]
```

spot-st-spotcompute2-[1-2] ノードと efa-st-efacompute1-1 ノードには、すでにバッキングインスタンスが設定されており、使用可能です。ondemand-dy-ondemandcompute1-[1-2] ノードは POWER_UP 状態であり、数分以内に利用可能になるはずです。gpu-dy-gpucompute1-1 ノードは POWER_DOWN 状態であり、POWER_SAVING (デフォルトは 10 分) 後に ScaledownIdletime 状態へ遷移します。

他のノードはすべて POWER_SAVING 状態で、EC2 インスタンスのバックアップはありません。

使用可能なノードの操作

利用可能なノードは Amazon EC2 インスタンスによってバッキングされます。デフォルトでは、ノード名を使用してインスタンスに直接 SSH 接続することができます (例: ssh efa-st-efacompute1-1)。インスタンスのプライベート IP アドレスは、次のコマンドを使用して取得することができます。

```
$ scontrol show nodes nodename
```

返された NodeAddr フィールドの IP アドレスを確認します。

ノードが利用可能でない場合、NodeAddr フィールドは稼働中の Amazon EC2 インスタンスを指すべきではありません。ノード名と同じにする必要があります。

ジョブの状態および送信

通常、送信されたジョブはすぐにシステム内のノードに割り当てられ、すべてのノードが割り当てられている場合は保留状態になります。

ジョブに割り当てられたノードに POWER_SAVING 状態のノードが含まれる場合、ジョブは、CF または CONFIGURING 状態で開始されます。このとき、ジョブは POWER_SAVING 状態のノードが POWER_UP 状態に遷移し、利用可能になるのを待ちます。

ジョブに割り当てられたすべてのノードが利用可能になると、RUNNING (R) 状態になります。

デフォルトでは、すべてのジョブはデフォルトのキュー (Slurm ではパーティションと呼ばれます) に送信されます。これは、キュー名の後に * サフィックスが付くことで示されます。-p ジョブ送信オプションを使用してキューを選択することができます。

すべてのノードには、ジョブ送信コマンドで使用可能な次の機能が設定されています。

- インスタンスタイプ (例: c5.xlarge)
- ノードタイプ (dynamic または static のいずれか)。

次のコマンドを使用して、特定のノードの機能を確認することができます。

```
$ scontrol show nodes nodename
```

返された AvailableFeatures リストを確認します。

クラスターの初期状態を考えてみましょう。sinfo コマンドを実行することで確認できます。

```
$ sinfo
  PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
  efa          up   infinite      4  idle~ efa-dy-efacompute1-[1-4]
  efa          up   infinite      1   idle efa-st-efacompute1-1
  gpu          up   infinite     10  idle~ gpu-dy-gpucompute1-[1-10]
  ondemand     up   infinite     20  idle~ ondemand-dy-ondemandcompute1-
[1-10],ondemand-dy-ondemandcompute2-[1-10]
  spot*        up   infinite     13  idle~ spot-dy-spotcompute1-[1-10],spot-dy-
spotcompute2-[1-3]
  spot*        up   infinite      2   idle spot-st-spotcompute2-[1-2]
```

なお、spot はデフォルトのキューです。* というサフィックスで表示されます。

デフォルトキュー (spot) の 1 つの静的ノードにジョブを送信します。

```
$ sbatch --wrap "sleep 300" -N 1 -C static
```

EFA キューのある動的ノードにジョブを送信します。

```
$ sbatch --wrap "sleep 300" -p efa -C dynamic
```

ondemand キューの c5.2xlarge ノード 8 台、t2.xlarge ノード 2 台にジョブを送信します。

```
$ sbatch --wrap "sleep 300" -p ondemand -N 10 -C "[c5.2xlarge*8&t2.xlarge*2]"
```

gpu キューのある GPU ノードにジョブを送信します。

```
$ sbatch --wrap "sleep 300" -p gpu -G 1
```

squeue コマンドを使ったジョブの状態を考えてみます。

```
$ squeue
 JOBID PARTITION     NAME   USER   ST       TIME  NODES NODELIST(REASON)
  12   ondemand      wrap   ubuntu CF       0:36     10 ondemand-dy-ondemandcompute1-
[1-8],ondemand-dy-ondemandcompute2-[1-2]
  13        gpu      wrap   ubuntu CF       0:05      1 gpu-dy-gpucompute1-1
   7       spot      wrap   ubuntu  R       2:48      1 spot-st-spotcompute2-1
   8        efa      wrap   ubuntu  R       0:39      1 efa-dy-efacompute1-1
```

ジョブ 7 と 8 (spot と efa のキュー) はすでに実行中です (R)。ジョブ 12 と 13 はまだ設定中 (CF) で、インスタンスが利用可能になるのを待っている可能性があります。

```
# Nodes states corresponds to state of running jobs
$ sinfo
 PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
 efa          up   infinite      3  idle~ efa-dy-efacompute1-[2-4]
 efa          up   infinite      1    mix efa-dy-efacompute1-1
 efa          up   infinite      1   idle efa-st-efacompute1-1
 gpu          up   infinite      1   mix~ gpu-dy-gpucompute1-1
 gpu          up   infinite      9  idle~ gpu-dy-gpucompute1-[2-10]
 ondemand     up   infinite     10   mix# ondemand-dy-ondemandcompute1-[1-8],ondemand-
dy-ondemandcompute2-[1-2]
```

```
 ondemand     up   infinite     10  idle~ ondemand-dy-ondemandcompute1-[9-10],ondemand-
dy-ondemandcompute2-[3-10]
 spot*        up   infinite     13  idle~ spot-dy-spotcompute1-[1-10],spot-dy-
spotcompute2-[1-3]
 spot*        up   infinite      1    mix spot-st-spotcompute2-1
 spot*        up   infinite      1   idle spot-st-spotcompute2-2
```

ノードの状態および機能

ほとんどの場合、ノードの状態は、このトピックで前述したクラウドノードライフサイクルの特定のプロセス AWS ParallelCluster に従って によって完全に管理されます。

ただし、 は、 DOWNおよび DRAINEDの状態の異常なノードと、異常なバッキングインスタンスを持つノード AWS ParallelCluster も置き換えるか終了します。詳細については、「clustermgtd」を参照してください。

パーティションの状態

AWS ParallelCluster では、次のパーティション状態がサポートされています。Slurm パーティションは AWS ParallelClusterのキューです。

- UP: パーティションがアクティブ状態であることを示します。これは、パーティションのデフォルトの状態です。この状態では、パーティション内のすべてのノードがアクティブで、使用可能な状態です。
- INACTIVE: パーティションが非アクティブ状態であることを示す。この状態では、非アクティブなパーティションのノードをバックアップしているインスタンスはすべて終了します。非アクティブなパーティションのノードには、新しいインスタンスは起動しません。

pcluster update-compute-fleet

- コンピューティングフリートの停止 - 次のコマンドを実行すると、すべてのパーティションは INACTIVE状態に移行し、 AWS ParallelCluster プロセスはパーティションを INACTIVE状態に保ちます。

```
$ pcluster update-compute-fleet --cluster-name testSlurm \
   --region eu-west-1 --status STOP_REQUESTED
```

- コンピューティングフリートの起動 - 次のコマンドを実行すると、すべてのパーティションが最初に UP の状態に遷移します。ただし、 AWS ParallelCluster プロセスはパーティションを UP状態に

保持しません。パーティションの状態を手動で変更する必要があります。数分後、すべての静的ノードが使用可能になります。パーティションを UP に設定しても、動的容量は向上しません。

```
$ pcluster update-compute-fleet --cluster-name testSlurm \
   --region eu-west-1 --status START_REQUESTED
```

update-compute-fleet が動作している場合、pcluster describe-compute-fleet コマンドを実行して Status を確認することで、クラスターの状態を確認することができます。考えられる状態を以下に示します。

- STOP_REQUESTED: コンピューティングフリートの停止リクエストがクラスターに送信されます。
- STOPPING: pcluster プロセスは現在、コンピューティングフリートを停止しています。
- STOPPED: pcluster プロセスは停止処理を終了し、すべてのパーティションは INACTIVE 状態になり、すべてのコンピューティングインスタンスは終了しました。
- START_REQUESTED: コンピューティングフリートの開始リクエストがクラスターに送信されます。
- STARTING: pcluster プロセスは現在、クラスターを起動中です。
- RUNNING: pcluster プロセスは起動処理を終了し、すべてのパーティションが UP 状態になり、数分後に静的ノードが利用可能になります。
- PROTECTED：このステータスは、一部のパーティションでブートストラップが失敗し続けていることを示しています。影響を受けたパーティションは非アクティブになります。問題を調査し、update-compute-fleet を実行して、フリートを再度有効化してください。

キューの手動制御

場合によっては、クラスター内のノードやキュー (Slurm ではパーティションと呼ばれます) を手動で制御したいことがあります。次の共通の手順で scontrol コマンドを使用してクラスター内のノードを管理することができます。

- **POWER_SAVING** 状態の動的ノードの電源を入れる

  su ルートユーザーとしてコマンドを実行します。

  ```
  $ scontrol update nodename=nodename state=power_up
  ```

特定の数のノードをリクエストするプレースホルダー `sleep 1` のジョブを送信し、Slurm に依存して必要な数のノードの電源を入れることもできます。

- **`ScaledownIdletime`** の前に動的ノードの電源を切る

  su ルートユーザーとしてコマンドを使用して、動的ノードを DOWN に設定することをお勧めします。

  ```
  $ scontrol update nodename=nodename state=down reason="manually draining"
  ```

  AWS ParallelCluster は、ダウンした動的ノードを自動的に終了およびリセットします。

  一般的に、`scontrol update nodename=nodename state=power_down` コマンドを使用してノードを直接 POWER_DOWN に設定することはお勧めしません。これは、 AWS ParallelCluster が自動的にパワーダウン処理を行うためです。

- キュー (パーティション) を無効にするか、特定のパーティションのすべての静的ノードを停止する

  su ルートユーザーとしてコマンドを使用して、特定のキューを INACTIVE に設定します。

  ```
  $ scontrol update partition=queuename state=inactive
  ```

  この操作により、パーティション内のすべてのインスタンスバッキングノードが終了します。

- キュー (パーティション) を有効にする

  su ルートユーザーとしてコマンドを使用して、特定のキューを UP に設定します。

  ```
  $ scontrol update partition=queuename state=up
  ```

スケーリング動作および調整

ここでは、通常のスケーリングワークフローの例を示します。

- スケジューラは、2 つのノードを必要とするジョブを受け取ります。
- スケジューラは 2 つのノードを POWER_UP 状態に遷移させ、ノード名 (例: `queue1-dy-spotcompute1-[1-2]`) で `ResumeProgram` を呼び出す。

- ResumeProgram は Amazon EC2 インスタンスを 2 つ起動し、queue1-dy-spotcompute1-[1-2] のプライベート IP アドレスとホストネームを割り当て、ResumeTimeout (デフォルトの期間は 30 分) を待ってからノードをリセットします。
- インスタンスが設定され、クラスターに参加します。インスタンスでジョブの実行を開始します。
- ジョブは完了し、実行は停止します。
- 設定された SuspendTime が経過した後 (ScaledownIdletime に設定されています)、スケジューラはインスタンスを POWER_SAVING 状態に設定します。次に、スケジューラは queue1-dy-spotcompute1-[1-2] を POWER_DOWN 状態に設定し、ノード名で SuspendProgram を呼び出します。
- SuspendProgram は 2 つのノードに対して呼び出されます。ノードは、例えば SuspendTimeout のために idle% を維持することで、POWER_DOWN 状態を維持します (デフォルトの期間は 120 秒 (2 分))。clustermgtd は、ノードがパワーダウンしていることを検出した後、バッキングインスタンスを終了します。次に、queue1-dy-spotcompute1-[1-2] はアイドル状態に遷移し、プライベート IP アドレスとホスト名はリセットされ、今後のジョブに備えて電源を入れる準備をします。

処理がうまくいかず、何らかの理由で特定のノードのインスタンスが起動できない場合、次のようなことが起こります。

- スケジューラが、2 つのノードを必要とするジョブを受け取る。
- スケジューラが 2 つのクラウドでのバーストノードを POWER_UP 状態に遷移させ、ノード名 (例: queue1-dy-spotcompute1-[1-2]) で ResumeProgram を呼び出す。
- ResumeProgram は 1 つの Amazon EC2 インスタンスだけを起動して queue1-dy-spotcompute1-1 を設定し、別のインスタンス queue1-dy-spotcompute1-2 は起動に失敗する。
- queue1-dy-spotcompute1-1 は影響を受けず、POWER_UP 状態に到達した後、オンラインになる。
- queue1-dy-spotcompute1-2 が POWER_DOWN 状態に遷移し、Slurm がノード障害を検出するため、自動的にジョブが再キューされる。
- queue1-dy-spotcompute1-2 が SuspendTimeout の後に利用可能になる (デフォルトは 120 秒 (2 分))。その間、ジョブは再キューされ、別のノードで実行を開始することができます。
- 上記のプロセスが、使用可能なノードで障害が発生せずにジョブを実行できるようになるまで繰り返される。

必要に応じて調整可能な 2 つのタイミングパラメータがあります。

- **`ResumeTimeout`** (デフォルトは 30 分): `ResumeTimeout` は、ノードがダウン状態に遷移するまで Slurm が待機する時間を制御します。
  - インストール前後の処理に時間がかかる場合は、`ResumeTimeout` を拡張するのも有効です。
  - `ResumeTimeout` は、また、問題が発生した場合にノードを交換またはリセットするまでに、AWS ParallelCluster が待機する最大時間です。起動またはセットアップ中にエラーが発生した場合、コンピューティングノードは自己終了します。 AWS ParallelCluster プロセスは、終了したインスタンスの検出時にノードを置き換えます。
- **`SuspendTimeout`** (デフォルトは 120 秒 (2 分)): `SuspendTimeout` は、ノードがシステムに戻され、再び使用できるようになるまでの時間を制御します。
  - `SuspendTimeout` が短いと、ノードのリセットが早くなり、Slurm はより頻繁にインスタンスの起動を試みることができます。
  - `SuspendTimeout` が長いと、故障したノードのリセットが遅くなります。その間、Slurm は他のノードの使用を試みます。`SuspendTimeout` が数分以上であれば、Slurm はシステム内の全ノードの循環を試みます。1,000 ノード以上の大規模システムでは、失敗したジョブを頻繁に再キューする場合に Slurm のストレスを軽減するために、より長い `SuspendTimeout` が有効である場合があります。
  - `SuspendTimeout` は、 がノードのバッキングインスタンスの終了を AWS ParallelCluster 待機する時間を参照しないことに注意してください。`POWER_DOWN` ノードのバックアップインスタンスは即座に終了します。通常、終了プロセスは数分で完了します。ただし、この間、ノードは `POWER_DOWN` 状態にあり、スケジューラで利用することはできません。

アーキテクチャのログ

次のリストには、キーのログが含まれています。Amazon CloudWatch Logs で使用されるログストリーム名は、*`{hostname}`*`.`*`{instance_id}`*`.`*`{logIdentifier}`* という形式になっており、*`logIdentifier`* がログ名の後に続きます。

- `ResumeProgram`: `/var/log/parallelcluster/slurm_resume.log` (`slurm_resume`)
- `SuspendProgram`: `/var/log/parallelcluster/slurm_suspend.log` (`slurm_suspend`)
- `clustermgtd`: `/var/log/parallelcluster/clustermgtd.log` (`clustermgtd`)
- `computemgtd`: `/var/log/parallelcluster/computemgtd.log` (`computemgtd`)
- `slurmctld`: `/var/log/slurmctld.log` (`slurmctld`)
- `slurmd`: `/var/log/slurmd.log` (`slurmd`)

よくある問題とデバッグの方法:

起動、パワーアップ、またはクラスターへの参加に失敗したノード

- 動的ノード:
  - `ResumeProgram` ログを確認し、ノードで `ResumeProgram` が呼び出されたかどうかを確認します。そうでない場合は、`slurmctld` ログを確認し、Slurm がノードで `ResumeProgram` を呼び出したかどうかを確認します。`ResumeProgram` のパーミッションが正しくない場合、サイレントで失敗することがあります。
  - `ResumeProgram` が呼び出された場合、そのノードに対してインスタンスが起動されたかどうかを確認します。インスタンスが起動しなかった場合は、インスタンスの起動に失敗した理由を示す明確なエラーメッセージが表示されます。
  - インスタンスが起動した場合、ブートストラッププロセスの実行時に何らかの問題が発生した可能性があります。`ResumeProgram` ログから対応するプライベート IP アドレスとインスタンス ID を探し、CloudWatch Logs で特定のインスタンスの対応するブートストラップのログを見ます。
- 静的ノード:
  - `clustermgtd` ログをチェックして、ノードに対してインスタンスが起動されたかどうかを確認します。インスタンスが起動しなかった場合は、インスタンスの起動に失敗した原因について、明確なエラーが表示されるはずです。
  - インスタンスを起動していた場合、ブートストラップ処理で何らかの問題が発生しています。`clustermgtd` ログから対応するプライベート IP とインスタンス ID を探し、CloudWatch Logs で特定のインスタンスの対応するブートストラップのログを見ます。

ノードが予期せず置換または終了し、ノードに障害が生じる

- ノードが予期せず置換または終了した。
  - 通常、`clustermgtd` はすべてのノードのメンテナンスアクションを処理します。`clustermgtd` がノードを置換または終了させたかどうかを確認するには、`clustermgtd` のログを確認します。
  - `clustermgtd` がノードを置換または終了させた場合、その理由を示すメッセージが表示されます。スケジューラ関連 (ノードが `DOWN` だった場合など) の場合は、`slurmctld` ログで詳細を確認します。理由が Amazon EC2 関連の場合は、Amazon CloudWatch コンソール、Amazon EC2 コンソール、CLI、SDK などのツールを使用して、そのインスタンスのステータスまたはログを確認します。例えば、インスタンスにスケジュールされたイベントがあったかどうか、Amazon EC2 ヘルスステータスのチェックに失敗したかどうかを確認できます。

  - clustermgtd がノードを終了させなかった場合、computemgtd がノードを終了させたか、EC2 がスポットインスタンスを再要求するためにインスタンスを終了させたかどうかを確認します。
- ノードの障害。
  - 通常、ノードに障害が発生した場合、ジョブは自動的に再キューされます。slurmctld ログを見て、ジョブやノードがなぜ失敗したかを確認し、そこから状況を評価します。

インスタンスの置換やターミネーション時の障害、ノードのパワーダウン時の障害

- 一般に、clustermgtd は期待されるすべてのインスタンス終了アクションを処理します。clustermgtd ログを見て、ノードの置換または終了に失敗した理由を確認する。
- ScaledownIdletime に失敗した動的ノードの場合、SuspendProgram のログから、slurmctld プロセスが特定のノードを引数にした呼び出しを行ったかどうかを確認します。SuspendProgram は実際に特定のアクションを実行するわけではありません。呼び出されたときだけログに記録されます。すべてのインスタンスの終了と NodeAddr リセットは clustermgtd により完了します。Slurm は SuspendTimeout の後、ノードを IDLE に遷移させます。

その他の問題:

- AWS ParallelCluster はジョブの割り当てやスケーリングの決定を行いません。Slurm の指示に従い、リソースを起動、終了、維持を試みるだけです。

  ジョブの割り当て、ノードの割り当て、スケーリングの決定に関する問題については、slurmctld ログを見てエラーを確認します。

Slurm クラスター保護モード

保護モードを有効にしてクラスターを実行すると、 はコンピューティングノードの起動時にコンピューティングノードのブートストラップ障害 AWS ParallelCluster を監視および追跡します。これは、これらの障害が継続的に発生しているかどうかを検出するために行われます。

キュー (パーティション) で以下が検出されると、クラスターは保護ステータスになります。

1. コンピューティングノードが正常に起動せず、コンピューティングノードのブートストラップ障害が連続して発生する。

2. 障害カウントが事前に定義されたしきい値に達した。

クラスターが保護ステータスになると、 は、事前定義されたしきい値以上で障害が発生したキューを AWS ParallelCluster 無効にします。

Slurm クラスター保護モードが AWS ParallelCluster バージョン 3.0.0 に追加されました。

保護モードを使用すると、コンピューティングノードのブートストラップ障害サイクルに費やす時間とリソースを削減できます。

保護モードパラメータ

protected_failure_count

`protected_failure_count` は、クラスターの保護ステータスを有効にするキュー (パーティション) の連続した障害の数を指定します。

デフォルトの `protected_failure_count` は 10 で、保護モードは有効になっています。

`protected_failure_count` が 0 より大きい場合、保護モードは有効になります。

`protected_failure_count` が 0 以下の場合、保護モードは無効になります。

`protected_failure_count` の値は、HeadNode の `/etc/parallelcluster/slurm_plugin/parallelcluster_clustermgtd.conf` にある `clustermgtd` 設定ファイルにパラメータを追加することで変更できます。

このパラメータはいつでも更新でき、そのためにコンピューティングフリートを停止する必要はありません。障害カウントが `protected_failure_count` に達する前にキューで起動が成功すると、障害カウントはゼロにリセットされます。

保護ステータスでのクラスターステータスの確認

クラスターが保護ステータスの場合、コンピューティングフリートのステータスとノードのステータスを確認できます。

コンピューティングフリートのステータス

保護ステータスで動作しているクラスターでは、コンピューティングフリートのステータスは `PROTECTED` です。

```
$ pcluster describe-compute-fleet --cluster-name <cluster-name> --region <region-id>
{
```

```
    "status": "PROTECTED",
    "lastStatusUpdatedTime": "2022-04-22T00:31:24.000Z"
}
```

ノードのステータス

ブートストラップ障害が発生し、保護ステータスをアクティブにしたキュー (パーティション) を調べるには、クラスターにログインして sinfo コマンドを実行します。protected_failure_count 以上のブートストラップ障害があるパーティションは、INACTIVE 状態です。protected_failure_count 以上のブートストラップ障害が発生していないパーティションは、UP 状態にあり、想定どおりに動作します。

PROTECTED ステータスは実行中のジョブには影響しません。protected_failure_count 以上のブートストラップ障害があるパーティションでジョブが実行されている場合、パーティションは実行中のジョブの完了後に INACTIVE に設定されます。

次の例で示されるノードの状態を考慮します。

```
$ sinfo
PARTITION AVAIL TIMELIMIT NODES STATE NODELIST
queue1* inact infinite 10 down% queue1-dy-c5xlarge-[1-10]
queue1* inact infinite 3490 idle~ queue1-dy-c5xlarge-[11-3500]
queue2 up infinite 10 idle~ queue2-dy-c5xlarge-[1-10]
```

コンピューティングノードのブートストラップ障害が 10 回連続で検出されたため、パーティション queue1 は INACTIVE になりました。

ノード queue1-dy-c5xlarge-[1-10] の背後にあるインスタンスは起動しましたが、異常状態のためクラスターに参加できませんでした。

クラスターは保護ステータスです。

パーティション queue2 は、queue1 のブートストラップ障害の影響を受けていません。UP 状態であり、まだジョブを実行することができます。

保護ステータスを非アクティブ化する方法

ブートストラップエラーが解決されたら、以下のコマンドを実行してクラスターを保護ステータスから解除できます。

```
$ pcluster update-compute-fleet --cluster-name <cluster-name> \
  --region <region-id> \
```

```
  --status START_REQUESTED
```

保護ステータスをアクティブ化するブートストラップ障害

保護ステータスをアクティブ化するブートストラップエラーは、次の 3 つのタイプに分割されます。タイプと問題を特定するには、ログが AWS ParallelCluster 生成されたかどうかを確認します。ログが生成された場合は、そのログでエラーの詳細を確認できます。詳細については、「ログの取得と保存」を参照してください。

1. インスタンスが自己終了する原因となるブートストラップエラー

   インスタンスが `SlurmQueues`/`CustomActions`/`OnNodeStart`/`OnNodeConfigured` スクリプトのエラーが原因で自己終了するなど、インスタンスはブートストラッププロセスの早い段階で失敗します。

   動的ノードの場合は、次のようなエラーを探します。

   ```
   Node bootstrap error: Node ... is in power up state without valid backing instance
   ```

   静的ノードの場合は、`clustermgtd` ログ (`/var/log/parallelcluster/clustermgtd`) に次のようなエラーがないか確認します。

   ```
   Node bootstrap error: Node ... is in power up state without valid backing instance
   ```

2. ノード **`resume_timeout`** または **`node_replacement_timeout`** が期限切れになります。

   インスタンスは `resume_timeout` (動的ノードの場合) または `node_replacement_timeout` (静的ノードの場合) 内のクラスターに参加できません。タイムアウト前に自己終了することはありません。例えば、クラスターに対してネットワークが正しく設定されておらず、タイムアウトが期限切れになった後にノードが Slurm によって `DOWN` 状態に設定されます。

   動的ノードの場合は、次のようなエラーを探します。

   ```
   Node bootstrap error: Resume timeout expires for node
   ```

   静的ノードの場合は、`clustermgtd` ログ (`/var/log/parallelcluster/clustermgtd`) に次のようなエラーがないか確認します。

   ```
   Node bootstrap error: Replacement timeout expires for node ... in replacement.
   ```

3. ノードはヘルスチェックを失敗します。

   ノードの背後にあるインスタンスが Amazon EC2 ヘルスチェックまたはスケジュールされたイベントのヘルスチェックに失敗し、ノードはブートストラップ障害ノードとして扱われます。この場合、インスタンスは制御できない理由で終了します AWS ParallelCluster。

   `clustermgtd` ログ (`/var/log/parallelcluster/clustermgtd`) に次のようなエラーがないか確認します。

   ```
   Node bootstrap error: Node %s failed during bootstrap when performing health check.
   ```

4. コンピューティングノードは Slurm の登録に失敗します。

   `slurmd` デーモンと Slurm コントロールデーモン (`slurmctld`) の登録が失敗し、コンピューティングノードの状態が `INVALID_REG` 状態に変更します。`CustomSlurmSettings` コンピューティングノードの仕様エラーで設定されたコンピューティングノードなど、間違えて設定された Slurm コンピューティングノードによって、このエラーが発生する可能性があります。

   ヘッドノードの `slurmctld` ログファイル (`/var/log/slurmctld.log`)、または障害が発生したコンピューティングノードの `slurmd` ログファイル (`/var/log/slurmd.log`) に次のようなエラーがないか確認します。

   ```
   Setting node %s to INVAL with reason: ...
   ```

保護モードをデバッグする方法

クラスターが保護ステータスにあり、 から AWS ParallelCluster 生成された`clustermgtd`ログ`HeadNode`と問題のあるコンピューティングノードから`cloud-init-output`生成されたログがある場合は、ログでエラーの詳細を確認できます。ログの取得方法の詳細については、「ログの取得と保存」を参照してください。

ヘッドノードの **`clustermgtd`** ログ (**`/var/log/parallelcluster/clustermgtd`**)

ログメッセージには、ブートストラップ障害が発生したパーティションと、対応するブートストラップ障害カウントが表示されます。

```
[slurm_plugin.clustermgtd:_handle_protected_mode_process] - INFO - Partitions
bootstrap failure count: {'queue1': 2}, cluster will be set into protected mode if
 protected failure count reach threshold.
```

clustermgtd ログで、Found the following bootstrap failure nodes を検索して、ブートストラップに失敗したノードを見つけます。

```
[slurm_plugin.clustermgtd:_handle_protected_mode_process] - WARNING -
Found the following bootstrap failure nodes: (x2)  ['queue1-st-
c5large-1(192.168.110.155)',  'broken-st-c5large-2(192.168.65.215)']
```

clustermgtd ログで、Node bootstrap error を検索して障害の原因を見つけます。

```
[slurm_plugin.clustermgtd:_is_node_bootstrap_failure] - WARNING - Node bootstrap
 error:
Node broken-st-c5large-2(192.168.65.215) is currently in  replacement and no backing
 instance
```

コンピューティングノードの **cloud-init-output** ログ (**/var/log/cloud-init-output.log**)

clustermgtd ログでブートストラップ障害ノードのプライベート IP アドレスを取得したら、コンピューティングノードにログインするか、ログの取得と保存 のガイダンスに従ってログを取得することで、対応するコンピューティングノードのログを確認することができます。ほとんどの場合、問題のあるノードの /var/log/cloud-init-output ログには、コンピューティングノードのブートストラップ障害の原因となった手順が示されています。

Slurm クラスタ高速容量不足フェイルオーバー

AWS ParallelCluster バージョン 3.2.0 以降、クラスターは、デフォルトで高速容量不足フェイルオーバーモードが有効になっていて実行されます。これにより、Amazon EC2 の容量不足エラーが検出されたときに、ジョブのキューイングの再試行時間を最小限に抑えることができます。これは、異なるインスタンスタイプを使用する複数のコンピューティングリソースでキューを設定する場合に特に効果的です。

Amazon EC2 が検出する容量不足エラー：

- InsufficientInstanceCapacity
- InsufficientHostCapacity
- InsufficientReservedInstanceCapacity
- MaxSpotInstanceCountExceeded
- SpotMaxPriceTooLow: 設定したスポットリクエスト料金が、スポットリクエストに最低限必要な料金を下回る場合、有効化します。

- Unsupported: 特定の でサポートされていないインスタンスタイプを使用してアクティブ化されます AWS リージョン。

高速容量不足フェイルオーバーモードでは、ジョブが / SlurmQueues に割り当てられたときに容量不足エラーが検出されるとcompute resource、 は次の AWS ParallelCluster 操作を行います。

1. コンピューティングリソースを、あらかじめ定義された期間、無効 (DOWN) 状態に設定します。
2. POWER_DOWN_FORCE を使用し、コンピューティングリソースで障害が発生したノードジョブをキャンセルし、障害が発生したノードを一時停止します。障害が発生したノードを IDLE および POWER_DOWN (!) の状態に設定し、次いで POWERING_DOWN (%) に設定します。
3. ジョブを別のコンピューティングリソースに再キューします。

無効化されたコンピューティングリソースの静的なノードや電源が入っているノードには影響しません。ジョブはこれらのノードで完了できます。

このサイクルは、ジョブが 1 つまたは複数のコンピューティングリソースノードに正常に割り当てられるまで繰り返されます。ノードの状態についての詳細は、「マルチキューモードの Slurm ガイド」を参照してください。

ジョブを実行するコンピューティングリソースが見つからない場合、ジョブはあらかじめ定義された期間が経過するまで PENDING の状態に設定されます。この場合、次のセクションで説明するように、定義済みの期間を変更できます。

容量不足のタイムアウトパラメータ

insufficient_capacity_timeout

insufficient_capacity_timeout では、容量不足エラーが検出されたときに、コンピューティングリソースが無効 (down) 状態に保たれる時間 (秒) が指定されます。

デフォルトでは、insufficient_capacity_timeout は有効です。

デフォルトは 600 秒 (10 分) です。

insufficient_capacity_timeout の値が 0 以下の場合、高速容量不足フェイルオーバーモードは無効になります。

insufficient_capacity_timeout の値は、HeadNode の /etc/parallelcluster/slurm_plugin/parallelcluster_clustermgtd.confにある clustermgtd 設定ファイルにパラメータを追加することで変更できます。

このパラメータは、コンピューティングフリートを停止せずにいつでも更新できます。

例:

- `insufficient_capacity_timeout=600`:

  容量不足エラーが検出されると、コンピューティングリソースは無効 (DOWN) に設定されます。10 分後、障害が発生したノードは idle~ (POWER_SAVING) 状態に設定されます。
- `insufficient_capacity_timeout=60`:

  容量不足エラーが検出されると、コンピューティングリソースは無効になります (DOWN)。1 分後、障害が発生したノードは idle~ の状態に設定されます。
- `insufficient_capacity_timeout=0`:

  高速容量不足フェイルオーバーモードは無効になります。コンピューティングリソースは無効になりません。

> **Note**
>
> 容量不足エラーでノードに障害が発生してから、クラスター管理デーモンがノード障害を検出するまでの間には、最大 1 分の遅延が発生する可能性があります。これは、クラスター管理デーモンがノードの容量不足の障害をチェックし、コンピューティングリソースを 1 分間隔で down状態に設定するためです。

高速容量不足フェイルオーバーモードのステータス

クラスターが高速容量不足フェイルオーバーモードの場合、その状態とノードの状態を確認できます。

ノードの状態

コンピューティングリソースのダイナミックノードにジョブが投入され、容量不足エラーが検出されると、ノードは down# の状態になり、理由が提供されます。

```
(Code:InsufficientInstanceCapacity)Failure when resuming nodes.
```

その後、電源がオフになっているノード (idle~ の状態にあるノード) down~ に設定され、理由が提供されます。

```
(Code:InsufficientInstanceCapacity)Temporarily disabling node due to insufficient
 capacity.
```

ジョブはキュー内の他のコンピューティングリソースに再キューされます。

コンピューティングリソースの静的ノードと、UP であるノードは、高速容量不足フェイルオーバーモードの影響を受けません。

ここで、次の例で示されるノードの状態を考えてみます。

```
$ sinfo
PARTITION AVAIL TIMELIMIT NODES STATE NODELIST
queue1*   up    infinite    30  idle~ queue1-dy-c-1-[1-15],queue1-dy-c-2-[1-15]
queue2    up    infinite    30  idle~ queue2-dy-c-1-[1-15],queue2-dy-c-2-[1-15]
```

1 つのノードを必要とするジョブを queue1 に送信します。

```
$ sinfo
PARTITION AVAIL TIMELIMIT NODES STATE NODELIST
queue1*   up   infinite  1   down# queue1-dy-c-1-1
queue1*   up   infinite  15  idle~ queue1-dy-c-2-[1-15]
queue1*   up   infinite  14  down~ queue1-dy-c-1-[2-15]
queue2    up   infinite  30  idle~ queue2-dy-c-1-[1-15],queue2-dy-c-2-[1-15]
```

ノード queue1-dy-c-1-1 を起動してジョブを実行します。しかし、容量不足エラーが発生し、インスタンスを起動できませんでした。ノード queue1-dy-c-1-1 は down に設定されます。コンピューティングリソース (queue2-dy-c-1) 内の電源がオフになっている動的ノードは down に設定されます。

ノードの原因は `scontrol show nodes` で確認できます。

```
$ scontrol show nodes queue1-dy-c-1-1
NodeName=broken-dy-c-2-1 Arch=x86_64 CoresPerSocket=1
CPUAlloc=0 CPUTot=96 CPULoad=0.00
...
ExtSensorsJoules=n/s ExtSensorsWatts=0 ExtSensorsTemp=n/s
Reason=(Code:InsufficientInstanceCapacity)Failure when resuming nodes
 [root@2022-03-10T22:17:50]

$ scontrol show nodes queue1-dy-c-1-2
NodeName=broken-dy-c-2-1 Arch=x86_64 CoresPerSocket=1
CPUAlloc=0 CPUTot=96 CPULoad=0.00
```

```
...
ExtSensorsJoules=n/s ExtSensorsWatts=0 ExtSensorsTemp=n/s
Reason=(Code:InsufficientInstanceCapacity)Temporarily disabling node due to
 insufficient capacity [root@2022-03-10T22:17:50]
```

ジョブはキューコンピューティングリソース内の別のインスタンスタイプにキューイングされます。

`insufficient_capacity_timeout` の時間が経過すると、コンピューティングリソース内のノードは `idle~` の状態にリセットされます。

```
$ sinfo
PARTITION AVAIL TIMELIMIT NODES STATE NODELIST
queue1*   up    infinite    30  idle~ queue1-dy-c-1-[1-15],queue1-dy-c-2-[1-15]
queue2    up    infinite    30  idle~ queue2-dy-c-1-[1-15],queue2-dy-c-2-[1-15]
```

`insufficient_capacity_timeout` の時間が経過し、コンピューティングリソース内のノードが `idle~` の状態にリセットされると、Slurm スケジューラはノードの優先度を低くします。スケジューラは、以下のいずれかが起こらない限り、重みの大きい他のキューコンピューティングリソースからノードを選択し続けます。

- ジョブの送信要件は、回復されたコンピューティングリソースと一致します。
- 他のコンピューティングリソースはキャパシティに達しているため、使用できません。
- `slurmctld` が再起動されます。
- AWS ParallelCluster コンピューティングフリートが停止し、すべてのノードの電源をオフにして電源をオンにします。

関連ログ

容量不足エラーと高速容量不足フェイルオーバーモードに関連するログは、ヘッドノードの Slurm の `resume` ログと `clustermgtd` ログにあります。

Slurm **`resume`** (**`/var/log/parallelcluster/slurm_resume.log`**)

容量が不足しているためにノードを起動できない場合のエラーメッセージ。

```
[slurm_plugin.instance_manager:_launch_ec2_instances] - ERROR - Failed RunInstances
 request: dcd0c252-90d4-44a7-9c79-ef740f7ecd87
[slurm_plugin.instance_manager:add_instances_for_nodes] - ERROR - Encountered
 exception when launching instances for nodes (x1) ['queue1-dy-c-1-1']: An error
 occurred
```

```
(InsufficientInstanceCapacity) when calling the RunInstances operation (reached max
 retries: 1): We currently do not have sufficient p4d.24xlarge capacity in the
Availability Zone you requested (us-west-2b). Our system will be working on
 provisioning additional capacity. You can currently get p4d.24xlarge capacity by
 not
specifying an Availability Zone in your request or choosing us-west-2a, us-west-2c.
```

Slurm **clustermgtd** (**/var/log/parallelcluster/clustermgtd**)

Queue1 のコンピューティングリソース c-1 は、容量が不足しているため無効になっています。

```
[slurm_plugin.clustermgtd:_reset_timeout_expired_compute_resources] - INFO - The
 following compute resources are in down state
due to insufficient capacity: {'queue1': {'c-1':
 ComputeResourceFailureEvent(timestamp=datetime.datetime(2022, 4, 14, 23, 0, 4,
 769380, tzinfo=datetime.timezone.utc),
error_code='InsufficientInstanceCapacity')}}, compute resources are reset after
 insufficient capacity timeout (600 seconds) expired
```

容量不足タイムアウトの期限が切れると、コンピューティングリソースはリセットされ、コンピューティングリソース内のノードは `idle~` に設定されます。

```
[root:_reset_insufficient_capacity_timeout_expired_nodes] - INFO - Reset the
 following compute resources because insufficient capacity
timeout expired: {'queue1': ['c-1']}
```

Slurm メモリベースのスケジューリング

バージョン 3.2.0 以降、 は / `SlurmSettings` `EnableMemoryBasedScheduling`クラスター設定パラメータを使用したSlurmメモリベースのスケジューリング AWS ParallelCluster をサポートしています。

> **Note**
>
> AWS ParallelCluster バージョン 3.7.0 以降では、インスタンスで複数のインスタンスタイプを設定すると、 を有効に`EnableMemoryBasedScheduling`できます。
> AWS ParallelCluster バージョン 3.2.0 から 3.6.*x* では、インスタンスで複数のインスタンスタイプを設定した場合、 を有効にする`EnableMemoryBasedScheduling`ことはできません。

> **Warning**
>
> `EnableMemoryBasedScheduling` が有効になっている Slurm キューコンピューティングリソースで複数のインスタンスタイプを指定すると、`RealMemory` の値はすべてのインスタンスタイプで使用できる最小メモリ量になります。これにより、メモリ容量が大きく異なるインスタンスタイプを指定すると、大量の未使用メモリが発生する可能性があります。

`EnableMemoryBasedScheduling: true` では、Slurm スケジューラは各ジョブが各ノードで必要とするメモリ量を追跡します。次に、Slurm スケジューラはこの情報を使用して、同じコンピューティングノード上の複数のジョブをスケジュールします。ノード上でジョブが必要とするメモリの合計量は、使用可能なノードメモリを超えることはできません。スケジューラは、ジョブが、ジョブ送信時に要求された量よりも多くのメモリを使用するのを防ぎます。

`EnableMemoryBasedScheduling: false` を使用すると、ジョブが共有ノード上のメモリを奪い合い、ジョブの失敗や `out-of-memory` のイベントが発生する可能性があります。

> **Warning**
>
> Slurm ラベルには MB や GB などの 2 のべき乗表記を使用します。これらのラベルをそれぞれ MiB と GiB として読み取ります。

Slurm 設定とメモリベースのスケジューリング

`EnableMemoryBasedScheduling: true` では、Slurm は以下の Slurm 設定パラメータを設定します。

- `SelectTypeParameters=CR_CPU_Memory`」(`slurm.conf`) を参照してください。このオプションは、ノードメモリを Slurm で消費可能なリソースとして設定します。
- Slurm `cgroup.conf` の`ConstrainRAMSpace=yes`。このオプションでは、ジョブのメモリへのアクセスは、ジョブが送信時に要求したメモリ量に制限されます。

> **Note**
> これら 2 つのオプションを設定すると、Slurm スケジューラとリソースマネージャの動作に影響する Slurm 設定パラメータが他にもいくつかあります。詳細については、Slurm のドキュメント を参照してください。

Slurm スケジューラとメモリベースのスケジューリング

`EnableMemoryBasedScheduling: false` (デフォルト)

デフォルトで `EnableMemoryBasedScheduling` は false に設定されます。false の場合、Slurm はスケジューリングアルゴリズムにメモリをリソースとして含めず、ジョブが使用するメモリも追跡しません。ユーザーは、ジョブに必要なノードあたりの最小メモリ量を設定する `--mem MEM_PER_NODE` オプションを指定できます。これにより、スケジューラはジョブをスケジューリングするときに、`RealMemory` 値が少なくとも `MEM_PER_NODE` のノードを選択する必要があります。

例えば、ユーザーが `--mem=5GB` のジョブを 2 つ送信したとします。CPU や GPU などの要求されたリソースが使用可能な場合、ジョブは 8 GiB のメモリのノードで同時に実行できます。この 2 つのジョブは、`RealMemory` が 5 GiB 未満のコンピューティングノードではスケジュールされません。

> **Warning**
> メモリベースのスケジューリングが無効になっている場合、Slurm はジョブが使用するメモリ量を追跡しません。同じノードで実行されるジョブがメモリリソースを奪い合い、他のジョブが失敗する可能性があります。
> メモリベースのスケジューリングが無効になっている場合、`--mem-per-cpu` または `--mem-per-gpu` オプションを指定しないことをお勧めします。これらのオプションにより、Slurm のドキュメントに記載している内容とは異なる動作が発生する場合があります。

`EnableMemoryBasedScheduling: true`

`EnableMemoryBasedScheduling` を true に設定すると、Slurm は各ジョブのメモリ使用量を追跡し、ジョブが `--mem` 送信オプションで要求された量よりも多くのメモリを使用するのを防ぎます。

前述の例を使用して、ユーザーは `--mem=5GB` のジョブを 2 つ送信します。メモリが 8 GiB のノードでは、ジョブを同時に実行することはできません。これは、必要なメモリの合計量がノードで使用可能なメモリ量よりも多いためです。

メモリベースのスケジューリングが有効になっている場合は、`--mem-per-cpu` と `--mem-per-gpu` は「Slurm documentation」に記載されている内容と一貫した動作をします。例えば、ジョブは `--ntasks-per-node=2 -c 1 --mem-per-cpu=2GB` で送信されます。この場合、Slurm はジョブに合計 4 GiB をノードごとに割り当てます。

> **Warning**
>
> メモリベースのスケジューリングが有効になっている場合は、ジョブを送信する際に `--mem` 仕様を含めることをお勧めします。に含まれているSlurmデフォルト設定では AWS ParallelCluster、メモリオプション (`--mem`、、または `--mem-per-gpu`) が含まれていない場合`--mem-per-cpu`、 CPUs は CPU や GPU などの他のリソースの一部のみをリクエストしても、割り当てられたノードのメモリ全体をジョブにSlurm割り当てます。 GPUs これにより、他のジョブに使用できるメモリがなくなるため、ジョブが終了するまでノードを共有できなくなります。これは、ジョブの送信時にメモリ仕様が提供されていないとき、Slurm がジョブのノードあたりのメモリを `DefMemPerNode` に設定するためです。このパラメータのデフォルト値は 0 で、ノードのメモリに無制限にアクセスできるようになっています。メモリ量の異なる複数のタイプのコンピューティングリソースが同じキューにある場合、メモリオプションなしで送信されたジョブには、異なるノードに異なる量のメモリが割り当てられる可能性があります。これは、スケジューラがどのノードをジョブに使用できるようにするかによって異なります。ユーザーは、Slurm 設定ファイルのクラスターまたはパーティションレベルで、`DefMemPerNode` や `DefMemPerCPU` のようなオプションのカスタム値を定義して、この動作を防ぐことができます。

Slurm RealMemory と AWS ParallelCluster SchedulableMemory

に同梱されているSlurm設定では AWS ParallelCluster、 Slurm は RealMemory をジョブで使用できるノードあたりのメモリ量として解釈します。バージョン 3.2.0 以降、デフォルトでは、 `RealMemory`は Amazon EC2 インスタンスタイプにリストされ、Amazon EC2 API DescribeInstanceTypes によって返されるメモリの 95% AWS ParallelCluster に設定します。

メモリベースのスケジューリングが無効な場合、`--mem` を指定したジョブをユーザーが送信すると、Slurm スケジューラは `RealMemory` を使用しノードをフィルタリングします。

メモリーベースのスケジューリングが有効な場合、Slurm スケジューラは RealMemory をコンピューティングノード上で実行中のジョブに使用可能な最大メモリー容量と解釈します。

デフォルト設定は、すべてのインスタンスタイプに最適であるとは限りません。

- この設定は、ノードが実際にアクセスできるメモリ量よりも多い場合があります。これは、コンピューティングノードが小さいインスタンスタイプである場合に発生する可能性があります。
- この設定は、ノードが実際にアクセスできるメモリ量よりも少ない場合があります。これは、コンピューティングノードが大きいインスタンスタイプの場合に発生し、大量のメモリが未使用のままになる可能性があります。

SlurmQueues /ComputeResources/ を使用して、コンピューティングノード AWS ParallelCluster 用に によってRealMemory設定された の値をSchedulableMemory微調整できます。デフォルトをオーバーライドするには、クラスター設定専用に SchedulableMemory のカスタム値を定義します。

コンピューティングノードの実際に使用可能なメモリを確認するには、ノード上で /opt/slurm/sbin/slurmd -C コマンドを実行します。このコマンドは、RealMemory 値を含むノードのハードウェア設定を返します。詳細については、「slurmd -C」を参照してください。

コンピューティングノードのオペレーティングシステムプロセスに十分なメモリがあることを確認してください。そのためには、SchedulableMemory 値を、slurmd -C コマンドが返した RealMemory 値よりも小さく設定してジョブが使用できるメモリを制限します。

Slurm による複数のインスタンスタイプの割り当て

AWS ParallelCluster バージョン 3.3.0 以降では、コンピューティングリソースの定義されたインスタンスタイプのセットから を割り当てるようにクラスターを設定できます。割り当ては、Amazon EC2 Fleet の低コスト戦略や最適容量戦略に基づいて行うことができます。

この定義済みのインスタンスタイプセットは、すべて同じ数の vCPU を備えているか、マルチスレッドが無効な場合は同じ数のコアを備えている必要があります。さらに、このインスタンスタイプセットには、同じ製造元の同じ数のアクセラレータが必要です。Efa/Enabled が true に設定されている場合、インスタンスは EFA をサポートしている必要があります。要件の詳細については、「Scheduling/SlurmQueues/AllocationStrategy」および「ComputeResources/Instances」を参照してください。

CapacityType の設定に応じて、AllocationStrategy を lowest-price または capacity-optimized に設定できます。

Instances では、一連のインスタンスタイプを設定できます。

> **Note**
>
> AWS ParallelCluster バージョン 3.7.0 以降では、インスタンスで複数のインスタンスタイプを設定すると、 を有効にEnableMemoryBasedSchedulingできます。
> AWS ParallelCluster バージョン 3.2.0 から 3.6.*x* では、インスタンスで複数のインスタンスタイプを設定した場合、 を有効にするEnableMemoryBasedSchedulingことはできません。

以下の例では、vCPUs、EFA サポート、アーキテクチャのインスタンスタイプをクエリする方法を示しています。

96 個の vCPU と x86_64 アーキテクチャで InstanceTypes をクエリする。

```
$ aws ec2 describe-instance-types --region region-id \
  --filters "Name=vcpu-info.default-vcpus,Values=96" "Name=processor-info.supported-
architecture,Values=x86_64" \
  --query "sort_by(InstanceTypes[*].
{InstanceType:InstanceType,MemoryMiB:MemoryInfo.SizeInMiB,CurrentGeneration:CurrentGeneration,V
 &InstanceType)" \
  --output table
```

64 コア、EFA サポート、arm64 アーキテクチャで、InstanceTypes をクエリする。

```
$ aws ec2 describe-instance-types --region region-id \
  --filters "Name=vcpu-info.default-cores,Values=64" "Name=processor-
info.supported-architecture,Values=arm64" "Name=network-info.efa-
supported,Values=true" --query "sort_by(InstanceTypes[*].
{InstanceType:InstanceType,MemoryMiB:MemoryInfo.SizeInMiB,CurrentGeneration:CurrentGeneration,V
 &InstanceType)" \
  --output table
```

次のクラスター設定スニペット例は、これらの InstanceType および AllocationStrategy プロパティの使用方法を示しています。

```
...
 Scheduling:
  Scheduler: slurm
```

```
  SlurmQueues:
    - Name: queue-1
      CapacityType: ONDEMAND
      AllocationStrategy: lowest-price
      ...
      ComputeResources:
        - Name: computeresource1
          Instances:
            - InstanceType: r6g.2xlarge
            - InstanceType: m6g.2xlarge
            - InstanceType: c6g.2xlarge
          MinCount: 0
          MaxCount: 500
        - Name: computeresource2
          Instances:
            - InstanceType: m6g.12xlarge
            - InstanceType: x2gd.12xlarge
          MinCount: 0
          MaxCount: 500
...
```

動的ノードのクラスタースケーリング

ParallelCluster は、Slurm の省電力プラグインを使用してクラスターを動的にスケールする Slurm のメソッドをサポートしています。詳細については、「Slurm ドキュメント」の「クラウドスケジューリングガイド」と「Slurm 省電力ガイド」を参照してください。以下のトピックでは、各バージョンの Slurm 戦略について説明します。

トピック

- バージョン 3.8.0 の Slurm 動的ノード割り当て戦略
- バージョン 3.7.x での Slurm 動的ノード割り当て戦略
- バージョン 3.6.x 以前での Slurm 動的ノード割り当て戦略

バージョン 3.8.0 の Slurm 動的ノード割り当て戦略

ParallelCluster は、バージョン 3.8.0 以降、ジョブレベルの再開またはジョブレベルのスケーリングをデフォルトの動的ノード割り当て戦略として使用してクラスターをスケールします。ParallelCluster は、各ジョブの要件、ジョブに割り当てられたノードの数、再開する必要があるノードに基づいてクラスターをスケールアップします。ParallelCluster は、この情報を SLURM_RESUME_FILE 環境変数から取得します。

動的ノードのスケーリングプロセスは、Amazon EC2 インスタンスを起動するステップと、起動した EC2 インスタンスを Slurm ノードに割り当てるステップの 2 つで構成されます。これら 2 つのステップは、どちらもオールオアナッシングロジックまたはベストエフォートロジックを使用して実行できます。

Amazon EC2 インスタンスを起動する場合:

- オールオアナッシングでは、最小ターゲット容量が合計ターゲット容量に等しい場合に Amazon EC2 API を起動します。
- ベストエフォートでは、最小ターゲット容量が 1 に等しく、合計ターゲット容量がリクエスト容量に等しい場合に Amazon EC2 API を起動します。

Amazon EC2 インスタンスを Slurm ノードに割り当てる場合:

- オールオアナッシングでは、リクエストされたすべての Slurm ノードに Amazon EC2 インスタンスを割り当てることができる場合にのみ、Amazon EC2 インスタンスをノードに割り当てます。
- ベストエフォートでは、リクエストされたすべての Slurm ノードを Amazon EC2 インスタンス容量でカバーできない場合でも、Amazon EC2 インスタンスをノードに割り当てます。

  上の戦略の可能な組み合わせが、ParallelCluster の起動戦略になります。

Example

<caption>The available ParallelCluster 起動戦略 that can be set into the ScalingStrategy cluster configuration to be used with ジョブレベルのスケーリング are:</caption>

オールオアナッシングのスケーリング:

この戦略では、ジョブごとに Amazon EC2 起動インスタンス API コール AWS ParallelCluster を開始します。この呼び出しでは、リクエストされたコンピューティングノードが正常に起動するために必要なすべてのインスタンスが必要です。ジョブごとの必要な容量が利用可能な場合にのみクラスターがスケールするため、スケーリングプロセスの最後にアイドルインスタンスが残ることはありません。

この戦略では、ジョブごとに Amazon EC2 インスタンスを起動するのにオールオアナッシングロジックを使用するとともに、Amazon EC2 インスタンスを Slurm ノードに割り当てるのにもオールオアナッシングロジックを使用します。

戦略グループは、リクエストをバッチにまとめて起動します。リクエストされたコンピューティングリソースごとに 1 つのバッチを使用し、バッチごとのノード数は最大 500 です。リクエストが複数のコンピューティングリソースにまたがる場合や、500 ノードを超える場合、ParallelCluster は複数のバッチを使用して順次処理します。

いずれかのリソースのバッチに障害が発生すると、すべての関連する未使用の容量が終了するため、スケーリングプロセスの終了時にアイドルインスタンスは残りません。

制限

- スケーリングにかかる時間は、Slurm 再開プログラムの実行ごとに送信されるジョブの数に正比例します。
- スケーリングオペレーションは、デフォルトで 1,000 インスタンスに設定されている、RunInstances リソースアカウントの上限によって制限されます。この制限は、 AWSの EC2 API スロットリングポリシーに準拠しています。詳細については、Amazon EC2 API スロットリングドキュメント」を参照してください。
- 単一のインスタンスタイプを持つコンピューティングリソースのジョブを、複数のアベイラビリティーゾーンにまたがるキューに送信する場合、オールオアナッシングの EC2 起動 API コールは、すべての容量を単一のアベイラビリティーゾーンで提供できる場合にのみ成功します。
- 複数のインスタンスタイプを持つコンピューティングリソースのジョブを、単一のアベイラビリティーゾーンのキューに送信する場合、オールオアナッシングの Amazon EC2 起動 API コールは、すべての容量を単一のインスタンスタイプで提供できる場合にのみ成功します。
- 複数のインスタンスタイプを持つコンピューティングリソースのジョブを、複数のアベイラビリティーゾーンにまたがるキューに送信する場合、オールオアナッシングの Amazon EC2 起動 API コールはサポートされず、ParallelCluster は代わりにベストエフォートのスケーリングを実行します。

貪欲なオールオアナッシングのスケーリング:

このオールオアノッシング戦略のバリアントは、ジョブごとの必要な容量が利用可能な場合にのみクラスターをスケールし、スケーリングプロセスの最後にアイドルインスタンスが残らないようにする点は同じですが、ParallelCluster が開始する Amazon EC2 起動インスタンス API コールでは、最小ターゲット容量を 1 とし、リクエストされた容量までノードの起動数を最大化しようとします。この戦略では、すべてのジョブで EC2 インスタンスを起動するためにベストエフォートロジックを使用するとともに、ジョブごとに Amazon EC2 インスタンスを Slurm ノードに割り当てるためにもオールオアノッシングロジックを使用します。

戦略グループは、リクエストをバッチにまとめて起動します。リクエストされたコンピューティングリソースごとに 1 つのバッチを使用し、バッチごとのノード数は最大 500 です。複数のコンピューティングリソースにまたがるリクエスト、または 500 ノードを超えるリクエストの場合、ParellelCluster は複数のバッチを順次処理します。

これにより、スケーリングプロセス中に一時的なオーバースケーリングが発生してもスループットを最大化することで、スケーリングプロセスの最後にアイドルインスタンスが残らないようにします。

制限

- 一時的なオーバースケーリングが発生し、スケーリングの完了前にインスタンスが実行中の状態に移行した場合は追加コストが発生することがあります。
- all-or-nothing戦略と同じインスタンス制限が適用されます。ただし、 AWSの RunInstances リソースアカウント制限が適用されます。

ベストエフォートのスケーリング:

この戦略で開始する Amazon EC2 起動インスタンス API コールでは、最小容量を 1 とし、リクエストされた容量のすべてには対応できずにアイドルインスタンスがスケーリングプロセスの実行後に残るとしても、リクエストされた合計容量を達成しようとします。この戦略では、すべてのジョブで Amazon EC2 インスタンスを起動するためにベストエフォートロジックを使用するとともに、ジョブごとに Amazon EC2 インスタンスを Slurm ノードに割り当てるためにもベストエフォートロジックを使用します。

戦略グループは、リクエストをバッチにまとめて起動します。リクエストされたコンピューティングリソースごとに 1 つのバッチを使用し、バッチごとのノード数は最大 500 です。リクエストが複数のコンピューティングリソースにまたがる場合や、500 ノードを超える場合、ParallelCluster は複数のバッチを使用して順次処理します。

この戦略では、さまざまなスケーリングプロセスでアイドルインスタンスが発生するとしても、複数のスケーリングプロセスの実行に対するデフォルトの 1,000 インスタンスの制限をはるかに超えてスケーリングできます。

制限

- ジョブがリクエストしているノードの一部を割り当てることができない場合、スケーリングプロセスの最後にアイドル状態のインスタンスが発生する可能性があります。

ParallelCluster 起動戦略別に動的ノードのスケーリングがどのように動作するかについて、以下に例を示します。2 つのジョブを送信し、同じタイプのノードをそれぞれ 20 個、合計 40 個リクエストしたとします。しかし、リクエストした容量をカバーするために EC2 で利用できる Amazon EC2 インスタンスは 30 個のみです。

オールオアナッシングのスケーリング:

- 最初のジョブでは、オールオアナッシングの Amazon EC2 起動インスタンス API を呼び出し、20 個のインスタンスをリクエストします。呼び出しは成功して、20 個のインスタンスが起動します。
- 最初のジョブでは、起動した 20 個のインスタンスを Slurm ノードに割り当てるオールオアナッシングロジックも成功します。
- 2 番目のジョブで、別のオールオアナッシングの Amazon EC2 起動インスタンス API を呼び出し、20 個のインスタンスをリクエストします。10 個のインスタンスの容量しか残っていないため、この呼び出しは失敗します。この時点ではインスタンスは起動されません

貪欲なオールオアナッシングのスケーリング:

- ベストエフォートの Amazon EC2 起動インスタンス API を呼び出し、40 個のインスタンスをリクエストします。これは、すべてのジョブがリクエストしている合計容量です。結果として、30 個のインスタンスが起動します。
- 最初のジョブで、起動したインスタンスのうち 20 個を Slurm ノードに割り当てるオールオアナッシングロジックは成功します。
- 2 番目のジョブでは、起動した残りのインスタンスを Slurm ノードに割り当てる別のオールオアナッシングロジックを試行しますが、ジョブがリクエストしている合計 20 個に対して利用可能なインスタンスは 10 個しかないため、割り当ては失敗します。
- 割り当てられない 10 個の起動済みインスタンスは終了します。

ベストエフォートのスケーリング:

- ベストエフォートの Amazon EC2 起動インスタンス API を呼び出し、40 個のインスタンスをリクエストします。これは、すべてのジョブがリクエストしている合計容量です。結果として、30 個のインスタンスが起動します。
- 最初のジョブで、起動した 20 個のインスタンスを Slurm ノードに割り当てるベストエフォートロジックが成功します。

- 2 番目のジョブで、起動した残りの 10 個のインスタンスを Slurm ノードに割り当てる別のベストエフォートロジックは、リクエストされた合計容量が 20 個であっても、成功します。ただし、ジョブは 20 個のノードをリクエストしており、そのうちの 10 個のみに Amazon EC2 インスタンスを割り当てることができたため、スケーリングプロセスの今後のコールで欠落している 10 個のインスタンスを起動するのに十分な容量が見つかるか、スケジューラが他の既に実行中のコンピューティングノードでジョブをスケジュールするまで、ジョブは開始できず、インスタンスはアイドル状態のままになります。

バージョン 3.7.x での Slurm 動的ノード割り当て戦略

ParallelCluster は 次の 2 種類の動的ノード割り当て戦略を使用してクラスターをスケールします。

- リクエストされた利用可能なノード情報に基づく割り当て:
  - 全ノード再開またはノードリストのスケーリング:

    ParallelCluster は、Slurm の `ResumeProgram` の実行時に、Slurm がリクエストしたノードリストの名前のみに基づいてクラスターをスケールアップします。コンピューティングリソースはノード名のみでノードに割り当てられます。ノード名のリストは複数のジョブにまたがる場合があります。
  - ジョブレベルの再開またはジョブレベルのスケーリング。

    ParallelCluster は、各ジョブの要件、ジョブに割り当てられている現在のノード数、再開する必要があるノードに基づいて、クラスターをスケールアップします。ParallelCluster は、この情報を `SLURM_RESUME_FILE` 環境変数から取得します。
- Amazon EC2 起動戦略による割り当て:
  - ベストエフォートのスケーリング:

    ParallelCluster は、最小ターゲット容量が 1 に等しい Amazon EC2 起動インスタンス API コールを使用してクラスターをスケールアップし、リクエストされたノードをサポートするのに必要なインスタンスのすべてとは言わないまでも一部を起動します。
  - オールオアナッシングスケーリング:

    ParallelCluster は、リクエストされたノードをサポートするのに必要なすべてのインスタンスが起動された場合にのみ成功する Amazon EC2 起動インスタンス API コールを使用してクラスターをスケールアップします。この場合、最小ターゲット容量がリクエストされた容量の合計に等しい Amazon EC2 起動インスタンス API を呼び出します。

デフォルトでは、ParallelCluster はベストエフォートの Amazon EC2 起動戦略でノードリストのスケーリングを使用して、リクエストされたノードをサポートするのに必要なインスタンスのすべてとは言わないまでも一部を起動します。送信されたワークロードに対応できるように、できるだけ多くの容量をプロビジョニングしようとします。

ParallelCluster は、バージョン 3.7.0 以降、排他モードで送信されたジョブに対し、オールオアナッシングの EC2 起動戦略でジョブレベルのスケーリングを採用しています。排他モードでジョブを送信すると、そのジョブは割り当てられたノードに排他的にアクセスできるようになります。詳細については、「Slurm ドキュメント」の「EXCLUSIVE」を参照してください。

排他モードでジョブを送信するには。

- Slurm ジョブをクラスターに送信する際に排他フラグを渡します。例えば、`sbatch ... --exclusive`。

  または

- `JobExclusiveAllocation` を `true` に設定したクラスターキューにジョブを送信します。

排他モードでジョブを送信する場合。

- ParallelCluster は、現在、最大 500 個のノードを含むように起動リクエストをバッチ処理します。ジョブが 500 個を超えるノードをリクエストすると、ParallelCluster は 500 個のノードのセットごとにオールオアナッシングの起動リクエストを行い、残りのノードに対しては追加の起動リクエストを行います。
- ノードの割り当てが単一のコンピューティングリソース内で行われる場合、ParallelCluster は 500 個のノードのセットごとにオールオアナッシングの起動リクエストを行い、残りのノードに対しては追加の起動リクエストを行います。1 つの起動リクエストが失敗すると、ParallelCluster はすべての起動リクエストによって作成された未使用の容量を終了します。
- ノードの割り当てが複数のコンピューティングリソースにまたがる場合、ParallelCluster はコンピューティングリソースごとにオールオアナッシングの起動リクエストを行う必要があります。これらのリクエストもバッチ処理されます。1 つのコンピューティングリソースの起動リクエストが失敗すると、ParallelCluster はすべてのコンピューティングリソースの起動リクエストによって作成された未使用の容量を終了します。

オールオアナッシングの起動戦略を使用したジョブレベルスケーリングの既知の制限事項

- 単一のインスタンスタイプを持つコンピューティングリソースのジョブを、複数のアベイラビリティーゾーンにまたがるキューに送信する場合、オールオアナッシングの EC2 起動 API コールは、すべての容量を単一のアベイラビリティーゾーンで提供できる場合にのみ成功します。
- 複数のインスタンスタイプを持つコンピューティングリソースのジョブを、単一のアベイラビリティーゾーンのキューに送信する場合、オールオアナッシングの Amazon EC2 起動 API コールは、すべての容量を単一のインスタンスタイプで提供できる場合にのみ成功します。
- 複数のインスタンスタイプを持つコンピューティングリソースのジョブを、複数のアベイラビリティーゾーンにまたがるキューに送信する場合、オールオアナッシングの Amazon EC2 起動 API コールはサポートされず、ParallelCluster は代わりにベストエフォートのスケーリングを実行します。

バージョン 3.6.x 以前での Slurm 動的ノード割り当て戦略

AWS ParallelCluster は 1 種類の動的ノード割り当て戦略のみを使用してクラスターをスケールします。

- リクエストされた利用可能なノード情報に基づく割り当て:
  - 全ノード再開またはノードリストのスケーリング: ParallelCluster は、Slurm の `ResumeProgram` の実行時に Slurm のリクエストされたノードリストの名前のみに基づいてクラスターをスケールアップします。コンピューティングリソースはノード名のみでノードに割り当てられます。ノード名のリストは複数のジョブにまたがる場合があります。
- Amazon EC2 起動戦略による割り当て:
  - ベストエフォートのスケーリング: ParallelCluster は、最小ターゲット容量が 1 に等しい Amazon EC2 起動インスタンス API コールを使用してクラスターをスケールアップし、リクエストされたノードをサポートするのに必要なインスタンスのすべてとは言わないまでも一部を起動します。

ParallelCluster は、ベストエフォートの Amazon EC2 起動戦略でノードリストのスケーリングを使用して、リクエストされたノードをサポートするのに必要なインスタンスのすべてとは言わないまでも一部を起動します。送信されたワークロードに対応できるように、できるだけ多くの容量をプロビジョニングしようとします。

制限

- ジョブがリクエストしているノードの一部を割り当てることができない場合、スケーリングプロセスの最後にアイドル状態のインスタンスが発生する可能性があります。

Slurm による アカウンティング AWS ParallelCluster

バージョン 3.3.0 以降、 はクラスター設定パラメータ SlurmSettings / Database でのSlurmアカウンティング AWS ParallelCluster をサポートしています。

バージョン 3.10.0 以降、 は、クラスター設定パラメータ SlurmSettings / ExternalSlurmdbd ExternalSlurmdbd とのSlurmアカウンティング AWS ParallelCluster をサポートしています。複数のクラスターが同じデータベースを共有する場合は、外部 Slurmdbd を使用することをお勧めします。

Slurm アカウンティングを使用すると、外部のアカウンティングデータベースを統合して次のことを行うことができます。

- クラスターユーザーまたはユーザーのグループとその他のエンティティを管理する。この機能を使用すると、リソース制限の適用、公平配分、QOSs など、 Slurmのより高度な機能を使用できます。
- ジョブを実行したユーザー、ジョブの期間、および使用するリソースなどのジョブデータを収集して保存する。保存したデータは `sacct` ユーティリティを使用して表示できます。

> **Note**
>
> AWS ParallelCluster は、Slurmサポートされている MySQL データベースサーバーのSlurmアカウンティングをサポートしています。

v3.10.0 AWS ParallelCluster 以降Slurmdbdで外部 を使用してSlurmアカウンティングを操作する

Slurm アカウンティングを設定する前に、既存の外部データベースサーバーに接続する既存の外部 Slurmdbd データベースサーバーが必要です。

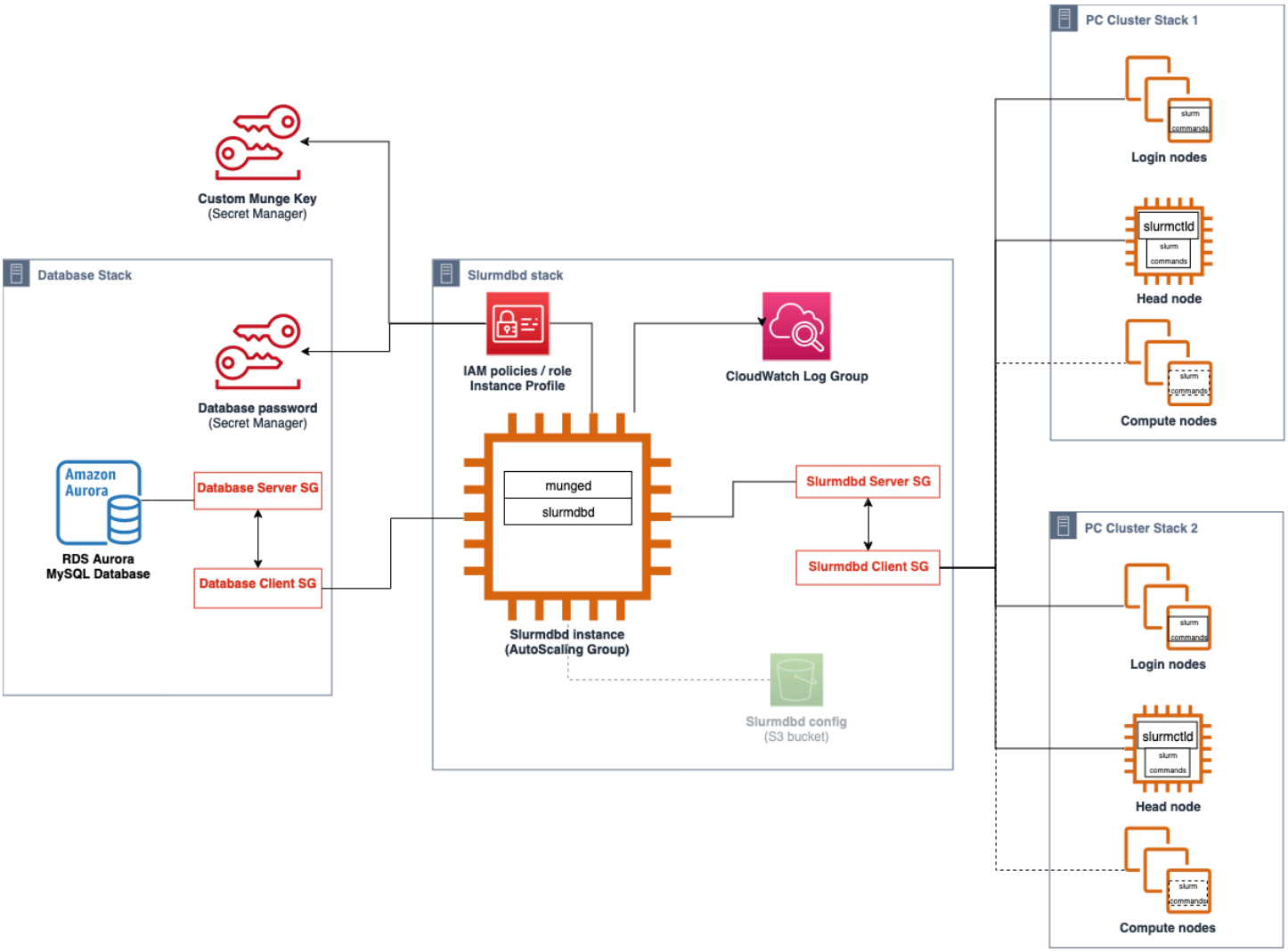
これを設定するには、以下を定義します。

- ExternalSlurmdbd/Host 内の外部 Slurmdbd サーバーのアドレス。このサーバーが存在していて、ヘッドノードからアクセスできることが必要です。
- MungeKeySecretArn で外部 Slurmdbd サーバーと通信するための munge キー。

チュートリアルを完了するには、「外部 Slurmdbd アカウンティングによるクラスターの作成」を参照してください。

> **Note**
>
> Slurm データベースのアカウンティングエンティティを管理する責任はユーザーにあります。

AWS ParallelCluster 外部SlurmDBサポート機能のアーキテクチャにより、複数のクラスターが同じSlurmDBデータベースを共有できます。



> **Warning**
>
> AWS ParallelCluster と外部との間のトラフィックSlurmDBは暗号化されません。クラスターと外部 SlurmDB は、信頼できるネットワークで実行することをお勧めします。

v3.3.0 AWS ParallelCluster 以降のヘッドノードを使用したSlurmアカウンティングSlurmdbdの操作

Slurm アカウンティングを設定する前に、既存の外部データベースサーバーと `mysql` プロトコルを使用するデータベースが必要です。

でSlurmアカウンティングを設定するには AWS ParallelCluster、以下を定義する必要があります。

- Database/Uri の形式で表す外部データベースサーバーの URI。サーバーが存在し、ヘッドノードから到達できる必要があります。
- Database / PasswordSecretArn および Database / UserName. で定義されている外部データベースにアクセスするための認証情報は、この情報 AWS ParallelCluster を使用して、 Slurmレベルでアカウンティングを設定し、ヘッドノード上の `slurmdbd`サービスを設定します。 `slurmdbd`は、クラスターとデータベースサーバー間の通信を管理するデーモンです。

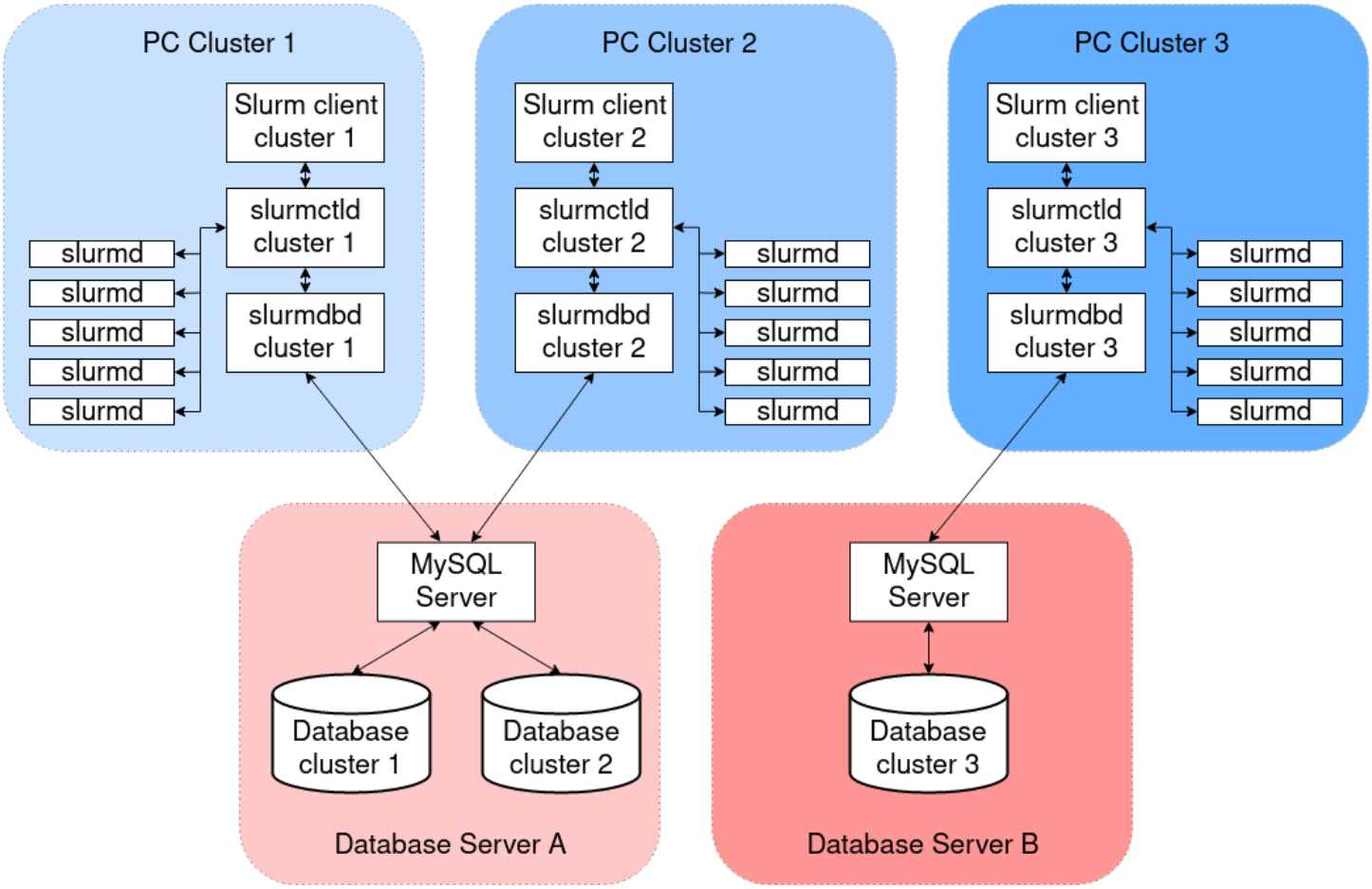
チュートリアルを完了するには、「Slurm アカウンティングによるクラスターの作成」を参照してください。

> **Note**
>
> AWS ParallelCluster は、デフォルトのクラスターユーザーをデータベースのデータベース管理者として設定することで、アSlurmカウンティングSlurmデータベースの基本的なブートストラップを実行します。 AWS ParallelCluster は、アカウンティングデータベースに他のユーザーを追加しません。Slurm データベースのアカウンティングエンティティを管理する責任はお客様にあります。

AWS ParallelCluster は`slurmdbd`、クラスターがSlurmデータベースサーバー上に独自のデータベースを持つように を設定します。同じデータベースサーバーを複数のクラスターで使用できますが、各クラスターには個別のデータベースがあります。 はクラスター名 AWS ParallelCluster を使用して、`slurmdbd`設定ファイル`StorageLoc`パラメータでデータベースの名前を定義します。次の状況を考えてみます。データベースサーバーに存在するデータベースに、アクティブなクラスター名にマッピングされていないクラスター名が含まれています。この場合、そのクラスター名を使用して新しいクラスターを作成し、そのデータベースにマッピングできます。Slurm は新しいクラスターにデータベースを再利用します。

> **Warning**
>
> - 一度に同じデータベースを使用するように複数のクラスターを設定することはお勧めしません。これにより、パフォーマンスの問題またはデータベースのデッドロック状態が発生する可能性があります。
> - クラスターのヘッドノードで Slurm アカウンティングが有効になっている場合、強力な CPU、より多くのメモリ、およびより高いネットワーク帯域幅を使用するインスタンスタイプを使用することをお勧めします。Slurm アカウンティングによりクラスターのヘッドノードに負担が加わる可能性があります。

アカウンティング機能の現在のアーキテクチャ AWS ParallelCluster Slurmでは、次の図の設定例に示すように、各クラスターには`slurmdbd`デーモンの独自のインスタンスがあります。



クラスター環境に Slurm カスタムマルチクラスター機能またはフェデレーション機能を追加する場合、すべてのクラスターが同じ `slurmdbd` インスタンスを参照する必要があります。この代替方法

として、1 つのクラスターでアカウンティングを有効に AWS ParallelCluster Slurmし、最初のクラスターでホストslurmdbdされている に接続するように他のクラスターを手動で設定することをお勧めします。

AWS ParallelCluster バージョン 3.3.0 より前のバージョンを使用している場合は、この HPC ブログ投稿で説明されているアSlurmカウンティングを実装するための代替方法を参照してください。

Slurm のアカウンティングに関する考慮事項

異なる VPC のデータベースとクラスター

Slurm のアカウンティングを有効にするには、slurmdbd デーモンが実行する読み取り操作と書き込み操作のバックエンドとして機能するデータベースサーバーが必要です。クラスターを作成または更新して Slurm アカウンティングを有効にする前に、ヘッドノードがデータベースサーバーに到達できる必要があります。

クラスターが使用していない VPC にデータベースサーバーをデプロイする必要がある場合は、次の点を考慮します。

- クラスター側の slurmdbd とデータベースサーバー間の通信を有効にするには、2 つの VPC 間の接続を設定する必要があります。詳細については、「Amazon Virtual Private Cloud ユーザーガイド」の「VPC ピア機能とは」を参照してください。
- クラスターの VPC のヘッドノードにアタッチするセキュリティグループを作成する必要があります。2 つの VPC がピアリング接続されると、データベース側とクラスター側のセキュリティグループ間のクロスリンクが使用可能になります。詳細については、「Amazon Virtual Private Cloud ユーザーガイド」の「セキュリティグループのルール」を参照してください。

slurmdbd とデータベースサーバー間の TLS 暗号化を設定する

AWS ParallelCluster が提供するデフォルトのSlurmアカウンティング設定では、サーバーが Amazon RDS などの TLS encryption. AWS database サービスをサポートし、デフォルトで TLS 暗号化 Amazon Aurora をサポートしている場合、 はデータベースサーバーへの TLS 暗号化接続slurmdbdを確立します。

データベースサーバーで require_secure_transport パラメータを設定することにより、サーバー側の安全な接続を要求できます。これは、提供されている CloudFormation テンプレートで設定されます。

セキュリティのベストプラクティスに従って、slurmdbd クライアントのサーバー ID 検証も有効にすることをお勧めします。これを行うには、slurmdbd.conf で StorageParameters

を設定します。サーバー CA 証明書をクラスターのヘッドノードにアップロードします。次に、`slurmdbd.conf` の `StorageParameters` の SSL_CA オプションをヘッドノードのサーバー CA 証明書のパスに設定します。これにより、`slurmdbd` 側でのサーバー ID 検証が有効になります。これらの変更を行った後、`slurmdbd` サービスを再起動して ID 検証が有効になっているデータベースサーバーへの接続を再確立します。

データベース認証情報を更新する

Database/UserName または PasswordSecretArn の値を更新するには、まずコンピューティングフリートを停止する必要があります。シークレットに保存されている AWS Secrets Manager シークレット値が変更され、その ARN が変更されないとします。この状況では、クラスターは自動的にデータベースのパスワードを新しい値に更新しません。新しいシークレット値のクラスターを更新するには、ヘッドノードから次のコマンドを実行します。

```
$ sudo /opt/parallelcluster/scripts/slurm/update_slurm_database_password.sh
```

> **Warning**
> 財務データが失われないように、コンピューティングフリートが停止している場合にのみデータベースパスワードを変更することをお勧めします。

データベースのモニタリング

AWS データベースサービスのモニタリング機能を有効にすることをお勧めします。詳細については、「Amazon RDS のモニタリング」または「Amazon Aurora のモニタリング」のドキュメントを参照してください。

Slurm 設定のカスタマイズ

AWS ParallelCluster バージョン 3.6.0 以降では、 AWS ParallelCluster クラスター`slurm.conf`Slurm設定で設定をカスタマイズできます。

クラスター設定では、以下の クラスター構成設定を使用してSlurm 設定パラメータをカスタマイズできます。

- / または両方を指定した場合は `CustomSlurmSettings` parameter. AWS ParallelCluster fails `SlurmSettings` を使用してSlurm、クラスター全体の`CustomSlurmSettingsIncludeFile`パラメータをカスタマイズします。

- SlurmQueues/CustomSlurmSettings (Slurm パーティションにマッピングされています) を使用してキューの Slurm パラメータをカスタマイズします。
- SlurmQueues/ComputeResources/CustomSlurmSettings (Slurm ノードにマッピングされています) を使用してコンピューティングリソースの Slurm パラメータをカスタマイズします。

Slurm を使用する際の設定カスタマイズの制限と考慮事項 AWS ParallelCluster

- CustomSlurmSettings および CustomSlurmSettingsIncludeFile設定では、クラスターの設定に使用しているSlurmバージョンでサポートされている AWS ParallelCluster バージョンに含まれるslurm.confパラメータのみを指定および更新できます。
- パラメータのいずれかでカスタムSlurm設定を指定するとCustomSlurmSettings、 は検証チェック AWS ParallelCluster を実行し、 AWS ParallelCluster ロジックと競合するSlurm設定パラメータの設定または更新を防止します。と競合することが知られているSlurm設定パラメータは、拒否リストで識別 AWS ParallelCluster されます。拒否リストは、他のSlurm機能が追加されると、将来の AWS ParallelCluster バージョンで変更される可能性があります。詳細については、「CustomSlurmSettings で拒否リストに記載されている Slurm 設定パラメータ」を参照してください。
- AWS ParallelCluster は、パラメータが拒否リストにあるかどうかのみをチェックします。カスタムSlurm設定パラメータの構文またはセマンティクスは検証 AWS ParallelCluster されません。カスタム Slurm 設定パラメータはお客様の責任で検証していただく必要があります。無効なカスタム Slurm 設定パラメータは、クラスターの作成や更新の失敗につながる Slurm デーモンの障害を引き起こす可能性があります。
- でカスタムSlurm設定を指定した場合CustomSlurmSettingsIncludeFile、 AWS ParallelCluster は検証を実行しません。
- CustomSlurmSettings および CustomSlurmSettingsIncludeFile は、コンピューティングフリートを停止および起動することなく更新できます。この場合、 はslurmctldデーモン AWS ParallelCluster を再起動し、 scontrol reconfigure コマンドを実行します。

  一部の Slurm 設定パラメータでは、クラスター全体に変更が登録される前に異なる操作が必要になる場合があります。例えば、クラスター内のすべてのデーモンを再起動する必要がある場合があります。更新中にカスタムSlurm設定パラメータ設定を伝達するのに AWS ParallelCluster オペレーションが十分かどうかを検証するのはお客様の責任です。 AWS ParallelCluster オペレーションが不十分な場合は、Slurmドキュメントで推奨されているように、更新された設定を伝達するために必要な追加のアクションを提供する責任があります。

`CustomSlurmSettings` で拒否リストに記載されている Slurm 設定パラメータ

次の表は、 AWS ParallelCluster バージョン 3.6.0 以降の使用を拒否するバージョンのパラメータを示しています。 `CustomSlurmSettings`はバージョン 3.6.0 より前の AWS ParallelCluster バージョンではサポートされていません。

クラスターレベルで拒否リストに登録されているパラメータ:

Slurm パラメータ	AWS ParallelCluster バージョンに拒否リスト
CommunicationParameters	3.6.0
Epilog	3.6.0
GresTypes	3.6.0
LaunchParameters	3.6.0
Prolog	3.6.0
ReconfigFlags	3.6.0
ResumeFailProgram	3.6.0
ResumeProgram	3.6.0
ResumeTimeout	3.6.0
SlurmctldHost	3.6.0
SlurmctldLogFile	3.6.0
SlurmctldParameters	3.6.0
SlurmdLogfile	3.6.0
SlurmUser	3.6.0
SuspendExcNodes	3.6.0
SuspendProgram	3.6.0
SuspendTime	3.6.0

Slurm パラメータ	AWS ParallelCluster バージョンに拒否リスト
TaskPlugin	3.6.0
TreeWidth	3.6.0

native Slurm accounting integration がクラスター設定で設定されている場合の、クラスターレベルで拒否リストに登録されているパラメータ:

Slurm パラメータ	AWS ParallelCluster バージョンに拒否リスト
AccountingStorageType	3.6.0
AccountingStorageHost	3.6.0
AccountingStoragePort	3.6.0
AccountingStorageUser	3.6.0
JobAcctGatherType	3.6.0

によって管理されるキューのキュー (パーティション) レベルで拒否リストに登録されたパラメータ AWS ParallelCluster：

Slurm パラメータ	AWS ParallelCluster バージョンに拒否リスト
ノード	3.6.0
PartitionName	3.6.0
ResumeTimeout	3.6.0
状態	3.6.0
SuspendTime	3.6.0

以下によって管理されるコンピューティングリソースのコンピューティングリソース (ノード) レベルで拒否リストに登録されたパラメータ AWS ParallelCluster：

Slurm パラメータ	AWS ParallelCluster バージョン 以降のバージョンで拒否リストに登録
CPUs	3.6.0
機能	3.6.0
Gres	3.6.0
NodeAddr	3.6.0
NodeHostname	3.6.0
NodeName	3.6.0
(重量)	3.7.0

Slurm**prolog** と **epilog**

AWS ParallelCluster バージョン 3.6.0 以降、 で AWS ParallelCluster デプロイされるSlurm設定には、 `Prolog`および `Epilog`設定パラメータが含まれています。

```
# PROLOG AND EPILOG
Prolog=/opt/slurm/etc/scripts/prolog.d/*
Epilog=/opt/slurm/etc/scripts/epilog.d/*
SchedulerParameters=nohold_on_prolog_fail
BatchStartTimeout=180
```

詳細については、Slurm ドキュメントの「Prolog and Epilog Guide」を参照してください。

AWS ParallelCluster には、次のプロログスクリプトとエピックログスクリプトが含まれています。

- `90_plcuster_health_check_manager` (`Prolog` フォルダ内)
- `90_pcluster_noop` (`Epilog` フォルダ内)

> **Note**
>
> Prolog および Epilog フォルダの両方に、少なくとも 1 つのファイルが含まれている必要があります。

独自のカスタム prolog または epilog スクリプトを対応する Prolog および Epilog フォルダに追加して使用できます。

> **Warning**
>
> Slurm はフォルダ内のすべてのスクリプトをアルファベットの降順で実行します。

prolog および epilog スクリプトの実行時間は、ジョブの実行に必要な時間に影響します。複数または長時間実行される prolog スクリプトを実行する場合は、BatchStartTimeout 設定を更新します。デフォルトは 3 分です。

カスタム prolog と epilog スクリプトを使用している場合は、それぞれの Prolog および Epilog フォルダでスクリプトを見つけます。すべてのカスタムスクリプトの前に実行される 90_plcuster_health_check_manager スクリプトを実行し続けることを推奨します。詳細については、「Slurm 設定のカスタマイズ」を参照してください。

クラスター容量のサイズと更新

クラスターの容量は、クラスターがスケールできるコンピューティングノードの数によって定義されます。コンピューティングノードは、 AWS ParallelCluster 設定 のコンピューティングリソース内で定義された Amazon EC2 インスタンスによってバックアップされ(Scheduling/SlurmQueues/ ComputeResources)、Slurmパーティションに 1:1 をマッピング(Scheduling/SlurmQueues)するキューに編成されます。

コンピューティングリソース内では、クラスターで実行し続ける必要があるコンピューティングノード (インスタンス) の最小数 (MinCount) と、コンピューティングリソースがスケールできるインスタンスの最大数 (MaxCount 3) を設定できます。

クラスターの作成時、またはクラスターの更新時に、 はクラスターで定義された各コンピューティングリソース (Scheduling/SlurmQueues/ ComputeResources) MinCount に対して で設定された数だけ Amazon EC2 インスタンス AWS ParallelCluster を起動します。クラスター内のコンピューティングリソースの最小ノード数をカバーするために起動されたインスタンスは、静的

ノードと呼ばれます。静的ノードは、起動されるとクラスター内で永続的となり、特定のイベントや条件が発生しない限り、システムによって終了されることはありません。このようなイベントには、Slurm や Amazon EC2 ヘルスチェックの失敗、Slurm ノードステータスの DRAIN や DOWN への変更などがあります。

クラスターの負荷の増加に対応するために、1 から ‘MaxCount - MinCount’ (MaxCount マイナス MinCount) までの範囲で、オンデマンドで起動される Amazon EC2 インスタンスは、動的ノードと呼ばれます。動的ノードはエフェメラルであり、保留中のジョブを処理するために起動され、クラスター設定の Scheduling/SlurmSettings/ScaledownIdletime で定義した期間 (デフォルトは 10 分) にわたってアイドル状態が続くと終了されます。

静的ノードと動的ノードは、次の命名スキーマに準拠します。

- 静的ノード <Queue/Name>-st-<ComputeResource/Name>-<num> (この場合、<num> = 1..ComputeResource/MinCount となります)
- 動的ノード <Queue/Name>-dy-<ComputeResource/Name>-<num> (この場合、<num> = 1..(ComputeResource/MaxCount - ComputeResource/MinCount) となります)

たとえば、次の AWS ParallelCluster 設定があるとします。

```
Scheduling:
    Scheduler: Slurm
    SlurmQueues:
        - Name: queue1
            ComputeResources:
                - Name: c5xlarge
                    Instances:
                        - InstanceType: c5.xlarge
                        MinCount: 100
                        MaxCount: 150
```

Slurm では、以下のノードが定義されます。

```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
queue1*      up   infinite     50  idle~ queue1-dy-c5xlarge-[1-50]
queue1*      up   infinite    100   idle queue1-st-c5xlarge-[1-100]
```

コンピューティングリソースに MinCount == MaxCount がある場合、すべての対応するコンピューティングノードは静的になり、すべてのインスタンスはクラスターの作成/更新時に起動されて、稼働状態が維持されます。次に例を示します。

```
Scheduling:
  Scheduler: slurm
  SlurmQueues:
    - Name: queue1
      ComputeResources:
        - Name: c5xlarge
          Instances:
            - InstanceType: c5.xlarge
          MinCount: 100
          MaxCount: 100
```

```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
queue1*      up   infinite    100   idle queue1-st-c5xlarge-[1-100]
```

クラスター容量の更新

クラスター容量の更新には、キューやコンピューティングリソースの追加または削除、コンピューティングリソースの MinCount/MaxCount の変更などが含まれます。 AWS ParallelCluster バージョン 3.9.0 以降では、キューのサイズを小さくするには、クラスターの更新が行われる前にコンピューティングフリートを停止するか、QueueUpdateStrategy を TERMINATE に設定する必要があります。以下の場合は、コンピューティングフリートを停止したり、QueueUpdateStrategy を TERMINATE に設定したりする必要はありません。

- 新しいキューをスケジューリング/SlurmQueues に追加する
- 新しいコンピューティングリソースを Scheduling/SlurmQueues/ComputeResources をキューに追加する
- コンピューティングリソースの MaxCount を増やす
- コンピューティングリソースの MinCount を増やし、同じコンピューティングリソースの MaxCount を同等以上に増やす

考慮事項と制限事項

このセクションでは、クラスター容量のサイズを変更する際に考慮すべき重要な要因、制約、または制限について概説します。

- Scheduling/SlurmQueues からキューを削除すると、名前が <Queue/Name>-* であるすべてのコンピューティングノード (静的と動的の両方) が Slurm 設定から削除され、対応する Amazon EC2 インスタンスが終了します。
- キューからコンピューティングリソース Scheduling/SlurmQueues/ComputeResources を削除すると、名前が <Queue/Name>-*-<ComputeResource/Name>-* であるすべてのコンピューティングノード (静的と動的の両方) が Slurm 設定から削除され、対応する Amazon EC2 インスタンスが終了します。

コンピューティングリソースの MinCount パラメータを変更するときは、MaxCount が MinCount と等しい場合 (静的容量のみ) と MaxCount が MinCount より大きい場合 (静的容量と動的容量の混合) の 2 つの異なるシナリオを区別できます。

静的ノードのみの容量変更

- MinCount == MaxCount の場合、MinCount (および MaxCount) を増やすには、静的ノードの数を MinCount の新しい値まで拡張するようにクラスターを設定します (<Queue/Name>-st-<ComputeResource/Name>-<new_MinCount>)。システムは新しい静的容量を満たすまで必要な数の Amazon EC2 インスタンスを起動し続けます。
- MinCount == MaxCount の場合、MinCount (および MaxCount) を量 N だけ減らすには、最後の N 個の静的ノードを削除するようにクラスターを設定します (<Queue/Name>-st-<ComputeResource/Name>-<old_MinCount - N>...<old_MinCount>])。システムは対応する数の Amazon EC2 インスタンスを終了します。
  - 初期状態: MinCount = MaxCount = 100
  -
    ```
    $ sinfo
    PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
    queue1*      up   infinite    100   idle queue1-st-c5xlarge-[1-100]
    ```
  - 更新 (-30) を適用した MinCount および MaxCount: MinCount = MaxCount = 70
  -

```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
queue1*      up   infinite     70   idle queue1-st-c5xlarge-[1-70]
```

混合ノードの容量変更

MinCount < MaxCount の場合、MinCount を量 N だけ増やすには (MaxCount は変更しないと仮定)、静的ノードの数を MinCount の新しい値 (old_MinCount + N) まで拡張するようにクラスターを設定します (<Queue/Name>-st-<ComputeResource/Name>-<old_MinCount + N>)。システムは新しい静的容量を満たすまで必要な数の Amazon EC2 インスタンスを起動し続けます。さらに、コンピューティングリソースの MaxCount 容量を保持するために、最後の N 個の動的ノードを削除するようにクラスター設定を更新します (<Queue/Name>-dy-<ComputeResource/Name>-[<MaxCount - old_MinCount - N>...<MaxCount - old_MinCount>])。システムは対応する数の Amazon EC2 インスタンスを終了します。

- 初期状態: MinCount = 100; MaxCount = 150
-
```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
queue1*      up   infinite     50  idle~ queue1-dy-c5xlarge-[1-50]
queue1*      up   infinite    100   idle queue1-st-c5xlarge-[1-100]
```
- 更新 (+30) を適用した MinCount : MinCount = 130 (MaxCount = 150)
-
```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
queue1*      up   infinite     20  idle~ queue1-dy-c5xlarge-[1-20]
queue1*      up   infinite    130   idle queue1-st-c5xlarge-[1-130]
```

MinCount < MaxCount の場合、MinCount と MaxCount を同じ量 N だけ増やすには、静的ノードの数を MinCount の新しい値 (old_MinCount + N) まで拡張するようにクラスターを設定します (<Queue/Name>-st-<ComputeResource/Name>-<old_MinCount + N>)。システムは新し

い静的容量を満たすまで必要な数の Amazon EC2 インスタンスを起動し続けます。さらに、動的ノードの数は変更せずに、新しい

MaxCount の値を保持します。

- 初期状態: MinCount = 100; MaxCount = 150
-
```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
queue1*      up   infinite     50  idle~ queue1-dy-c5xlarge-[1-50]
queue1*      up   infinite    100   idle queue1-st-c5xlarge-[1-100]
```
- 更新 (+30) を適用した MinCount : MinCount = 130 (MaxCount = 180)
-
```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
queue1*      up   infinite     20  idle~ queue1-dy-c5xlarge-[1-50]
queue1*      up   infinite    130   idle queue1-st-c5xlarge-[1-130]
```

MinCount < MaxCount の場合、MinCount を量 N だけ減らすには (MaxCount は変更しないと仮定)、最後の N 個の静的ノードを削除するようにクラスターを設定します (<Queue/Name>-st-<ComputeResource/Name>-[<old_MinCount - N>...<old_MinCount>)。システムは対応する数の Amazon EC2 インスタンスを終了します。さらに、コンピューティングリソースの MaxCount 容量を保持するために、動的ノードの数を拡張してギャップを埋めるようにクラスター設定を更新します (MaxCount - new_MinCount: <Queue/Name>-dy-<ComputeResource/Name>-[1..<MazCount - new_MinCount>])。この場合、これらは動的ノードであるため、新しいノードでスケジューラに保留中のジョブがない限り、新しい Amazon EC2 インスタンスは起動されません。

- 初期状態: MinCount = 100; MaxCount = 150
-
```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
queue1*      up   infinite     50  idle~ queue1-dy-c5xlarge-[1-50]
```

```
queue1*      up   infinite    100   idle queue1-st-c5xlarge-[1-100]
```

- 更新 (-30) を適用した MinCount : MinCount = 70 (MaxCount = 120)
-

```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
queue1*      up   infinite     80  idle~ queue1-dy-c5xlarge-[1-80]
queue1*      up   infinite     70   idle queue1-st-c5xlarge-[1-70]
```

MinCount < MaxCount の場合、MinCount と MaxCount を同じ量 N だけ減らすには、最後の N 個の静的ノードを削除するようにクラスターを設定します (<Queue/Name>-st-<ComputeResource/Name>-<old_MinCount - N>...<oldMinCount>])。システムは対応する数の Amazon EC2 インスタンスを終了します。

さらに、動的ノードの数は変更せずに、新しい MaxCount の値を保持します。

- 初期状態: MinCount = 100; MaxCount = 150
-

```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
queue1*      up   infinite     50  idle~ queue1-dy-c5xlarge-[1-50]
queue1*      up   infinite    100   idle queue1-st-c5xlarge-[1-100]
```

- 更新 (-30) を適用した MinCount : MinCount = 70 (MaxCount = 120)
-

```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
queue1*      up   infinite     80  idle~ queue1-dy-c5xlarge-[1-50]
queue1*      up   infinite     70   idle queue1-st-c5xlarge-[1-70]
```

MinCount < MaxCount の場合、MaxCount を量 N だけ減らすには (MinCount は変更しないと仮定)、最後の N 個の動的ノードを削除するようにクラスターを設定します (<Queue/Name>-dy-<ComputeResource/Name>-<old_MaxCount - N...<oldMaxCount>])。システムは対応する数の Amazon EC2 インスタンス (実行している場合) を終了します。静的ノードには影響がないと想定されます。

- 初期状態: MinCount = 100; MaxCount = 150
-
```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
queue1*      up   infinite     50  idle~ queue1-dy-c5xlarge-[1-50]
queue1*      up   infinite    100   idle queue1-st-c5xlarge-[1-100]
```
- 更新 (-30) を適用した MaxCount : MinCount = 100 (MaxCount = 120)
-
```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
queue1*      up   infinite     20  idle~ queue1-dy-c5xlarge-[1-20]
queue1*      up   infinite    100   idle queue1-st-c5xlarge-[1-100]
```

ジョブへの影響

ノードを削除して Amazon EC2 インスタンスを終了すると、削除したノードで実行していた sbatch ジョブは常に再びキューに入れられます。ただし、ジョブ要件を満たす他のノードがない場合を除きます。この場合、ジョブはステータス NODE_FAIL で失敗し、キューから消えるため、手動で再送信する必要があります。

クラスターのサイズ変更の更新を実行する予定がある場合は、予定した更新中に削除対象のノードでジョブが実行されないようにすることができます。これを行うには、ノードをメンテナンス中に削除するように設定します。メンテナンス中にノードを設定しても、ノードで既に実行しているジョブには最終的に影響しないことに注意してください。

予定したクラスターのサイズ変更の更新で、ノード qeueu-st-computeresource-[9-10] を削除するとします。次のコマンドを使用して Slurm 予約を作成できます。

```
sudo -i scontrol create reservation ReservationName=maint_for_update user=root
 starttime=now duration=infinite flags=maint,ignore_jobs nodes=qeueu-st-
computeresource-[9-10]
```

これにより、Slurm 予約が `maint_for_update` という名前でノード `qeueu-st-computeresource-[9-10]` に作成されます。予約を作成した時点から、ノード `qeueu-st-computeresource-[9-10]` では以降のジョブを実行できなくなります。予約では、ノード `qeueu-st-computeresource-[9-10]` へのジョブの割り当てを最終的には阻止できないことに注意してください。

Slurm 予約を設定しておいたノードのみがサイズ変更の更新中に削除された場合、クラスターのサイズ変更の更新後に、メンテナンス予約は自動的に削除されます。一方、Slurm 予約を作成しておいたノードがクラスターのサイズ変更の更新後も存続する場合は、次のコマンドを使用して、サイズ変更の更新の実行後にノードのメンテナンス予約を削除できます。

```
sudo -i scontrol delete ReservationName=maint_for_update
```

Slurm 予約の詳細については、こちらで公式の SchedMD ドキュメントを参照してください。

容量変更時のクラスター更新プロセス

スケジューラ設定を変更すると、クラスターの更新プロセス中に次の手順が実行されます。

- 停止 AWS ParallelCluster `clustermgtd` (`supervisorctl stop clustermgtd`)
- 更新された Slurm パーティション設定を AWS ParallelCluster 設定から生成する
- `slurmctld` を再起動する (Chef サービスレシピを使用して実行)
- `slurmctld` のステータスを確認する (`systemctl is-active --quiet slurmctld.service`)
- Slurm 設定を再ロードする (`scontrol reconfigure`)
- 起動する: `clustermgtd` (`supervisorctl start clustermgtd`)

での AWS Batch (**awsbatch**) スケジューラの使用 AWS ParallelCluster

AWS ParallelCluster はス AWS Batch ケジューラもサポートしています。以下のトピックでは、AWS Batchの使用方法について説明します。詳細については AWS Batch、「」を参照してくださいAWS Batch。ドキュメントについては、「AWS Batch IAM ユーザーガイド」を参照してください。

AWS ParallelCluster の CLI コマンド AWS Batch

スawsbatchケジューラを使用すると、 の AWS ParallelCluster AWS Batch CLI コマンドが AWS ParallelCluster ヘッドノードに自動的にインストールされます。CLI は AWS Batch API オペレーションを使用し、次のオペレーションを許可します。

- ジョブの送信と管理
- ジョブ、キュー、ホストのモニタリング
- 従来のスケジューラコマンドのミラーリング

> **Important**
>
> AWS ParallelCluster は の GPU ジョブをサポートしていません AWS Batch。詳細については、「GPU jobs」を参照してください。

この CLI は個別のパッケージとして配布されます。詳細については、「スケジューラのサポート」を参照してください。

トピック

- awsbsub
- awsbstat
- awsbout
- awsbkill
- awsbqueues
- awsbhosts

awsbsub

ジョブをクラスターのジョブキューに送信します。

```
awsbsub [-h] [-jn JOB_NAME] [-c CLUSTER] [-cf] [-w WORKING_DIR]
        [-pw PARENT_WORKING_DIR] [-if INPUT_FILE] [-p VCPUS] [-m MEMORY]
        [-e ENV] [-eb ENV_DENYLIST] [-r RETRY_ATTEMPTS] [-t TIMEOUT]
        [-n NODES] [-a ARRAY_SIZE] [-d DEPENDS_ON]
        [command] [arguments [arguments ...]]
```

> **Important**
>
> AWS ParallelCluster は の GPU ジョブをサポートしていません AWS Batch。詳細については、「GPU jobs」を参照してください。

位置引数

command

ジョブを送信するか (指定したコマンドがコンピューティングインスタンスで使用可能である必要があります)、転送するファイル名を指定します。「`--command-file`」も参照してください。

`arguments`

(オプション) コマンドまたはコマンドファイルの引数を指定します。

名前付き引数

`-jn` *`JOB_NAME`*, `--job-name` *`JOB_NAME`*

ジョブの名前を指定します。最初の文字はアルファベットまたは数字でなければなりません。ジョブ名には、アルファベット (大文字、小文字)、数字、ハイフン、アンダースコアを含めることができ、最大 128 文字まで使用可能です。

`-c` *`CLUSTER`*, `--cluster` *`CLUSTER`*

使用するクラスターを指定します。

`-cf, --command-file`

コマンドがコンピューティングインスタンスに転送されるファイルであることを示します。

デフォルト: False

`-w` *`WORKING_DIR`*, `--working-dir` *`WORKING_DIR`*

ジョブの作業ディレクトリとして使用するフォルダを指定します。作業ディレクトリが指定されない場合、ジョブは、ユーザーのホームディレクトリの `job-<AWS_BATCH_JOB_ID>` サブフォルダで実行されます。このパラメータ、または `--parent-working-dir` パラメータを使用できます。

-pw *PARENT_WORKING_DIR*, --parent-working-dir *PARENT_WORKING_DIR*

ジョブの作業ディレクトリの親フォルダを指定します。親の作業ディレクトリが指定されない場合、デフォルトはユーザーのホームディレクトリに設定されます。job-*<AWS_BATCH_JOB_ID>* という名前のサブフォルダが親の作業ディレクトリに作成されます。このパラメータ、または --working-dir パラメータを使用できます。

-if *INPUT_FILE*, --input-file *INPUT_FILE*

コンピューティングインスタンスに転送するファイル (ジョブの作業ディレクトリ内) を指定します。複数の入力ファイルのパラメータを指定できます。

-p *VCPUS*, --vcpus *VCPUS*

コンテナ用に予約する vCPU の数を指定します。–nodes を指定すると、ノードごとの vCPU の数が識別されます。

デフォルト: 1

-m *MEMORY*, --memory *MEMORY*

ジョブに送信するメモリのハード制限 (MiB 単位) を指定します。ここで指定したメモリ制限を超えようとすると、ジョブは強制終了されます。

デフォルト: 128

-e *ENV*, --env *ENV*

ジョブ環境にエクスポートする環境変数名のカンマ区切りリストを指定します。すべての環境変数をエクスポートするには、「all」を指定します。「all」の環境変数のリストには、–env-blacklist パラメータに一覧表示されている変数や、プレフィックスが PCLUSTER_* または AWS_* の変数は含まれません。

-eb *ENV_DENYLIST*, --env-blacklist *ENV_DENYLIST*

ジョブ環境にエクスポートしない環境変数名のカンマ区切りリストを指定します。HOME、PWD、USER、PATH、LD_LIBRARY_PATH、TERM、および TERMCAP はデフォルトでエクスポートされません。

-r *RETRY_ATTEMPTS*, --retry-attempts *RETRY_ATTEMPTS*

ジョブを RUNNABLE ステータスに移行する回数を指定します。1〜10 回の試行を指定できます。試行回数の設定値が 1 より大きい場合にジョブが失敗すると、RUNNABLE ステータスに変わるまで、指定された回数分、再試行します。

デフォルト: 1

-t *TIMEOUT*, --timeout *TIMEOUT*

ジョブが終了していない場合に がジョブ AWS Batch を終了するまでの時間 (ジョブ試行のstartedAtタイムスタンプから測定) を秒単位で指定します。タイムアウト値は 60 秒以上に指定する必要があります。

-n *NODES*, --nodes *NODES*

ジョブ用に予約するノード数を指定します。マルチノード並列送信が有効になるように、このパラメータに値を指定します。

> **Note**
>
> Scheduler/AwsBatchQueues/CapacityType パラメータが SPOT に設定されている場合、マルチノード並列ジョブはサポートされません。さらに、アカウントには AWSServiceRoleForEC2Spot のサービスにリンクされたロールが必要です。このロールは、次の AWS CLI コマンドを使用して作成できます。
>
> ```
> $ aws iam create-service-linked-role --aws-service-name spot.amazonaws.com
> ```
>
> 詳細については、「Linux インスタンス用 Amazon Elastic Compute Cloud ユーザーガイド」の「スポットインスタンスリクエスト向けのサービスにリンクされたロール」を参照してください。

-a *ARRAY_SIZE*, --array-size *ARRAY_SIZE*

配列のサイズを示します。2 から 10,000 までの値を指定できます。ジョブの配列プロパティを指定した場合は、配列ジョブになります。

-d *DEPENDS_ON*, --depends-on *DEPENDS_ON*

ジョブの依存関係のセミコロン区切りリストを指定します。ジョブは最大 20 個のジョブに依存します。配列ジョブのジョブ ID を指定せずに、SEQUENTIAL タイプの依存関係を指定できます。シーケンシャルな依存関係では、各子配列ジョブがインデックス 0 から開始して順番に完了します。また、配列ジョブのジョブ ID を使用して N_TO_N タイプの依存関係を指定することもできます。N_TO_N の依存関係では、このジョブの各インデックスの子は各依存関係の対応するインデックスの子が完了するまで待機してから開始されます。このパラメータの構文は、「jobId=*<string>*,type=*<string>*;...」です。

awsbstat

クラスターのジョブキューに送信されたジョブを表示します。

```
awsbstat [-h] [-c CLUSTER] [-s STATUS] [-e] [-d] [job_ids [job_ids ...]]
```

位置引数

job_ids

出力に表示するジョブ ID のスペース区切りリストを指定します。ジョブがジョブ配列の場合は、すべての子ジョブが表示されます。単一のジョブがリクエストされた場合は、詳細バージョンで表示されます。

名前付き引数

-c *CLUSTER*, --cluster *CLUSTER*

使用するクラスターを示します。

-s *STATUS*, --status *STATUS*

含めるジョブステータスのカンマ区切りリストを指定します。
デフォルトのジョブのステータスは「active」です。有効な値は、SUBMITTED、PENDING、RUNNABLE、STARTING、RUNNING、SUCCEEDED、FAILED、ALL です。

デフォルト値は、SUBMITTED、PENDING、RUNNABLE、STARTING、RUNNING です。

-e, --expand-children

子を含むジョブを拡張します (配列とマルチノードの並列ジョブのいずれも)。

デフォルト: False

-d, --details

ジョブの詳細を表示します。

デフォルト: False

awsbout

指定されたジョブの出力を表示します。

```
awsbout [-h] [-c CLUSTER] [-hd HEAD] [-t TAIL] [-s] [-sp STREAM_PERIOD] job_id
```

位置引数

job_id

ジョブ ID を指定します。

名前付き引数

-c *CLUSTER*, --cluster *CLUSTER*

使用するクラスターを示します。

-hd *HEAD*, --head *HEAD*

ジョブ出力の最初の *HEAD* 行を取得します。

-t *TAIL*, --tail *TAIL*

ジョブ出力の最後の <tail> 行を取得します。

-s, --stream

ジョブ出力を取得してから、追加の出力が生成されるのを待ちます。ジョブ出力の最新の <tail> 行から開始するには、この引数に –tail を指定します。

デフォルト: False

-sp *STREAM_PERIOD*, --stream-period *STREAM_PERIOD*

ストリーミング期間を設定します。

デフォルト: 5

awsbkill

クラスターに送信されたジョブをキャンセルし、終了します。

```
awsbkill [-h] [-c CLUSTER] [-r REASON] job_ids [job_ids ... ]
```

位置引数

job_ids

キャンセルまたは終了するジョブ ID のスペース区切りリストを指定します。

名前付き引数

-c *CLUSTER*, --cluster *CLUSTER*

使用するクラスターの名前を示します。

-r *REASON*, --reason *REASON*

キャンセル理由と合わせて、ジョブにアタッチするメッセージを示します。

デフォルト:「Terminated by the user」

awsbqueues

クラスターに関連付けられているジョブキューを表示します。

```
awsbqueues [-h] [-c CLUSTER] [-d] [job_queues [job_queues ... ]]
```

位置引数

job_queues

表示するキュー名のスペース区切りリストを指定します。単一のキューがリクエストされた場合は、詳細バージョンで表示されます。

名前付き引数

-c *CLUSTER*, --cluster *CLUSTER*

使用するクラスターの名前を指定します。

-d, --details

キューの詳細を表示するかどうかを示します。

デフォルト: False

awsbhosts

クラスターのコンピューティング環境に属するホストを表示します。

```
awsbhosts [-h] [-c CLUSTER] [-d] [instance_ids [instance_ids ... ]]
```

位置引数

instance_ids

インスタンス ID のスペース区切りリストを指定します。単一のインスタンスがリクエストされた場合は、詳細バージョンで表示されます。

名前付き引数

-c *CLUSTER*, --cluster *CLUSTER*

使用するクラスターの名前を指定します。

-d, --details

ホストの詳細を表示するかどうかを示します。

デフォルト: False

共有ストレージ

AWS ParallelCluster は、Amazon EBS、FSx for ONTAP、FSx for OpenZFS 共有ストレージボリューム、Amazon EFS and FSx for Lustre 共有ストレージファイルシステム、またはファイルキャッシュの使用をサポートしています。AWS Well-Architected フレームワークの信頼性の柱ガイダンスに従い、ボリュームとファイルシステムをバックアップすることをお勧めします。

HPC アプリケーション I/O 要件を満たすストレージシステムを選択します。特定のユースケースに基づいて、各ファイルシステムを最適化できます。詳細については、「storage options overview」を参照してください。

Amazon EBS ボリュームはヘッドノードにアタッチされ、NFS を介してコンピューティングノードと共有されます。このオプションはコスト効率が高いですが、ストレージのニーズがスケールするにつれて、パフォーマンスはヘッドノードのリソースによって異なります。クラスターに追加されるコ

ンピューティングノードの数が増え、スループットの需要が高まるにつれて、これがボトルネックになる可能性があります。

Amazon EFS ファイルシステムは、ストレージのニーズの変更に応じてスケールします。これらのファイルシステムは、さまざまなユースケースに合わせて設定できます。Amazon EFS ファイルシステムを使用して、並列化された、レイテンシーの影響を受けやすいアプリケーションをクラスター上で実行します。

FSx for Lustre ファイルシステムでは、最大数百ギガバイト/秒のスループット、数百万の IOPS、サブミリ秒のレイテンシーで大規模なデータセットを処理できます。FSx for Lustre ファイルシステムは、要求の厳しい高性能コンピューティング環境で使用します。

では`SharedStorage` セクション、外部ストレージまたは AWS ParallelCluster マネージドストレージを定義できます。

- 外部ストレージとは、ユーザーが管理する既存のボリュームまたはファイルシステムを指します。このストレージは作成または削除 AWS ParallelCluster されません。
- マネージドストレージとは、 が AWS ParallelCluster 作成し、削除できるボリュームまたはファイルシステムを指します。

外部ストレージ

クラスターの作成時または更新時に外部ストレージをクラスターにアタッチ AWS ParallelCluster するように を設定できます。同様に、クラスターが削除または更新されたときに、外部ストレージをクラスターからデタッチするように設定できます。データは保持され、クラスターのライフサイクル外の長期的な永続的共有ストレージとして使用できます。

> **Note**
>
> 3.8 AWS ParallelCluster より前のバージョンの では、外部管理のファイルシステムを にマウントすることはできません/home。バージョン 3.8 以降の AWS ParallelCluster では、外部管理ファイルシステムのマウントポイントとして /home を使用できます。の下にある MountDir パラメータの値/homeとして を指定/homeすることで、外部管理のファイルシステムを にマウントできます`SharedStorage` セクション。
> Amazon File Cache はシステムの /home ディレクトリとして使用するのに適していないため、現時点では /home へのマウントはサポートされていません。
> /home ディレクトリを `SharedStorage` セクションで指定すると、SharedStorageType 設定オプションが上書きされます。つまり、代わりに `SharedStorage` セクションの設定が使用されます。

外部ファイルシステムを /home ディレクトリにマウントすると、外部ストレージ上の既存のファイルを上書きすることなく、ヘッドノードの/home内容を外部ファイルシステム AWS ParallelCluster にコピーします。これには、クラスターのデフォルトユーザーの SSH キーの転送が含まれます (このキーが外部ファイルシステムにない場合)。詳細については、「AWS ParallelCluster 共有ストレージに関する考慮事項」を参照してください。

AWS ParallelCluster マネージドストレージ

AWS ParallelCluster マネージドストレージは、設定のデフォルトでクラスターのライフサイクルによって異なります。`SharedStorage DeletionPolicy` 設定パラメータは、デフォルトで `Delete` に設定されています。

デフォルトでは、次のいずれかが true の場合、 AWS ParallelCluster マネージドファイルシステムまたはボリュームとそのデータは削除されます。

- クラスターを削除した場合。
- マネージド共有ストレージ設定 `Name` を変更した場合。
- マネージド共有ストレージを設定から削除した場合。

`DeletionPolicy` を `Retain` に設定して、マネージド共有ファイルシステムまたはボリュームとデータを保持します。データの損失を避けるために、データを定期的にバックアップすることをお勧めします。AWS Backup を使用すると、すべてのストレージオプションのバックアップを一元管理できます。

構成設定でライフサイクルの依存関係を削除できます。詳細については、「AWS ParallelCluster マネージドストレージを外部ストレージに変換する」を参照してください。

共有ストレージのクォータの詳細については、「共有ストレージのクォータ」を参照してください。

共有ストレージと新しい AWS ParallelCluster バージョンへの切り替えの詳細については、「」を参照してくださいベストプラクティス: AWS ParallelCluster クラスターを新しいマイナーバージョンまたはパッチバージョンに移動する。

クラスターの作成時または更新時に外部ストレージをクラスターにアタッチ AWS ParallelCluster するように を設定できます。同様に、クラスターが削除または更新されたときに、外部ストレージをクラスターからデタッチするように設定できます。データは保持され、クラスターのライフサイクルとは関係なく、長期的な永続的共有ストレージソリューションに使用できます。

デフォルトでは、マネージドストレージはクラスターのライフサイクルに依存します。AWS ParallelCluster マネージドストレージを外部ストレージに変換する で説明されている構成設定で、この依存関係を削除できます。

特定の設定で、サポートされている各ストレージソリューションをユースケースに合わせて最適化できます。

共有ストレージクォータについては、「共有ストレージのクォータ」を参照してください。

共有ストレージと新しい AWS ParallelCluster バージョンへの切り替えの詳細については、「」を参照してくださいベストプラクティス: AWS ParallelCluster クラスターを新しいマイナーバージョンまたはパッチバージョンに移動する。

以下のトピックでは、 が AWS ParallelCluster サポートするストレージサービスごとに共有ストレージを設定する方法について説明します。

トピック

- Amazon Elastic Block Store
- Amazon Elastic File System
- Amazon FSx for Lustre
- FSx for ONTAP、FSx for OpenZFS、およびファイルキャッシュ共有ストレージの設定
- での共有ストレージの使用 AWS ParallelCluster
- 共有ストレージのクォータ

Amazon Elastic Block Store

クラスターのライフサイクルとは関係なく、既存の外部 Amazon EBS ボリュームを長期的な永続的ストレージに使用するには、`EbsSettings`/`VolumeId` を指定します。

`VolumeId` を指定しない場合、デフォルトでは、 AWS ParallelCluster はクラスターの作成時に `EbsSettings` からマネージド EBS ボリュームを作成します。また、 AWS ParallelCluster はクラスターが削除されたり、クラスター設定からボリュームが削除されたりすると、ボリュームとデータを削除します。

AWS ParallelCluster マネージド EBS ボリュームの場合、クラスターが削除されたとき、またはボリュームSnapshotがクラスター設定から削除されたときに、`EbsSettings`/ を使用して AWS ParallelCluster `Delete`、`Retain`、または ボリュームに`DeletionPolicy`指示できます。デフォルトで、`DeletionPolicy` は `Delete` に設定されています。

> **Warning**
>
> AWS ParallelCluster マネージド共有ストレージの場合、 `DeletionPolicy`は`Delete`デフォルトで に設定されます。
> つまり、以下のいずれかに該当する場合、マネージドボリュームとそのデータは削除されます。
>
> - クラスターを削除した場合。
> - マネージド共有ストレージの設定 `SharedStorage`/`Name` を変更した場合。
> - マネージド共有ストレージを設定から削除した場合。
>
> データの損失を防ぐために、スナップショットを使用して共有ストレージを定期的にバックアップすることをお勧めします。Amazon EBS スナップショットの詳細については、「Linux インスタンス用 Amazon Elastic Compute Cloud ユーザーガイド」の「Amazon EBS スナップショット」を参照してください。全体のデータバックアップを管理する方法については AWS のサービス、「 AWS Backup デベロッパーガイド」のAWS 「バックアップ」を参照してください。

Amazon Elastic File System

既存の外部 Amazon EFS ファイルシステムをクラスターライフサイクル外の長期永続的ストレージに使用するには、`EfsSettings`/ を指定します。デフォルトでは`FileSystemId`、クラスターの作成`EfsSettings`時に からマネージド Amazon EFS ファイルシステム AWS ParallelCluster を作成します。 AWS ParallelCluster また、クラスターが削除されたとき、またはファイルシステムがクラスター設定から削除されたときに、 はファイルシステムとデータを削除します。

AWS ParallelCluster マネージド Amazon EFS ファイルシステムの場合、/ `EfsSettings` を使用して`DeletionPolicy`、クラスターが削除されたとき`Delete`、またはファイルシステムがクラスター設定から削除されたときに、 または `Retain` AWS ParallelCluster に指示できます。デフォルトで、`DeletionPolicy` は `Delete` に設定されています。

> **Warning**
>
> AWS ParallelCluster マネージド共有ストレージの場合、 `DeletionPolicy`は`Delete`デフォルトで に設定されます。

つまり、以下のいずれかに該当する場合、マネージドファイルシステムとそのデータは削除されます。

- クラスターを削除した場合。
- マネージド共有ストレージの設定 `SharedStorage`/`Name` を変更した場合。
- マネージド共有ストレージを設定から削除した場合。

データ損失を防ぐために、共有ストレージを定期的にバックアップすることをお勧めします。個々の Amazon EFS ボリュームをバックアップする方法の詳細については、「Amazon Elastic File System ユーザーガイド」の「Backing up your Amazon EFS file systems」を参照してください。全体のデータバックアップを管理する方法については AWS のサービス、AWS Backup デベロッパーガイドのAWS 「バックアップ」を参照してください。

Amazon FSx for Lustre

既存の外部 FSx for Lustre ファイルシステムをクラスターライフサイクル外で長期的な永続ストレージに使用するには、`FsxLustreSettings`/`FileSystemId` を指定します。

`FsxLustreSettings` / を指定しない場合`FileSystemId`、デフォルトでは、 はクラスターの作成`FsxLustreSettings`時に からマネージド FSx for Lustre ファイルシステム AWS ParallelCluster を作成します。 AWS ParallelCluster また、 は、クラスターが削除されたとき、またはファイルシステムがクラスター設定から削除されたときに、ファイルシステムとデータを削除します。

AWS ParallelCluster マネージド FSx for Lustre ファイルシステムの場合、/ `FsxLustreSettings` を使用して、クラスターが削除されたとき、 AWS ParallelCluster `Delete`または`Retain`ファイルシステムがクラスター設定から削除されたときに、 または ファイルシステムに`DeletionPolicy`指示できます。デフォルトで、`DeletionPolicy` は `Delete` に設定されています。

> **Warning**
>
> AWS ParallelCluster マネージド共有ストレージの場合、 `DeletionPolicy`は`Delete`デフォルトで に設定されます。
> つまり、以下のいずれかに該当する場合、マネージドファイルシステムとそのデータは削除されます。
>
> - クラスターを削除した場合。
> - マネージド共有ストレージの設定 `SharedStorage`/`Name` を変更した場合。

- マネージド共有ストレージを設定から削除した場合。

データ損失を防ぐために、共有ストレージを定期的にバックアップすることをお勧めします。SharedStorage/FsxLustreSettings/AutomaticBackupRetentionDays/DailyAutomati を使用してクラスター内のバックアップを定義できます。全体のデータバックアップを管理する方法については AWS のサービス、「 AWS Backup デベロッパーガイド」のAWS「バックアップ」を参照してください。

FSx for ONTAP、FSx for OpenZFS、およびファイルキャッシュ共有ストレージの設定

FSx for ONTAP、FSx for OpenZFS、およびファイルキャッシュでは、FsxOntapSettings/VolumeId、FsxOpenZfsSettings/VolumeId、FileCacheSettings/FileC を使用して、クラスターの外部の既存ボリュームまたはファイルキャッシュのマウントを指定できます。

AWS ParallelCluster マネージド共有ストレージは、FSx for ONTAP、FSx for OpenZFS、およびファイルキャッシュではサポートされていません。

での共有ストレージの使用 AWS ParallelCluster

以下のセクションでは、共有ストレージに関する考慮事項や、マネージドストレージを外部ストレージに変換する方法など、 AWS ParallelCluster と共有ストレージの使用について説明します。

トピック

- AWS ParallelCluster 共有ストレージに関する考慮事項
- AWS ParallelCluster マネージドストレージを外部ストレージに変換する

AWS ParallelCluster 共有ストレージに関する考慮事項

AWS ParallelClusterで共有ストレージを操作する場合は、以下を考慮します。

- AWS Backup または別の方法でファイルシステムデータをバックアップし、すべてのストレージシステムのバックアップを管理します。
- 共有ストレージを追加するには、設定ファイルに共有ストレージセクションを追加し、クラスターを作成または更新します。

- 共有ストレージを削除するには、設定ファイルから共有ストレージセクションを削除し、クラスターを更新します。
- 既存の AWS ParallelCluster マネージド共有ストレージを新しいマネージドストレージに置き換えるには、/ SharedStorage の値を変更Nameし、クラスターを更新します。

> **Warning**
>
> デフォルトでは、新しいNameパラメータを使用してクラスターの更新を実行すると、既存の AWS ParallelCluster マネージドストレージとデータが削除されます。Name を変更して既存のマネージド共有ストレージデータを保持する必要がある場合は、DeletionPolicy を Retain に設定するか、クラスターを更新する前に、データをバックアップします。

- AWS ParallelCluster マネージドストレージデータをバックアップせず、 DeletionPolicyが の場合Delete、クラスターが削除されるか、クラスター設定からマネージドストレージが削除されてクラスターが更新されると、データは削除されます。
- AWS ParallelCluster マネージドストレージデータをバックアップせず、 DeletionPolicyが の場合Retain、クラスターが削除される前にファイルシステムがデタッチされ、外部ファイルシステムとして別のクラスターに再アタッチできます。データは保持されます。
- AWS ParallelCluster マネージドストレージがクラスター設定から削除され、DeletionPolicyである場合はRetain、クラスターデータを保存した外部ファイルシステムとしてクラスターに再アタッチできます。
- AWS ParallelCluster バージョン 3.4.0 以降では、// EfsSettings EncryptionInTransitIamAuthorizationおよび SharedStorage を設定することで、Amazon EFS ファイルシステムマウントのセキュリティを強化できます。
- 外部ファイルシステムを /home ディレクトリにマウントすると、 はヘッドノードの /home ディレクトリの内容を外部ファイルシステム AWS ParallelCluster にコピーします。外部ストレージの既存のファイルやディレクトリを上書きすることなく、/home ディレクトリ内の既存のデータがコピーされます。これには、クラスターのデフォルトユーザーの SSH キーも含まれます (このキーが外部ファイルシステムにまだ存在しない場合)。結果として、同じ外部ファイルシステムをそれぞれの /home ディレクトリにマウントする他のすべてのクラスターも、クラスターのデフォルトユーザーと同じ SSH キーを持つことになります。
- クラスターの /home ディレクトリに同じ外部ファイルシステムをマウントするマルチクラスター環境では、ヘッドノードで作成されたコンピューティングノードへのアクセスを許可する SSH キーは AWS ParallelCluster、最初のクラスターが外部ファイルシステムを /home にマウントしたときに 1 回だけ生成されます。他のすべてのクラスターは、同じ SSH キーを使用します。その結果、これらの共有クラスターのデフォルトユーザーの SSH キーを所有していれば、誰でもすべて

のクラスターにアクセスできます。すべてのコンピューティングノードは、最初に生成されたキーを使用した接続を許可します。

AWS ParallelCluster マネージドストレージを外部ストレージに変換する

AWS ParallelCluster マネージドストレージを外部ストレージに変換する方法について説明します。

手順は、以下の設定ファイルスニペットの例に基づいています。

```
...
  - MountDir: /fsx
    Name: fsx
    StorageType: FsxLustre
    FsxLustreSettings:
      StorageCapacity: 1200
      DeletionPolicy: Delete
...
```

AWS ParallelCluster マネージドストレージを外部ストレージに変換する

1. クラスター設定ファイルで、`DeletionPolicy` を `Retain` に設定します。

   ```
   ...
      - MountDir: /fsx
        Name: fsx
        StorageType: FsxLustre
        FsxLustreSettings:
          StorageCapacity: 1200
          DeletionPolicy: Retain
   ...
   ```

2. `DeletionPolicy` の変更を設定するには、以下のコマンドを実行します。

   ```
   pcluster update-cluster -n cluster-name -c cluster-config.yaml
   ```

3. クラスター設定ファイルから `SharedStorage` セクションを削除します。

   ```
   ...
   ...
   ```

4. マネージド SharedStorage を外部 SharedStorage に変更してクラスターからデタッチするには、以下のコマンドを実行します。

```
pcluster update-cluster -n cluster-name -c cluster-config.yaml
```

5. これで、共有ストレージは外部になり、クラスターからデタッチされました。
6. 外部ファイルシステムを元のクラスターまたは別のクラスターにアタッチするには、以下の手順に従います。

   a. FSx for Lustre ファイルシステム ID を取得します。

      i. を使用するには、次のコマンド AWS CLI を実行し、元のクラスターの名前を含む名前のファイルシステムを見つけ、ファイルシステム ID をメモします。

      ```
      aws fsx describe-file-systems
      ```

      ii. を使用するには AWS Management Console、ログインして https://console.aws.amazon.com/fsx/ に移動します。ファイルシステムのリストで、元のクラスター名を含む名前のファイルシステムを検索し、ファイルシステム ID を書き留めます。

   b. ファイルシステムのセキュリティグループのルールを更新して、ファイルシステムとクラスターサブネットとの間のアクセスを許可します。ファイルシステムのセキュリティグループ名と ID は Amazon FSx コンソールで確認できます。

      ヘッドノードとコンピューティングノードの IP CIDR 範囲またはプレフィックスとのインバウンドおよびアウトバウンド TCP トラフィックを許可するルールをファイルシステムセキュリティグループに追加します。インバウンドとアウトバウンドの TCP トラフィックに TCP ポート 988、1021、1022、1023 を指定します。

      詳細については、「AWS Command Line Interface バージョン 2 用ユーザーガイド」の「SharedStorage」/「FsxLustreSettings」/「FileSystemId」と「Amazon EC2 のセキュリティグループを作成、設定、および削除する」を参照してください。

   c. クラスター設定に SharedStorage セクションを追加します。

      ```
      ...
         - MountDir: /fsx
           Name: fsx-external
           StorageType: FsxLustre
           FsxLustreSettings:
      ```

```
        FileSystemId: fs-02e5b4b4abd62d51c
...
```

d. 外部共有ストレージをクラスターに追加するには、次のコマンドを実行します。

```
pcluster update-cluster -n cluster-name -c cluster-config.yaml
```

共有ストレージのクォータ

クラスター SharedStorage を設定して、既存の共有ファイルストレージをマウントし、次の表に示すクォータに基づいて新しい共有ファイルストレージを作成します。

各クラスターにマウントされたファイルストレージのクォータ

ファイル共有ストレージタイプ	AWS ParallelCluster マネージドストレージ	外部ストレージ	クォータネット合計
Amazon EBS	5	5	5
RAID	1	0	1
Amazon EFS	1	20	21
Amazon FSx †	1 FSx for Lustre	20	21

> **Note**
> このクォータの表は、 AWS ParallelCluster バージョン 3.2.0 で追加されています。

† は、既存の Amazon FSx for NetApp ONTAP、Amazon FSx for OpenZFS、およびファイルキャッシュシステムのマウント AWS ParallelCluster のみをサポートしています。FSx for ONTAP、FSx for OpenZFS、およびファイルキャッシュシステムの作成はサポートされていません。

> **Note**
> スケジューラ AWS Batch として を使用する場合、FSx for Lustre はクラスターヘッドノードでのみ使用できます。

ファイルキャッシュはス AWS Batch ケジューラをサポートしていません。

AWS ParallelCluster リソースとタグ付け

AWS ParallelCluster を使用すると、 AWS ParallelCluster タグを作成してリソースを追跡および管理できます。 AWS CloudFormation クラスター設定ファイルの で、作成してすべてのTags セクションクラスターリソースに伝達するタグを定義します。また、 AWS ParallelCluster が自動的に生成するタグを使用して、 リソースを追跡および管理することもできます。

クラスターを作成すると、クラスターとそのリソースには、このセクションで定義されている AWS ParallelCluster および AWS システムタグがタグ付けされます。

AWS ParallelCluster は、クラスターインスタンス、ボリューム、リソースにタグを適用します。クラスタースタックを識別するために、 はクラスターインスタンスに AWS システムタグ AWS CloudFormation を適用します。クラスターの Amazon EC2 起動テンプレートを識別するために、Amazon EC2 はインスタンスにシステムタグを適用します。これらのタグを使用して、 AWS ParallelCluster リソースを表示および管理できます。

> **Warning**
> システム機能への影響を避けるため、すべての AWS ParallelCluster タグは必須であり、変更しないでください。このため、 AWS システムタグを変更することはできません。

以下は、 AWS ParallelCluster リソースの AWS システムタグの例です。

```
"aws:cloudformation:stack-name"="clustername"
```

以下は、リソースに適用される AWS ParallelCluster タグの例です。

```
"parallelcluster:cluster-name"="clustername"
```

これらのタグは、 AWS Management Consoleの Amazon EC2 セクションで表示できます。

タグの表示

AWS Management Consoleの Amazon EC2 セクションでタグを表示するには、次の手順を実行します。

タグの表示

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) に移動します。
2. クラスタータグをすべて表示するには、ナビゲーションペインで [タグ] を選択します。
3. クラスタータグをインスタンス別に表示するには、ナビゲーションペインで [インスタンス] を選択します。
4. クラスターインスタンスを選択します。
5. インスタンス詳細の [タグを管理] タブを選択し、タグを表示します。
6. インスタンス詳細の [ストレージ] タブを選択します。
7. [ボリューム ID] を選択します。
8. [ボリューム] で、ボリュームを選択します。
9. ボリューム詳細の [タグ] タブを選択し、タグを表示します。

AWS ParallelCluster ヘッドノードインスタンスタグ

キー	タグ値
parallelcluster:cluster-name	*clustername*
Name	HeadNode
aws:ec2launchtemplate:id	*lt-1234567890abcdef0*
aws:ec2launchtemplate:version	*1*
parallelcluster:node-type	HeadNode
aws:cloudformation:stack-name	*clustername*
aws:cloudformation:logical-id	HeadNode
aws:cloudformation:stack-id	arn:aws:cloudformation: *region-id* :*ACCOUNTID* :stack/*clustername* /*1234abcd-12ab-12ab-12ab-1234567890abcdef0*
parallelcluster:version	*3.13.2*

AWS ParallelCluster ヘッドノードルートボリュームタグ

タグキー	タグ値
parallelcluster:cluster-name	*clustername*
parallelcluster:node-type	HeadNode
parallelcluster:version	*3.13.2*

AWS ParallelCluster コンピューティングノードインスタンスタグ

キー	タグ値
parallelcluster:cluster-name	*clustername*
parallelcluster:compute-resource-name	*compute-resource-name*
aws:ec2launchtemplate:id	*lt-1234567890abcdef0*
aws:ec2launchtemplate:version	*1*
parallelcluster:node-type	Compute
parallelcluster:queue-name	*queue-name*
parallelcluster:version	*3.13.2*

AWS ParallelCluster コンピューティングノードのルートボリュームタグ

タグキー	タグ値
parallelcluster:cluster-name	*clustername*
parallelcluster:compute-resource-name	*compute-resource-name*
parallelcluster:node-type	Compute

タグキー	タグ値
parallelcluster:queue-name	*queue-name*
parallelcluster:version	*3.13.2*

PCUI タグ

タグキー	タグ値
parallelcluster-ui	true

AWS ParallelCluster と ログのモニタリング

モニタリングは、 およびその他の AWS ParallelCluster AWS ソリューションの信頼性、可用性、パフォーマンスを維持する上で重要な部分です。 AWS には、監視 AWS ParallelCluster、問題発生時の報告、必要に応じて自動アクションを実行するための以下のモニタリングツールが用意されています。

- Amazon CloudWatch は、 AWS リソースと で実行しているアプリケーションを AWS リアルタイムでモニタリングします。メトリクスの収集と追跡、カスタマイズしたダッシュボードの作成、および指定したメトリクスが指定したしきい値に達したときに通知またはアクションを実行するアラームの設定を行うことができます。例えば、CloudWatch で Amazon EC2 インスタンスの CPU 使用率などのメトリクスを追跡し、必要に応じて新しいインスタンスを自動的に起動できます。詳細については、「Amazon CloudWatch ユーザーガイド」を参照してください。
- Amazon CloudWatch Logs では、Amazon EC2 インスタンス、CloudTrail、その他ソースから得たログファイルのモニタリング、保存、およびアクセスが可能です。CloudWatch Logs は、ログファイル内の情報をモニタリングし、特定のしきい値が満たされたときに通知します。高い耐久性を備えたストレージにログデータをアーカイブすることも可能です。詳細については、『Amazon CloudWatch Logs ユーザーガイド』を参照してください。
- AWS CloudTrailは、 AWS アカウント により、またはそのアカウントに代わって行われた API コールや関連イベントを取得し、指定した Amazon S3 バケットにログファイルを配信します。AWSを呼び出したユーザーとアカウント、呼び出し元の IP アドレス、および呼び出しの発生日時を特定できます。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。
- Amazon EventBridge は、アプリケーションをさまざまなイベントソースのデータに簡単に接続できるようにするサーバーレスイベントバスサービスです。EventBridge は、独自のアプリケーショ

ン、Software-as-a-Service (SaaS) アプリケーション、および AWS のサービスからリアルタイムデータのストリームを配信し、そのデータを Lambda などのターゲットにルーティングします。これにより、サービスで発生したイベントをモニタリングし、イベント駆動型アーキテクチャを構築できます。詳細については、「Amazon EventBridge ユーザーガイド」を参照してください。

トピック

- Amazon CloudWatch Logs との統合
- Amazon CloudWatch ダッシュボード
- クラスターメトリクス用の Amazon CloudWatch アラーム
- AWS ParallelCluster ログローテーションの設定
- pcluster CLI ログ
- Amazon EC2 コンソール出力ログ
- PCUI と AWS ParallelCluster ランタイムログを取得する
- ログの取得と保存

Amazon CloudWatch Logs との統合

CloudWatch Logs の詳細については、「Amazon CloudWatch Logs User Guide」(Amazon CloudWatch Logs ユーザーガイド) を参照してください。CloudWatch Logs の統合を設定するには、「`Monitoring`」セクションを参照してください。`append-config` を使用して CloudWatch 設定にカスタムログを追加する方法については、「Amazon CloudWatch ユーザーガイド」の「CloudWatch エージェント設定ファイル」を参照してください。

Amazon CloudWatch Logs のクラスターログ

ロググループは、クラスターごとに `/aws/parallelcluster/`*`cluster-name-<timestamp>`* という名前 (例: `/aws/parallelcluster/testCluster-202202050215`) で作成されます。ノード別の各ログ (またはパスに `*` が含まれている場合はログのセット) には、*`{hostname}`*`.`*`{instance_id}`*`.`*`{logIdentifier}`* という名前のログストリームが存在します。(例: `ip-172-31-10-46.i-02587cf29cc3048f3.nodewatcher`) ログデータは、すべてのクラスターインスタンス上で `root` として実行される CloudWatch エージェントによって CloudWatch に送信されます。

Amazon CloudWatch ダッシュボードは、クラスター作成時に作成されます。このダッシュボードでは、CloudWatch Logs に保存されているログを確認することができます。詳細については、「Amazon CloudWatch ダッシュボード」を参照してください。

このリストには、プラットフォーム、スケジューラー、ノードで使用できるログストリームの *logIdentifier* とパスが含まれています。

プラットフォーム、スケジューラー、ノードで使用できるログストリーム

[Platforms] (プラットフォーム)	スケジューラ	ノード	ログストリーム
Amazon redhat Ubuntu	awsbatch slurm	HeadNode	dcv-authenticator: `/var/log/parallelcluster/pcluster_dcv_authenticator.log` dcv-ext-authenticator: `/var/log/parallelcluster/pcluster_dcv_connect.log` dcv-agent: `/var/log/dcv/agent.*.log` dcv-xsession: `/var/log/dcv/dcv-xsession.*.log` dcv-server: `/var/log/dcv/server.log` dcv-session-launcher: `/var/log/dcv/sessionlauncher.log` Xdcv: `/var/log/dcv/Xdcv.*.log` cfn-init: `/var/log/cfn-init.log` chef-client: `/var/log/chef-client.log`
Amazon redhat Ubuntu	awsbatch slurm	ComputeNode HeadNode	cloud-init: `/var/log/cloud-init.log` supervisord: `/var/log/supervisord.log`
Amazon redhat	slurm	ComputeNode	cloud-init-output: `/var/log/cloud-init-output.log` computemgtd: `/var/log/parallelcluster/computemgtd`

[Platforms] (プラットフォーム)	スケジューラ	ノード	ログストリーム
Ubuntu			slurmd: `/var/log/slurmd.log` slurm_prolog_epilog: `/var/log/parallelcluster/slurm_prolog_epilog.log`
Amazon redhat Ubuntu	slurm	HeadNo	sssd: `/var/log/sssd/sssd.log` sssd_domain_default: `/var/log/sssd/sssd_default.log` pam_ssh_key_generator: `/var/log/parallelcluster/pam_ssh_key_generator.log` clusterstatusmgtd: `/var/log/parallelcluster/clusterstatusmgtd` clustermgtd: `/var/log/parallelcluster/clustermgtd` compute_console_output: `/var/log/parallelcluster/compute_console_output` slurm_resume: `/var/log/parallelcluster/slurm_resume.log` slurm_suspend: `/var/log/parallelcluster/slurm_suspend.log` slurmctld: `/var/log/slurmctld.log` slurm_fleet_status_manager: `/var/log/parallelcluster/slurm_fleet_status_manager.log`

[Platforms] (プラットフォーム)	スケジューラ	ノード	ログストリーム
Amazon redhat	awsbatch slurm	Computeet HeadNc	system-messages: `/var/log/messages`
Ubuntu	awsbatch slurm	Computeet HeadNc	syslog: `/var/log/syslog`

を使用するクラスター内のジョブは、`RUNNING`、、`SUCCEEDED`または の状態に達したジョブの出力を CloudWatch Logs `FAILED`に AWS Batch 保存します。ロググループは `/aws/batch/job`、ログストリーム名形式は *`jobDefinitionName`*`/default/`*`ecs_task_id`* です。デフォルトでは、このログは失効しませんが、保持期間を変更することもできます。詳細については、「Amazon CloudWatch Logs User Guide」(Amazon CloudWatch Logs ユーザーガイド) の「Change log data retention in CloudWatch Logs」(CloudWatch ログでのログデータ保管期間の変更) を参照してください。

Amazon CloudWatch Logs のビルドイメージログ

カスタムビルドイメージごとに `/aws/imagebuilder/ParallelClusterImage-`*`<image-id>`* という名前が付けられたロググループが作成されます。*`{pcluster-version}`*/1 という名前のユニークなログストリームには、ビルドイメージプロセスの出力が含まれます。

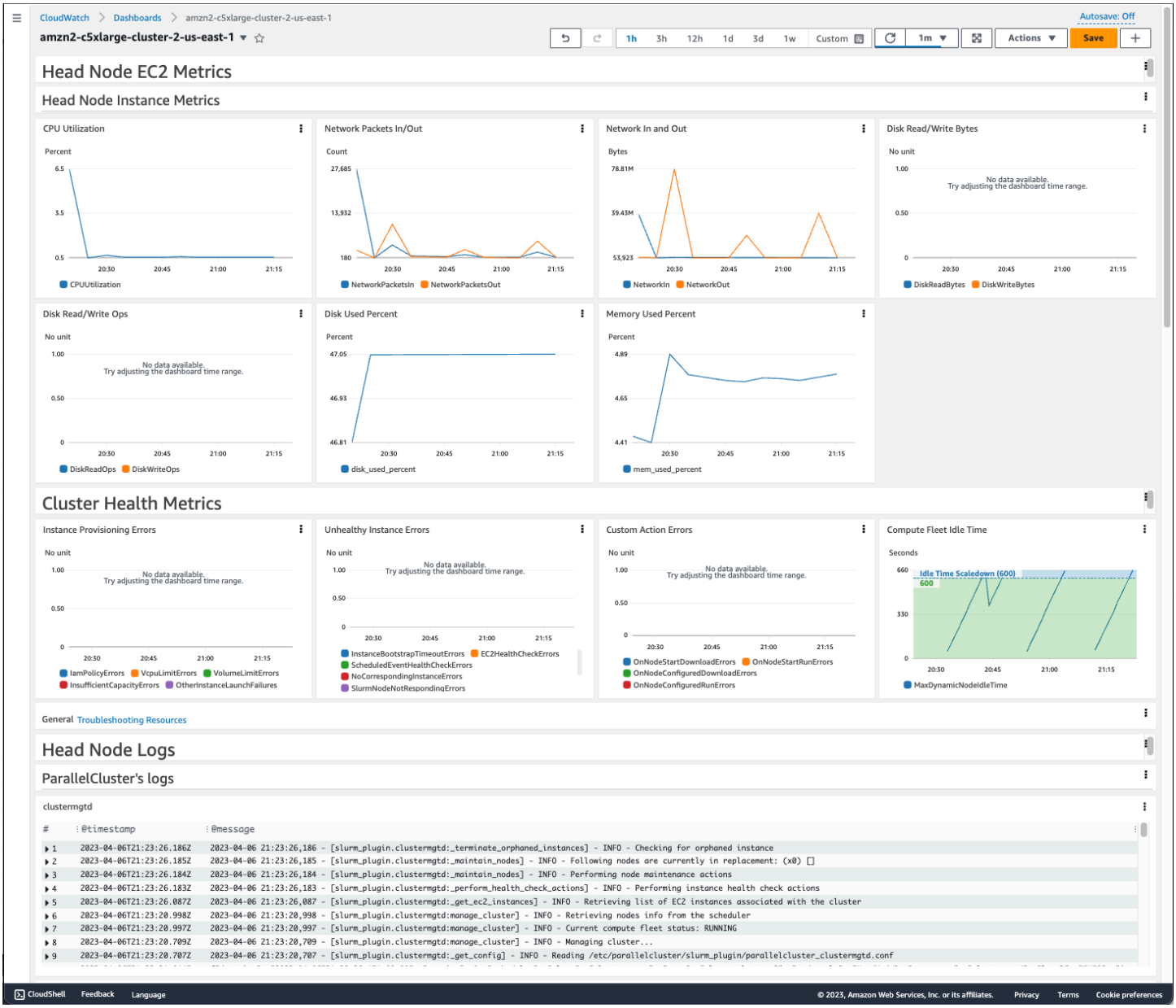
ログには、`pcluster` イメージコマンドを使用してアクセスできます。詳細については、「AWS ParallelCluster AMI のカスタマイズ」を参照してください。

Amazon CloudWatch ダッシュボード

Amazon CloudWatch ダッシュボードは、クラスター作成時に作成されます。これにより、クラスター内のノードのモニタリングや、Amazon CloudWatch Logs に保存されたログの確認が

容易になります。ダッシュボードの名前は *ClusterName*-*Region* です。*ClusterName* はクラスターの名前、*Region* はクラスターが存在する AWS リージョン です。ダッシュボードはコンソールから、または `https://console.aws.amazon.com/cloudwatch/home?region=`*Region*`#dashboards:name=`*ClusterName*`-`*Region* を開くことでアクセスできます。

次の図は、クラスターの CloudWatch ダッシュボード例を示しています。



ヘッドノードインスタンスメトリクス

ダッシュボードの最初のセクションには、ヘッドノードの Amazon EC2 メトリクスのグラフが表示されます。

クラスターに共有ストレージがある場合、次のセクションには共有ストレージメトリクスが表示されます。

クラスターヘルスメトリクス

クラスターが Slurm をスケジューリングに使用している場合、クラスターヘルスメトリクスグラフにはクラスターコンピュートノードエラーがリアルタイムで表示されます。詳細については、「クラスターヘルスメトリクスのトラブルシューティング」を参照してください。クラスターのヘルスメトリクスは、 AWS ParallelCluster バージョン 3.6.0 からダッシュボードに追加されます。

ヘッドノードログ

最後のセクションでは、ヘッドノードログを AWS ParallelCluster、 のログ、スケジューラのログ、Amazon DCV 統合ログ、およびシステムのログ別にグループ化して一覧表示します。

CloudWatch メトリクスの操作方法の詳細については、「Amazon CloudWatch User Guide」(Amazon CloudWatch ユーザーガイド) の「Using Amazon CloudWatch dashboards」(Amazon CloudWatch ダッシュボードの使用) を参照してください。

Amazon CloudWatch ダッシュボードを作成したくない場合は、`Monitoring`/`Dashboards`/`CloudWatch`/`Enabled` を `false` に設定することでダッシュボードをオフにできます。

> **Note**
>
> Amazon CloudWatch ダッシュボードの作成を無効にすると、クラスターの Amazon CloudWatch `disk_used_percent` および `memory_used_percent` アラームも無効になります。詳細については、「クラスターメトリクス用の Amazon CloudWatch アラーム」を参照してください。
>
> `disk_used_percent` および `memory_used_percent`アラームは、 AWS ParallelCluster バージョン 3.6 から追加されます。

クラスターメトリクス用の Amazon CloudWatch アラーム

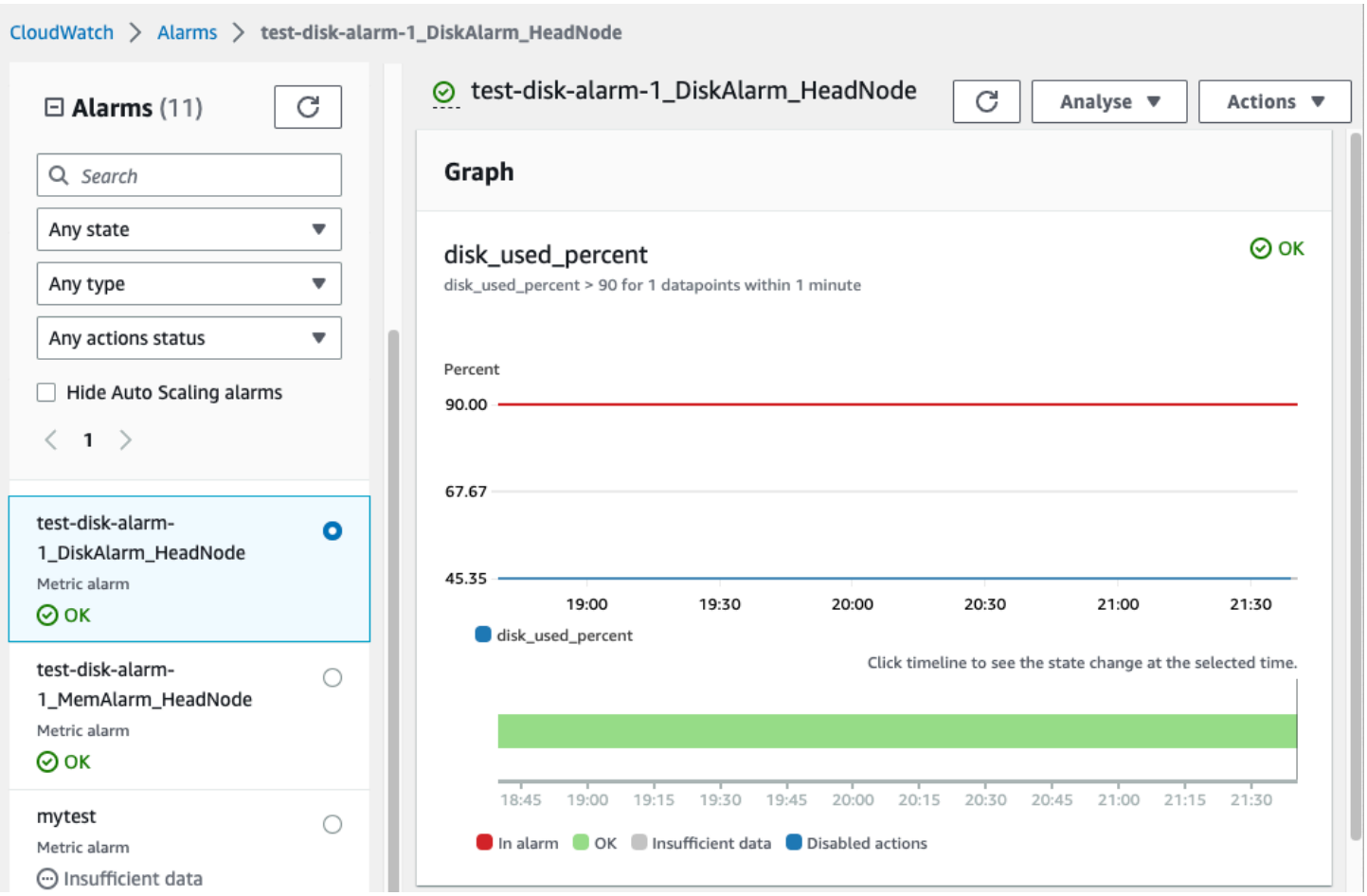
AWS ParallelCluster バージョン 3.6 以降では、ヘッドノードをモニタリングするための Amazon CloudWatch アラームを使用してクラスターを設定できます。1 つのアラームがルートボリューム `disk_used_percent` を監視します。もう 1 つのアラームは `mem_used_percent` メトリクスを監視します。詳細については、Amazon CloudWatch ユーザーガイドの「CloudWatch エージェントにより収集されるメトリクス」を参照してください。

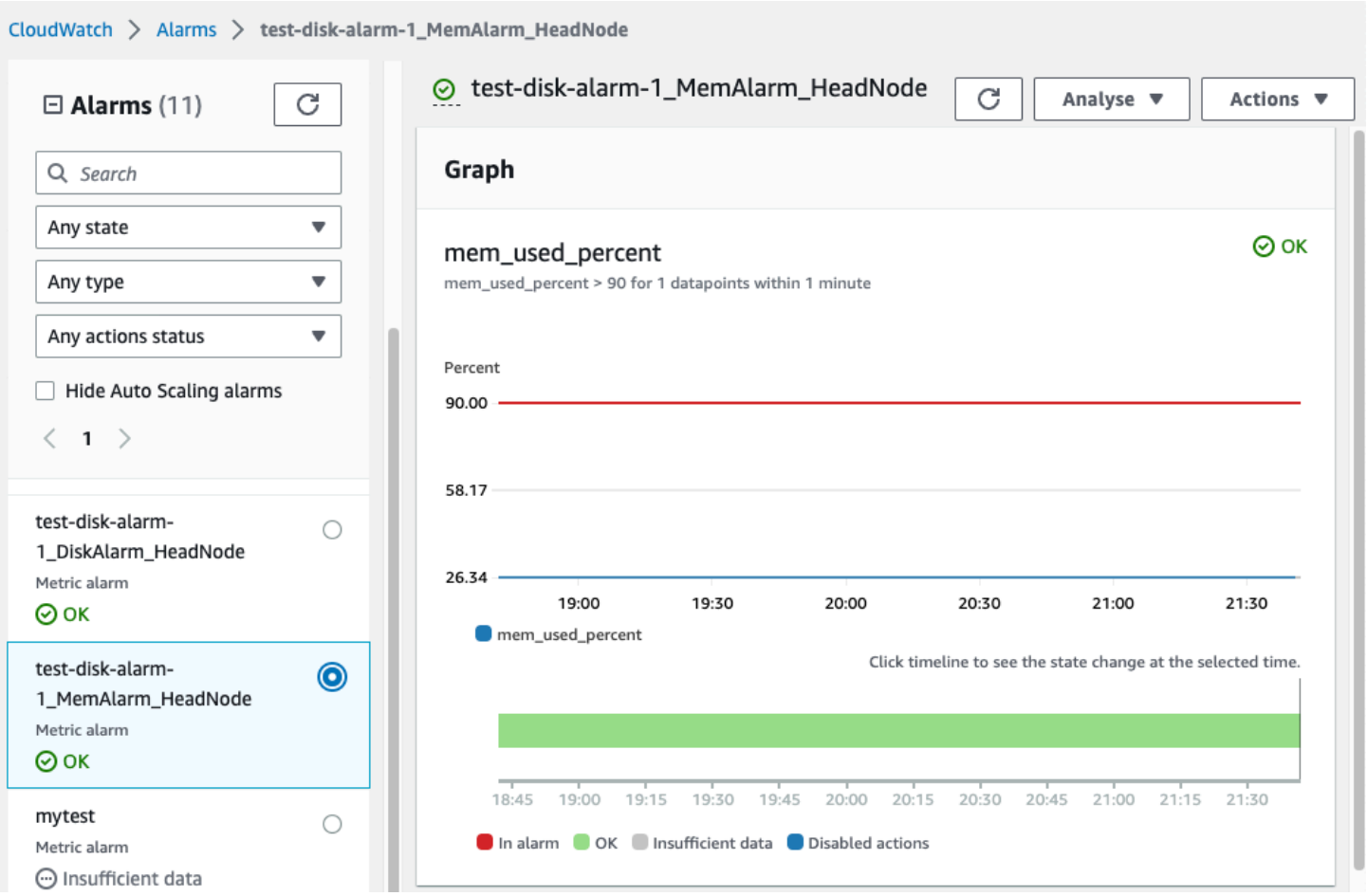
アラームの名前は以下のとおりです。

- *cluster-name*_DiskAlarm_HeadNode
- *cluster-name*_MemAlarm_HeadNode

cluster-name は、クラスターの名前です。

ナビゲーションペインで [アラーム] を選択して、CloudWatch コンソールのアラームにアクセスします。以下の画像は、クラスターのディスク使用量アラームとメモリ使用量アラームを示しています。





ディスク使用量アラームは、1 分以内に 1 つのデータポイントのディスク使用率が 90% を超えると ALARM の状態になります。

メモリ使用量アラームは、1 分以内に 1 つのデータポイントのメモリ使用率が 90% を超えると ALARM の状態になります。

> **Note**
>
> AWS ParallelCluster では、デフォルトではアラームアクションは設定されません。通知の送信など、アラームアクションの設定方法については、「アラームアクション」を参照してください。Amazon CloudWatch アラームの使用の詳細については、「Amazon CloudWatch ユーザーガイド」の「Amazon CloudWatch アラームの使用」を参照してください。

これらの Amazon CloudWatch アラームを作成したくない場合は、クラスター設定で `Monitoring`/`Dashboards`/`CloudWatch`/`Enabled` を `false` に設定して非アクティブ化します。

これにより、Amazon CloudWatch ダッシュボードの作成も無効になります。詳細については、「Amazon CloudWatch ダッシュボード」を参照してください。

> **Note**
>
> Amazon CloudWatch ダッシュボードの作成を非アクティブ化すると、クラスターの Amazon CloudWatch `disk_used_percent` および `memory_used_percent` アラームも非アクティブ化されます。

AWS ParallelCluster ログローテーションの設定

AWS ParallelCluster ログローテーション設定は `/etc/logrotate.d/parallelcluster_*_log_rotation` ファイルにあります。設定したログがローテーションされると、現在のログコンテンツは単一のバックアップに保存され、空になったログはログ記録を再開します。

設定したログごとに 1 つのバックアップだけが保持されます。

AWS ParallelCluster は、50 MB のサイズに達したときにローテーションするように、急成長するログを設定します。急速に増大するログはスケーリングおよび Slurm と関連し、`/var/log/parallelcluster/clustermgtd`、`/var/log/parallelcluster/slurm_resume.log`、`/var/log/slurmctld.log` が含まれます。

AWS ParallelCluster は、サイズが 10 MB に達したときにローテーションするように、成長の遅いログを設定します。

CloudFormation ログ記録を有効にした状態で、クラスター設定の `Logs/CloudWatch/RetentionInDays` 設定で定義された日数だけ保持されている以前のログを表示できます。`RetentionInDays` 設定を確認して、ユースケースに合わせて日数を増やす必要があるかどうかを確認してください。

AWS ParallelCluster は、次のログを設定およびローテーションします。

ヘッドノードログ

```
/var/log/cloud-init.log
/var/log/supervisord.log
/var/log/cfn-init.log
/var/log/chef-client.log
/var/log/dcv/server.log
```

```
/var/log/dcv/sessionlauncher.log
/var/log/dcv/agent.*.log
/var/log/dcv/dcv-xsession.*.log
/var/log/dcv/Xdcv.*.log
/var/log/parallelcluster/pam_ssh_key_generator.log
/var/log/parallelcluster/clustermgtd
/var/log/parallelcluster/clusterstatusmgtd
/var/log/parallelcluster/slurm_fleet_status_manager.log
/var/log/parallelcluster/slurm_resume.log
/var/log/parallelcluster/slurm_suspend.log
/var/log/slurmctld.log
/var/log/slurmdbd.log
/var/log/parallelcluster/compute_console_output.log
```

コンピューティングノードログ

```
/var/log/cloud-init.log
/var/log/supervisord.log
/var/log/cloud-init-output.log
/var/log/parallelcluster/computemgtd
/var/log/slurmd.log
```

ログインノードログ

```
/var/log/cloud-init.log
/var/log/cloud-init.log
/var/log/cloud-init-output.log
/var/log/supervisord.log
/var/log/parallelcluster/pam_ssh_key_generator.log
```

`pcluster` CLI ログ

`pcluster` CLI はコマンドのログを `/home/user/.parallelcluster/` 内の `pcluster.log.#` ファイルに書き込みます。

各コマンドのログには、通常、入力されたコマンド、コマンドの作成に使用された CLI API バージョンのコピー、応答、および情報とエラーメッセージがそれぞれ含まれます。create および build コマンドの場合、ログには設定ファイル、設定ファイルの検証操作、CloudFormation テンプレート、スタックコマンドも含まれます。

これらのログを使用して、エラー、入力、バージョン、および `pcluster` CLI コマンドを確認できます。また、コマンドが実行されたときの記録としても役立ちます。

Amazon EC2 コンソール出力ログ

は、静的コンピューティングノードインスタンスが予期せず終了したこと AWS ParallelCluster を検出すると、一定期間が経過した後に、終了したノードインスタンスから Amazon EC2 コンソール出力を取得しようとします。これにより、コンピューティングノードが Amazon CloudWatch と通信できなかった場合でも、ノードが終了した理由に関する有用なトラブルシューティング情報をコンソール出力から取得できます。このコンソール出力はヘッドノードの `/var/log/parallelcluster/compute_console_output` ログに記録されます。Amazon EC2 コンソール出力の詳細については、「Linux インスタンス用 Amazon EC2 ユーザーガイド」の「インスタンスコンソール出力」を参照してください。

デフォルトでは、 は終了したノードのサンプルサブセットから AWS ParallelCluster のみコンソール出力を取得します。これにより、多数の終了による複数のコンソール出力要求でクラスターヘッドノードが圧倒されるのを防ぐことができます。デフォルトでは、 は終了検出からコンソール出力の取得まで 5 分 AWS ParallelCluster 待機し、Amazon EC2 にノードから最終的なコンソール出力を取得する時間を与えます。

ヘッドノード上の `/etc/parallelcluster/slurm_plugin/parallelcluster_clustermgtd.conf` ファイルで、サンプルサイズと待機時間のパラメータ値を編集できます。

この機能は AWS ParallelCluster バージョン 3.5.0 で追加されています。

Amazon EC2 コンソール出力パラメータ

ヘッドノードの `/etc/parallelcluster/slurm_plugin/parallelcluster_clustermgtd.conf` ファイルで、以下の EC2 コンソール出力パラメータの値を編集できます。

`compute_console_logging_enabled`

コンソール出力ログの収集を無効にするには、`compute_console_logging_enabled` を `false` に設定します。デフォルトは `true` です。

このパラメータは、コンピューティングフリートを停止せずにいつでも更新することができます。

`compute_console_logging_max_sample_size`

`compute_console_logging_max_sample_size` は、予期しない終了を検出するたびに がコンソール出力を AWS ParallelCluster 収集するコンピューティングノードの最大数を設定します。この

値が より小さい場合1、 は終了したすべてのノードからコンソール出力 AWS ParallelCluster を取得します。デフォルト値は 1 です。

このパラメータは、コンピューティングフリートを停止せずにいつでも更新することができます。

compute_console_wait_time

compute_console_wait_time は、ノード障害を検出してからそのノードからコンソール出力を収集するまで AWS ParallelCluster 待機する時間を秒単位で設定します。終了したノードから最終出力を収集するためにより多くの時間が Amazon EC2 に必要だと判断した場合は、待機時間を増やすことができます。デフォルト値は 300 秒 (5 分) です。

このパラメータは、コンピューティングフリートを停止せずにいつでも更新することができます。

PCUI と AWS ParallelCluster ランタイムログを取得する

トラブルシューティングのために PCUI と AWS ParallelCluster ランタイムログを取得する方法について説明します。まず、関連する PCUI および AWS ParallelCluster スタックの名前を見つけます。スタック名を使用してインストールロググループを探します。終了するには、ログをエクスポートします。これらのログは AWS ParallelCluster ランタイムに固有のものです。クラスターログの場合には、「ログの取得と保存」を参照してください。

前提条件

- AWS CLI がインストールされます。
- PCUI AWS アカウント がオンになっている で AWS CLI コマンドを実行するための認証情報があります。
- PCUI がオン AWS アカウント になっている で Amazon CloudWatch コンソールにアクセスできます。

ステップ 1: 関連するスタックのスタック名を見つける

以下の例では、赤く強調表示されたテキストを実際の値に置き換えます。

PCUI をインストール AWS リージョン した を使用して、スタックを一覧表示します。

```
$ aws cloudformation list-stacks --region aws-region-id
```

以下のスタックのスタック名を書き留めておきます。

- アカウントに PCUI をデプロイしたスタックの名前。PCUI をインストールしたときに入力した名前です (`pcluster-ui` など)。
- 入力した AWS ParallelCluster スタック名のプレフィックスが付いたスタック。例: `pcluster-ui-ParallelClusterApi-ABCD1234EFGH`。

ステップ 2: ロググループを検索する

次の例に示すように、PCUI スタックのロググループを一覧表示します。

```
$ aws cloudformation describe-stack-resources \
   --region aws-region-id \
   --stack-name pcluster-ui \
   --query "StackResources[?ResourceType == 'AWS::Logs::LogGroup' &&
 (LogicalResourceId == 'ApiGatewayAccessLog' || LogicalResourceId ==
 'ParallelClusterUILambdaLogGroup')].PhysicalResourceId" \
   --output text
```

次の例に示すように、 AWS ParallelCluster API スタックのロググループを一覧表示します。

```
$ aws cloudformation describe-stack-resources \
   --region aws-region-id \
   --stack-name pcluster-ui-ParallelCluster-Api-ABCD1234EFGH \
   --query "StackResources[?ResourceType == 'AWS::Logs::LogGroup' && LogicalResourceId
 == 'ParallelClusterFunctionLogGroup'].PhysicalResourceId" \
   --output text
```

後のステップで使用するために、ロググループのリストを書き留めます。

ステップ 3: ログをエクスポートする

ログを収集してエクスポートするには、次の手順に従います。

1. にログインし AWS Management Console、PCUI がオンになっている の Amazon CloudWatch AWS アカウント コンソールに移動します。
2. ナビゲーションペインで、[ログ]、[ログインサイト] を選択します。
3. 前のステップでリストされたすべてのロググループを選択します。
4. 12 時間などの時間範囲を選択します。
5. 次のクエリを実行します。

```
$ fields @timestamp, @message
| sort @timestamp desc
| limit 10000
```

6. [結果のエクスポート]、[テーブルをダウンロード (JSON)] を選択します。

ログの取得と保存

AWS ParallelCluster は、HeadNode および Compute インスタンスとストレージの Amazon EC2 メトリクスを作成します。CloudWatch コンソールのカスタムダッシュボードでメトリクスを表示できます。 AWS ParallelCluster また、 はロググループにクラスター CloudWatch ログストリームを作成します。これらのログは、CloudWatch コンソールのカスタムダッシュボードまたはロググループで表示できます。モニタリングクラスター設定セクションでは、クラスターの CloudWatch ログとダッシュボードを変更する方法について説明されています。詳細については、「Amazon CloudWatch Logs との統合」および「Amazon CloudWatch ダッシュボード」を参照してください。

ログは問題を解決するための有用なリソースです。障害が発生したクラスターを削除する場合は、まずクラスターログのアーカイブを作成すると役立つことがあります。ログのアーカイブ の手順に従ってアーカイブを作成します。

トピック

- CloudWatch ではクラスターログを使用できません
- ログのアーカイブ
- 保存されたログ
- 終了したノードログ

CloudWatch ではクラスターログを使用できません

CloudWatch でクラスターログを使用できない場合は、設定にカスタムログを追加するときに AWS ParallelCluster CloudWatch ログ設定が上書きされていないことを確認してください。

CloudWatch 設定にカスタムログを追加するには、取得して上書きするのではなく、必ず設定に追加してください。`fetch-config` および `append-config` の詳細については、「CloudWatch ユーザーガイド」の「CloudWatch エージェント設定ファイル」を参照してください。

AWS ParallelCluster CloudWatch ログ設定を復元するには、 AWS ParallelCluster ノード内で次のコマンドを実行します。

```
$ PLATFORM="$(ohai platform | jq -r ".[]")"
LOG_GROUP_NAME="$(cat /etc/chef/dna.json | jq -r ".cluster.log_group_name")"
SCHEDULER="$(cat /etc/chef/dna.json | jq -r ".cluster.scheduler")"
NODE_ROLE="$(cat /etc/chef/dna.json | jq -r ".cluster.node_type")"
CONFIG_DATA_PATH="/usr/local/etc/cloudwatch_agent_config.json"
/opt/parallelcluster/pyenv/versions/cookbook_virtualenv/bin/python /usr/local/bin/
write_cloudwatch_agent_json.py --platform $PLATFORM --config $CONFIG_DATA_PATH --log-
group $LOG_GROUP_NAME --scheduler $SCHEDULER --node-role $NODE_ROLE
/opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2
 -c file:/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json -s
```

ログのアーカイブ

ログは S3 またはローカルファイルにアーカイブできます (`--output-file` パラメータに依存します)。

> **Note**
>
> AWS ParallelCluster 3.12.0 以降では、ログをデフォルトの AWS ParallelCluster バケットにエクスポートできます。この場合、バケットのアクセス許可を設定する必要はありません。

> **Note**
>
> CloudWatch へのアクセスを許可するため、Amazon S3 バケットポリシーにアクセス許可を追加しします。詳細については、「CloudWatch Logs ユーザーガイド」の「Amazon S3 バケットのアクセス許可を設定する」を参照してください。

```
$ pcluster export-cluster-logs --cluster-name mycluster --region eu-west-1 \
  --bucket bucketname --bucket-prefix logs
{
  "url": "https://bucketname.s3.eu-west-1.amazonaws.com/export-log/mycluster-
logs-202109071136.tar.gz?..."
}

# use the --output-file parameter to save the logs locally
$ pcluster export-cluster-logs --cluster-name mycluster --region eu-west-1 \
  --bucket bucketname --bucket-prefix logs --output-file /tmp/archive.tar.gz
{
```

```
  "path": "/tmp/archive.tar.gz"
}
```

アーカイブには、設定または export-cluster-logs コマンドのパラメータで明示的に指定されていない限り、過去 14 日間のヘッドノードとコンピューティングノードからの Amazon CloudWatch Logs ストリームと AWS CloudFormation スタックイベントが含まれます。クラスター内のノード数や CloudWatch Logs で利用できるログストリームの数によっては、コマンドの実行に時間がかかる場合があります。利用できるすべてのログストリームについての詳細は、「Amazon CloudWatch Logs との統合」を参照してください。

保存されたログ

バージョン 3.0.0 以降、クラスターが削除されると、 はデフォルトで CloudWatch Logs AWS ParallelCluster を保持します。クラスターを削除してそのログを保存したい場合は、クラスター設定で Monitoring/Logs/CloudWatch/DeletionPolicy が Delete に設定されていないことを確認してください。そうでない場合は、このフィールドの値を Retain に変更して pcluster update-cluster コマンドを実行します。その後、pcluster delete-cluster --cluster-name *<cluster_name>* を実行してクラスターを削除しますが、Amazon CloudWatch に保存されているロググループを保持します。

終了したノードログ

静的コンピューティングノードが予期せず終了し、CloudWatch にログがない場合、 AWS ParallelCluster がそのコンピューティングノードのコンソール出力をヘッドノードに/var/log/parallelcluster/compute_console_outputログに記録しているかどうかを確認します。詳細については、「デバッグ用のキーログ」を参照してください。

/var/log/parallelcluster/compute_console_output ログが利用できない場合、またはノードの出力が含まれていない場合は、 を使用して AWS CLI 、障害が発生したノードからコンソール出力を取得します。クラスターヘッドノードにログインし、障害が発生したノード instance-id を /var/log/parallelcluster/slurm_resume.log ファイルから取得します。

コンソール出力を取得するには、次のコマンドを instance-id と共に使用します。

```
$ aws ec2 get-console-output --instance-id i-abcdef01234567890
```

動的コンピューティングノードが起動後に自動的に終了し、CloudWatch にそのログがない場合は、クラスタースケーリングアクションを有効にするジョブを送信します。インスタンスに障害が発生するのを待ち、インスタンスコンソールログを取得します。

クラスターヘッドノードにログインし、`/var/log/parallelcluster/slurm_resume.log` ファイルからコンピューティングノード `instance-id` を取得します。

インスタンスコンソールログを取得するには、次のコマンドを使用します。

```
$ aws ec2 get-console-output --instance-id i-abcdef01234567890
```

コンソール出力ログは、コンピューティングノードログが利用できない場合にコンピューティングノード障害の根本原因をデバッグするのに役立ちます。

AWS CloudFormation カスタムリソース

AWS ParallelCluster バージョン 3.6.0 以降では、 AWS CloudFormation スタックで AWS ParallelCluster CloudFormation カスタムリソースを使用できます。カスタムリソースはホスト AWS ParallelCluster されたスタックです。これにより、CloudFormation を使用してクラスターを設定および管理できます。例えば、CloudFormation スタックでネットワーク、共有ストレージ、セキュリティグループインフラストラクチャなどのクラスター外部リソースを設定できます。さらに、CloudFormation インフラストラクチャをコードパイプラインとして使用してクラスターを管理できます。

以下を実行して、CloudFormation テンプレートに AWS ParallelCluster カスタムリソースを追加します。

1. によって所有およびホストされているカスタムリソースプロバイダースタックを追加します AWS ParallelCluster。
2. CloudFormation テンプレートのプロバイダースタックをカスタムリソースとして参照します。

カスタムリソースプロバイダースタックは、CloudFormation のリクエストを処理して応答します。例えば、CloudFormation スタックをデプロイする際は、クラスターの設定と作成も行います。クラスターを更新するには、CloudFormation スタックを更新します。スタックを削除すると、クラスターも削除されます。CloudFormation カスタムリソースの詳細については、「AWS CloudFormation ユーザーガイド」の「カスタムリソース」を参照してください。

> **Warning**
>
> CloudFormation は、カスタムリソースのドリフトを検出しません。CloudFormation は、クラスター設定の更新およびクラスターの削除にのみ使用してください。

`pcluster` CLI または AWS ParallelCluster UI を使用してクラスターの状態をモニタリングしたり、コンピューティングフリートを更新したりできますが、クラスター設定の更新やクラスターの削除には使用しないでください。

> **Note**
>
> 間違って削除しないようにするため、スタックに削除保護を追加することをお勧めします。

によってホストされるプロバイダースタック AWS ParallelCluster

カスタムリソースプロバイダースタックは、次の CloudFormation テンプレートスニペットに示すようにフォーマットされています。

```
PclusterClusterProvider:
  Type: AWS::CloudFormation::Stack
  Properties:
    Parameters:
      CustomLambdaRole: # (Optional) RoleARN to override default
      AdditionalIamPolicies: # (Optional) comma-separated list of IAM policies to add
    TemplateURL: !Sub
      - https://${AWS::Region}-aws-parallelcluster.s3.${AWS::Region}.${AWS::URLSuffix}/
parallelcluster/${Version}/templates/custom_resource/cluster.yaml
      - { Version: 3.13.2 }
```

プロパティ:

パラメータ :

CustomLambdaRole (オプション):

クラスターの作成および管理を行う AWS Lambda を実行するためのアクセス許可を持つカスタムロール。デフォルトでは、ロールは「AWS ParallelCluster documentation」でデフォルトで定義されているのと同じポリシーを使用します。

AdditionalIamPolicies (オプション):

Lambda が使用するロールに追加する IAM ポリシーの Amazon リソースネーム (ARN) のカンマ区切りリスト。これは `CustomLambdaRole` が指定されていない場合にのみ使用され、空白のままでも構いません。

ヘッドノード、コンピューティングノード、または Amazon S3 バケットへのアクセスに追加のポリシーが必要な場合、`CustomLambdaRole` または `AdditionalIamPolicy` プロパティにポリシーを追加します。

追加のポリシーをヘッドノードにアタッチする必要がある場合は、それらのポリシーをヘッドノードに関連付けられた IAM ロールにアタッチまたはデタッチするために必要なアクセス許可も付与する必要があります。具体的には、「iam:AttachRolePolicy」および「iam:DetachRolePolicy」のアクセス許可 (またはマネージドポリシーの同等のもの) を、ヘッドノードで使用される IAM ロールにアタッチする必要があります。詳細については、「AWS ParallelCluster IAM リソースを管理するためのユーザーポリシーの例」を参照してください。

デフォルトポリシーの詳細については、「AWS Identity and Access Management での アクセス許可 AWS ParallelCluster」を参照してください。

TemplateURL (必須):

AWS ParallelCluster カスタムリソースファイルの URL。

出力:

ServiceToken:

カスタムリソース `ServiceToken` プロパティとして使用できる値。カスタムリソースは、がリクエスト AWS CloudFormation を送信する場所`ServiceToken`を指定します。これは、AWS CloudFormation テンプレートに含めるクラスターリソースに必要な入力です。

LogGroupArn:

基になるリソースのログの出力先となる CloudWatch `LogGroup` の ARN。

LambdaLayerArn:

AWS ParallelCluster オペレーションの実行に使用される Lambda レイヤーの ARN。

クラスターリソース

CloudFormation クラスターリソースは、次の CloudFormation テンプレートスニペットに示すようにフォーマットされています。

```
PclusterCluster:
```

```
  Type: Custom::PclusterCluster
  Properties:
    ServiceToken: !GetAtt [ PclusterClusterProvider , Outputs.ServiceToken ]
    ClusterName: !Sub 'c-${AWS::StackName}' # Must be different from StackName
    ClusterConfiguration:
    # Your Cluster Configuration
```

プロパティ:

ServiceToken:

AWS ParallelCluster プロバイダースタックの`ServiceToken`出力。

ClusterName:

作成および管理するクラスターの名前。名前は、CloudFormation スタックの名前と一致してはいけません。クラスターの作成後に名前を変更することはできません。

ClusterConfiguration:

クラスター設定ファイル で説明されているクラスター設定 YAML ファイル。ただし、組み込み関数など通常の CloudFormation コンストラクトを使用することができます。

DeletionPolicy:

ルートスタックが削除された際にクラスターを削除するかどうかを定義します。デフォルトは `Delete` です。

Retain:

カスタムリソースが削除されても、クラスターは保持されます。

> **Note**
> 保持されたクラスターの機能を維持するには、ストレージやネットワークなどクラスターに依存するリソースの削除ポリシーが保持に設定されている必要があります。

Delete:

カスタムリソースが削除された場合、クラスターは削除されます。

`Fn::GetAtt` 戻り値:

`Fn::GetAtt` 組み込み関数は、このタイプの指定された属性の値を返します。`Fn::GetAtt` `intrinsic` 関数の使用方法の詳細については、「Fn::GetAtt」を参照してください。

ClusterProperties:

`pcluster describe-cluster` オペレーションの値。

validationMessages:

前回の作成または更新操作中に発生したすべての検証メッセージを含む文字列。

logGroupName:

Lambda クラスターオペレーションのログ記録に使用されるロググループの名前。ログイベントは 90 日間保持され、ロググループはクラスターの削除後も保持されます。

例: `Fn::GetAtt`:

```
# Provide the public IP address of the head node as an output of a stack
Outputs:
  HeadNodeIp:
    Description: The public IP address of the head node
    Value: !GetAtt [ PclusterCluster, headNode.publicIpAddress ]
```

例: AWS ParallelCluster カスタムリソースを使用したシンプルで完全な CloudFormation テンプレート：

```
AWSTemplateFormatVersion: '2010-09-09'
Description: > AWS ParallelCluster CloudFormation Template

Parameters:
  HeadNodeSubnet:
    Description: Subnet where the HeadNode will run
    Type: AWS::EC2::Subnet::Id

  ComputeSubnet:
    Description: Subnet where the Compute Nodes will run
    Type: AWS::EC2::Subnet::Id

  KeyName:
    Description: KeyPair to login to the head node
    Type: AWS::EC2::KeyPair::KeyName
```

```
Resources:
  PclusterClusterProvider:
    Type: AWS::CloudFormation::Stack
    Properties:
      TemplateURL: !Sub
        - https://${AWS::Region}-aws-parallelcluster.s3.${AWS::Region}.
${AWS::URLSuffix}/parallelcluster/${Version}/templates/custom_resource/cluster.yaml
        - { Version: 3.13.2 }

  PclusterCluster:
    Type: Custom::PclusterCluster
    Properties:
      ServiceToken: !GetAtt [ PclusterClusterProvider , Outputs.ServiceToken ]
      ClusterName: !Sub 'c-${AWS::StackName}'
      ClusterConfiguration:
        Image:
          Os: alinux2
        HeadNode:
          InstanceType: t2.medium
          Networking:
            SubnetId: !Ref HeadNodeSubnet
          Ssh:
            KeyName: !Ref KeyName
        Scheduling:
          Scheduler: slurm
          SlurmQueues:
          - Name: queue0
            ComputeResources:
            - Name: queue0-cr0
              InstanceType: t2.micro
            Networking:
              SubnetIds:
              - !Ref ComputeSubnet

Outputs:
  HeadNodeIp:
    Description: The Public IP address of the HeadNode
    Value: !GetAtt [ PclusterCluster, headNode.publicIpAddress ]
  ValidationMessages:
    Description: Any warnings from cluster create or update operations.
    Value: !GetAtt PclusterCluster.validationMessages
```

CloudFormation AWS ParallelCluster カスタムリソースの使用方法の詳細については、「」を参照してください[を使用したクラスターの作成 AWS CloudFormation](#)。

クラスターオペレーション

クラスターカスタムリソースを CloudFormation スタックに追加すると、CloudFormation は次のクラスターオペレーションを実行できます。

- CloudFormation は、 AWS ParallelCluster カスタムリソースを含むスタックをデプロイするときに、新しい別のスタックにクラスターを作成します。
- スタックで定義されているクラスター設定を設定更新ポリシーに従って更新すると、CloudFormation はクラスターを更新します。 AWS ParallelCluster カスタムリソースプロバイダーは、クラスターを更新する前にコンピューティングフリートを停止しません。クラスターの更新には `QueueUpdateStrategy` 設定を使用することをお勧めします。これにより、 AWS ParallelCluster カスタムリソースを使用するときに、更新の前後に明示的な`pcluster update-compute-fleet`呼び出しを行うことを回避できます。
- スタックを削除すると、クラスターは削除されます。

AWS ParallelCluster カスタムリソースを含むスタックのトラブルシューティング

AWS ParallelCluster カスタムリソースを使用すると、CloudFormation は新しい個別のスタックからクラスターをデプロイします。以下の手順を実行して、クラスターの作成をモニタリングできます。

1. で CloudFormation に移動 AWS Management Console し、ナビゲーションペインで スタックを選択します。
2. クラスター名に定義した名前のスタックを選択します。
3. スタックの状態が `ROLLBACK_COMPLETE` の場合、クラスターの作成中にエラーが発生しました。
4. [スタックの詳細] を選択し、[イベント] タブを選択します。
5. クラスター名として定義した名前の [論理 ID] で [イベント] を検索します。問題の原因を表す `Status reason` が示されます。
6. [スタック] ドロップダウンメニューから [削除] を選択して、削除されたスタックのリストを表示することもできます。クラスター名を含むスタックを選択し、[イベント] から詳細を確認してください。
7. クラスターを管理するカスタムリソースプロバイダーからの出力を表示するには、説明AWS ParallelCluster「クラスターカスタムリソース」でスタックを選択します。[リソース] タブを選択

し、[論理 ID] `PclusterCfnFunctionLogGroup` を持つリソースを探して、指定されたリンクに従います。Lambda デバッグ出力を示すログストリームを表示します。

8. クラスターのトラブルシューティングについては、「AWS ParallelCluster トラブルシューティング」を参照してください。

Elastic Fabric Adapter

Elastic Fabric Adapter (EFA) は、同じサブネット上の他のインスタンスとの低レイテンシーのネットワーク通信用の OS バイパス機能を備えたネットワークデバイスです。EFA は Libfabric を使用して公開され、メッセージングパッシングインターフェイス (MPI) を使用するアプリケーションで使用できます。

AWS ParallelCluster とスSlurmケジューラで EFA `ComputeResources` を使用するには、`SlurmQueues`///`Efa`/ `Enabled`を に設定しますtrue。

EFA をサポートする Amazon EC2 インスタンスのリストを確認するには、「Linux インスタンス用 Amazon EC2 ユーザーガイド」の「サポートされるインスタンスタイプ」を参照してください。

EFA 対応インスタンスはプレイスメントグループで実行することをお勧めします。これにより、インスタンスは、1 つのアベイラビリティーゾーン内の低レイテンシーグループに起動されます。 AWS ParallelClusterを使用してプレイスメントグループを設定する方法の詳細については、「`SlurmQueues`」/「`Networking`」/「`PlacementGroup`」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「Elastic Fabric Adapter」と、「AWS Open Source Blog」の「Scale HPC workloads with elastic fabric adapter and AWS ParallelCluster」を参照してください。

> **Note**
>
> Elastic Fabric Adapter (EFA) は、異なるアベイラビリティーゾーン間ではサポートされていません。詳細については、「スケジューリング」/「SlurmQueues」/「ネットワーク」/「SubnetIds」を参照してください。

> **Note**
>
> デフォルトでは、Ubuntu ディストリビューションは ptrace (プロセストレース) 保護を有効にします。Libfabric が正常に動作するように ptrace 保護は無効になっています。詳細につ

いては、「Amazon EC2 ユーザーガイド」の「ptrace 保護を無効にする」を参照してください。

Intel MPI を有効にする

Intel MPI は AWS ParallelCluster AMIsで使用できます。

> **Note**
> インテル MPI を使用するには、「Intel simplified software license」(インテル簡易ソフトウェアライセンス) の条項を確認し、同意する必要があります。

デフォルトでは、Open MPI はパス上に配置されます。Open MPI の代わりにインテル MPI を有効にするには、まずインテル MPI モジュールをロードする必要があります。その後、`module load intelmpi` を使用して最新版をインストールする必要があります。モジュールの正確な名前は、更新ごとに変更されます。使用可能なモジュールを確認するには、`module avail` を実行します。出力は次のとおりです。

```
$ module avail
-----------------------------/usr/share/Modules/modulefiles
 --------------------------------
dot                             modules
libfabric-aws/1.16.0~amzn3.0    null
module-git                      openmpi/4.1.4
module-info                     use.own

----------------------------/opt/intel/mpi/2021.6.0/modulefiles
 ---------------------------
intelmpi
```

モジュールをロードするには、`module load` *`modulename`* を実行します。`mpirun` を実行するために使用するスクリプトにこれを追加できます。

```
$ module load intelmpi
```

ロードされているモジュールを確認するには、`module list` を実行します。

```
$ module list
Currently Loaded Modulefiles:
  1) intelmpi
```

Intel MPI が有効になっていることを確認するには、`mpirun --version` を実行します。

```
$ mpirun --version
Intel(R) MPI Library for Linux* OS, Version 2021.6 Build 20220227 (id: 28877f3f32)
Copyright 2003-2022, Intel Corporation.
```

Intel MPI モジュールがロードされると、Intel MPI ツールを使用するように複数のパスが変更されます。Intel MPI ツールでコンパイルされたコードを実行するには、まず Intel MPI モジュールをロードします。

> **Note**
>
> Intel MPI は Graviton AWS ベースのインスタンスと互換性がありません。

> **Note**
>
> AWS ParallelCluster バージョン 2.5.0 以前は、Intel MPI は中国 (北京) および中国 (寧夏) リージョンの AWS ParallelCluster AMIs では使用できませんでした。

AWS ParallelCluster API

AWS ParallelCluster API とは

AWS ParallelCluster API はサーバーレスアプリケーションであり、 にデプロイされると AWS アカウント、 API を介して AWS ParallelCluster 機能へのプログラムによるアクセスを提供します。

AWS ParallelCluster API は、Amazon API Gateway エンドポイントを含む自己完結型AWS CloudFormationテンプレートとして配布されます。Amazon API Gateway エンドポイントは、 AWS ParallelCluster 機能を公開し、 AWS Lambda関数は、呼び出された機能を処理します。 Amazon API Gateway

次の図は、 AWS ParallelCluster API インフラストラクチャの概要アーキテクチャ図を示しています。

AWS ParallelCluster API ドキュメント

AWS ParallelCluster API を記述する OpenAPI 仕様ファイルは、以下からダウンロードできます。

```
https://<REGION>-aws-parallelcluster.s3.<REGION>.amazonaws.com/
parallelcluster/<VERSION>/api/ParallelCluster.openapi.yaml
```

OpenAPI 仕様ファイルから、Swagger UI や Redoc など、利用可能な多くのツールのいずれかを使用して AWS ParallelCluster API のドキュメントを生成できます。

AWS ParallelCluster API をデプロイする方法

AWS ParallelCluster API をデプロイするには、 の管理者である必要があります AWS アカウント。

API のデプロイに使用したテンプレートは、以下の URL から入手できます。

```
https://<REGION>-aws-parallelcluster.s3.<REGION>.amazonaws.com/
parallelcluster/<VERSION>/api/parallelcluster-api.yaml
```

ここで、 *<REGION>* は API をデプロイする必要がある AWS リージョン で、 *<VERSION>*は AWS ParallelCluster バージョン (例: 3.13.2) です。

AWS Lambda は、 で Lambda レイヤーインターフェイスAWS ParallelCluster Python ライブラリ APIを使用して API で呼び出された機能を処理します。

> **Warning**
>
> AWS Lambda または Amazon API Gateway サービスへの特権アクセスを持つ のユーザーは AWS アカウント、 AWS ParallelCluster API リソースを管理するアクセス許可を自動的に継承します。

を使用して AWS ParallelCluster API をデプロイする AWS CLI

このセクションでは、 を使用してデプロイする方法について説明します AWS CLI。

CLI AWS で使用する認証情報をまだ設定していない場合は、設定します。

```
$ aws configure
```

以下のコマンドを実行して、API をデプロイします。

```
$ REGION=<region>
$ API_STACK_NAME=<stack-name>  # This can be any name
$ VERSION=3.13.2
$ aws cloudformation create-stack \
    --region ${REGION} \
    --stack-name ${API_STACK_NAME} \
    --template-url https://${REGION}-aws-parallelcluster.s3.${REGION}.amazonaws.com/
parallelcluster/${VERSION}/api/parallelcluster-api.yaml \
    --capabilities CAPABILITY_NAMED_IAM CAPABILITY_AUTO_EXPAND
$ aws cloudformation wait stack-create-complete --stack-name ${API_STACK_NAME} --region
 ${REGION}
```

デプロイのカスタマイズ

テンプレートによって公開されている AWS CloudFormation パラメータを使用して、API デプロイをカスタマイズできます。CLI を使用してデプロイするときにパラメータの値を設定するには、次のオプションを使用できます`--parameters ParameterKey=KeyName,ParameterValue=Value`。

以下のパラメータはオプションです。

- リージョン - `Region`パラメータを使用して、API が all AWS リージョン （ デフォルト) または 1 つの でリソースを制御できるかどうかを指定します AWS リージョン。アクセスを制限するために AWS リージョン 、この値を API がデプロイされている に設定します。
- ParallelClusterFunctionRole - 機能を実装する AWS Lambda 関数に割り当てられた IAM ロールを上書きします AWS ParallelCluster 。パラメータは、IAM ロールの ARN を受け取ります。このロールは、IAM プリンシパル AWS Lambda として を持つように設定する必要があります。また、このロールは API Lambda 関数のデフォルトロールを置き換えるため、 にリストされている API に必要なデフォルトのアクセス許可が少なくとも必要です[AWS ParallelCluster `pcluster` ユーザーポリシーの例](#)。
- ParallelClusterFunctionAdditionalPolicies - AWS ParallelCluster API 関数ロールにアタッチされる追加の IAM ポリシーの ARN。指定できるポリシーは 1 つだけです。
- CustomDomainName, CustomDomainCertificate, CustomDomainHostedZoneId - これらのパラメータを使用して、Amazon API Gateway のエンドポイントにカスタムドメインを設定することができます。`CustomDomainName` は使用するドメインの名前、`CustomDomainCertificate` はこのドメイン名に対する AWS マネージド証明書の ARN、`CustomDomainHostedZoneId` はレコードを作成する [Amazon Route 53](#) ホストゾーンの ID です。

> **Warning**
>
> API に最低限の Transport Layer Security (TLS) バージョンを強制適用するようにカスタムドメイン設定を構成できます。詳細については、「API Gateway におけるカスタムドメインの TLS の最小バージョンの選択」を参照してください。

- EnableIamAdminAccess - デフォルトでは、 AWS Lambda AWS ParallelCluster API オペレーションを処理する関数は、特権 IAM アクセス () を禁止する IAM ロールで設定されます`EnableIamAdminAccess=false`。このため、IAM ロールやポリシーの作成を必要とするオペレーションを API で処理することができません。このため、クラスターやカスタムイメージの作成は、IAM ロールがリソース構成のインプットの場合にのみ可能です。

  `EnableIamAdminAccess`を に設定すると`true`、 AWS ParallelCluster クラスターのデプロイまたはカスタム AMIs。

  > **Warning**
  >
  > これを true に設定すると、 AWS ParallelCluster オペレーションを処理する AWS Lambda 関数に IAM 管理者権限が付与されます。

  このモードを有効にするときにロック解除できる機能の詳細については、AWS ParallelCluster IAM リソースを管理するためのユーザーポリシーの例「」を参照してください。

- PermissionsBoundaryPolicy - このオプションのパラメータは、PC API インフラストラクチャで作成するすべての IAM ロールのアクセス許可の境界として、および管理 IAM アクセス許可の条件として、設定される既存の IAM ポリシー ARN を受け入れ、このポリシーを持つロールのみを PC API によって作成できるようにします。

  このモードによる制限の詳細については、「`PermissionsBoundary` モード」を参照してください。

- CreateApiUserRole - デフォルトでは、 AWS ParallelCluster API のデプロイには、API の呼び出しを許可された唯一のロールとして設定された IAM ロールの作成が含まれます。Amazon API Gateway エンドポイントには、作成されたユーザーのみに呼び出しアクセス許可を付与するようにリソースベースのポリシーが設定されています。これを変更するには、 を設定し、選択した IAM ユーザーに API アクセス`CreateApiUserRole=false`を付与します。詳細については、Amazon API Gateway API Gateway デベロッパーガイド」の「API を呼び出すためのアクセスの制御」を参照してください。

> **Warning**
>
> API エンドポイント`CreateApiUserRole=true`へのアクセスが Amazon API Gateway リソースポリシーによって制限されていない場合、制約のない`execute-api:Invoke`アクセス許可を持つすべての IAM ロールが AWS ParallelCluster 機能にアクセスできます。詳細については、「API Gateway デベロッパーガイド」の「API Gateway リソースポリシーを使用した API へのアクセスの制御」を参照してください。

> **Warning**
>
> `ParallelClusterApiUserRole` は、すべての AWS ParallelCluster API オペレーションに対する呼び出し権限を有しています。API リソースのサブセットへのアクセスを制限するには、「API Gateway デベロッパーガイド」の「IAM ポリシーを使用して API Gateway API メソッドを呼び出すことができるコントロール」を参照してください。

- IAMRoleAndPolicyPrefix - このオプションパラメータは、PC API インフラストラクチャの一部として作成された IAM ロールとポリシーの両方のプレフィックスとして使用される最大 10 文字の文字列を受け入れます。

API の更新

このセクションでは、使用可能な 2 つのオプションのいずれかを使用して API を更新する方法について説明します。

新しい AWS ParallelCluster バージョンへのアップグレード

オプション 1: 既存の API を削除するには、対応する AWS CloudFormation スタックを削除し、上記のように新しい API をデプロイします。

オプション 2: 既存の API を更新するには、次のコマンドを実行します。

```
$ REGION=<region>
$ API_STACK_NAME=<stack-name>  # This needs to correspond to the existing API stack
 name
$ VERSION=3.13.2
$ aws cloudformation update-stack \
    --region ${REGION} \
```

```
    --stack-name ${API_STACK_NAME} \
    --template-url https://${REGION}-aws-parallelcluster.s3.${REGION}.amazonaws.com/
parallelcluster/${VERSION}/api/parallelcluster-api.yaml \
    --capabilities CAPABILITY_NAMED_IAM CAPABILITY_AUTO_EXPAND
$ aws cloudformation wait stack-update-complete --stack-name ${API_STACK_NAME} --region
 ${REGION}
```

AWS ParallelCluster API の呼び出し

AWS ParallelCluster Amazon API Gateway エンドポイントは`AWS_IAM認可タイプ`に設定されているため、すべてのリクエストに有効な IAM 認証情報による SigV4 署名が必要となります (API リファレンス: http リクエストの作成)。

デフォルトの設定でデプロイされた場合、API の呼び出し許可は、API で作成されたデフォルトの IAM ユーザーにのみ付与されます。

デフォルトの IAM ユーザーの ARN を取得するには、以下を実行します。

```
$ REGION=<region>
$ API_STACK_NAME=<stack-name>
$ aws cloudformation describe-stacks --region ${REGION} --stack-name ${API_STACK_NAME}
 --query "Stacks[0].Outputs[?OutputKey=='ParallelClusterApiUserRole'].OutputValue" --
output text
```

デフォルトの IAM ユーザーの一時的な認証情報を取得するには、STS AssumeRole コマンドを実行します。

AWS ParallelCluster API エンドポイントを取得するには、次のコマンドを実行します。

```
$ REGION=<region>
$ API_STACK_NAME=<stack-name>
$ aws cloudformation describe-stacks --region ${REGION} --stack-name ${API_STACK_NAME}
 --query "Stacks[0].Outputs[?OutputKey=='ParallelClusterApiInvokeUrl'].OutputValue" --
output text
```

AWS ParallelCluster API は、OpenAPI 仕様に準拠する任意の HTTP クライアントで呼び出すことができます。この仕様については、以下を参照してください。

```
https://<REGION>-aws-parallelcluster.s3.<REGION>.amazonaws.com/
parallelcluster/<VERSION>/api/ParallelCluster.openapi.yaml
```

ここのドキュメントにある通り、リクエストには SigV4 署名が必要です。

現時点では、公式な API クライアントの実装はありません。ただし、OpenAPI Generator を使用して、OpenAPI モデルから API OpenAPI クライアントを簡単に生成できます。クライアントが生成されたら、すぐに指定しない場合は SigV4 署名を追加する必要があります。

Python API クライアントのリファレンス実装は、AWS ParallelCluster のリポジトリにあります。Python API クライアントの使い方については、「AWS ParallelCluster API の使用 チュートリアル」を参照してください。

Amazon Cognito や Lambda オーソライザーなどのより高度なアクセスコントロールメカニズムを実装したり、 AWS WAF または API キーを使用して API をさらに保護したりするには、Amazon API Gateway のドキュメントに従ってください。

> **Warning**
>
> AWS ParallelCluster API を呼び出す権限を持つ IAM ユーザーは、 AWS ParallelCluster で によって管理されるすべての AWS リソースを間接的に制御できます AWS アカウント。これには、ユーザー IAM ポリシーの制限によりユーザーが直接制御できない AWS リソースの作成が含まれます。たとえば、 AWS ParallelCluster クラスターの作成は、その設定に応じて、Amazon EC2 インスタンス、Amazon Route 53、Amazon Elastic File System ファイルシステム、Amazon FSx ファイルシステム、IAM ロール、および AWS ParallelCluster ユーザーが直接制御できない可能性がある他の AWS のサービス のリソースのデプロイが含まれる場合があります。

> **Warning**
>
> 設定で`AdditionalIamPolicies`指定された を使用してクラスターを作成する場合、追加のポリシーは次のいずれかのパターンと一致する必要があります。
>
> ```
> - !Sub arn:${AWS::Partition}:iam::${AWS::AccountId}:policy/parallelcluster*
> - !Sub arn:${AWS::Partition}:iam::${AWS::AccountId}:policy/parallelcluster/*
> - !Sub arn:${AWS::Partition}:iam::aws:policy/CloudWatchAgentServerPolicy
> - !Sub arn:${AWS::Partition}:iam::aws:policy/AmazonSSMManagedInstanceCore
> - !Sub arn:${AWS::Partition}:iam::aws:policy/AWSBatchFullAccess
> - !Sub arn:${AWS::Partition}:iam::aws:policy/AmazonS3ReadOnlyAccess
> - !Sub arn:${AWS::Partition}:iam::aws:policy/service-role/AWSBatchServiceRole
> - !Sub arn:${AWS::Partition}:iam::aws:policy/service-role/
> AmazonEC2ContainerServiceforEC2Role
> ```

```
- !Sub arn:${AWS::Partition}:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy
- !Sub arn:${AWS::Partition}:iam::aws:policy/service-role/
AmazonEC2SpotFleetTaggingRole
- !Sub arn:${AWS::Partition}:iam::aws:policy/EC2InstanceProfileForImageBuilder
- !Sub arn:${AWS::Partition}:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole
```

その他の追加ポリシーが必要な場合は、以下のいずれかを行うことができます。

- `DefaultParallelClusterIamAdminPolicy` を以下で編集する。

  ```
  https://<REGION>-aws-parallelcluster.s3.<REGION>.amazonaws.com/
  parallelcluster/<VERSION>/api/parallelcluster-api.yaml
  ```

  ポリシーを `ArnLike/iam:PolicyARN` セクションに追加します。
- 設定ファイル`AdditionalIamPolicies`で のポリシーを指定せず、クラスター内で作成された AWS ParallelCluster インスタンスロールにポリシーを手動で追加します。

API ログとメトリクスへのアクセス

API ログは Amazon CloudWatch に公開されますが、その保存期間は 30 日間です。API デプロイに関連する LogGroup 名を取得するには、次のコマンドを実行します。

```
$ REGION=<region>
$ API_STACK_NAME=<stack-name>
$ aws cloudformation describe-stacks --region ${REGION} --
stack-name ${API_STACK_NAME} --query "Stacks[0].Outputs[?
OutputKey=='ParallelClusterLambdaLogGroup'].OutputValue" --output text
```

また、Lambda のメトリクス、ログ、AWS X-Ray トレースログは Lambda コンソールからもアクセスできます。API デプロイに関連する Lambda 関数の ARN を取得するには、次のコマンドを実行します。

```
$ REGION=<region>
$ API_STACK_NAME=<stack-name>
```

```
$ aws cloudformation describe-stacks --region ${REGION} --stack-name ${API_STACK_NAME}
 --query "Stacks[0].Outputs[?OutputKey=='ParallelClusterLambdaArn'].OutputValue" --
output text
```

AWS ParallelCluster Terraform の

AWS ParallelCluster 3.8.0 以降では、Terraform を使用してクラスターとカスタムイメージをデプロイできます。この機能の使用を開始するには、Terraform Registry の「Terraform Provider for AWS ParallelCluster」を参照してください。

> **Note**
>
> プロバイダーを使用するには、ParallelCluster API をアカウントにデプロイしている必要があります。

次のチャートを使用して、プロバイダーと AWS ParallelCluster のバージョン間の互換性を判断します。

プロバイダーのバージョン	AWS ParallelCluster バージョン
1.0.0	3.8.0 ~ 3.10.1
1.1.0	3.11.0 以降

プロバイダーの使用方法の例を参照してください。

さらにスムーズなエクスペリエンスを実現するには、Terraform Registry の公式 Terraform Module for AWS ParallelCluster を使用します。このモジュールでは、以下をデプロイできます。

1. ParallelCluster API
2. YAML 設定ファイルおよび HCL で定義した ParallelCluster クラスター
3. ParallelCluster クラスターに必要なネットワークインフラストラクチャ

モジュールの使用方法の例を参照してください。

Amazon DCV 経由でヘッドノードとログインノードに接続する

Amazon DCV は、リモートの高性能サーバーでホストされているグラフィック集約型の 3D アプリケーションにユーザーが安全に接続できるようにするリモート可視化テクノロジーです。詳細については、「Amazon DCV」を参照してください。

Amazon DCV ソフトウェアは、ヘッドノードに自動的にインストールされ、HeadNode から Dcv セクションを使用して有効にすることができます。

AWS ParallelCluster バージョン 3.11 以降では、/ Pools 設定の Dcv セクションを使用して、ログインノードプールに対して Amazon DCV LoginNodes セクション を有効にできます。

```
LoginNodes:
  Pools:
    Dcv:
      Enabled: true
```

AWS ParallelCluster は DCV サーバーストレージフォルダ/home/<DEFAULT_AMI_USER>として設定されます。Amazon DCV の設定パラメータの詳細については、「HeadNode」または「Dcv」を参照してください。

ヘッドノードの Amazon DCV セッションに接続するには、dcv-connect コマンドを使用します。ログインノードに接続するには、dcv-connect で --login-node-ip パラメータを使用し、接続先のログインノードのパブリック IP アドレスまたはプライベート IP アドレスを渡します。

Amazon DCV の HTTPS 証明書

Amazon DCV は、Amazon DCV クライアント と Amazon DCV サーバー間のトラフィックを保護するための自己署名証明書を自動的に生成します。

デフォルトの自己署名 Amazon DCV 証明書を別の証明書に置き換えるには、まずヘッドノードに接続します。次に、pcluster dcv-connect コマンドを実行する前に、証明書とキーの両方を /etc/dcv フォルダにコピーします。

詳細については、「Amazon DCV 管理者ガイド」の「TLS 証明書の変更」を参照してください。

Amazon DCV のライセンス

Amazon DCV は、Amazon EC2 インスタンスで実行する場合、ライセンスサーバーを必要としません。ただし、Amazon DCV サーバーは定期的に Amazon S3 バケットに接続して、有効なライセンスが利用可能かどうかを判断する必要があります。

AWS ParallelCluster は、必要なアクセス許可をヘッドノードの IAM ポリシーに自動的に追加します。カスタム IAM インスタンスポリシーを使用する場合は、「Amazon DCV 管理者ガイド」の「Amazon EC2 上の Amazon DCV」で説明しているアクセス許可を使用します。

トラブルシューティングのヒントについては、「Amazon DCV の問題のトラブルシューティング」を参照してください。

pcluster update-cluster を使用する

AWS ParallelCluster 3.x では、現在のクラスターの作成に使用される設定と、設定ファイルの設定で問題がないか`pcluster update-cluster`を分析します。問題が発見された場合は報告され、問題を解決するための手順が表示されます。例えば、コンピューティング `InstanceType` が変更された場合、アップデートを行う前にコンピューティングフリートを停止する必要があります。この問題は、発見された時点で報告されます。ブロッキング問題が発見されなければ、アップデートプロセスが開始され、変更が報告されます。

`pcluster update-cluster` `--dryrun` `option` を使用して、実行前に変更を確認できます。詳細については、「`pcluster update-cluster` の例」を参照してください。

トラブルシューティングヘルプについては、「AWS ParallelCluster トラブルシューティング」を参照してください。

ポリシー定義を更新します

更新ポリシー: この設定を変更して更新するには、クラスターのログインノードを停止する必要があります。

> クラスター内のログインノードが使用中である場合、これらの設定を変更することはできません。変更を元に戻すか、クラスターのログインノードを停止する必要があります（各プールのカウントを 0 に設定することで、クラスター内のログインノードを停止できます）。クラスターのログインノードが停止したら、クラスター (pcluster update-cluster) を更新して変更をアクティブ化できます。

> **Note**
>
> この更新ポリシーは、 AWS ParallelCluster バージョン 3.7.0 以降でサポートされています。

更新ポリシー: ログインノードプールを追加できますが、プールを削除するには、クラスター内のすべてのログインノードを停止する必要があります。

プールを削除するには、クラスター内のすべてのログインノードを停止する必要があります (各プールのカウントを 0 に設定することで、クラスター内のログインノードを停止できます)。クラスターのログインノードを停止したら、クラスターを更新 (`pcluster update-cluster`) して変更を有効にすることができます。

> **Note**
>
> この更新ポリシーは、 AWS ParallelCluster バージョン 3.11.0 以降でサポートされています。

更新ポリシー: この設定を変更して更新するには、プール内のログインノードを停止する必要があります。

プール内のログインノードが使用中の場合、これらの設定を変更することはできません。変更を元に戻すか、プールのログインノードを停止する必要があります (プール内のログインノードを停止するには、プールのカウントを 0 に設定します)。プールのログインノードを停止したら、クラスターを更新 (`pcluster update-cluster`) して変更を有効にすることができます。

> **Note**
>
> この更新ポリシーは、 AWS ParallelCluster バージョン 3.11.0 以降でサポートされています。

更新ポリシー: この設定は、更新中に変更できます。

この設定を変更すると、`pcluster update-cluster` を使ってクラスターを更新できるようになります。

更新ポリシー: この設定が変更された場合、更新は許可されません。

この設定を変更すると、クラスターの更新ができなくなります。元のクラスターの設定を元に戻し、更新した設定で新しいクラスターを作成する必要があります。元のクラスターは後日削除できます。`pcluster create-cluster` を使用して新しいクラスターを作成します。`pcluster delete-cluster` を使用して元のクラスターを削除します。

更新ポリシー: この設定は、更新中には分析されません。

これらの設定は、pcluster update-cluster を使用して変更し、クラスターを更新することができます。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

これらの設定は、コンピューティングフリートが存在する間は変更できません。変更を元に戻すか、コンピューティングフリートを停止する必要があります (pcluster update-compute-fleet を使用します)。コンピューティングフリートを停止したら、クラスター (pcluster update-cluster) を更新して変更を有効にすることができます。例えば、Slurm のスケジューラを SlurmQueues/ComputeResources/ - Name/MinCount > 0 で使用している場合、コンピューティングフリートが開始されます。

更新ポリシー: この設定を変更して更新するには、コンピューティングフリートとログインノードを停止する必要があります。

コンピューティングフリートが存在する間、またはログインノードが使用中の場合、これらの設定を変更することはできません。変更を元に戻すか、コンピューティングフリートとログインノードを停止する必要があります (コンピューティングフリートを停止するには、pcluster update-compute-fleet を使用できます)。コンピューティングフリートとログインノードを停止したら、クラスターを更新 (pcluster update-cluster) して変更を有効にすることができます。

更新ポリシー: これらの設定は、更新中に減らすことはできません。

これらの設定は変更できますが、減らすことはできません。これらの設定を減らす必要がある場合は、元のクラスターの設定を元に戻し、更新した設定で新しいクラスターを作成する必要があります。元のクラスターは後日削除できます。pcluster create-cluster を使用して新しいクラスターを作成します。pcluster delete-cluster を使用して元のクラスターを削除します。

更新ポリシー: この設定が変更された場合、更新は許可されません。強制的に更新した場合、新しい値は無視され、既存の値が使用されます。

この設定を変更すると、クラスターの更新ができなくなります。元のクラスターの設定を元に戻し、更新した設定で新しいクラスターを作成する必要があります。元のクラスターは後日削除できます。pcluster create-cluster を使用して新しいクラスターを作成します。pcluster delete-cluster を使用して元のクラスターを削除します。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、**`QueueUpdateStrategy`** が設定されている必要があります。

これらの設定は変更することができます。コンピューティングフリートを停止するか (`pcluster update-compute-fleet` を使用します)、`QueueUpdateStrategy` を設定する必要があります。コンピューティングフリートを停止されるか、`QueueUpdateStrategy` が設定された後、クラスター (`pcluster update-cluster`) を更新して変更を有効にすることができます。

> **Note**
>
> この更新ポリシーは、 AWS ParallelCluster バージョン 3.2.0 以降でサポートされています。

更新ポリシー:このリスト値の設定では、更新中に新しい値を追加することができ、既存の値を削除する場合はコンピューティングフリートを停止する必要があります。

これらの設定には更新時に新しい値を追加できます。リストに新しい値を追加すると、クラスターは (`pcluster update-cluster`) を使用して更新できます。

リストから既存の値を削除するには、コンピューティングフリートを (`pcluster update-compute-fleet` を使用して) 停止する必要があります。

例えば、Slurm スケジューラを使用していて、新しいインスタンスタイプを Instances/InstanceType に追加する場合は、コンピューティングフリートを停止せずにクラスターを更新できます。既存のインスタンスタイプを Instances/InstanceType から削除するには、最初にコンピューティングフリートを停止する必要があります (pcluster update-compute-fleet を使用します)。

> **Note**
>
> この更新ポリシーは、 AWS ParallelCluster バージョン 3.2.0 以降でサポートされています。

更新ポリシー: キューのサイズを減らすには、コンピューティングフリートを停止するか、QueueUpdateStrategy を TERMINATE に設定して、この設定を変更して更新する必要があります。

これらの設定は変更できますが、変更に伴ってキューのサイズが小さくなる場合は、コンピューティングフリートを停止するか (pcluster update-compute-fleet を使用)、QueueUpdateStrategy を TERMINATE に設定する必要があります。コンピューティングフリートを停止するか、QueueUpdateStrategy を TERMINATE に設定したら、クラスターを更新 (pcluster update-cluster) して変更を有効にすることができます。

クラスターの容量のサイズを変更するときに設定した TERMINATE は、ノードリストの最後尾にあるノードのみを終了し、同じパーティションの他のすべてのノードはそのまま残します。

例えば、クラスターの初期容量が `MinCount = 5` および `MaxCount = 10` である場合、ノードは `st-[1-5]; dy-[1-5]` です。クラスターのサイズを `MinCount = 3` および `MaxCount = 5` に変更すると、新しいクラスター容量はノード `st-[1-3]; dy-[1-2]` によって構成され、ノード は更新中に変更されません。更新中に終了されるのは、ノード `st-[4-5]; dy-[3-5]` のみです。

以下の変更がサポートされており、コンピューティングフリートを停止したり、QueueUpdateStrategy を TERMINATE に設定したりする必要はありません。

- 新しい SlurmQueue の追加
- 新しい ComputeResource の追加
- MaxCount の増加
- MinCount の増加と MaxCount の同等以上の増加

注： この更新ポリシーは、 AWS ParallelCluster バージョン 3.9.0 以降でサポートされています。

更新ポリシー: このリスト値の設定では、コンピューティングフリートを停止するか、**`QueueUpdateStrategy`** が設定されている必要があります。既存の値を削除する場合は、コンピューティングフリートを停止する必要があります。

これらの設定には更新時に新しい値を追加できます。コンピューティングフリートを停止するか (`pcluster update-compute-fleet` を使用します)、`QueueUpdateStrategy` を設定する必要があります。コンピューティングフリートを停止されるか、`QueueUpdateStrategy` が設定された後、クラスター (`pcluster update-cluster`) を更新して変更を有効にすることができます。

リストから既存の値を削除するには、コンピューティングフリートを (`pcluster update-compute-fleet` を使用して) 停止する必要があります。

> **Note**
>
> この更新ポリシーは、 AWS ParallelCluster バージョン 3.3.0 以降でサポートされています。

更新ポリシー: マネージドプレースメントグループを削除するには、すべてのコンピューティングノードを停止する必要があります。この設定を更新で変更するためには、コンピューティングフリートが停止しているか、**`QueueUpdateStrategy`** が設定されている必要があります。

マネージドプレイスメントグループを削除するには、コンピューティングフリートを停止 (`pcluster update-compute-fleet` を使用します) する必要があります。コンピューティングフリートを停止する前にクラスター更新を実行してマネージドプレイスメントグループを削除すると、無効な設定メッセージが返され、更新は続行されません。コンピューティングフリートを停止すると、インスタンスが実行されていないことが保証されます。

`pcluster update-cluster` の例

これらの設定は変更できますが、変更に伴ってキューのサイズが小さくなる場合は、コンピューティングフリートを停止するか (pcluster update-compute-fleet を使用)、QueueUpdateStrategy を TERMINATE に設定する必要があります。コンピューティングフリートを停止するか、QueueUpdateStrategy を TERMINATE に設定したら、クラスター (pcluster update-cluster) を更新して変更を有効にすることができます。

- この例では、いくつかの許可された変更を伴う更新を実際に行い、更新を直接開始します。

```
$ pcluster update-cluster --cluster-name cluster_name --cluster-config
 ~/.parallelcluster/test_cluster --region us-east-1
{
  "cluster": {
    "clusterName": cluster_name,
    "cloudformationStackStatus": "UPDATE_IN_PROGRESS",
    "cloudformationStackArn": stack_arn,
    "region": "us-east-1",
    "version": "3.13.2",
    "clusterStatus": "UPDATE_IN_PROGRESS"
```

```
  },
  "changeSet": [
    {
      "parameter": "HeadNode.Networking.AdditionalSecurityGroups",
      "requestedValue": [
        "sg-0cd61884c4ad11234"
      ],
      "currentValue": [
        "sg-0cd61884c4ad16341"
      ]
    }
  ]
}
```

- この例では、いくつかの変更が許可されたドライランアップデートを行います。ドライランは、アップデートを開始せずに変更内容を報告するのに便利です。

```
$ pcluster update-cluster --cluster-name cluster_name --cluster-config
 ~/.parallelcluster/test_cluster --region us-east-1 --dryrun true
{
  "message": "Request would have succeeded, but DryRun flag is set.",
  "changeSet": [
    {
      "parameter": "HeadNode.Networking.AdditionalSecurityGroups",
      "requestedValue": [
        "sg-0cd61884c4ad11234"
      ],
      "currentValue": [
        "sg-0cd61884c4ad16341"
      ]
    }
  ]
}
```

- この例では、アップデートをブロックするいくつかの変更を加えたアップデートです。

```
$ pcluster update-cluster --cluster-name cluster_name --cluster-config
 ~/.parallelcluster/test_cluster --region us-east-1
{
  "message": "Update failure",
  "updateValidationErrors": [
    {
      "parameter": "HeadNode.Ssh.KeyName",
```

```
      "requestedValue": "mykey_2",
      "message": "Update actions are not currently supported for the 'KeyName'
 parameter. Restore 'KeyName' value to 'jenkinsjun'. If you need this change, please
 consider creating a new cluster instead of updating the existing one.",
      "currentValue": "mykey_1"
    },
    {
      "parameter": "Scheduling.SlurmQueues[queue1].ComputeResources[queue1-
t2micro].InstanceType",
      "requestedValue": "c4.xlarge",
      "message": "All compute nodes must be stopped. Stop the compute fleet with the
 pcluster update-compute-fleet command",
      "currentValue": "t2.micro"
    },
    {
      "parameter": "SharedStorage[ebs1].MountDir",
      "requestedValue": "/my/very/very/long/shared_dir",
      "message": "Update actions are not currently supported for the 'MountDir'
 parameter. Restore 'MountDir' value to '/shared'. If you need this change, please
 consider creating a new cluster instead of updating the existing one.",
      "currentValue": "/shared"
    }
  ],
  "changeSet": [
    {
      "parameter": "HeadNode.Networking.AdditionalSecurityGroups",
      "requestedValue": [
        "sg-0cd61884c4ad11234"
      ],
      "currentValue": [
        "sg-0cd61884c4ad16341"
      ]
    },
    {
      "parameter": "HeadNode.Ssh.KeyName",
      "requestedValue": "mykey_2",
      "currentValue": "mykey_1"
    },
    {
      "parameter": "Scheduling.SlurmQueues[queue1].ComputeResources[queue1-
t2micro].InstanceType",
      "requestedValue": "c4.xlarge",
      "currentValue": "t2.micro"
    },
```

```
    {
      "parameter": "SharedStorage[ebs1].MountDir",
      "requestedValue": "/my/very/very/long/shared_dir",
      "currentValue": "/shared"
    }
  ]
}
```

AWS ParallelCluster AMI のカスタマイズ

のカスタム AMI の構築 AWS ParallelCluster が必要なシナリオがあります。このセクションでは、カスタム AMI AWS ParallelCluster を構築する際の考慮事項について説明します。

カスタム AMI AWS ParallelCluster は、次のいずれかの方法を使用して構築できます。

1. ビルドイメージ設定ファイルを作成し、`pcluster` CLI を使用して EC2 Image Builder でイメージを構築します。このプロセスは自動化され、繰り返し可能で、モニタリングにも対応しています。詳細については、「`pcluster` イメージコマンド」を参照してください。
2. AWS ParallelCluster AMI からインスタンスを作成し、そのインスタンスにログインして手動で変更します。最後に、Amazon EC2 を使用して、変更したインスタンスから新しい AMI を作成します。このプロセスにかかる時間は短くなります。ただし、自動化されておらず、繰り返し可能でもなく、`pcluster` CLI イメージモニタリングコマンドの使用もサポートされていません。

これらの方法の詳細については、「カスタム AMI AWS ParallelCluster の構築」を参照してください。

AWS ParallelCluster AMI のカスタマイズに関する考慮事項

カスタムイメージをどのように作成するかにかかわらず、事前検証テストを実施し、作成中のイメージのステータスを監視するための対策を含めることをお勧めします。

`pcluster` を使用してカスタム AMI を構築するには、EC2 Image Builder がカスタムイメージのビルドに使用する `Build` および `Image` セクションを含むビルドイメージ設定ファイルを作成します。`Build` セクションでは、Image Builder がイメージを構築するために必要なものを指定します。これには `ParentImage` (ベースイメージ)、および `Components` が含まれます。Image Builder コンポーネントは、イメージが作成される前にインスタンスをカスタマイズしたり、作成されたイメージによって起動されたインスタンスをテストしたりするために必要な一連の手順を定義します。詳細に

ついては、EC2 Image Builder ユーザーガイドの「Image Builder を使用してカスタムコンポーネントを作成する」を参照してください。 EC2

pcluster から呼び出`build-image`されてカスタムイメージを作成すると、Image Builder はク AWS ParallelCluster ックブックでビルドイメージ設定を使用して AWS ParallelCluster でブートストラップします`ParentImage`。Image Builder はコンポーネントをダウンロードし、ビルドフェーズと検証フェーズを実行し、AMI を作成し、AMI からインスタンスを起動し、テストを実行します。プロセスが完了すると、Image Builder は新しいイメージまたは停止メッセージを生成します。

カスタムコンポーネントの検証テストを実行する

設定に Image Builder コンポーネントを含める前に、以下の方法のいずれかを使用してテストと検証を行います。Image Builder の処理には最大 1 時間かかることがあるため、事前にコンポーネントをテストすることをお勧めします。これにより大幅に時間を節約できます。

スクリプトのケース

ビルドイメージプロセス外の実行中のインスタンスでスクリプトをテストし、スクリプトが終了コード 0 で終了することを確認します。

Amazon リソースネーム (ARN) のケース

ビルドイメージプロセスの外で、実行中のインスタンスでコンポーネントドキュメントをテストします。要件のリストについては、「Image Builder ユーザーガイド」の「コンポーネントマネージャ」を参照してください。

検証に成功後、ビルドイメージ設定にコンポーネントを追加する

カスタムコンポーネントが動作していることを確認したら、ビルドイメージ設定ファイルに追加します。

デバッグに役立つ **pcluster** コマンドを使用して Image Builder プロセスを監視する

`describe-image`

このコマンドを使用すると、ビルドイメージのステータスを監視できます。

`list-image-log-streams`

`get-image-log-events` と併せて、このコマンドを使用すると、ログイベントの取得に使用できるログストリームの ID を取得します。

get-image-log-events

このコマンドを使用すると、ビルドイメージプロセスイベントのログストリームを取得できます。

例えば、以下のコマンドを使用して、ビルドイメージのイベントを追跡できます。

```
$ watch -n 1 'pcluster get-image-log-events -i <image-id> \
        --log-stream-name/1 <pcluster-version> \
        --query "events[*].message" | tail -n 50'
```

get-image-stack-events

このコマンドを使用すると、Image Builder が作成するスタックのイメージスタックイベントを取得できます。

export-image-logs

このコマンドを使用すると、イメージログを保存できます。

AWS ParallelCluster ログと Amazon CloudWatch の詳細については、Amazon CloudWatch Logs のビルドイメージログ「」および「」を参照してくださいAmazon CloudWatch ダッシュボード。

その他の考慮事項

新しい AWS ParallelCluster リリースとカスタム AMIs

カスタム AMI を構築した場合は、新しい AWS ParallelCluster のリリースごとに、カスタム AMI を作成したときの手順を繰り返す必要があります。

カスタムブートストラップアクション

カスタムブートストラップアクション セクションを確認して、今後の AWS ParallelCluster リリースで変更をスクリプト化してサポートできるかどうかを確認します。

カスタム AMI を使用する

`Image`/`CustomAmi` および `Scheduling`/`SlurmQueues`/ - `Name`/`Image`/`CustomAmi` セクションのクラスター設定でカスタム AMI を指定できます。

カスタム AMI 検証警告のトラブルシューティングについては、「カスタム AMI の問題のトラブルシューティング」を参照してください。

オンデマンドキャパシティ予約 (ODCR) を使用してインスタンスを起動する

オンデマンドキャパシティ予約 (ODCR) を使用すると、特定のアベイラビリティーゾーンでクラスターの Amazon EC2 インスタンスのキャパシティを予約できます。これにより、Savings Plans またはリージョンリザーブドインスタンスが提供する請求アカウントとは関係なく、キャパシティの予約を作成および管理できます。

open または `targeted` の ODCR を設定できます。オープン ODCR は、ODCR 属性に一致するすべてのインスタンスを対象としています。これらの属性はインスタンスタイプ、プラットフォーム、およびアベイラビリティーゾーンです。ターゲット ODCR はクラスター設定で明示的に定義する必要があります。ODCR が open または `targeted` であるかどうかを確認するには、 AWS CLI の Amazon EC2 `describe-capacity-reservation` コマンドを実行します。

クラスタープレイスメントグループオンデマンドキャパシティ予約 (CPG ODCR) と呼ばれるクラスタープレイスメントグループに ODCR を作成することもできます。

複数の ODCR を 1 つのリソースグループにまとめることができます。これはクラスター設定ファイルで定義できます。リソースグループの詳細については、「Resource Groups とタグのユーザーガイド」の「What are resource groups?」を参照してください。

での ODCR の使用 AWS ParallelCluster

AWS ParallelCluster はオープン ODCR をサポートしています。オープン ODCR を使用する場合は、 AWS ParallelClusterで何も指定する必要はありません。インスタンスはクラスター用に自動的に選択されます。既存のプレイスメントグループを指定するか、 に新しいプレイスメントグループ AWS ParallelCluster を作成させることができます。

クラスター設定の ODCR

AWS ParallelCluster バージョン 3.3.0 以降では、クラスター設定ファイルで ODCRs を定義できます。Amazon EC2 ランインスタンスの上書きを指定する必要はありません。

まず、それぞれのリンク先のドキュメントで説明されている方法を使用して、キャパシティ予約とリソースグループを作成します。キャパシティ予約グループを作成するには、 AWS CLI メソッドを使用する必要があります。を使用する場合は AWS Management Console、タグベースまたはスタックベースのリソースグループのみを作成できます。キャパシティ予約でインスタンスを起動 AWS CLI する場合、タグベースおよびスタックベースのリソースグループは AWS ParallelCluster または ではサポートされていません。

キャパシティ予約とリソースグループを作成したら、以下のクラスター設定例に示すように、SlurmQueues/CapacityReservationTarget または SlurmQueues/ComputeResources/CapacityReservationTarget で指定します。赤で強調表示された##有効な値に置き換えます。

```
Image:
  Os: os
HeadNode:
  InstanceType: head_node_instance
  Networking:
    SubnetId: public_subnet_id
  Ssh:
    KeyName: key_name
Scheduling:
  Scheduler: scheduler
  SlurmQueues:
    - Name: queue1
      Networking:
        SubnetIds:
          - private_subnet_id
      ComputeResources:
        - Name: cr1
          Instances:
            - InstanceType: instance
          MaxCount: max_queue_size
          MinCount: max_queue_size
          Efa:
            Enabled: true
          CapacityReservationTarget:
            CapacityReservationResourceGroupArn: capacity_reservation_arn
```

廃止/非推奨 – Amazon EC2 インスタンスオーバーライドによるターゲット ODCR

> **Warning**
>
> - AWS ParallelCluster バージョン 3.3.0 以降では、この方法はお勧めしません。このセクションは、以前のバージョンを使用した実装のリファレンスとして残しています。
> - このメソッドは Slurm による複数インスタンスタイプ割り当てとは互換性がありません。

ODCRs のサポートが targeted AWS ParallelCluster 3.1.1 に追加されました。このリリースでは、EC2 の RunInstances パラメータをオーバーライドして、設定されている AWS ParallelClusterの各コンピューティングリソースに使用する予約に関する情報を渡すメカニズムが導入されました。このメカニズムは targeted ODCR と互換性があります。ただし、targeted ODCR を使用する場合は、run-instances オーバーライド設定を指定する必要があります。ターゲット ODCR は AWS CLI の Amazon EC2 run-instances コマンドで明示的に定義する必要があります。ODCR が であるかどうかを判断するopenか、 AWS CLI Amazon EC2 コマンド targetedを実行しますdescribe-capacity-reservation。

複数の ODCR を 1 つのリソースグループにまとめることができます。これを実行インスタンスオーバーライドで使用すると、同時に複数の ODCR をターゲットにすることができます。

targeted ODCR を使用している場合は、プレイスメントグループを指定できます。ただし、run-instances オーバーライド設定も指定する必要があります。

が ODCR targeted を AWS 作成したか、特定のリザーブドインスタンスのセットがあるとします。そうなると、プレイスメントグループを指定できなくなります。によって設定されたルールは、プレイスメントグループ設定と競合 AWS する可能性があります。そのため、アプリケーションにプレイスメントグループが必要な場合は、CPG ODCR を使用してください。いずれの場合も、run-instances オーバーライド設定を指定する必要があります。

CPG ODCR を使用している場合は、run-instances オーバーライド設定を指定し、クラスター設定でも同じプレイスメントグループを指定する必要があります。

でのリザーブドインスタンスの使用 AWS ParallelCluster

リザーブドインスタンスはキャパシティ予約 (ODCR) とは異なります。リザーブドインスタンスには 2 つのタイプがあります。リージョンのリザーブドインスタンスでは、キャパシティは予約されません。ゾーンのリザーブドインスタンスでは、指定されたアベイラビリティーゾーンでキャパシティが予約されます。

リージョンのリザーブドインスタンスがある場合、キャパシティが予約されず、容量不足エラーが発生する可能性があります。ゾーンのリザーブドインスタンスがある場合、キャパシティ予約はできますが、その指定に使用できる run-instances API パラメータはありません。

リザーブドインスタンスは任意の AWS ParallelCluster バージョンでサポートされています。で何も指定する必要はありません。 AWS ParallelCluster インスタンスが自動的に選択されます。

ゾーンのリザーブドインスタンスを使用するときは、クラスター設定でプレイスメントグループの指定を省略することで、発生する可能性のある容量不足エラーを回避できます。

廃止/非推奨 - オンデマンドキャパシティ予約 (ODCR) に AWS ParallelCluster 3 **targeted** で**RunInstances**カスタマイズを使用する

> **Warning**
>
> - AWS ParallelCluster バージョン 3.3.0 以降では、この方法はお勧めしません。このセクションは、以前のバージョンを使用した実装のリファレンスとして残しています。
> - このメソッドは Slurm による複数インスタンスタイプ割り当てとは互換性がありません。

クラスターキューに設定されているコンピューティングリソースごとに Amazon EC2 の `RunInstances` パラメータをオーバーライドできます。そのためには、以下のコードスニペットコンテンツを含む `/opt/slurm/etc/pcluster/run_instances_overrides.json` ファイルをクラスターのヘッドノードに作成します。

- `${queue_name}` はオーバーライドを適用するキューの名前です。
- `${compute_resource_name}` オーバーライドを適用するコンピューティングリソースです。
- `${overrides}` は、キューとインスタンスタイプの特定の組み合わせに使用する`RunInstances`オーバーライドのリストを含む任意の JSON オブジェクトです。オーバーライドの構文は、run_instances boto3 呼び出しで説明されているのと同じ仕様に従う必要があります。

```
{
    "${queue_name}": {
        "${compute_resource_name}": {
            ${overrides}
        },
        ...
    },
    ...
}
```

例えば、次の JSON は、`my-queue` と `my-compute-resource` で設定された `p4d.24xlarge` インスタンスに ODCR グループ `group_arn` を使用するように設定します。

```
{
    "my-queue": {
        "my-compute-resource": {
```

```
            "CapacityReservationSpecification": {
                "CapacityReservationTarget": {
                    "CapacityReservationResourceGroupArn": "group_arn"
                }
            }
        }
    }
}
```

この JSON ファイルが生成されると、クラスターのスケーリングを担当する AWS ParallelCluster デーモンは、インスタンス起動のオーバーライド設定を自動的に使用します。指定したパラメータがインスタンスのプロビジョニングに使用されていることを確認するには、以下のログファイルを確認してください。

- `/var/log/parallelcluster/clustermgtd` (静的容量)
- `/var/log/parallelcluster/slurm_resume.log` (動的容量)

パラメータが正しければ、次の内容を含むログエントリが見つかります。

```
Found RunInstances parameters override. Launching instances with: <parameters_list>
```

廃止/非推奨 – **targeted** オンデマンドキャパシティ予約 (ODCR) を使用してクラスターを作成する

> **Warning**
>
> - AWS ParallelCluster バージョン 3.3.0 以降では、この方法はお勧めしません。このセクションは、以前のバージョンを使用した実装のリファレンスとして残しています。
> - このメソッドは Slurm による複数のインスタンスタイプの割り当て と互換性がありません。

1. リソースグループを作成し、容量をグループ化します。

```
$ aws resource-groups create-group --name EC2CRGroup \
    --configuration '{"Type":"AWS::EC2::CapacityReservationPool"}'
 '{"Type":"AWS::ResourceGroups::Generic", "Parameters": [{"Name": "allowed-
resource-types", "Values": ["AWS::EC2::CapacityReservation"]}]}'
```

> **Note**
>
> リソースグループは、他のアカウントが共有するリソースをサポートしていません。ターゲット ODCR が別のアカウントで共有されている場合は、リソースグループを作成する必要はありません。ステップ 3 では、リソースグループの代わりに `CapacityReservationId` を使用します。

```
#!/bin/bash
set -e

# Override run_instance attributes
cat > /opt/slurm/etc/pcluster/run_instances_overrides.json << EOF
{
    "my-queue": {
        "my-compute-resource": {
            "CapacityReservationSpecification": {
                "CapacityReservationTarget": {
                    "CapacityReservationId": "cr-abcdef01234567890"
                }
            }
        }
    }
}
EOF
```

リソースグループにキャパシティ予約を追加します。新しい ODCR を作成するたびに、それをグループ予約に追加します。*ACCOUNT_ID* をアカウント ID、*PLACEHOLDER_CAPACITY_RESERVATION*をキャパシティ予約 ID、 を AWS リージョン ID (us-east-1 など) *REGION_ID*に置き換えます。

```
$ aws resource-groups group-resources --region REGION_ID --group EC2CRGroup \
    --resource-arns arn:aws:ec2:REGION_ID:ACCOUNT_ID:capacity-
reservation/PLACEHOLDER_CAPACITY_RESERVATION
```

ローカルコンピュータでポリシードキュメントを作成します。をアカウント ID *ACCOUNT_ID*に、 を AWS リージョン ID (us-east-1 など) *REGION_ID*に置き換えます。

```
cat > policy.json << EOF
```

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RunInstancesInCapacityReservation",
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": [
                "arn:aws:ec2:REGION_ID:ACCOUNT_ID:capacity-reservation/*",
                "arn:aws:resource-groups:REGION_ID:ACCOUNT_ID:group/*"
            ]
        }
    ]
}
EOF
```

2. 作成した json ファイル AWS アカウント を使用して、 に IAM ポリシーを作成します。

```
$ aws iam create-policy --policy-name RunInstancesCapacityReservation --policy-
document file://policy.json
```

3. 次のポストインストールスクリプトをインスタンス上でローカルに作成し、**postinstall.sh** という名前を付けます。

   を AWS アカウント ID *ACCOUNT_ID*に、 を AWS リージョン ID (us-east-1 など) *REGION_ID*に置き換えます。

```
#!/bin/bash
set -e

# Override run_instance attributes
cat > /opt/slurm/etc/pcluster/run_instances_overrides.json << EOF
{
    "my-queue": {
        "my-compute-resource": {
            "CapacityReservationSpecification": {
                "CapacityReservationTarget": {
                    "CapacityReservationResourceGroupArn": "arn:aws:resource-
groups:REGION_ID:ACCOUNT_ID:group/EC2CRGroup"
                }
            }
        }
    }
```

```
}
EOF
```

Amazon S3 バケットにファイルをアップロードします。*amzn-s3-demo-bucket* は、特定の S3 バケット名に置き換えます。

```
$ aws s3 mb s3://amzn-s3-demo-bucket
aws s3 cp postinstall.sh s3://amzn-s3-demo-bucket/postinstall.sh
```

4. プレースホルダーを独自の値に置き換えて、ローカルクラスター設定を作成します。

```
Region: REGION_ID
Image:
  Os: alinux2
HeadNode:
  InstanceType: c5.2xlarge
  Ssh:
    KeyName: YOUR_SSH_KEY
  Iam:
    S3Access:
      - BucketName: amzn-s3-demo-bucket
    AdditionalIamPolicies:
      - Policy: arn:aws:iam::ACCOUNT_ID:policy/RunInstancesCapacityReservation
  ## This post-install script is executed after the node is configured.
  ## It is used to install scripts at boot time and specific configurations
  ## In the script below we are overriding the calls to RunInstance to force
  ## the provisioning of our my-queue partition to go through
  ## the On-Demand Capacity Reservation
  CustomActions:
    OnNodeConfigured:
      Script: s3://amzn-s3-demo-bucket/postinstall.sh
  Networking:
    SubnetId: YOUR_PUBLIC_SUBNET_IN_TARGET_AZ

Scheduling:
  Scheduler: slurm
  SlurmQueues:
    - Name: my-queue
      ComputeResources:
        - MinCount: 0
          MaxCount: 100
          InstanceType: p4d.24xlarge
          Name: my-compute-resource
```

```
          Efa:
            Enabled: true
      Networking:
        ## PlacementGroup:
        ##   Enabled: true ## Keep PG disabled if using targeted ODCR
        SubnetIds:
          - YOUR_PRIVATE_SUBNET_IN_TARGET_AZ
```

5. クラスターを作成します。

   次のコマンドを使用してクラスターを作成します。*cluster-config.yaml* を構成ファイル名に、*cluster-dl* をクラスター名に、*REGION_ID* をリージョン ID (us-east-1 など) に置き換えます。

```
$ pcluster create-cluster --cluster-configuration cluster-config.yaml --cluster-
name cluster-dl --region REGION_ID
```

   クラスターが作成されると、ポストインストールスクリプトはヘッドノードで実行されます。このスクリプトは `run_instances_overrides.json` ファイルを作成し、`RunInstances` の呼び出しをオーバーライドして、パーティションのプロビジョニングがオンデマンドキャパシティ予約を経由するように強制します。

   クラスターのスケーリングを担当する AWS ParallelCluster デーモンは、起動された新しいインスタンスに対してこの設定を自動的に使用します。指定したパラメータがインスタンスのプロビジョニングに使用されていることを確認するには、以下のログファイルを確認します。

   - `/var/log/parallelcluster/clustermgtd` (静的容量 - `MinCount > 0`)
   - `/var/log/parallelcluster/slurm_resume.log` (動的容量)

   パラメータが正しければ、ログエントリには以下が含まれます。

```
Found RunInstances parameters override. Launching instances with: <parameters_list>
```

`RunInstances` オーバーライドの更新

生成された JSON 設定は、コンピューティングフリートを停止せずにいつでも更新できます。変更が適用されると、すべての新しいインスタンスは更新された設定で起動します。実行中のノードに更新された設定を適用する必要がある場合は、インスタンスの終了を強制してノードをリサイクルし、

がそれらのノードを置き換える AWS ParallelCluster のを待ちます。そのためには、Amazon EC2 コンソールまたは AWS CLIからインスタンスを終了するか、Slurm ノードを DOWN 状態または DRAIN 状態に設定します。

次のコマンドを使用して、Slurm ノードを DOWN または DRAIN に設定します。

```
$ scontrol update nodename=my-queue-dy-my-compute-resource-1 state=down
 reason=your_reason
scontrol update nodename=my-queue-dy-my-compute-resource-1 state=drain
 reason=your_reason
```

キャパシティブロック (CB) を使用してインスタンスを起動する

AWS ParallelCluster は、機械学習のオンデマンドキャパシティ予約 (ODCR) とキャパシティブロック (CB) をサポートしています。 Machine Learning ODCR とは異なり、CB は将来の開始時間を持つ場合があり、期限があります。ODCR を使用した起動の詳細については、「オンデマンドキャパシティ予約 (ODCR) を使用したインスタンスの起動」を参照してください。

での CB の使用 AWS ParallelCluster

CB を使用するように新規または既存のクラスターを設定するには、まず AWS アカウントに有効な CB が必要です。 AWS Management Console、 AWS Command Line Interface、または SDK を使用して、公式ドキュメントに従って利用可能な CB を検索して購入できます。有効な CB を取得したら、 AWS ParallelCluster 設定ファイルに CB Amazon リソースネーム (ARN) と関連するパラメータを設定できます。詳細については、「キャパシティブロック (CB) の検索と購入」を参照してください。

クラスター設定の CB

特定のキューで CB を使用するには、`CapacityReservationId` パラメータを使用する必要があります。それを既存の CB ID に設定します。CB ARN は AWS Management Console、 AWS CLI、または CB の作成に使用した SDK から取得できます。

CB を使用する先のキューに `CapacityType = CAPACITY_BLOCK` を設定する必要があります。それをコンピューティングリソースの `InstanceType` (CB と同じ Amazon Elastic Compute Cloud インスタンスタイプ) に設定します。

`CapacityReservationId` をコンピューティングリソースレベルで指定した場合は、予約から自動的に取得されるため、`InstanceType` はオプションです。

`CapacityType = CAPACITY_BLOCK` を使用する場合、CB 予約の一部であるすべてのインスタンスは静的ノードとして管理されるため、`MaxCount` は `MinCount` と等しく、0 より大きい必要があります。

クラスターの作成時に、ヘッドノードはすべての静的ノードの準備が整うのを待ってから、クラスターの作成の成功を通知します。ただし、`CapacityType = CAPACITY_BLOCK` を使用する場合は、関連するコンピューティングリソースの一部であるノードは、このチェックでは考慮されません。設定されているノードの一部がアクティブでなくても、クラスターは作成されます。

次の設定ファイルスニペットは、 AWS ParallelCluster 設定ファイルで を有効にするために必要なパラメータを示しています。

```
SlurmQueues:
 - Name: string
   CapacityType: CAPACITY_BLOCK
   ComputeResources:
   - Name: string
     InstanceType: String (EC2 Instance type of the CB)
     MinCount: integer (<= total capacity of the CB)
     MaxCount: integer (equal to MinCount)
     CapacityReservationTarget:
        CapacityReservationId: String (CB id)
```

がキャパシティブロック (CB) AWS ParallelCluster を使用する方法

AWS ParallelCluster は、 に関連付けられた静的ノードを特殊な方法で管理します。 は、CB がまだアクティブでなくてもクラスター AWS ParallelCluster を作成し、CB がアクティブになるとインスタンスが自動的に起動されます。

関連するコンピューティングリソースに対応する Slurm ノードは、まだアクティブではない場合、CB 開始時刻に達するまでメンテナンス状態になります。Slurm ノードは予約/メンテナンス状態のままで、slurm 管理者ユーザーに関連付けられます。つまり、ジョブを受け入れることはできますが、予約が削除されるまで、ジョブは `pending` 状態になります。

AWS ParallelCluster はSlurm予約を自動的に更新し、関連する CB ノードをメンテナンス (CB 状態に対応) します。CB がアクティブになると、Slurm 予約が削除され、ノードが起動して、保留中のジョブまたは新しいジョブの送信に利用可能になります。

CB の終了時間に達すると、ノードは予約/メンテナンス状態に戻ります。CB がアクティブでなくなり、インスタンスが終了したときに、ジョブを新しいキュー/コンピューティングリソースに再送信/再キューするかどうかは、ユーザーが決めます。

AMI のパッチ適用と Amazon EC2 インスタンスの置換

動的に起動されたすべてのクラスターコンピューティングノードが一貫した方法で動作するように、はクラスターインスタンスの自動 OS 更新 AWS ParallelCluster を無効にします。さらに、 AWS ParallelCluster および関連する CLI のバージョンごとに特定の AWS ParallelCluster AMIs セットが構築されます。この特定の AMIs セットは変更されず、リリースされた AWS ParallelCluster バージョンの AWS ParallelCluster AMIs 用に構築されたバージョンでのみサポートされます。AMI は更新されません。

ただし、緊急のセキュリティ問題により、お客様はこれらの AMI にパッチを追加し、パッチが適用された AMI でクラスターを更新したい場合があります。これはAWS ParallelCluster 責任共有モデルと一致しています。

現在使用している CLI AWS ParallelCluster バージョンでサポートされている特定の AWS ParallelCluster AMIs のセットを表示するには、以下を実行します。

```
$ pcluster version
$ pcluster list-official-images
```

AWS ParallelCluster ヘッドノードは静的インスタンスであり、手動で更新できます。ヘッドノードの再起動と再起動は、 AWS ParallelCluster バージョン 3.0.0 以降で完全にサポートされています。

インスタンスにエフェメラルインスタンスストアがある場合は、手動で更新する前に必ずインスタンスストアデータを保存する必要があります。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「HeadNode/LocalStorage/EphemeralVolume クラスター設定」と「インスタンスストアボリュームによるインスタンスタイプ」を参照してください。

コンピューティングノードはエフェメラルインスタンスです。デフォルトでは、ヘッドノードからのみアクセスできます。 AWS ParallelCluster バージョン 3.0.0 以降では、 でコンピューティングフリートを停止した後、/SchedulingSlurmQueues/// Image CustomAmiパラメータを変更して pcluster update-cluster コマンドを実行することで、コンピューティングインスタンスに関連付けられた AMI を更新できますpcluster update-compute-fleet。

```
$ pcluster update-compute-fleet-status --status STOP_REQUESTED
```

以下の方法のいずれかを使用して、コンピューティングノード用の更新されたカスタム AMI の作成を自動化できます。

- pcluster build-image コマンドを、更新された Build/ParentImage と共に使用する。

- Build/UpdateOsPackages/Enabled: true を使用してビルドを実行する。

ヘッドノードインスタンスの更新または交換

状況によっては、ヘッドノードの再開または再起動が必要になることがあります。例えば、OS を手動で更新する場合や、ヘッドノードインスタンスの再起動を強制する AWS インスタンスのリタイアの予定がある場合などに必要です。

インスタンスにエフェメラルドライブがない場合は、その後いつでも再び起動できます。リタイアが予定されている場合、停止したインスタンスを起動すると、新しいハードウェアを使用するように移行されます。

同様に、インスタンスストアがないインスタンスは手動で停止して起動できます。この場合や、エフェメラルボリュームのない他のインスタンスについては、クラスターのヘッドノードを停止して起動する に進んでください。

インスタンスにエフェメラルドライブがあり、停止されている場合、インスタンスストアのデータは失われます。ヘッドノードに使用されているインスタンスタイプにインスタンスストアがあるかどうかは、インスタンスストアボリュームにあるテーブルで確認できます。

エフェメラルドライブからデータを保存する

AWS ParallelCluster バージョン 3.0.0 以降、ヘッドノードの再起動と再起動はすべてのインスタンスタイプで完全にサポートされています。ただし、インスタンスにエフェメラルドライブがあると、そのデータは失われます。ヘッドノードを再開または再起動する前に、次の手順に従ってデータを保存します。

保存する必要のあるデータがあるかどうかを確認するには、EphemeralVolume/MountDir フォルダー (デフォルトは /scratch) の内容を確認します。

データは、ルートボリュームまたはクラスターに接続されている共有ストレージシステム (Amazon FSx、Amazon EFS、Amazon EBS など) に転送できます。リモートストレージへのデータ転送には追加コストが発生する可能性があることに注意してください。

データを保存したら、続けて、クラスターのヘッドノードを停止して起動する を行います。

クラスターのヘッドノードを停止して起動する

1. クラスターに実行中のジョブがないことを確認します。

Slurm スケジューラーを使用する場合。

- `sbatch --no-requeue` オプションが指定されていない場合、実行中のジョブはキューに入れられます。
- `--no-requeue` オプションを指定すると、実行中のジョブは失敗します。

2. クラスターコンピューティングフリートの停止を要求します。

```
$ pcluster update-compute-fleet --cluster-name cluster-name --status STOP_REQUESTED
{
  "status": "STOP_REQUESTED",
  ...
}
```

3. コンピューティングフリートのステータスが STOPPED になるまで待ちます。

```
$ pcluster update-compute-fleet --cluster-name cluster-name --status STOP_REQUESTED
{
  "status": "STOPPED",
  ...
}
```

4. OS の再起動またはインスタンスの再起動による手動更新には、 AWS Management Console または を使用できます AWS CLI。以下に示しているのは、 AWS CLIを使用した例です。

```
# Retrieve head node instance id
$ pcluster describe-cluster --cluster-name cluster-name --status STOP_REQUESTED
{
  "headNode": {
  "instanceId": "i-1234567890abcdef0",
  ...
},
  ...
}
# stop and start the instance
$ aws ec2 stop-instances --instance-ids 1234567890abcdef0
{
  "StoppingInstances": [
    {
      "CurrentState": {
        "Name": "stopping"
        ...
```

```
      },
      "InstanceId": "i-1234567890abcdef0",
      "PreviousState": {
        "Name": "running"
        ...
      }
    }
  ]
}
$ aws ec2 start-instances --instance-ids 1234567890abcdef0
{
  "StartingInstances": [
    {
      "CurrentState": {
        "Name": "pending"
        ...
      },
      "InstanceId": "i-1234567890abcdef0",
      "PreviousState": {
        "Name": "stopped"
        ...
      }
    }
  ]
}
```

5. クラスターコンピューティングフリートの開始。

```
$ pcluster update-compute-fleet --cluster-name cluster-name --status
 START_REQUESTED
{
  "status": "START_REQUESTED",
  ...
}
```

オペレーティングシステム

AWS ParallelCluster は、Amazon Linux 2、Amazon Linux 2023、Ubuntu24.04、Ubuntu 22.04、Ubuntu 20.04、Red Hat Enterprise Linux 8 (RHEL8)、Rocky 8、Red Hat Enterprise Linux 9 (RHEL9)、および Rocky 9. AWS ParallelCluster をサポートする一部のオペレーティングシステ

ム用の構築済み AMI をサポートしています。 が提供する AMIs の詳細については、 AMIs AWS ParallelCluster 「」を参照してください`Image` セクション。

オペレーティングシステムに関する考慮事項

Ubuntu 22.04 および Ubuntu 24.04

Ubuntu 22.04 および Ubuntu 24.04 では、ssh 用により安全なキーが必要であり、デフォルトでは RSA キーをサポートしていません。ed25519 キーを生成し、それをクラスターの作成に使用してください。

Ubuntu 22.04 は、そのカーネルに FSx クライアントがないため、最新のカーネルに更新できません。

RHEL 8

RedHat Enterprise Linux 8.7 (rhel8) が AWS ParallelCluster バージョン 3.6.0 から追加されました。rhel8 を使用するようにクラスターを設定した場合、任意のインスタンスタイプのオンデマンドコストは、他のサポートされているオペレーティングシステムを使用するようにクラスターを設定した場合よりも高くなります。

料金の詳細については、「オンデマンド料金」と「Amazon EC2 での Red Hat Enterprise Linux が提供される方法と料金体系について教えてください」を参照してください。

Rocky 8

AWS ParallelCluster 3.8.0 は Rocky Linux 8 をサポートしていますが、構築済みの Rocky Linux 8 AMIs (x86 および ARM アーキテクチャ用) は使用できません。 AWS ParallelCluster 3.8.0 は、CustomAmi プロパティを使用してカスタム AMIs を使用して Rocky Linux 8 でクラスターを作成することをサポートしています。カスタム AMI の構築の詳細については、「AWS ParallelCluster AMI のカスタマイズ」を参照してください。

ベースの Rocky Linux 8 AMI からカスタム AMI を構築するには、 AWS Marketplace で利用可能な Rocky Linux 8 AMIs へのサブスクライブを検討してください。 AWS Marketplace で Rocky Linux 8 AMIs の料金とサブスクリプションコストを確認してください。または、公式の Rocky Linux 8 AMIs をベース AMI として使用することもできます。

Rocky9

AWS ParallelCluster 3.9.0 は Rocky Linux 9 をサポートしていますが、構築済みの Rocky Linux 9 AMIs (x86 および ARM アーキテクチャ用) は使用できません。 AWS ParallelCluster 3.9.0

はCustomAmi プロパティを使用してカスタム AMIs を使用して Rocky Linux 9 でクラスターを作成することをサポートしています。カスタム AMI の構築の詳細については、「AWS ParallelCluster AMI のカスタマイズ」を参照してください。ベース Rocky Linux 9 AMI からカスタム AMI を構築するには、公式の Rocky Linux 9 AMIs をベース AMI として使用することもできます。ベースの AMI に最新のカーネルがない場合、カスタム Rocky Linux 9 AMI の構築は失敗する可能性があります。AMI を構築する前にカーネルをアップグレードするには、以下の方法に従います。

- こちら (https://rockylinux.org/cloud-images/) から Rocky9 AMI ID を使用してインスタンスを起動します。
- インスタンスに SSH で接続し、`sudo yum -y update` コマンドを実行します。
- インスタンスからイメージを作成し、`ParentImage` として使用します。

のリファレンス AWS ParallelCluster

トピック

- AWS ParallelCluster CLI コマンド
- 設定ファイル
- AWS ParallelCluster API リファレンス
- AWS ParallelCluster Python ライブラリ API

AWS ParallelCluster CLI コマンド

`pcluster` はプライマリ AWS ParallelCluster CLI コマンドです。`pcluster` を使用して、 で HPC クラスターを起動および管理 AWS クラウド し、カスタム AMI イメージを作成および管理します。

`pcluster3-config-converter` は、バージョン 2 形式のクラスター設定 AWS ParallelCluster を AWS ParallelCluster バージョン 3 形式に変換するために使用されます。

```
pcluster [-h] ( build-image | configure |
                create-cluster | dcv-connect |
                delete-cluster | delete-cluster-instances | delete-image |
                describe-cluster | describe-cluster-instances |
                describe-compute-fleet | describe-image |
                export-cluster-logs | export-image-logs |
                get-cluster-log-events | get-cluster-stack-events |
                get-image-log-events | get-image-stack-events |
                list-cluster-log-streams | list-clusters |
                list-images | list-image-log-streams | list-official-images |
                ssh | update-cluster |
                update-compute-fleet | version ) ...
pcluster3-config-converter [-h] [-t CLUSTER_TEMPLATE]
                [-c CONFIG_FILE]
                [--force-convert]
                [-o OUTPUT_FILE]
```

トピック

- pcluster
- pcluster3-config-converter

pcluster

pcluster はプライマリ AWS ParallelCluster CLI コマンドです。pcluster は、 AWS クラウド内の HPC クラスターを起動し、管理するために使用します。

pcluster は /home/user/.parallelcluster/ の pcluster.log.# ファイルにコマンドのログを書き込みます。詳細については、「pcluster CLI ログ」を参照してください。

pcluster の実行に必要なアクセス許可を持つ IAM ロールが必要です。

```
pcluster [-h]
```

引数

pcluster ***command***

可能な選択肢: build-image | configure | create-cluster | dcv-connect | delete-cluster | delete-cluster-instances | delete-image | describe-cluster | describe-cluster-instances | describe-compute-fleet | describe-image | export-cluster-logs | export-image-logs get-cluster-log-events || get-cluster-stack-events | get-image-log-events | get-image-stack-events || list-clusters | list-cluster-log-streams | list-images | ssh || list-image-log-streams list-official-images || update-cluster update-compute-fleet | version

サブコマンド：

トピック

- pcluster build-image
- pcluster configure
- pcluster create-cluster
- pcluster dcv-connect
- pcluster delete-cluster
- pcluster delete-cluster-instances
- pcluster delete-image

- pcluster describe-cluster
- pcluster describe-cluster-instances
- pcluster describe-compute-fleet
- pcluster describe-image
- pcluster export-cluster-logs
- pcluster export-image-logs
- pcluster get-cluster-log-events
- pcluster get-cluster-stack-events
- pcluster get-image-log-events
- pcluster get-image-stack-events
- pcluster list-clusters
- pcluster list-cluster-log-streams
- pcluster list-images
- pcluster list-image-log-streams
- pcluster list-official-images
- pcluster ssh
- pcluster update-cluster
- pcluster update-compute-fleet
- pcluster version

pcluster build-image

指定されたリージョンにカスタム AWS ParallelCluster イメージを作成します。

```
pcluster build-image [-h]
                  --image-configuration IMAGE_CONFIGURATION
                  --image-id IMAGE_ID
                 [--debug]
                 [--dryrun DRYRUN]
                 [--query QUERY]
                 [--region REGION]
                 [--rollback-on-failure ROLLBACK_ON_FAILURE]
```

```
                [--suppress-validators SUPPRESS_VALIDATORS [SUPPRESS_VALIDATORS ...]]
                [--validation-failure-level {INFO,WARNING,ERROR}]
```

名前付き引数

-h, --help

pcluster build-image のヘルプテキストを表示します。

--image-configuration, -c *IMAGE_CONFIGURATION*

イメージ設定ファイルを YAML ドキュメントで指定します。

--image-id, -i *IMAGE_ID*

構築されるイメージの ID を指定します。

--debug

デバッグログを有効にします。

--dryrun *DRYRUN*

true の場合、リソースを作成することなく、検証を行います。イメージの構成を検証するためにこれを使用することができます。(デフォルトは false です)

--query *QUERY*

出力時に実行する JMESPath クエリ。

--region, -r *REGION*

AWS リージョン 使用する を指定します。イメージ設定ファイルのリージョン設定、AWS_DEFAULT_REGION環境変数、~/.aws/configファイルの [default]セクションのregion設定、または --regionパラメータを使用して、 を指定 AWS リージョン する必要があります。

--rollback-on-failure *ROLLBACK_ON_FAILURE*

true の場合、失敗するとイメージスタックのロールバックが自動的に開始されます。(デフォルトは false です)

--suppress-validators *SUPPRESS_VALIDATORS* [*SUPPRESS_VALIDATORS ...*]

抑制する 1 つまたは複数のコンフィグバリデータを指定します。

形式: (ALL|type:[A-Za-z0-9]+)

--validation-failure-level {INFO,WARNING,ERROR}

作成が失敗する最小の検証レベルを指定します。(デフォルトは ERROR です)

AWS ParallelCluster バージョン 3.1.2 の使用例：

```
$ pcluster build-image --image-configuration image-config.yaml --image-id custom-alinux2-image
{
  "image": {
    "imageId": "custom-alinux2-image",
    "imageBuildStatus": "BUILD_IN_PROGRESS",
    "cloudformationStackStatus": "CREATE_IN_PROGRESS",
    "cloudformationStackArn": "arn:aws:cloudformation:us-east-1:123456789012:stack/
custom-alinux2-image/1234abcd-56ef-78gh-90ij-abcd1234efgh",
    "region": "us-east-1",
    "version": "3.1.2"
  }
}
```

> **Warning**
>
> `pcluster build-image` はデフォルトの VPC を使用します。 AWS Control Tower または AWS Landing Zone を使用してデフォルトの VPC が削除された場合は、イメージ設定ファイルでサブネット ID を指定する必要があります。詳細については、「SubnetId」を参照してください。

pcluster configure

AWS ParallelCluster バージョン 3 のインタラクティブ設定ウィザードを開始します。詳細については、「「Configure and create a cluster with the AWS ParallelCluster command line interface」」を参照してください。

```
pcluster configure [-h]
                 --config CONFIG
                [--debug]
                [--region REGION]
```

名前付き引数

-h, --help

pcluster configure のヘルプテキストを表示します。

--config *CONFIG*

生成された設定ファイルを出力するパス。

--debug

デバッグログを有効にします。

--region, -r *REGION*

AWS リージョン 使用する を指定します。リージョンは、イメージ設定ファイルのリージョン設定、環境変数 AWS_DEFAULT_REGION、~/.aws/config ファイルの [default] セクションにある region 設定、または --region パラメータを使って指定する必要があります。

pcluster create-cluster

AWS ParallelCluster クラスターを作成します。

```
pcluster create-cluster [-h]
                 --cluster-configuration CLUSTER_CONFIGURATION
                 --cluster-name CLUSTER_NAME
                [--debug]
                [--dryrun DRYRUN]
                [--query QUERY]
                [--region REGION]
                [--rollback-on-failure ROLLBACK_ON_FAILURE]
                [--suppress-validators SUPPRESS_VALIDATORS [SUPPRESS_VALIDATORS ...]]

                [--validation-failure-level {INFO,WARNING,ERROR}]
```

名前付き引数

-h, --help

pcluster create-cluster のヘルプテキストを表示します。

--cluster-configuration, -c *CLUSTER_CONFIGURATION*

YAML クラスター設定ファイルを指定します。

--cluster-name, -n *CLUSTER_NAME*

作成するクラスターの名前を指定します。

名前の最初は、アルファベットで始まっている必要があります。名前は最大 60 文字です。Slurm アカウンティングが有効になっている場合、名前の最大長は 40 文字です。

有効な文字は、A ~ Z、a ~ z、0 ~ 9、- (ハイフン) です。

--debug

デバッグログの有効化

--dryrun *DRYRUN*

`true` の場合、コマンドはリソースを作成することなく、検証を行います。クラスターの構成を検証するためにこれを使用することができます。(デフォルトは `false` です)

--query *QUERY*

出力に対して実行する JMESPath クエリを指定します。

--region, -r *REGION*

AWS リージョン 使用する を指定します。クラスター設定ファイルの Region設定、`AWS_DEFAULT_REGION`環境変数、 `~/.aws/config` ファイルの `[default]`セクションの `region`設定、または `--region`パラメータを使用して、 を指定 AWS リージョン する必要があります。

--rollback-on-failure *ROLLBACK_ON_FAILURE*

`true` の場合、障害発生時に自動的にクラスタースタックのロールバックを開始します。(デフォルトは `true` です)

--suppress-validators *SUPPRESS_VALIDATORS* [*SUPPRESS_VALIDATORS* ...]

抑制する 1 つまたは複数のコンフィグバリデータを指定します。

形式: (`ALL`|タイプ: `[A-Za-z0-9]+`)

--validation-failure-level {INFO,WARNING,ERROR}

作成が失敗する最小の検証レベルを指定します。(デフォルトは `ERROR` です)

AWS ParallelCluster バージョン 3.1.4 の使用例：

```
$ pcluster create-cluster -c cluster-config.yaml -n cluster-v3
{
  "cluster": {
    "clusterName": "cluster-v3",
    "cloudformationStackStatus": "CREATE_IN_PROGRESS",
    "cloudformationStackArn": "arn:aws:cloudformation:us-east-1:123456789012:stack/
cluster-v3/1234abcd-56ef-78gh-90ij-abcd1234efgh",
    "region": "us-east-1",
    "version": "3.1.4",
    "clusterStatus": "CREATE_IN_PROGRESS"
  }
}
```

pcluster dcv-connect

Amazon DCV を使用してインタラクティブセッションからヘッドノードに接続することを許可します。

```
pcluster dcv-connect [-h]
                 --cluster-name CLUSTER_NAME
                [--debug]
                [--key-path KEY_PATH]
                [--login-node-ip LOGIN_NODE_IP]
                [--region REGION]
                [--show-url]
```

名前付き引数

-h, --help

`pcluster dcv-connect` のヘルプテキストを表示します。

--cluster-name, -n *CLUSTER_NAME*

クラスターの名前を指定します。

--debug

デバッグログの有効化

--key-path *KEY_PATH*

接続に使用する SSH キーのパスを指定します。

`--login-node-ip`

クラスター内のログインノードのパブリック IP アドレスまたはプライベート IP アドレスを指定します。この引数を使用すると、DCV が有効になっているクラスター内のログインノードに接続できます。

> **Note**
>
> この引数は AWS ParallelCluster バージョン 3.11.0 で追加されています。

`--region, -r REGION`

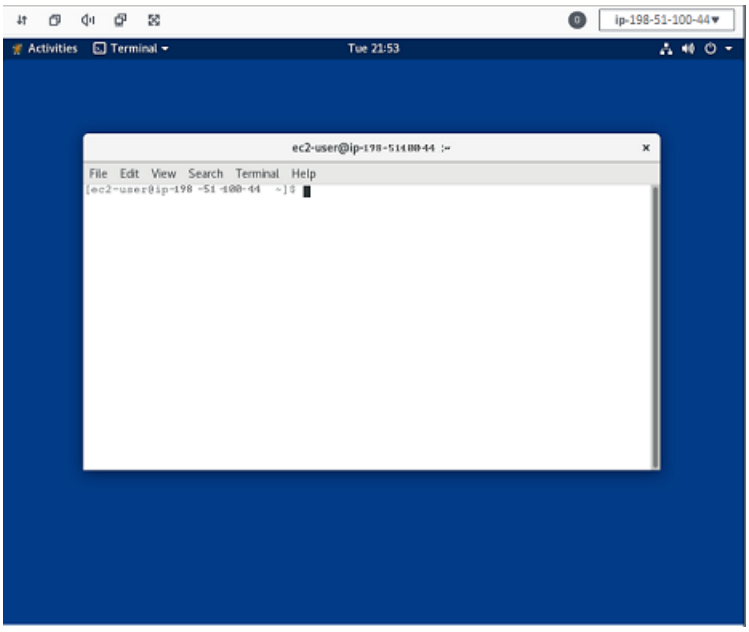
AWS リージョン 使用する を指定します。`AWS_DEFAULT_REGION` 環境変数、 `~/.aws/config` ファイルの `[default]`セクションの `region`設定、または `--region`パラメータを使用して、を指定 AWS リージョン する必要があります。

`--show-url`

DCV の接続に使用される URL を出力して終了します。

AWS ParallelCluster バージョン 3.11 の使用例

```
$ pcluster dcv-connect -n cluster-3Dcv --login-node-ip 198.51.100.44 -r us-east-1 --
key-path /home/user/.ssh/key.pem
```



pcluster delete-cluster

クラスターの削除を開始します。

```
pcluster delete-cluster [-h]
                 --cluster-name CLUSTER_NAME
                [--debug]
                [--query QUERY]
                [--region REGION]
```

名前付き引数

-h, --help

pcluster delete-cluster のヘルプテキストを表示します。

--cluster-name, -n *CLUSTER_NAME*

クラスターの名前を指定します。

--debug

デバッグログの有効化

--query *QUERY*

出力に対して実行する JMESPath クエリを指定します。

--region, -r *REGION*

AWS リージョン 使用する を指定します。リージョンは、AWS_DEFAULT_REGION の環境変数、~/.aws/config ファイルの [default] セクションにある region 設定、または --region のパラメータを使って指定する必要があります。

AWS ParallelCluster バージョン 3.1.4 の使用例：

```
$ pcluster delete-cluster -n cluster-v3
   {
  "cluster": {
    "clusterName": "cluster-v3",
    "cloudformationStackStatus": "DELETE_IN_PROGRESS",
    "cloudformationStackArn": "arn:aws:cloudformation:us-east-1:123456789012:stack/
cluster-v3/1234abcd-56ef-78gh-90ij-abcd1234efgh",
    "region": "us-east-1",
```

```
    "version": "3.1.4",
    "clusterStatus": "DELETE_IN_PROGRESS"
  }
}
```

pcluster delete-cluster-instances

すべてのクラスターコンピューティングノードの強制終了を開始します。これは AWS Batch クラスターでは機能しません。

```
pcluster delete-cluster-instances [-h]
                 --cluster-name CLUSTER_NAME
                [--debug]
                [--force FORCE]
                [--query QUERY]
                [--region REGION]
```

名前付き引数

-h, --help

pcluster delete-cluster-instances のヘルプテキストを表示します。

--cluster-name, -n *CLUSTER_NAME*

クラスターの名前を指定します。

--debug

デバッグログの有効化

--force *FORCE*

true の場合、検証エラーを無視して強制的に削除します。(デフォルトは false です)

--query *QUERY*

出力に対して実行する JMESPath クエリを指定します。

--region, -r *REGION*

AWS リージョン 使用する を指定します。AWS_DEFAULT_REGION 環境変数、 ~/.aws/config ファイルの [default]セクションの region設定、または --regionパラメータを使用して、を指定 AWS リージョン する必要があります。

```
$ pcluster delete-cluster-instances -n cluster-v3
```

pcluster delete-image

カスタム AWS ParallelCluster イメージの削除を開始します。

```
pcluster delete-image [-h]
                 --image-id IMAGE_ID
                [--debug]
                [--force FORCE]
                [--query QUERY]
                [--region REGION]
```

名前付き引数

-h, --help

`pcluster delete-image` のヘルプテキストを表示します。

--image-id, -i *IMAGE_ID*

削除するイメージの ID を指定します。

--debug

デバッグログの有効化

--force *FORCE*

`true` の場合、AMI を使用しているインスタンスがある場合や、AMI が共有されている場合は、強制的に削除されます。(デフォルトは `false` です)

--query *QUERY*

出力に対して実行する JMESPath クエリを指定します。

--region, -r *REGION*

AWS リージョン 使用する を指定します。`AWS_DEFAULT_REGION` 環境変数、 `~/.aws/config` ファイルの `[default]`セクションの `region`設定、または `--region`パラメータを使用して、を指定 AWS リージョン する必要があります。

AWS ParallelCluster バージョン 3.1.4 の使用例：

```
$ pcluster delete-image --image-id custom-alinux2-image
```

```
{
  "image": {
    "imageId": "custom-alinux2-image",
    "imageBuildStatus": "DELETE_IN_PROGRESS",
    "region": "us-east-1",
    "version": "3.1.4"
  }
}
```

pcluster describe-cluster

クラスターの詳細情報を取得します。

```
pcluster describe-cluster [-h]
                 --cluster-name CLUSTER_NAME
                [--debug]
                [--query QUERY]
                [--region REGION]
```

名前付き引数

-h, --help

pcluster describe-cluster のヘルプテキストを表示します。

--cluster-name, -n *CLUSTER_NAME*

クラスターの名前を指定します。

--debug

デバッグログの有効化

--query *QUERY*

出力に対して実行する JMESPath クエリを指定します。

--region, -r *REGION*

AWS リージョン 使用する を指定します。AWS_DEFAULT_REGION 環境変数、 ~/.aws/config ファイルの [default]セクションの region設定、または --regionパラメータを使用して、を指定 AWS リージョン する必要があります。

AWS ParallelCluster バージョン 3.1.4 を使用した例：

クラスターの詳細を説明します。

```
$ pcluster describe-cluster -n cluster-v3
{
  "creationTime": "2022-07-12T17:19:16.101Z",
  "headNode": {
    "launchTime": "2022-07-12T17:22:21.000Z",
    "instanceId": "i-1234567890abcdef0",
    "publicIpAddress": "198.51.100.44",
    "instanceType": "t2.micro",
    "state": "running",
    "privateIpAddress": "192.0.2.0.196"
  },
  "loginNodes": [
      {
          "status": "active",
          "poolName": "pool1",
          "address": "cluster-v3-eMr9BYRKZVDa-e5bb34f40b24f51d.elb.us-
east-1.amazonaws.com",
          "scheme": "internet-facing",
          "healthyNodes": 1,
          "unhealthyNodes": 0
      },
      {
          "status": "active",
          "poolName": "pool2",
          "address": "cluster-v3-PaQ7GgC27sic-aba10c890247b36b.elb.us-
east-1.amazonaws.com",
          "scheme": "internet-facing",
          "healthyNodes": 1,
          "unhealthyNodes": 0
      }
  ],
  "version": "3.1.4",
  "clusterConfiguration": {
    "url": "https://parallelcluster-e5ca74255d6c3886-v1-do-not-delete..."
  },
  "tags": [
    {
      "value": "3.11",
      "key": "parallelcluster:version"
    }
  ],
  "cloudFormationStackStatus": "CREATE_COMPLETE",
```

```
  "clusterName": "cluster-v3",
  "computeFleetStatus": "RUNNING",
  "cloudformationStackArn": "arn:aws:cloudformation:us-east-1:123456789012:stack/
cluster-v3/1234abcd-56ef-78gh-90ij-abcd1234efgh",
  "lastUpdatedTime": "2022-07-12T17:19:16.101Z",
  "region": "us-east-1",
  "clusterStatus": "CREATE_COMPLETE"
}
```

describe-cluster を使用してクラスター設定を取得します。

```
$ curl -o - $(pcluster describe-cluster -n cluster-v3 --query clusterConfiguration.url
 | xargs echo)
Region: us-east-1
Image:
  Os: alinux2
HeadNode:
  InstanceType: t2.micro
  Networking:
    SubnetId: subnet-abcdef01234567890
  Ssh:
    KeyName: adpc
  Iam:
    S3Access:
      - BucketName: cluster-v3-bucket
        KeyName: logs
        EnableWriteAccess: true
Scheduling:
  Scheduler: slurm
  SlurmQueues:
  - Name: queue1
    ComputeResources:
    - Name: t2micro
      InstanceType: t2.micro
      MinCount: 0
      MaxCount: 10
    Networking:
      SubnetIds:
      - subnet-021345abcdef6789
```

pcluster describe-cluster-instances

クラスター内のインスタンスについて説明します。

```
pcluster describe-cluster-instances [-h]
                 --cluster-name CLUSTER_NAME
                [--debug]
                [--next-token NEXT_TOKEN]
                [--node-type {HeadNode,ComputeNode,LoginNode}]
                [--query QUERY]
                [--queue-name QUEUE_NAME]
                [--region REGION]
```

名前付き引数

-h, --help

pcluster describe-cluster-instances のヘルプテキストを表示します。

--cluster-name, -n *CLUSTER_NAME*

クラスターの名前を指定します。

--debug

デバッグログの有効化

--next-token *NEXT_TOKEN*

次の結果セットのトークン。

--node-type {HeadNode,ComputeNode,LoginNode}

リストするノードタイプを指定します。サポートされる値は HeadNode、ComputeNode、LoginNode です。このパラメータが指定されていない場合、HeadNode、ComputeNode、LoginNode のインスタンスが記述されます。

--query *QUERY*

出力に対して実行する JMESPath クエリを指定します。

--queue-name *QUEUE_NAME*

一覧表示するキューの名前を指定します。このパラメータが指定されていない場合、すべてのキューのインスタンスが記述されます。

--region, -r *REGION*

AWS リージョン 使用する を指定します。AWS_DEFAULT_REGION 環境変数、 ~/.aws/config ファイルの [default]セクションの region設定、または --regionパラメータを使用して、を指定 AWS リージョン する必要があります。

AWS ParallelCluster バージョン 3.1.4 の使用例：

```
$ pcluster describe-cluster-instances -n cluster-v3
{
  "instances": [
    {
      "launchTime": "2022-07-12T17:22:21.000Z",
      "instanceId": "i-1234567890abcdef0",
      "publicIpAddress": "198.51.100.44",
      "instanceType": "t2.micro",
      "state": "running",
      "nodeType": "HeadNode",
      "privateIpAddress": "192.0.2.0.196"
    },
    {
      "launchTime": "2022-07-12T17:37:42.000Z",
      "instanceId": "i-021345abcdef6789",
      "queueName": "queue1",
      "publicIpAddress": "198.51.100.44",
      "instanceType": "t2.micro",
      "state": "pending",
      "nodeType": "ComputeNode",
      "privateIpAddress": "192.0.2.0.196"
    },
    {
        "launchTime": "2022-07-12T17:37:42.000Z",
        "instanceId": "i-021345abcdef6789",
        "poolName": "pool1",
        "publicIpAddress": "198.51.100.44",
        "instanceType": "t2.micro",
        "state": "pending",
        "nodeType": "loginNode",
        "privateIpAddress": "192.0.2.0.196"
    }
  ]
}
```

pcluster describe-compute-fleet

コンピューティングフリートの状況について説明してください。

```
pcluster describe-compute-fleet [-h]
                 --cluster-name CLUSTER_NAME
```

```
                 [--debug]
                 [--query QUERY]
                 [--region REGION]
```

名前付き引数

-h, --help

pcluster describe-compute-fleet のヘルプテキストを表示します。

--cluster-name, -n *CLUSTER_NAME*

クラスターの名前を指定します。

--debug

デバッグログの有効化

--query *QUERY*

出力に対して実行する JMESPath クエリを指定します。

--region, -r *REGION*

AWS リージョン 使用する を指定します。AWS_DEFAULT_REGION 環境変数、 ~/.aws/config ファイルの [default]セクションの region設定、または --regionパラメータを使用して、を指定 AWS リージョン する必要があります。

AWS ParallelCluster バージョン 3.1.4 の使用例：

```
$ pcluster describe-compute-fleet -n pcluster-v3
{
  "status": "RUNNING",
  "lastStatusUpdatedTime": "2022-07-12T17:24:26.000Z"
}
```

pcluster describe-image

イメージの詳細情報を取得します。

```
pcluster describe-image [-h]
                 --image-id IMAGE_ID
                [--debug]
                [--query QUERY]
```

```
                 [--region REGION]
```

名前付き引数

-h, --help

pcluster describe-image のヘルプテキストを表示します。

--image-id, -i *IMAGE_ID*

イメージの ID を指定します。

--debug

デバッグログの有効化

--query *QUERY*

出力に対して実行する JMESPath クエリを指定します。

--region, -r *REGION*

AWS リージョン 使用する を指定します。AWS_DEFAULT_REGION 環境変数、 ~/.aws/config ファイルの [default]セクションの region設定、または --regionパラメータを使用して、を指定 AWS リージョン する必要があります。

AWS ParallelCluster バージョン 3.1.2 の使用例：

```
$ pcluster describe-image --image-id custom-alinux2-image
{
  "imageConfiguration: {
    "url": "https://parallelcluster-1234abcd5678-v1-do-not-delete.../configs/image-
config.yaml"
  },
  "imageId": "custom-alinux2-image",
  "creationTime": "2022-04-05T20:23:07.000Z"
  "imageBuildStatus": "BUILD_COMPLETE",
  "region": "us-east-1",
  "ec2AmiInfo": {
    "amiName": "custom-alinux2-image 2022-04-05T19-55-22.518Z",
    "amiId": "ami-1234abcd5678efgh",
    "description": "AWS ParallelCluster AMI for alinux2,
 kernel-4.14.268-205.500.amzn2.x86_64, lustre-2.10.8-5.amzn2.x86_64,
 efa-1.14.2-1.amzn2.x86_64, dcv-2021.3.11591-1.el7.x86_64, slurm-21-08-6-1",
    "state": "AVAILABLE",
```

```
  "tags": [
      {
        "value": "arn:aws:imagebuilder:us-east-1:123456789012:image/
parallelclusterimage-custom-alinux2-image/3.1.2/1",
        "key": "Ec2ImageBuilderArn"
      },
      {
        "value": "parallelcluster-1234abcd5678efgh-v1-do-not-delete",
        "key": "parallelcluster:amzn-s3-demo-bucket"
      },
      {
        "value": "custom-alinux2-image",
        "key": "parallelcluster:image_name"
      },
      {
        "value": "available",
        "key": "parallelcluster:build_status"
      },
      {
        "value": "s3://amzn-s3-demo-bucket/parallelcluster/3.1.2/images/custom-alinux2-
image-1234abcd5678efgh/configs/image-config.yaml",
        "key": "parallelcluster:build_config"
      },
      {
        "value": "Amazon EC2 Image Builder",
        "key": "CreatedBy"
      },
      {
        "value": "arn:aws:logs:us-east-1:123456789012:log-group:/aws/imagebuilder/
ParallelClusterImage-custom-alinux2-image",
        "key": "parallelcluster:build_log"
      },
      {
        "value": "4.14.268-205.500.amzn2.x86_64",
        "key": "parallelcluster:kernel_version"
      },
      {
        "value": "arn:aws:imagebuilder:us-east-1:444455556666:image/amazon-linux-2-
x86/2022.3.16/1",
        "key": "parallelcluster:parent_image"
      },
      {
        "value": "3.1.2",
        "key": "parallelcluster:version"
```

```
      },
      {
        "value": "0.5.14",
        "key": "parallelcluster:munge_version"
      },
      {
        "value": "21-08-6-1",
        "key": "parallelcluster:slurm_version"
      },
      {
        "value": "2021.3.11591-1.el7.x86_64",
        "key": "parallelcluster:dcv_version"
      },
      {
        "value": "alinux2-image",
        "key": "parallelcluster:image_id"
      },
      {
        "value": "3.2.3",
        "key": "parallelcluster:pmix_version"
      },
      {
        "value": "parallelcluster/3.13.2/images/alinux2-image-abcd1234efgh56781234",
        "key": "parallelcluster:s3_image_dir"
      },
      {
        "value": "1.14.2-1.amzn2.x86_64",
        "key": "parallelcluster:efa_version"
      },
      {
        "value": "alinux2",
        "key": "parallelcluster:os"
      },
      {
        "value": "aws-parallelcluster-cookbook-3.1.2",
        "key": "parallelcluster:bootstrap_file"
      },
      {
        "value": "1.8.23-10.amzn2.1.x86_64",
        "key": "parallelcluster:sudo_version"
      },
      {
        "value": "2.10.8-5.amzn2.x86_64",
        "key": "parallelcluster:lustre_version"
```

```
      }
    ],
    "architecture": "x86_64"
  },
  "version": "3.1.2"
}
```

pcluster export-cluster-logs

クラスターのログを Amazon S3 バケットを経由して、ローカルの `tar.gz` アーカイブにエクスポートします。

```
pcluster export-cluster-logs [-h]
                 --cluster-name CLUSTER_NAME
                [--bucket BUCKET_NAME]
                [--bucket-prefix BUCKET_PREFIX]
                [--debug]
                [--end-time END_TIME]
                [--filters FILTER [FILTER ...]]
                [--keep-s3-objects KEEP_S3_OBJECTS]
                [--output-file OUTPUT_FILE]
                [--region REGION]
                [--start-time START_TIME]
```

> **Note**
>
> `export-cluster-logs` コマンドは CloudWatch Logs がログのエクスポートを完了するのを待つため、出力なしで一定期間が発生することが予想されます。

名前付き引数

`-h, --help`

`pcluster export-cluster-logs` のヘルプテキストを表示します。

`--bucket` *`BUCKET_NAME`*

クラスターログデータのエクスポート先の Amazon S3 バケットを指定します。クラスターと同じリージョンである必要があります。

> **Note**
>
> - CloudWatch にアクセスを許可するために、Amazon S3 バケットポリシーに対する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、CloudWatch Logs ユーザーガイドのAmazon S3バケットに対するアクセス許可の設定」を参照してください。CloudWatch
> - AWS ParallelCluster バージョン 3.12.0 以降、 `--bucket`オプションはオプションです。オプションが指定されていない場合は、リージョンの AWS ParallelCluster デフォルトバケット (`parallelcluster-hash-v1-DO-NOT-DELETE`) が使用されます。または、 が指す Amazon S3 バケット`CustomS3Bucket`がクラスター設定で指定されている場合は、 が使用されます。`--bucket` オプションを指定せず、デフォルトの AWS ParallelCluster バケットを使用する場合は、内部使用のために予約されている保護された`parallelcluster/`フォルダであるため、 フォルダにログをエクスポートすることはできません。

> **Important**
>
> AWS ParallelCluster デフォルトのバケットが使用されている場合、pcluster はバケットポリシーの設定を処理します。バケットポリシーをカスタマイズしてからバージョン 3.12.0 にアップグレード AWS ParallelCluster すると、バケットポリシーは上書きされ、変更を再適用する必要があります。

`--cluster-name, -n` *`CLUSTER_NAME`*

クラスターの名前を指定します。

`--bucket-prefix` *`BUCKET_PREFIX`*

エクスポートされたログデータが Amazon S3 バケットに格納されるパスを指定します。

デフォルトでは、バケットプレフィックスは次のとおりです。

```
cluster-name-logs-202209061743.tar.gz
```

`202209061743` は、 `%Y%m%d%H%M` 形式の時刻の例です。

> **Note**
>
> AWS ParallelCluster バージョン 3.12.0 以降では、 `--bucket`オプションを指定せず、デフォルトの AWS ParallelCluster バケットを使用する場合は、内部使用用に予約された保護されたフォルダであるため、 `parallelcluster/` フォルダにログをエクスポートすることはできません。

`--debug`

デバッグログの有効化

`--end-time` *`END_TIME`*

ログイベントを収集する時間範囲の終わりを指定し、ISO 8601 形式 (`YYYY-MM-DDThh:mm:ssZ`、例えば `2021-01-01T20:00:00Z`) で表します。この時間と同じかそれ以降のタイムスタンプを持つイベントは含まれません。時間要素 (分、秒など) は省略可能です。デフォルト値は現在の時刻です。

`--filters` *`FILTER`* `[`*`FILTER`* `...]`

ログのフィルタを指定します。形式: `Name=a,Values=1 Name=b,Values=2,3`。サポートされているフィルタ:

`private-dns-name`

インスタンスのプライベート DNS 名の短縮形を指定します (例: `ip-10-0-0-101`)。

`node-type`

ノードタイプを指定します。このフィルターで受け入れられる値は `HeadNode` のみです。

`--keep-s3-objects` *`KEEP_S3_OBJECTS`*

`true` の場合は、Amazon S3 にエクスポートされたオブジェクトが保持されます。(デフォルトは `false` です)

`--output-file` *`OUTPUT_FILE`*

ログアーカイブを保存するファイルパスを指定します。これが指定された場合、ログはローカルに保存されます。それ以外の場合は、Amazon S3 にアップロードされ、出力に URL が返されます。デフォルトでは、Amazon S3 にアップロードされます。

--region, -r *REGION*

AWS リージョン 使用する を指定します。AWS_DEFAULT_REGION 環境変数、 ~/.aws/config ファイルの [default]セクションの region設定、または --regionパラメータを使用して、を指定 AWS リージョン する必要があります。

--start-time *START_TIME*

時間範囲の開始時刻を ISO 8601 形式 (YYYY-MM-DDThh:mm:ssZ、例えば 2021-01-01T20:00:00Z) で指定します。タイムスタンプがこの時間に等しいか、この時間よりも遅いログイベントが含まれます。指定されていない場合、デフォルトはクラスターが作成された時刻です。

AWS ParallelCluster バージョン 3.1.4 の使用例：

```
$ pcluster export-cluster-logs --bucket cluster-v3-bucket -n cluster-v3
{
  "url": "https://cluster-v3-bucket..."
}
```

ログを取得できませんか？

export-cluster-logs コマンドを使用してログを取得できない場合は、次のいずれかを実行します。

- クラスターの CloudWatch ロググループからログを手動で取得します。
- ロググループが空の場合は、クラスターノードに SSH 接続し、 にリストされているログを取得します[ノードの初期化に関する問題のトラブルシューティング](#)。
- クラスターの作成に失敗したためにクラスターノードにアクセスできない場合は、 オプションを使用してクラスターを再作成--rollback-on-failure falseし、ノードからログを取得します。

pcluster export-image-logs

Image Builder スタックのログを Amazon S3 バケットを経由して、ローカルの tar.gz アーカイブにエクスポートします。

```
pcluster export-image-logs [-h]
```

```
                 --image-id IMAGE_ID
                [--bucket BUCKET]
                [--bucket-prefix BUCKET_PREFIX]
                [--debug]
                [--end-time END_TIME]
                [--keep-s3-objects KEEP_S3_OBJECTS]
                [--output-file OUTPUT_FILE]
                [--region REGION]
                [--start-time START_TIME]
```

> **Note**
>
> `export-image-logs` コマンドは CloudWatch Logs がログのエクスポートを完了するのを待つため、出力なしで一定期間が発生することが予想されます。

名前付き引数

`-h, --help`

`pcluster export-image-logs` のヘルプテキストを表示します。

`--bucket` *`BUCKET_NAME`*

イメージビルドログのエクスポート先の Amazon S3 バケット名を指定します。イメージと同じリージョンである必要があります。

> **Note**
>
> - CloudWatch にアクセスを許可するために、Amazon S3 バケットポリシーに対する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、「CloudWatch Logs ユーザーガイド」のAmazon S3バケットに対するアクセス許可の設定」を参照してください。 CloudWatch
> - AWS ParallelCluster バージョン 3.12.0 以降、 `--bucket`オプションはオプションです。オプションが指定されていない場合は、リージョンの AWS ParallelCluster デフォルトバケット (`parallelcluster-hash-v1-DO-NOT-DELETE`) を使用するか、イメージ設定で `CustomS3Bucket`が指定されている場合は、 が使用されます。

> **Important**
>
> AWS ParallelCluster デフォルトのバケットが使用されている場合、pcluster はバケットポリシーの設定を処理します。 AWS ParallelCluster バージョン 3.12.0 にアップグレードする前にバケットポリシーをカスタマイズすると、バケットポリシーは上書きされ、変更を再適用する必要があります。

`--image-id, -i IMAGE_ID`

ログがエクスポートされるイメージ ID。

`--bucket-prefix BUCKET_PREFIX`

エクスポートされたログデータが Amazon S3 バケットに格納されるパスを指定します。

デフォルトでは、バケットプレフィックスは次のとおりです。

```
ami-id-logs-202209061743.tar.gz
```

202209061743 は現在の時刻を `%Y%m%d%H%M` フォーマットで表示します。

> **Note**
>
> AWS ParallelCluster バージョン 3.12.0 以降、 `--bucket`オプションを指定せず、デフォルトの AWS ParallelCluster バケットを使用する場合は、内部使用用に予約された保護されたフォルダであるため、 `parallelcluster/` フォルダにログをエクスポートすることはできません。

`--debug`

デバッグログの有効化

`--end-time END_TIME`

ログイベントを収集する時間範囲の終わりを指定し、ISO 8601 形式 (`YYYY-MM-DDThh:mm:ssZ`、例えば `2021-01-01T20:00:00Z`) で表します。この時間と同じかそれ以降のタイムスタンプを持つイベントは含まれません。時間要素 (分、秒など) は省略可能です。デフォルト値は現在の時刻です。

--keep-s3-objects ***KEEP_S3_OBJECTS***

true の場合は、Amazon S3 にエクスポートされたオブジェクトが保持されます。(デフォルトは false です)

--output-file ***OUTPUT_FILE***

ログアーカイブを保存するファイルパスを指定します。これが指定された場合、ログはローカルに保存されます。それ以外の場合は、Amazon S3 にアップロードされ、出力に URL が返されます。デフォルトでは、Amazon S3 にアップロードされます。

--region, -r ***REGION***

AWS リージョン 使用する を指定します。AWS_DEFAULT_REGION 環境変数、 ~/.aws/config ファイルの [default]セクションの region設定、または --regionパラメータを使用して、を指定 AWS リージョン する必要があります。

--start-time ***START_TIME***

時間範囲の開始時刻を ISO 8601 形式 (YYYY-MM-DDThh:mm:ssZ、例えば 2021-01-01T20:00:00Z) で指定します。タイムスタンプがこの時間に等しいか、この時間よりも遅いログイベントが含まれます。指定されていない場合、デフォルトはクラスターが作成された時刻です。

AWS ParallelCluster バージョン 3.1.4 の使用例：

```
$ pcluster export-image-logs --bucket image-v3-bucket --image-id ami-1234abcd5678efgh
{
  "url": "https://image-v3-bucket..."
}
```

pcluster get-cluster-log-events

ログストリームに関連するイベントを取得します。

```
pcluster get-cluster-log-events [-h]
                 --cluster-name CLUSTER_NAME
                 --log-stream-name LOG_STREAM_NAME
                [--debug]
                [--end-time END_TIME]
                [--limit LIMIT]
                [--next-token NEXT_TOKEN]
```

```
                [--query QUERY]
                [--region REGION]
                [--start-from-head START_FROM_HEAD]
                [--start-time START_TIME]
```

名前付き引数

-h, --help

pcluster get-cluster-log-events のヘルプテキストを表示します。

--cluster-name, -n *CLUSTER_NAME*

クラスターの名前を指定します。

--log-stream-name *LOG_STREAM_NAME*

ログストリームの名前を指定します。list-cluster-log-streams コマンドを使用して、1 つまたは複数のイベントに関連するログストリームを取得できます。

--debug

デバッグログの有効化

--end-time *END_TIME*

時間範囲の終了時刻を指定し、ISO 8601 形式 (YYYY-MM-DDThh:mm:ssZ、例えば 2021-01-01T20:00:00Z) で表します。この時間と同じかそれ以降のタイムスタンプを持つイベントは含まれません。

--limit *LIMIT*

返されるログイベントの最大数数を指定します。値が指定されていない場合、最大値は 1 MB の応答サイズに収まるログイベントの数で、最大 10,000 ログイベントまでとなります。

--next-token *NEXT_TOKEN*

次の結果セットのトークン。

--query *QUERY*

出力に対して実行する JMESPath クエリを指定します。

--region, -r *REGION*

AWS リージョン 使用する を指定します。AWS_DEFAULT_REGION 環境変数、 ~/.aws/config ファイルの [default]セクションの region設定、または --regionパラメータを使用して、を指定 AWS リージョン する必要があります。

--start-from-head ***START_FROM_HEAD***

値が true の場合、最も古いログイベントが最初に返されます。値が false の場合、最も新しいログイベントが最初に返されます。(デフォルトは false です)

--start-time ***START_TIME***

時間範囲の開始時刻を ISO 8601 形式 (YYYY-MM-DDThh:mm:ssZ、例えば 2021-01-01T20:00:00Z) で指定します。タイムスタンプがこの時間と同じか、この時間よりも遅いイベントが含まれます。

AWS ParallelCluster バージョン 3.1.4 の使用例：

```
$ pcluster get-cluster-log-events \
    -c cluster-v3 \
    -r us-east-1 \
    --log-stream-name ip-198-51-100-44.i-1234567890abcdef0.clustermgtd \
    --limit 3
{
  "nextToken": "f/36966906399261933213029082268132291405859205452101451780/s",
  "prevToken": "b/36966906399239632467830551644990755687586557090595471362/s",
  "events": [
    {
      "message": "2022-07-12 19:16:53,379 - [slurm_plugin.clustermgtd:_maintain_nodes]
 - INFO - Performing node maintenance actions",
      "timestamp": "2022-07-12T19:16:53.379Z"
    },
    {
      "message": "2022-07-12 19:16:53,380 - [slurm_plugin.clustermgtd:_maintain_nodes]
 - INFO - Following nodes are currently in replacement: (x0) []",
      "timestamp": "2022-07-12T19:16:53.380Z"
    },
    {
      "message": "2022-07-12 19:16:53,380 -
 [slurm_plugin.clustermgtd:_terminate_orphaned_instances] - INFO - Checking for
 orphaned instance",
      "timestamp": "2022-07-12T19:16:53.380Z"
    }
  ]
}
```

pcluster get-cluster-stack-events

指定されたクラスターのスタックに関連するイベントを取得します。

> **Note**
>
> バージョン 3.6.0 以降、 はネストされたスタック AWS ParallelCluster を使用して、キューとコンピューティングリソースに関連付けられたリソースを作成します。`GetClusterStackEvents` API と `pcluster get-cluster-stack-events` コマンドはクラスターのメインスタックイベントのみを返します。キューやコンピュートリソースに関連するものを含むクラスタースタックイベントは、CloudFormation コンソールで表示できます。

```
pcluster get-cluster-stack-events [-h]
                 --cluster-name CLUSTER_NAME
                [--debug]
                [--next-token NEXT_TOKEN]
                [--query QUERY]
                [--region REGION]
```

名前付き引数

-h, --help

`pcluster get-cluster-stack-events` のヘルプテキストを表示します。

--cluster-name, -n *CLUSTER_NAME*

クラスターの名前を指定します。

--debug

デバッグログの有効化

--next-token *NEXT_TOKEN*

次の結果セットのトークン。

--query *QUERY*

出力に対して実行する JMESPath クエリを指定します。

--region, -r *REGION*

AWS リージョン 使用する を指定します。は、 `AWS_DEFAULT_REGION`環境変数、 `~/.aws/config` ファイルの `[default]`セクションの `region`設定、または `--region`パラメータを使用して指定 AWS リージョン する必要があります。

AWS ParallelCluster バージョン 3.1.4 の使用例：

```
$ pcluster get-cluster-stack-events \
    -n cluster-v3 \
    -r us-east-1 \
    --query "events[0]"
{
  "eventId": "1234abcd-56ef-78gh-90ij-abcd1234efgh",
  "physicalResourceId": "arn:aws:cloudformation:us-east-1:123456789012:stack/cluster-
v3/1234abcd-56ef-78gh-90ij-abcd1234efgh",
  "resourceStatus": "CREATE_COMPLETE",
  "stackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/cluster-
v3/1234abcd-56ef-78gh-90ij-abcd1234efgh",
  "stackName": "cluster-v3",
  "logicalResourceId": "cluster-v3",
  "resourceType": "AWS::CloudFormation::Stack",
  "timestamp": "2022-07-12T18:29:12.140Z"
}
```

pcluster get-image-log-events

イメージビルドに関連付けられたログイベントを取得します。

```
pcluster get-image-log-events [-h]
                 --image-id IMAGE_ID
                 --log-stream-name LOG_STREAM_NAME
                [--debug]
                [--end-time END_TIME]
                [--limit LIMIT]
                [--next-token NEXT_TOKEN]
                [--query QUERY]
                [--region REGION]
                [--start-from-head START_FROM_HEAD]
                [--start-time START_TIME]
```

名前付き引数

-h, --help

pcluster get-image-log-events のヘルプテキストを表示します。

--image-id, -i *IMAGE_ID*

イメージの ID を指定します。

--log-stream-name *LOG_STREAM_NAME*

ログストリームの名前を指定します。list-image-log-streams コマンドを使用して、1 つまたは複数のイベントに関連するログストリームを取得できます。

--debug

デバッグログの有効化

--end-time *END_TIME*

時間範囲の終了時刻を指定し、ISO 8601 形式 (YYYY-MM-DDThh:mm:ssZ、例えば 2021-01-01T20:00:00Z) で表します。この時間と同じかそれ以降のタイムスタンプを持つイベントは含まれません。

--limit *LIMIT*

返されるログイベントの最大数数を指定します。値が指定されていない場合、最大値は 1 MB の応答サイズに収まるログイベントの数で、最大 10,000 ログイベントまでとなります。

--next-token *NEXT_TOKEN*

次の結果セットのトークン。

--query *QUERY*

出力に対して実行する JMESPath クエリを指定します。

--region, -r *REGION*

AWS リージョン 使用する を指定します。は、 AWS_DEFAULT_REGION環境変数、 ~/.aws/config ファイルの [default]セクションの region設定、または --regionパラメータを使用して指定 AWS リージョン する必要があります。

--start-from-head *START_FROM_HEAD*

値が true の場合、最も古いログイベントが最初に返されます。値が false の場合、最も新しいログイベントが最初に返されます。(デフォルトは false です)

--start-time ***START_TIME***

時間範囲の開始時刻を ISO 8601 形式 (`YYYY-MM-DDThh:mm:ssZ`、例えば `2021-01-01T20:00:00Z`) で指定します。この時間と同じかそれ以降のタイムスタンプを持つイベントが含まれます。

AWS ParallelCluster バージョン 3.1.2 の使用例：

```
$ pcluster get-image-log-events --image-id custom-alinux2-image --region us-east-1 --
log-stream-name 3.1.2/1 --limit 3
   {
  "nextToken": "f/36778317771100849897800729464621464113270312017760944178/s",
  "prevToken": "b/36778317766952911290874033560295820514557716777648586800/s",
  "events": [
    {
      "message": "ExecuteBash: FINISHED EXECUTION",
      "timestamp": "2022-04-05T22:13:26.633Z"
    },
    {
      "message": "Document arn:aws:imagebuilder:us-east-1:123456789012:component/
parallelclusterimage-test-1234abcd-56ef-78gh-90ij-abcd1234efgh/3.1.2/1",
      "timestamp": "2022-04-05T22:13:26.741Z"
    },
    {
      "message": "TOE has completed execution successfully",
      "timestamp": "2022-04-05T22:13:26.819Z"
    }
  ]
}
```

pcluster get-image-stack-events

指定されたイメージビルドのスタックに関連するイベントを取得します。

```
pcluster get-image-stack-events [-h]
                 --image-id IMAGE_ID
                [--debug]
                [--next-token NEXT_TOKEN]
                [--query QUERY]
                [--region REGION]
```

名前付き引数

-h, --help

pcluster get-image-stack-events のヘルプテキストを表示します。

--image-id, -i *IMAGE_ID*

イメージの ID を指定します。

--debug

デバッグログの有効化

--next-token *NEXT_TOKEN*

次の結果セットのトークン。

--query *QUERY*

出力に対して実行する JMESPath クエリを指定します。

--region, -r *REGION*

AWS リージョン 使用する を指定します。は、 AWS_DEFAULT_REGION環境変数、 ~/.aws/config ファイルの [default]セクションの region設定、または --regionパラメータを使用して指定 AWS リージョン する必要があります。

AWS ParallelCluster バージョン 3.1.2 の使用例：

```
$ pcluster get-image-stack-events --image-id custom-alinux2-image --region us-east-1 --
query "events[0]"
   {
  "eventId": "ParallelClusterImage-CREATE_IN_PROGRESS-2022-04-05T21:39:24.725Z",
  "physicalResourceId": "arn:aws:imagebuilder:us-east-1:123456789012:image/
parallelclusterimage-custom-alinux2-image/3.1.2/1",
  "resourceStatus": "CREATE_IN_PROGRESS",
  "resourceStatusReason": "Resource creation Initiated",
  "resourceProperties": "{\"InfrastructureConfigurationArn\":
\"arn:aws:imagebuilder:us-east-1:123456789012:infrastructure-configuration/
parallelclusterimage-1234abcd-56ef-78gh-90ij-abcd1234efgh\",\"ImageRecipeArn
\":\"arn:aws:imagebuilder:us-east-1:123456789012:image-recipe/
parallelclusterimage-custom-alinux2-image/3.1.2\",\"DistributionConfigurationArn
\":\"arn:aws:imagebuilder:us-east-1:123456789012:distribution-
configuration/parallelclusterimage-1234abcd-56ef-78gh-90ij-abcd1234efgh\",
```

```
\"EnhancedImageMetadataEnabled\":\"false\",\"Tags\":{\"parallelcluster:image_name\":
\"custom-alinux2-image\",\"parallelcluster:image_id\":\"custom-alinux2-image\"}}",
  "stackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/custom-alinux2-
image/1234abcd-56ef-78gh-90ij-abcd1234efgh",
  "stackName": "custom-alinux2-image",
  "logicalResourceId": "ParallelClusterImage",
  "resourceType": "AWS::ImageBuilder::Image",
  "timestamp": "2022-04-05T21:39:24.725Z"
}
```

pcluster list-clusters

既存のクラスターのリストを取得します。

```
pcluster list-clusters [-h]
                [--cluster-status {CREATE_IN_PROGRESS,CREATE_FAILED,CREATE_COMPLETE,
                           DELETE_IN_PROGRESS,DELETE_FAILED,UPDATE_IN_PROGRESS,
                           UPDATE_COMPLETE,UPDATE_FAILED}
                           [{CREATE_IN_PROGRESS,CREATE_FAILED,CREATE_COMPLETE,
                             DELETE_IN_PROGRESS,DELETE_FAILED,UPDATE_IN_PROGRESS,
                             UPDATE_COMPLETE,UPDATE_FAILED} ...]]
                [--debug]
                [--next-token NEXT_TOKEN]
                [--query QUERY]
                [--region REGION]
```

名前付き引数

-h, --help

pcluster list-clusters のヘルプテキストを表示します。

--cluster-status {CREATE_IN_PROGRESS, CREATE_FAILED, CREATE_COMPLETE, DELETE_IN_PROGRESS, DELETE_FAILED, UPDATE_IN_PROGRESS, UPDATE_COMPLETE, UPDATE_FAILED} [{CREATE_IN_PROGRESS, CREATE_FAILED, CREATE_COMPLETE, DELETE_IN_PROGRESS, DELETE_FAILED, UPDATE_IN_PROGRESS, UPDATE_COMPLETE, UPDATE_FAILED} ...]

フィルタリングの対象となるクラスターステータスのリストを指定します。(デフォルトは all です)

--debug

デバッグログの有効化

--next-token ***NEXT_TOKEN***

次の結果セットのトークン。

--query ***QUERY***

出力に対して実行する JMESPath クエリを指定します。

--region, -r ***REGION***

AWS リージョン 使用する を指定します。は、 `AWS_DEFAULT_REGION`環境変数、 `~/.aws/config` ファイルの `[default]`セクションの `region`設定、または `--region`パラメータを使用して指定 AWS リージョン する必要があります。

AWS ParallelCluster バージョン 3.1.4 の使用例：

```
$ pcluster list-clusters
{
  "clusters": [
    {
      "clusterName": "cluster-v3",
      "cloudformationStackStatus": "CREATE_COMPLETE",
      "cloudformationStackArn": "arn:aws:cloudformation:us-east-1:123456789012:stack/
cluster-v3/1234abcd-56ef-78gh-90ij-abcd1234efgh",
      "region": "us-east-1",
      "version": "3.1.4",
      "clusterStatus": "CREATE_COMPLETE"
    }
  ]
}
```

pcluster list-cluster-log-streams

クラスターに関連付けられているログストリームのリストを取得します。

```
pcluster list-cluster-log-streams [-h]
                 --cluster-name CLUSTER_NAME
                [--filters FILTERS [FILTERS ...]]
                [--next-token NEXT_TOKEN] [--debug]
```

```
                [--query QUERY]
                [--region REGION]
```

名前付き引数

-h, --help

pcluster list-cluster-log-streams のヘルプテキストを表示します。

--cluster-name, -n *CLUSTER_NAME*

クラスターの名前を指定します。

--debug

デバッグログの有効化

--filters *FILTERS* [*FILTERS* ...]

ログストリームのフィルタを指定します。形式: Name=a,Values=1 Name=b,Values=2,3。サポートされているフィルタ:

private-dns-name

インスタンスのプライベート DNS 名の短縮形を指定します (例: ip-10-0-0-101)。

node-type

ノードタイプを指定します。このフィルターで受け入れられる値は HeadNode のみです。

--next-token *NEXT_TOKEN*

次の結果セットのトークン。

--query *QUERY*

出力に対して実行する JMESPath クエリを指定します。

--region, -r *REGION*

AWS リージョン 使用する を指定します。は、 AWS_DEFAULT_REGION環境変数、 ~/.aws/config ファイルの [default]セクションの region設定、または --regionパラメータを使用して指定 AWS リージョン する必要があります。

AWS ParallelCluster バージョン 3.1.4 の使用例：

```
$ pcluster list-cluster-log-streams \
```

```
    -n cluster-v3 \
    -r us-east-1 \
    --query 'logStreams[*].logStreamName'
[
  "ip-172-31-58-205.i-1234567890abcdef0.cfn-init",
  "ip-172-31-58-205.i-1234567890abcdef0.chef-client",
  "ip-172-31-58-205.i-1234567890abcdef0.cloud-init",
  "ip-172-31-58-205.i-1234567890abcdef0.clustermgtd",
  "ip-172-31-58-205.i-1234567890abcdef0.slurmctld",
  "ip-172-31-58-205.i-1234567890abcdef0.supervisord",
  "ip-172-31-58-205.i-1234567890abcdef0.system-messages"
]
```

pcluster list-images

既存のカスタムイメージのリストを取得します。

```
pcluster list-images [-h]
                 --image-status {AVAILABLE,PENDING,FAILED}
                [--debug]
                [--next-token NEXT_TOKEN]
                [--query QUERY]
                [--region REGION]
```

名前付き引数

-h, --help

pcluster list-images のヘルプテキストを表示します。

--image-status {AVAILABLE,PENDING,FAILED}

指定されたステータスでイメージを返します。

--debug

デバッグログの有効化

--next-token *NEXT_TOKEN*

次の結果セットのトークン。

--query *QUERY*

出力に対して実行する JMESPath クエリを指定します。

--region, -r *REGION*

AWS リージョン 使用する を指定します。は、 `AWS_DEFAULT_REGION`環境変数、 `~/.aws/config` ファイルの `[default]`セクションの `region`設定、または `--region`パラメータを使用して指定 AWS リージョン する必要があります。

AWS ParallelCluster バージョン 3.1.2 の使用例：

```
$ pcluster list-images --image-status AVAILABLE
{
  "images": [
    {
      "imageId": "custom-alinux2-image",
      "imageBuildStatus": "BUILD_COMPLETE",
      "ec2AmiInfo": {
        "amiId": "ami-1234abcd5678efgh"
      },
      "region": "us-east-1",
      "version": "3.1.2"
    }
  ]
}
```

pcluster list-image-log-streams

イメージに関連付けられたログストリームのリストを取得します。

```
pcluster list-image-log-streams [-h]
                 --image-id IMAGE_ID
                [--next-token NEXT_TOKEN] [--debug]
                [--query QUERY]
                [--region REGION]
```

名前付き引数

-h, --help

`pcluster list-image-log-streams` のヘルプテキストを表示します。

--image-id, -i *IMAGE_ID*

イメージの ID を指定します。

--debug

デバッグログの有効化

--next-token *NEXT_TOKEN*

次の結果セットのトークン。

--query *QUERY*

出力に対して実行する JMESPath クエリを指定します。

--region, -r *REGION*

AWS リージョン 使用する を指定します。は、 `AWS_DEFAULT_REGION`環境変数、 `~/.aws/config` ファイルの `[default]`セクションの `region`設定、または `--region`パラメータを使用して指定 AWS リージョン する必要があります。

AWS ParallelCluster バージョン 3.1.2 の使用例：

```
$ pcluster list-image-log-streams --image-id custom-alinux2-image --region us-east-1 --
query 'logStreams[*].logStreamName'
[
  "3.0.0/1",
  "3.1.2/1"
]
```

pcluster list-official-images

AWS ParallelCluster AMIs。

```
pcluster list-official-images [-h]
                [--architecture ARCHITECTURE]
                [--debug]
                [--os OS]
                [--query QUERY]
                [--region REGION]
```

名前付き引数

-h, --help

`pcluster list-official-images` のヘルプテキストを表示します。

--architecture ***ARCHITECTURE***

検索結果のフィルタリングに使用するアーキテクチャを指定します。このパラメータが指定されていない場合は、すべてのアーキテクチャが返されます。

--debug

デバッグログの有効化

--os ***OS***

結果のフィルタリングに使用するオペレーティングシステムを指定します。このパラメータが指定されていない場合は、すべてのオペレーティングシステムが返されます。

--query ***QUERY***

出力に対して実行する JMESPath クエリを指定します。

--region, -r ***REGION***

AWS リージョン 使用する を指定します。イメージ設定ファイルのリージョン設定、`AWS_DEFAULT_REGION`環境変数、`~/.aws/config`ファイルの `[default]`セクションの `region`設定、または `--region`パラメータを使用して、 を指定 AWS リージョン する必要があります。

AWS ParallelCluster バージョン 3.1.2 の使用例：

```
$ pcluster list-official-images
{
  "images": [
    {
      "amiId": "ami-015cfeb4e0d6306b2",
      "os": "ubuntu2004",
      "name": "aws-parallelcluster-3.1.2-ubuntu-2004-lts-hvm-x86_64-202202261505
 2022-02-26T15-08-34.759Z",
      "version": "3.1.2",
      "architecture": "x86_64"
    },
    {
      "amiId": "ami-036f23237ce49d25b",
      "os": "ubuntu2204",
      "name": "aws-parallelcluster-3.1.2-ubuntu-1804-lts-hvm-x86_64-202202261505
 2022-02-26T15-08-17.558Z",
      "version": "3.1.2",
```

```
      "architecture": "x86_64"
    },
    {
      "amiId": "ami-09e5327e694d89ef4",
      "os": "ubuntu2004",
      "name": "aws-parallelcluster-3.1.2-ubuntu-2004-lts-hvm-arm64-202202261505
 2022-02-26T15-08-45.736Z",
      "version": "3.1.2",
      "architecture": "arm64"
    },
    {
      "amiId": "ami-0b9b0874c35f626ae",
      "os": "alinux2",
      "name": "aws-parallelcluster-3.1.2-amzn2-hvm-x86_64-202202261505
 2022-02-26T15-08-31.311Z",
      "version": "3.1.2",
      "architecture": "x86_64"
    },
    {
      "amiId": "ami-0d0de4f95f56374bc",
      "os": "alinux2",
      "name": "aws-parallelcluster-3.1.2-amzn2-hvm-arm64-202202261505
 2022-02-26T15-08-46.088Z",
      "version": "3.1.2",
      "architecture": "arm64"
    },
    {
      "amiId": "ami-0ebf7bc54b8740dc6",
      "os": "ubuntu2204",
      "name": "aws-parallelcluster-3.1.2-ubuntu-1804-lts-hvm-arm64-202202261505
 2022-02-26T15-08-45.293Z",
      "version": "3.1.2",
      "architecture": "arm64"
    }
  ]
}
```

pcluster ssh

事前に入力されているクラスターのユーザー名と IP アドレスを使用して ssh コマンドを実行します。。任意の引数は ssh コマンドラインの末尾に付加されます。

```
pcluster ssh [-h]
```

```
                 --cluster-name CLUSTER_NAME
                [--debug]
                [--dryrun DRYRUN]
                [--region REGION]
```

名前付き引数

-h, --help

pcluster ssh のヘルプテキストを表示します。

--cluster-name, -n *CLUSTER_NAME*

接続するクラスターの名前を指定します。

--debug

デバッグログの有効化

--dryrun *DRYRUN*

true の場合、実行するコマンドラインを出力して終了します。(デフォルトは false です)

--region, -r *REGION*

AWS リージョン 使用する を指定します。は、 AWS_DEFAULT_REGION環境変数、 ~/.aws/config ファイルの [default]セクションの region設定、または --regionパラメータを使用して指定 AWS リージョン する必要があります。

例:

```
$ pcluster ssh --cluster-name mycluster -i ~/.ssh/id_rsa
```

事前に入力されているクラスターのユーザー名と IP アドレスを使用して ssh コマンドを実行します。

```
ssh ec2-user@1.1.1.1 -i ~/.ssh/id_rsa
```

pcluster update-cluster

既存のクラスターを、指定された設定ファイルの設定に合わせて更新します。

```
pcluster update-cluster [-h]
```

```
                 --cluster-configuration CLUSTER_CONFIGURATION
                 --cluster-name CLUSTER_NAME
                [--debug]
                [--dryrun DRYRUN]
                [--force-update FORCE_UPDATE]
                [--query QUERY]
                [--region REGION]
                [--suppress-validators SUPPRESS_VALIDATORS [SUPPRESS_VALIDATORS ...]]
                [--validation-failure-level {INFO,WARNING,ERROR}]
```

名前付き引数

-h, --help

pcluster update-cluster のヘルプテキストを表示します。

--cluster-configuration, -c *CLUSTER_CONFIGURATION*

YAML クラスター設定ファイルを指定します。

--cluster-name, -n *CLUSTER_NAME*

クラスターの名前を指定します。

--debug

デバッグログの有効化

--dryrun *DRYRUN*

true の場合、クラスターを更新やリソースを作成することなく、検証を実行します。イメージの構成や更新の要件を検証するために使用することができます。(デフォルトは false です)

--force-update *FORCE_UPDATE*

true の場合、更新検証のエラーを無視して、強制的に更新を行います。(デフォルトは false です)

--query *QUERY*

出力に対して実行する JMESPath クエリを指定します。

--region, -r *REGION*

AWS リージョン 使用する を指定します。クラスター設定ファイルの Region設定、AWS_DEFAULT_REGION環境変数、 ~/.aws/config ファイルの [default]セクションの

region設定、または --regionパラメータを使用して、 を指定 AWS リージョン する必要があります。

--suppress-validators *SUPPRESS_VALIDATORS* [*SUPPRESS_VALIDATORS ...*]

抑制する 1 つまたは複数のコンフィグバリデータを指定します。

形式: (ALL|type:[A-Za-z0-9]+)

--validation-failure-level *{INFO,WARNING,ERROR}*

更新時に報告される検証エラーのレベルを指定します。

AWS ParallelCluster バージョン 3.1.4 の使用例：

```
$ pcluster update-cluster -c cluster-config.yaml -n cluster-v3 -r us-east-1
{
  "cluster": {
    "clusterName": "cluster-v3",
    "cloudformationStackStatus": "UPDATE_IN_PROGRESS",
    "cloudformationStackArn": "arn:aws:cloudformation:us-east-1:123456789012:stack/
cluster-v3/1234abcd-56ef-78gh-90ij-abcd1234efgh",
    "region": "us-east-1",
    "version": "3.1.4",
    "clusterStatus": "UPDATE_IN_PROGRESS"
  },
  "changeSet": [
    {
      "parameter": "HeadNode.Iam.S3Access",
      "requestedValue": {
        "BucketName": "amzn-s3-demo-bucket1",
        "KeyName": "output",
        "EnableWriteAccess": false
      }
    },
    {
      "parameter": "HeadNode.Iam.S3Access",
      "currentValue": {
        "BucketName": "amzn-s3-demo-bucket2",
        "KeyName": "logs",
        "EnableWriteAccess": true
      }
    }
  ]
```

```
}
```

pcluster update-compute-fleet

クラスターコンピューティングフリートの状態を更新します。

```
pcluster update-compute-fleet [-h]
                 --cluster-name CLUSTER_NAME
                 --status {START_REQUESTED,STOP_REQUESTED,ENABLED,DISABLED}

                [--debug]
                [--query QUERY]
                [--region REGION]
```

名前付き引数

-h, --help

pcluster update-compute-fleet のヘルプテキストを表示します。

--cluster-name, -n *CLUSTER_NAME*

クラスターの名前を指定します。

--status {START_REQUESTED,STOP_REQUESTED,ENABLED,DISABLED}

クラスターコンピューティングフリートに適用されるステータスを指定します。ステータスSTART_REQUESTEDと STOP_REQUESTEDはスSlurmケジューラに対応し、ステータスENABLEDと はス AWS Batch ケジューラDISABLEDに対応します。

--debug

デバッグログの有効化

--query *QUERY*

出力に対して実行する JMESPath クエリを指定します。

--region, -r *REGION*

AWS リージョン 使用する を指定します。は、 AWS_DEFAULT_REGION環境変数、 ~/.aws/config ファイルの [default]セクションの region設定、または --regionパラメータを使用して指定 AWS リージョン する必要があります。

AWS ParallelCluster バージョン 3.1.4 の使用例：

```
$ pcluster update-compute-fleet -n cluster-v3 --status STOP_REQUESTED
{
  "status": "STOP_REQUESTED",
  "lastStatusUpdatedTime": "2022-07-12T20:19:47.653Z"
}
```

pcluster version

のバージョンを表示します AWS ParallelCluster。

```
pcluster version [-h] [--debug]
```

名前付き引数

-h, --help

`pcluster version` のヘルプテキストを表示します。

--debug

デバッグログの有効化

AWS ParallelCluster バージョン 3.1.4 の使用例：

```
$ pcluster version
{
  "version": "3.1.4"
}
```

pcluster3-config-converter

AWS ParallelCluster バージョン 2 の設定ファイルを読み取り、 AWS ParallelCluster バージョン 3 の設定ファイルを書き込みます。

```
pcluster3-config-converter [-h]
                [-t CLUSTER_TEMPLATE]
                [-c CONFIG_FILE]
                [--force-convert]
```

```
                [-o OUTPUT_FILE]
```

名前付き引数

`-h, --help`

`pcluster3-config-converter` のヘルプテキストを表示します。

`-t CLUSTER_TEMPLATE, --cluster-template CLUSTER_TEMPLATE`

変換するコンフィグレーションファイルの `[cluster]` セクションを指定します。指定されていない場合、スクリプトは `[global]` セクションの cluster-template パラメータを検索するか、`[cluster default]` を検索します。

`-c CONFIG_FILE, --config-file CONFIG_FILE`

読み取る AWS ParallelCluster バージョン 2 の設定ファイルを指定します。

`--force-convert`

1 つ以上の設定がサポートされておらず、推奨されていない場合でも、変換を有効にします。

`-o OUTPUT_FILE, --output-file OUTPUT_FILE`

書き込む AWS ParallelCluster バージョン 3 の設定ファイルを指定します。このパラメータが指定されていない場合、設定内容は標準出力に書き込まれます。

> **Note**
>
> `pcluster3-config-converter` コマンドが AWS ParallelCluster バージョン 3.0.1 に追加されました。

設定ファイル

AWS ParallelCluster は、設定パラメータに YAML 1.1 ファイルを使用します。

トピック

- クラスター設定ファイル
- ビルドイメージの設定ファイル

クラスター設定ファイル

AWS ParallelCluster バージョン 3 では、個別の設定ファイルを使用して、クラスターインフラストラクチャの定義とカスタム AMIs の定義を制御します。すべての設定ファイルは YAML 1.1 ファイルを使用しています。それぞれの設定ファイルの詳細については、以下のリンクを参照してください。設定例については、https://github.com/aws/aws-parallelcluster/tree/release-3.0/cli/tests/pcluster/example_configs を参照してください。

これらのオブジェクトは、 AWS ParallelCluster バージョン 3 クラスター設定に使用されます。

トピック

- クラスター設定ファイルのプロパティ
- Imds セクション
- Image セクション
- HeadNode セクション
- Scheduling セクション
- SharedStorage セクション
- Iam セクション
- LoginNodes セクション
- Monitoring セクション
- Tags セクション
- AdditionalPackages セクション
- DirectoryService セクション
- DeploymentSettings セクション

クラスター設定ファイルのプロパティ

`Region` (オプション、`String`)

クラスター AWS リージョン の を指定します。例えば、`us-east-2`。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`CustomS3Bucket` (オプション、`String`)

クラスター設定ファイルなど、クラスターで使用されるリソースを保存し、logs をエクスポートするために AWS アカウントで作成される Amazon S3 バケットの名前を指定します。 は、クラ

スターを作成する各 AWS リージョンで 1 つの Amazon S3 バケット AWS ParallelCluster を維持します。デフォルトでは、これらの Amazon S3 バケットには `parallelcluster-hash-v1-DO-NOT-DELETE` という名前が付けられます。

更新ポリシー: この設定が変更された場合、更新は許可されません。強制的に更新した場合、新しい値は無視され、既存の値が使用されます。

`AdditionalResources` (オプション、`String`)

クラスターとともに起動する追加の AWS CloudFormation テンプレートを定義します。この追加のテンプレートは、クラスターの外部にありながらクラスターのライフサイクルの一部であるリソースを作成するために使用されます。

この値は、すべてのパラメータが指定されているパブリックテンプレートへの HTTPS URL である必要があります。

デフォルト値はありません。

更新ポリシー: この設定は、更新中に変更できます。

`Imds` セクション

(オプション) グローバルインスタンスメタデータサービス (IMDS) の設定を指定します。

```
Imds:
  ImdsSupport: string
```

`Imds` のプロパティ

`ImdsSupport` (オプション、`String`)

クラスターノードでサポートされている IMDS バージョンを指定します。サポートされている値は`v1.0` および `v2.0` です。デフォルト値は `v2.0` です。

`ImdsSupport` が `v1.0` に設定されている場合、IMDSv1 と IMDSv2 の両方がサポートされます。

`ImdsSupport` が `v2.0` に設定されている場合、IMDSv2 のみがサポートされます。

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「IMDSv2 の使用」を参照してください。

更新ポリシー: この設定が変更された場合、更新は許可されません。

> **Note**
>
> AWS ParallelCluster 3.7.0 以降、`ImdsSupport`デフォルト値は `v2.0`です。`ImdsSupport` を `v2.0` に設定し、カスタムアクション呼び出しにおいて、IMDSv1 を IMDSv2 に置き換えることをお勧めします。
> AWS ParallelCluster バージョン `Imds` 3.3.0 で / のサポートが追加され`ImdsSupport`ました。

`Image` セクション

> **Note**
>
> によって配布される公式 AMIs のサポートされていないバージョン AWS ParallelCluster は、18 か月間非アクティブになると使用できなくなります。これらの古いイメージには古いソフトウェアが含まれており、問題が発生した場合はサポートを受けることができません。サポートされている最新バージョンへの移行を強くお勧めします。

(必須) クラスターのオペレーティングシステムを定義します。

```
Image:
  Os: string
  CustomAmi: string
```

`Image` のプロパティ

`Os` (必須)、`String`)

クラスターに使用するオペレーティングシステムを指定します。サポートされている値は、`alinux2`、、`alinux2023`、`ubuntu2404`、`ubuntu2204`、`ubuntu2004`、、`rhel8`、`rocky8rhel`ですrocky9。

> **Note**
>
> RedHat Enterprise Linux 8.7 (`rhel8`) が AWS ParallelCluster バージョン 3.6.0 から追加されました。

`rhel` を使用するようにクラスターを設定した場合、任意のインスタンスタイプのオンデマンドコストは、サポートされている他のオペレーティングシステムを使用するようにクラスターを設定した場合よりも高くなります。料金の詳細については、「オンデマンド料金」と「Amazon EC2 での Red Hat Enterprise Linux が提供される方法と料金体系について教えてください」を参照してください。
RedHat Enterprise Linux 9 (rhel9) が AWS ParallelCluster バージョン 3.9.0 から追加されました。

すべての AWS 商用リージョンは、以下のすべてのオペレーティングシステムをサポートしています。

パーティション (AWS リージョン)	**alinux2**	**ubuntu224** および **ubuntu204**	**ubuntu244**	**rhel8**	**rhel9**	**alinux2023**
商用 (特に記載 AWS リージョン されていないすべて)	真	True	True	True	True	真
AWS GovCloud (米国東部) (`us-gov-east-1`)	真	True	True	True	True	真
AWS GovCloud (米国西部) (`us-gov-west-1`)	真	True	True	True	True	真
中国 (北京) (`cn-north-1`)	真	True	True	True	True	真
中国 (寧夏) (`cn-northwest-1`)	真	True	True	True	True	真

更新ポリシー: この設定が変更された場合、更新は許可されません。

> **Note**
>
> AWS ParallelCluster 3.8.0 は Rocky Linux 8 をサポートしていますが、構築済みの Rocky Linux 8 AMIs (x86 および ARM アーキテクチャ用) は使用できません。 AWS ParallelCluster 3.8.0 は、カスタム AMIs を使用した Rocky Linux 8 でのクラスターの作成をサポートしています。詳細については、「」を参照してください[オペレーティングシステムに関する考慮事項]。 AWS ParallelCluster 3.9.0 は Rocky Linux 9 をサポートしていますが、構築済みの Rocky Linux 9 AMIs (x86 および ARM アーキテクチャ用) は使用できません。 AWS ParallelCluster 3.9.0 は、カスタム AMIs を使用した Rocky Linux 9 でのクラスターの作成をサポートしています。詳細については、「[オペレーティングシステムに関する考慮事項]」を参照してください。

`CustomAmi` (オプション、`String`)

デフォルト AMI の代わりにヘッドノードとコンピューティングノードに使用するカスタム AMI の ID を指定します。詳細については、「[AWS ParallelCluster AMI のカスタマイズ]」を参照してください。

カスタム AMI の起動に追加のアクセス許可が必要な場合は、ユーザーおよびヘッドノードポリシーの両方にそれらのアクセス許可を追加する必要があります。

例えば、カスタム AMI に暗号化されたスナップショットが関連付けられている場合、ユーザーおよびヘッドノードポリシーの両方に次の追加のポリシーが必要です。

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
           "Effect": "Allow",
           "Action": [
               "kms:DescribeKey",
               "kms:ReEncrypt*",
               "kms:CreateGrant",
               "kms:Decrypt"
           ],
           "Resource": [
               "arn:aws:kms:<AWS_REGION>:<AWS_ACCOUNT_ID>:key/<AWS_KMS_KEY_ID>"
           ]
       }
   ]
```

```
}
```

RedHat Enterprise Linux カスタム AMI を構築するには、RHUI (AWS) リポジトリが提供するパッケージの `rhel-<version>-baseos-rhui-rpms`、`rhel-<version>-appstream-rhui-rpms`、および `codeready-builder-for-rhel-<version>-rhui-rpms` をインストールするように OS を設定する必要があります。さらに、カスタム AMI のリポジトリには、実行中のカーネルバージョンと同じバージョンの `kernel-devel` パッケージが含まれている必要があります。

既知の制限事項:

- RHEL 8.2 以降のバージョンのみが FSx for Lustre をサポートしています。
- RHEL 8.7 カーネルバージョン 4.18.0-425.3.1.el8 は FSx for Lustre をサポートしていません。
- RHEL 8.4 以降のバージョンのみが EFA をサポートしています。
- AL23 は NICE DCV をサポートしていません。NICE DCV を実行するために必要なグラフィカルデスクトップ環境が含まれていないためです。詳細については、公式の NICE DCVドキュメントを参照してください。

カスタム AMI 検証の警告のトラブルシューティングについては、「カスタム AMI の問題のトラブルシューティング」を参照してください。

更新ポリシー: この設定が変更された場合、更新は許可されません。

HeadNode セクション

(必須) ヘッドノードの設定を指定します。

```
HeadNode:
  InstanceType: string
  Networking:
    SubnetId: string
    ElasticIp: string/boolean
    SecurityGroups:
      - string
    AdditionalSecurityGroups:
      - string
    Proxy:
      HttpProxyAddress: string
  DisableSimultaneousMultithreading: boolean
  Ssh:
    KeyName: string
```

```
    AllowedIps: string
  LocalStorage:
    RootVolume:
      Size: integer
      Encrypted: boolean
      VolumeType: string
      Iops: integer
      Throughput: integer
      DeleteOnTermination: boolean
    EphemeralVolume:
      MountDir: string
  SharedStorageType: string
  Dcv:
    Enabled: boolean
    Port: integer
    AllowedIps: string
  CustomActions:
    OnNodeStart:
      Sequence:
        - Script: string
          Args:
            - string
      Script: string
      Args:
        - string
    OnNodeConfigured:
      Sequence:
        - Script: string
          Args:
            - string
      Script: string
      Args:
        - string
    OnNodeUpdated:
      Sequence:
        - Script: string
          Args:
            - string
      Script: string
      Args:
        - string
  Iam:
    InstanceRole: string
    InstanceProfile: string
```

```
    S3Access:
      - BucketName: string
        EnableWriteAccess: boolean
        KeyName: string
    AdditionalIamPolicies:
      - Policy: string
  Imds:
    Secured: boolean
  Image:
    CustomAmi: string
```

HeadNode のプロパティ

`InstanceType` (必須、`String`)

ヘッドノードのインスタンスタイプを指定します。

ヘッドノードに使用される Amazon EC2 インスタンスタイプを定義します。インスタンスタイプのアーキテクチャは、 `InstanceType`または AWS Batch Slurm`InstanceType`設定に使用されるアーキテクチャと同じである必要があります。

> **Note**
>
> AWS ParallelCluster は、 `HeadNode`設定で次のインスタンスタイプをサポートしていません。
>
> - hpc6id

複数のネットワークインターフェイスまたはネットワークインターフェイスカードを持つ p4d インスタンスタイプまたは別のインスタンスタイプを定義する場合は、パブリックアクセスを提供する`true`ように `ElasticIp`を に設定する必要があります。 AWS パブリック IPsは、単一のネットワークインターフェイスで起動されたインスタンスにのみ割り当てることができます。この場合、NAT ゲートウェイを使用してクラスターコンピューティングノードへのパブリックアクセスを提供することをお勧めします。詳細については、「Amazon EC2 Linux インスタンス用ユーザーガイド」の「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。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`DisableSimultaneousMultithreading` (オプション、`Boolean`)

の場合`true`、ヘッドノードのハイパースレッディングを無効にします。デフォルト値は `false` です。

すべてのインスタンスタイプがハイパースレッディングを無効にできるわけではありません。ハイパースレッディングの無効化をサポートするインスタンスタイプのリストについては、Amazon EC2 ユーザーガイド」の「インスタンスタイプごとの各 CPU コアの CPU コアとスレッド」を参照してください。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`SharedStorageType` (オプション、`String`)

内部共有データで使用するストレージのタイプを指定します。内部共有データには、 がクラスターを管理するために AWS ParallelCluster 使用するデータと、共有ファイルシステムボリュームをマウントするためのマウントディレクトリ`SharedStorage` セクションとして で指定`/home`されていない場合にデフォルトで共有されるデータが含まれます。内部共有データの詳細については、「AWS ParallelCluster 内部ディレクトリ」を参照してください。

デフォルトのストレージタイプの `Ebs` である場合、ヘッドノードは NFS を使用してルートボリュームの一部をコンピューティングノードとログインノードの共有ディレクトリとしてエクスポートします。

の場合`Efs`、ParallelCluster は共有内部データと に使用する EFS ファイルシステムを作成します`/home`。

更新ポリシー: この設定が変更された場合、更新は許可されません。

> **Note**
>
> クラスターがスケールアウトすると、ヘッドノードが NFS エクスポートを使用してルートボリュームのデータをコンピューティングノードと共有するため、EBS ストレージタイプがパフォーマンスのボトルネックをもたらす可能性があります。EFS を使用すると、クラスターのスケールアウトに伴う NFS エクスポートを回避し、関連するパフォーマンスのボトルネックを防止できます。小さなファイルやインストールプロセスでの読み取り/書き込みの可能性を最大限に高めるには、EBS を選択することをお勧めします。スケールには EFS を選択します。

Networking

(必須) ヘッドノードのネットワーク設定を定義します。

```
Networking:
  SubnetId: string
  ElasticIp: string/boolean
  SecurityGroups:
    - string
  AdditionalSecurityGroups:
    - string
  Proxy:
    HttpProxyAddress: string
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

Networking のプロパティ

`SubnetId` (必須、`String`)

ヘッドノードをプロビジョニングする既存のサブネットの ID を指定します。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`ElasticIp` (オプション、`String`)

ヘッドノードに Elastic IP アドレスを作成または割り当てます。サポートされる値は、`true`、`false`、または既存の Elastic IP アドレスの ID です。デフォルトは `false` です。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`SecurityGroups` (オプション、`[String]`)

ヘッドノードに使用する Amazon VPC セキュリティグループ ID のリスト。これらは、このプロパティが含まれていない場合に が AWS ParallelCluster 作成するセキュリティグループを置き換えます。

セキュリティグループが SharedStorage システム用に正しく設定されていることを確認します。

更新ポリシー: この設定は、更新中に変更できます。

`AdditionalSecurityGroups` (オプション、`[String]`)

ヘッドノードに使用する追加の Amazon VPC セキュリティグループ ID のリスト。

更新ポリシー: この設定は、更新中に変更できます。

Proxy (オプション)

ヘッドノードのプロキシ設定を指定します。

```
Proxy:
                                HttpProxyAddress:
                                string
```

HttpProxyAddress (オプション、String)

HTTP または HTTPS プロキシサーバーを定義します。通常は、https://*x.x.x.x:8080* です。

デフォルト値はありません。

更新ポリシー: この設定が変更された場合、更新は許可されません。

Ssh

(オプション) ヘッドノードへの SSH アクセスの設定を定義します。

```
Ssh:
      KeyName: string
      AllowedIps: string
```

更新ポリシー: この設定は、更新中に変更できます。

Ssh のプロパティ

KeyName (オプション、String)

ヘッドノードへの SSH アクセスを有効にするための既存の Amazon EC2 キーペアを指定します。

更新ポリシー: この設定が変更された場合、更新は許可されません。

AllowedIps (オプション、String)

ヘッドノードへの SSH 接続の CIDR 形式の IP 範囲またはプレフィックスリスト ID を指定します。デフォルトは 0.0.0.0/0 です。

更新ポリシー: この設定は、更新中に変更できます。

LocalStorage

(オプション) ヘッドノードのローカルストレージ構成を定義します。

```
LocalStorage:
  RootVolume:
    Size: integer
    Encrypted: boolean
    VolumeType: string
    Iops: integer
    Throughput: integer
    DeleteOnTermination: boolean
  EphemeralVolume:
    MountDir: string
```

更新ポリシー: この設定は、更新中に変更できます。

LocalStorage のプロパティ

RootVolume (必須)

ヘッドノードのルートボリュームストレージを指定します。

```
RootVolume:
  Size: integer
  Encrypted: boolean
  VolumeType: string
  Iops: integer
  Throughput: integer
  DeleteOnTermination: boolean
```

更新ポリシー: この設定は、更新中に変更できます。

Size (オプション、Integer)

ヘッドノードルートボリュームサイズをジビバイト (GiB) 単位で指定します。デフォルトのサイズは AMI から取得されます。異なるサイズを使用するには、AMI が growroot をサポートしている必要があります。

更新ポリシー: この設定が変更された場合、更新は許可されません。

Encrypted (オプション、Boolean)

ルートボリュームを暗号化するかどうかを指定します。デフォルト値は true です。

更新ポリシー: この設定が変更された場合、更新は許可されません。

VolumeType (オプション、String)

Amazon EBS ボリュームタイプを指定します。サポートされている値は gp2、gp3、io1、io2、sc1、st1 および standard です。デフォルト値は gp3 です。

詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EBS ボリュームの種類」を参照してください。

更新ポリシー: この設定が変更された場合、更新は許可されません。

Iops (オプション、Integer)

io1、io2 および gp3 タイプボリュームの IOPS の数を定義します。

デフォルト値、対応値、volume_iops 対 volume_size の比率は、VolumeType と Size で異なります。

更新ポリシー: この設定が変更された場合、更新は許可されません。

VolumeType = io1

デフォルト Iops = 100

サポートする値 Iops = 100 — 64000 †

最大 Iops/Size 比率 = 50 IOPS/GiB。5000 IOPS には、少なくとも 100 GiB の Size が必要です。

VolumeType = io2

デフォルト Iops = 100

サポートする値 Iops = 100 — 64000 (io2 Block Express ボリュームの場合は 256000) †

最大 Iops/Size 比率 = 500 IOPS/GiB。5000 IOPS には、少なくとも 10 GiB の Size が必要です。

VolumeType = gp3

デフォルト Iops = 3000

サポートされる値 Iops = 3000 — 16000

最大 Iops/Size 比率 = 500 IOPS/GiB。5000 IOPS には、少なくとも 10 GiB の Size が必要です。

† 最大 IOPS は、32,000 IOPS 以上でプロビジョニングされた Nitro System 上に構築されたインスタンスでのみ保証されます。他のインスタンスは、最大 32,000 IOPS を保証します。古いio1ボリュームは、ボリュームを変更しない限り、完全なパフォーマンスに達しない場合があります。 io2Block Express ボリュームは、R5b インスタンスタイプで最大 256000 までの Iops 値をサポートします。詳細については、「Amazon EC2 ユーザーガイド」の「io2 Block Express ボリューム」を参照してください。

更新ポリシー: この設定は、更新中に変更できます。

Throughput (オプション、Integer)

gp3 ボリュームタイプのスループットを MiB/秒で定義します。この設定は、VolumeType が gp3 の場合のみ有効です。デフォルト値は 125 です。サポートされる値: 125 — 1000 MIB/秒

Throughput と Iops の比率は 0.25 以下にします。最大のスループットである 1000 MiB/秒を実現するためには、Iops の設定を 4000 以上にする必要があります。

更新ポリシー: この設定が変更された場合、更新は許可されません。

DeleteOnTermination (オプション、Boolean)

ヘッドノードの終了時にルートボリュームを削除するかどうかを指定します。デフォルト値は true です。

更新ポリシー: この設定が変更された場合、更新は許可されません。

EphemeralVolume (オプション)

任意のインスタンスストアボリュームの詳細を指定します。詳細については、Amazon EC2 ユーザーガイド」の「インスタンスストアボリューム」を参照してください。

```
EphemeralVolume:
  MountDir: string
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

MountDir (オプション、String)

インスタンスストアボリュームのマウントディレクトリを指定します。デフォルトは /scratch です。

更新ポリシー: この設定が変更された場合、更新は許可されません。

Dcv

（オプション）ヘッドノードで実行される Amazon DCV サーバーの設定を定義します。

詳細については、「Amazon DCV 経由でヘッドノードとログインノードに接続する」を参照してください。

```
Dcv:
  Enabled: boolean
  Port: integer
  AllowedIps: string
```

> **Important**
>
> デフォルトでは、 によって設定された Amazon DCV ポート AWS ParallelCluster はすべての IPv4 アドレスに対して開かれています。ただし、Amazon DCV ポートに接続できるのは、Amazon DCV セッションの URL があり、この URL が `pcluster dcv-connect` から返されてから 30 秒以内に Amazon DCV セッションに接続する場合のみです。`AllowedIps` 設定を使用して、CIDR 形式の IP 範囲により Amazon DCV ポートへのアクセスをさらに制限し、`Port` 設定を使用して非標準ポートを設定します。

更新ポリシー: この設定が変更された場合、更新は許可されません。

Dcv のプロパティ

`Enabled` (必須、`Boolean`)

ヘッドノードで Amazon DCV を有効にするかどうかを指定します。デフォルト値は `false` です。

更新ポリシー: この設定が変更された場合、更新は許可されません。

> **Note**
>
> Amazon DCV は、Amazon DCV クライアントとヘッドノードで実行される Amazon DCV サーバー間のトラフィックを保護するために使用される自己署名証明書を自動的に生成し

ます。独自の証明書を設定するには、「Amazon DCV の HTTPS 証明書」を参照してください。

`Port` (オプション、`Integer`)

Amazon DCV のポートを指定します。デフォルト値は 8443 です。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`AllowedIps` (オプション、推奨)、`String`)

Amazon DCV に接続するための CIDR 形式の IP 範囲を指定します。この設定は、 がセキュリティグループ AWS ParallelCluster を作成する場合にのみ使用されます。デフォルト値は `0.0.0.0/0` で、すべてのインターネットアドレスからのアクセスを許可します。

更新ポリシー: この設定は、更新中に変更できます。

CustomActions

(オプション) ヘッドノードで実行するカスタムスクリプトを指定します。

```
CustomActions:
  OnNodeStart:
    Sequence:
      - Script: string
        Args:
          - string
    Script: string
    Args:
      - string
  OnNodeConfigured:
    Sequence:
      - Script: string
        Args:
          - string
    Script: string
    Args:
      - string
  OnNodeUpdated:
    Sequence:
      - Script: string
```

```
        Args:
          - string
    Script: string
    Args:
      - string
```

CustomActions のプロパティ

OnNodeStart (オプション)

ノードデプロイのブートストラップアクションが開始される前にヘッドノードで実行する単一のスクリプトまたは一連のスクリプトを指定します。詳細については、「カスタムブートストラップアクション」を参照してください。

Sequence (オプション)

実行するスクリプトのリスト。 は、設定ファイルに記載されているのと同じ順序でスクリプト AWS ParallelCluster を実行します。最初から開始します。

Script (必須、String)

使用するファイルを指定します。ファイルパスは `https://` または `s3://` で始まる必要があります。

Args (オプション、[String])

スクリプトに渡す引数のリスト。

Script (必須、String)

単一のスクリプトに使用するファイルを指定します。ファイルパスは `https://` または `s3://` で始まる必要があります。

Args (オプション、[String])

単一のスクリプトに渡す引数のリスト。

更新ポリシー: この設定が変更された場合、更新は許可されません。

OnNodeConfigured (オプション)

ノードのブートストラップアクションが完了した後に、ヘッドノードで実行する単一のスクリプトまたは一連のスクリプトを指定します。詳細については、「カスタムブートストラップアクション」を参照してください。

Sequence (オプション)

実行するスクリプトのリストを指定します。

Script (必須、String)

使用するファイルを指定します。ファイルパスは https:// または s3:// で始まる必要があります。

Args (オプション、[String])

スクリプトに渡す引数のリスト。

Script (必須、String)

単一のスクリプトに使用するファイルを指定します。ファイルパスは https:// または s3:// で始まる必要があります。

Args (オプション、[String])

単一のスクリプトに渡す引数のリスト。

更新ポリシー: この設定が変更された場合、更新は許可されません。

OnNodeUpdated (オプション)

ノードの更新アクションが完了した後に、ヘッドノードで実行する単一のスクリプトまたは一連のスクリプトを指定します。詳細については、「カスタムブートストラップアクション」を参照してください。

Sequence (オプション)

実行するスクリプトのリストを指定します。

Script (必須、String)

使用するファイルを指定します。ファイルパスは https:// または s3:// で始まる必要があります。

Args (オプション、[String])

スクリプトに渡す引数のリスト。

Script (必須、String)

単一のスクリプトに使用するファイルを指定します。ファイルパスは https:// または s3:// で始まる必要があります。

`Args` (オプション、`[String]`)

単一のスクリプトに渡す引数のリスト。

更新ポリシー: この設定は、更新中に変更できます。

> **Note**
>
> `OnNodeUpdated` が AWS ParallelCluster 3.4.0 から追加されました。
> `Sequence` バージョン 3.6.0 AWS ParallelCluster 以降で が追加されました。を指定すると`Sequence`、カスタムアクションの複数のスクリプトを一覧表示できます。 は、を含めずに 1 つのスクリプトでカスタムアクションの設定をサポートするように AWS ParallelCluster 続行します`Sequence`。
> AWS ParallelCluster は、1 つのスクリプトと の両方を同じカスタムアクション`Sequence`に含めることをサポートしていません。

`Iam`

(オプション) クラスターのデフォルトのインスタンスロールまたはインスタンスプロファイルをオーバーライドするために、ヘッドノードで使用するインスタンスロールまたはインスタンスプロファイルのいずれかを指定します。

```
Iam:
  InstanceRole: string
  InstanceProfile: string
  S3Access:
    - BucketName: string
      EnableWriteAccess: boolean
      KeyName: string
  AdditionalIamPolicies:
    - Policy: string
```

更新ポリシー: この設定は、更新中に変更できます。

`Iam` のプロパティ

`InstanceProfile` (オプション、`String`)

デフォルトのヘッドノードのインスタンスプロファイルをオーバーライドするインスタンスプロファイルを指定します。`InstanceProfile` と `InstanceRole` の両方を

指定することはできません。形式は arn:*Partition*:iam::*Account*:instance-profile/*InstanceProfileName* です。

これを指定すると、S3Access と AdditionalIamPolicies の設定を指定することはできません。

AWS ParallelCluster に追加された機能は新しいアクセス許可を必要とすることが多いため、S3Access と AdditionalIamPolicies の設定のいずれかまたは両方を指定することをお勧めします。

更新ポリシー: この設定が変更された場合、更新は許可されません。

InstanceRole (オプション、String)

デフォルトのヘッドノードのインスタンスロールをオーバーライドするインスタンスロールを指定します。InstanceProfile と InstanceRole の両方を指定することはできません。形式は arn:*Partition*:iam::*Account*:role/*RoleName* です。

これを指定すると、S3Access と AdditionalIamPolicies の設定を指定することはできません。

AWS ParallelCluster に追加された機能は新しいアクセス許可を必要とすることが多いため、S3Access と AdditionalIamPolicies の設定のいずれかまたは両方を指定することをお勧めします。

更新ポリシー: この設定は、更新中に変更できます。

S3Access

S3Access (オプション)

バケットを指定します。これは、指定されたアクセスをバケットに付与するためのポリシーを生成するために使用されます。

これを指定すると、InstanceProfile と InstanceRole の設定を指定することはできません。

AWS ParallelCluster に追加された機能は新しいアクセス許可を必要とすることが多いため、S3Access と AdditionalIamPolicies の設定のいずれかまたは両方を指定することをお勧めします。

```
S3Access:
  - BucketName: string
    EnableWriteAccess: boolean
    KeyName: string
```

更新ポリシー: この設定は、更新中に変更できます。

BucketName (必須)、String)

バケットの名前。

更新ポリシー: この設定は、更新中に変更できます。

KeyName (オプション、String)

バケットのキーです。デフォルト値は「*」です。

更新ポリシー: この設定は、更新中に変更できます。

EnableWriteAccess (オプション、Boolean)

バケットに対して書き込みアクセスが可能かどうかを示す。デフォルト値は false です。

更新ポリシー: この設定は、更新中に変更できます。

AdditionalIamPolicies

AdditionalIamPolicies (オプション)

Amazon EC2 の IAM ポリシーの Amazon リソースネーム (ARN) のリストを指定します。このリストは、 に必要なアクセス許可に加えて、ヘッドノードに使用されるルートロールにアタッチされます AWS ParallelCluster。

IAM ポリシー名とその ARN が異なります。名前を使用することはできません。

これを指定すると、InstanceProfile と InstanceRole の設定を指定することはできません。

AWS ParallelCluster が必要とするパーミッションに AdditionalIamPolicies が追加され、InstanceRole には必要なパーミッションがすべて含まれている必要があるため、AdditionalIamPolicies を使用することをお勧めします。必要な権限は、機能が追加されるにつれ、リリースごとに変更されることがよくあります。

デフォルト値はありません。

```
AdditionalIamPolicies:
  - Policy: string
```

更新ポリシー: この設定は、更新中に変更できます。

`Policy` (オプション、`[String]`)

IAM ポリシーの一覧。

更新ポリシー: この設定は、更新中に変更できます。

`Imds`

(オプション)インスタンスメタデータサービス (IMDS) のプロパティを指定します。詳細については、Amazon EC22 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 の仕組み」を参照してください。

```
Imds:
    Secured: boolean
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

`Imds` のプロパティ

`Secured` (オプション、`Boolean`)

`true` の場合、ヘッドノードの IMDS (およびインスタンスプロファイルの認証情報) へのアクセスが一部のスーパーユーザーに制限されます。

`false` の場合、ヘッドノードのすべてのユーザーがヘッドノードの IMDS にアクセスできます。

次のユーザーにはヘッドノードの IMDS へのアクセスが許可されています。

- ルートユーザー
- クラスターの管理者ユーザー (デフォルトでは `pc-cluster-admin`)
- オペレーティングシステム固有のデフォルトのユーザー (Amazon Linux 2 と RedHat の `ec2-user`、Ubuntu 18.04 の `ubuntu`)

デフォルトは `true` です。

default ユーザーは、クラスターが AWS リソースとやり取りするために必要なアクセス許可を持っていることを確認する責任があります。default ユーザー IMDS アクセスを無効にすると、 AWS ParallelCluster はコンピューティングノードを管理できず、動作が停止します。default ユーザーの IMDS アクセスを無効にしないでください。

ユーザーにヘッドノードの IMDS へのアクセス許可を付与すると、ヘッドノードのインスタンスプロファイルに含まれるアクセス許可を使用できます。例えば、これらのアクセス許可を使用して Amazon EC2 インスタンスを起動したり、クラスターを設定して認証のために使用する AD ドメインのパスワードを読み取ったりすることができます。

IMDS アクセスを制限するには、 が のチェーン AWS ParallelCluster を管理しますiptables。

sudo アクセスを持つクラスターユーザーは、コマンドを実行することで、default ユーザーを含む他の個々のユーザーのヘッドノードの IMDS へのアクセスを選択的に有効または無効にできます。

```
$ sudo /opt/parallelcluster/scripts/imds/imds-access.sh --allow <USERNAME>
```

このコマンドの --deny オプションを使用して、ユーザーの IMDS アクセスを無効にできます。

default ユーザーの IMDS アクセスを知らないうちに無効にしてしまった場合は、--allow オプションを使用してアクセス許可を復元できます。

> **Note**
>
> iptables または ip6tables ルールのカスタマイズは、ヘッドノードの IMDS アクセスを制限するために使用するメカニズムを妨げる可能性があります。

更新ポリシー: この設定が変更された場合、更新は許可されません。

Image

(オプション) ヘッドノードのカスタムイメージを定義します。

```
Image:
    CustomAmi: string
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

Image のプロパティ

CustomAmi (オプション、String)

デフォルト AMI の代わりにヘッドノードに使用するカスタム AMI の ID を指定します。詳細については、「AWS ParallelCluster AMI のカスタマイズ」を参照してください。

カスタム AMI の起動に追加のアクセス許可が必要な場合は、ユーザーおよびヘッドノードポリシーの両方にそれらのアクセス許可を追加する必要があります。

例えば、カスタム AMI に暗号化されたスナップショットが関連付けられている場合、ユーザーおよびヘッドノードポリシーの両方に次の追加のポリシーが必要です。

```
{
     "Version": "2012-10-17",
     "Statement": [
         {
             "Effect": "Allow",
             "Action": [
                 "kms:DescribeKey",
                 "kms:ReEncrypt*",
                 "kms:CreateGrant",
                 "kms:Decrypt"
             ],
             "Resource": [
                 "arn:aws:kms:<AWS_REGION>:<AWS_ACCOUNT_ID>:key/<AWS_KMS_KEY_ID>"
             ]
         }
     ]
  }
```

カスタム AMI 検証の警告のトラブルシューティングについては、「カスタム AMI の問題のトラブルシューティング」を参照してください。

更新ポリシー: この設定が変更された場合、更新は許可されません。

Scheduling セクション

(必須) クラスターで使用されるジョブスケジューラと、ジョブスケジューラが管理するコンピューティングインスタンスを定義します。Slurm または ス AWS Batch ケジューラを使用できます。それぞれがサポートする一連の設定とプロパティは異なります。

トピック

- Scheduling のプロパティ
- AwsBatchQueues
- SlurmQueues
- SlurmSettings

```
Scheduling:
  Scheduler: slurm
  ScalingStrategy: string
  SlurmSettings:
    MungeKeySecretArn: string
    ScaledownIdletime: integer
    QueueUpdateStrategy: string
    EnableMemoryBasedScheduling: boolean
    CustomSlurmSettings: [dict]
    CustomSlurmSettingsIncludeFile: string
    Database:
      Uri: string
      UserName: string
      PasswordSecretArn: string
      DatabaseName: string
    ExternalSlurmdbd: boolean
      Host: string
      Port: integer
    Dns:
      DisableManagedDns: boolean
      HostedZoneId: string
      UseEc2Hostnames: boolean
  SlurmQueues:
    - Name: string
      ComputeSettings:
        LocalStorage:
          RootVolume:
            Size: integer
            Encrypted: boolean
            VolumeType: string
            Iops: integer
            Throughput: integer
          EphemeralVolume:
            MountDir: string
      CapacityReservationTarget:
```

```
        CapacityReservationId: string
        CapacityReservationResourceGroupArn: string
      CapacityType: string
      AllocationStrategy: string
      JobExclusiveAllocation: boolean
      CustomSlurmSettings: dict
      Tags:
        - Key: string
          Value: string
      HealthChecks:
        Gpu:
          Enabled: boolean
      Networking:
        SubnetIds:
          - string
        AssignPublicIp: boolean
        SecurityGroups:
          - string
        AdditionalSecurityGroups:
          - string
        PlacementGroup:
          Enabled: boolean
          Id: string
          Name: string
        Proxy:
          HttpProxyAddress: string
      ComputeResources:
        - Name: string
          InstanceType: string
          Instances:
            - InstanceType: string
          MinCount: integer
          MaxCount: integer
          DynamicNodePriority: integer
          StaticNodePriority: integer
          SpotPrice: float
          DisableSimultaneousMultithreading: boolean
          SchedulableMemory: integer
          HealthChecks:
            Gpu:
              Enabled: boolean
          Efa:
            Enabled: boolean
            GdrSupport: boolean
```

```
          CapacityReservationTarget:
            CapacityReservationId: string
            CapacityReservationResourceGroupArn: string
          Networking:
            PlacementGroup:
              Enabled: boolean
              Name: string
          CustomSlurmSettings: dict
          Tags:
            - Key: string
              Value: string
      CustomActions:
        OnNodeStart:
          Sequence:
            - Script: string
              Args:
                - string
          Script: string
          Args:
            - string
        OnNodeConfigured:
          Sequence:
            - Script: string
              Args:
                - string
          Script: string
          Args:
            - string
      Iam:
        InstanceProfile: string
        InstanceRole: string
        S3Access:
          - BucketName: string
            EnableWriteAccess: boolean
            KeyName: string
        AdditionalIamPolicies:
          - Policy: string
      Image:
        CustomAmi: string
```

```
Scheduling:
  Scheduler: awsbatch
  AwsBatchQueues:
```

```
    - Name: string
      CapacityType: string
      Networking:
        SubnetIds:
          - string
        AssignPublicIp: boolean
        SecurityGroups:
          - string
        AdditionalSecurityGroups:
          - string
      ComputeResources:  # this maps to a Batch compute environment (initially we
 support only 1)
        - Name: string
          InstanceTypes:
            - string
          MinvCpus: integer
          DesiredvCpus: integer
          MaxvCpus: integer
          SpotBidPercentage: float
```

Scheduling のプロパティ

Scheduler (必須)、**String**)

使用するスケジューラのタイプを指定します。サポートされている値はslurm および awsbatch です。

更新ポリシー: この設定が変更された場合、更新は許可されません。

> **Note**
>
> awsbatch は alinux2 オペレーティングシステムと x86_64 プラットフォームのみをサポートします。

ScalingStrategy (オプション、**String**)

動的 Slurm ノードのスケールアップ方法を選択できます。サポートされている値は all-or-nothing、greedy-all-or-nothing、best-effort です。デフォルト値は all-or-nothing です。

更新ポリシー: この設定は、更新中に変更できます。

> **Note**
> スケーリング戦略は、Slurm で再開するノードにのみ適用され、既に実行中であるノードには最終的に適用されません。

- `all-or-nothing`: この戦略は、オールオアナッシングアプローチに厳密に従い、スケーリングプロセスの最後にアイドルインスタンスが発生しないようにすることを目的としています。オールオアナッシングベースで動作し、完全にスケールアップするか、まったくスケールアップしないかのいずれかになります。ジョブに必要なノードが 500 を超える場合や、ジョブが複数のコンピューティングリソースにまたがる場合は、一時的に起動されるインスタンスがあるため、追加コストがあり得ることに注意してください。この戦略は、3 つの可能なスケーリング戦略の中でスループットが最も低くなります。スケーリング時間は、Slurm 再開プログラムの実行ごとに送信されるジョブの数によって異なります。また、実行あたりのデフォルトの RunInstances リソースアカウント制限 (デフォルトでは 1,000 インスタンス) をはるかに超えてスケールすることはできません。詳細については、Amazon EC2 API スロットリングに関するドキュメントを参照してください。
- `greedy-all-or-nothing` ：オールオアナッシング戦略と同様に、スケーリング後のアイドルインスタンスを回避することを目的としています。この戦略では、スケーリングプロセス中に一時的なオーバースケーリングが可能であるため、オールオアナッシング戦略よりも高いスループットを実現できますが、RunInstances リソースアカウント制限と同じ 1,000 インスタンスのスケーリング制限が適用されます。
- `best-effort` ：この戦略では、スケーリングプロセスの最後に一部のインスタンスがアイドル状態になる可能性があるとしても、高スループットを優先します。ジョブがリクエストしている数だけのノードを割り当てようとしますが、リクエスト全体は満たせない可能性があります。他の戦略とは異なり、ベストエフォートアプローチでは、複数のスケーリングプロセスの実行時にアイドルリソースが発生するという代償が伴いますが、標準の RunInstances 制限よりも多くのインスタンスを蓄積できます。

各戦略は、さまざまなスケーリングニーズを満たすように設計されているため、特定の要件や制約に合った戦略を選択できます。

`AwsBatchQueues`

（オプション）AWS Batch キュー設定。サポートされているキューは 1 つだけです。このセクションは、`Scheduler` が `awsbatch` に設定されている場合に必要です。`awsbatch` スケジューラの詳

細については、「Networking Setup」と「での AWS Batch（awsbatch) スケジューラの使用 AWS ParallelCluster」を参照してください。

```
AwsBatchQueues:
  - Name: string
    CapacityType: string
    Networking:
      SubnetIds:
        - string
      AssignPublicIp: boolean
      SecurityGroups:
        - string
      AdditionalSecurityGroups:
        - string
    ComputeResources:  # this maps to a Batch compute environment (initially we support
 only 1)
      - Name: string
        InstanceTypes:
          - string
        MinvCpus: integer
        DesiredvCpus: integer
        MaxvCpus: integer
        SpotBidPercentage: float
```

更新ポリシー: この設定は、更新中に変更できます。

AwsBatchQueues のプロパティ

Name (必須)、**String**)

AWS Batch キューの名前。

更新ポリシー: この設定が変更された場合、更新は許可されません。

CapacityType (オプション、**String**)

AWS Batch キューが使用するコンピューティングリソースのタイプ。サポートされている値は ONDEMAND、SPOT、または CAPACITY_BLOCK です。デフォルト値は ONDEMAND です。

> **Note**
>
> `CapacityType` を `SPOT` に設定した場合、アカウントに `AWSServiceRoleForEC2Spot` サービスにリンクされたロールが含まれている必要があります。このロールは、次の AWS CLI コマンドを使用して作成できます。
>
> ```
> $ aws iam create-service-linked-role --aws-service-name spot.amazonaws.com
> ```
>
> 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「スポットインスタンスリクエスト向けのサービスにリンクされたロール」を参照してください。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

`Networking`

（必須）AWS Batch キューのネットワーク設定を定義します。

```
Networking:
  SubnetIds:
    - string
  AssignPublicIp: boolean
  SecurityGroups:
    - string
  AdditionalSecurityGroups:
    - string
```

`Networking` のプロパティ

`SubnetIds` (必須)、**`[String]`**)

AWS Batch キューをプロビジョニングする既存のサブネットの ID を指定します。現在、サポートされているのは 1 つのサブネットのみです。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

AssignPublicIp (オプション、**String**)

AWS Batch キュー内のノードにパブリック IP アドレスを作成または割り当てます。サポートされている値は`true` および `false` です。デフォルトは、指定されたサブネットによって異なります。

更新ポリシー: この設定が変更された場合、更新は許可されません。

SecurityGroups (オプション、**[String]**)

AWS Batch キューが使用するセキュリティグループのリスト。セキュリティグループを指定しない場合、 は新しいセキュリティグループ AWS ParallelCluster を作成します。

更新ポリシー: この設定は、更新中に変更できます。

AdditionalSecurityGroups (オプション、**[String]**)

AWS Batch キューが使用するセキュリティグループのリスト。

更新ポリシー: この設定は、更新中に変更できます。

ComputeResources

（必須）AWS Batch キューの ComputeResources 設定を定義します。

```
ComputeResources:  # this maps to a Batch compute environment (initially we support
 only 1)
  - Name: string
    InstanceTypes:
      - string
    MinvCpus: integer
    DesiredvCpus: integer
    MaxvCpus: integer
    SpotBidPercentage: float
```

ComputeResources のプロパティ

Name (必須)、**String**)

AWS Batch キューコンピューティング環境の名前。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

InstanceTypes (必須)、**[String]**)

インスタンスタイプの AWS Batch コンピューティング環境配列。すべてのインスタンスタイプが x86_64 アーキテクチャを使用する必要があります。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

MinvCpus (オプション、**Integer**)

AWS Batch コンピューティング環境で使用できる VCPUs の最小数。

更新ポリシー: この設定は、更新中に変更できます。

DesiredVcpus (オプション、**Integer**)

AWS Batch コンピューティング環境で必要な VCPUs の数。 AWS Batch は、ジョブキューの需要`MaxvCpus`に基づいて、この値を `MinvCpus` と の間で調整します。

更新ポリシー: この設定は、更新中には分析されません。

MaxvCpus (オプション、**Integer**)

AWS Batch コンピューティング環境VCPUs の最大数。これを `DesiredVcpus` より小さい値に設定することはできません。

更新ポリシー: これらの設定は、更新中に減らすことはできません。

SpotBidPercentage (オプション、**Float**)

インスタンスの起動前に、Amazon EC2 スポットインスタンス料金がオンデマンドインスタンスタイプ料金に到達可能な最大パーセンテージ。デフォルト値は `100` (100%) です。対応する範囲は `1` ~ `100` です。

更新ポリシー: この設定は、更新中に変更できます。

SlurmQueues

(オプション) Slurm キューの設定。このセクションは、`Scheduler` が `slurm` に設定されている場合に必要です。

```
SlurmQueues:
  - Name: string
    ComputeSettings:
      LocalStorage:
```

```
        RootVolume:
          Size: integer
          Encrypted: boolean
          VolumeType: string
          Iops: integer
          Throughput: integer
        EphemeralVolume:
          MountDir: string
    CapacityReservationTarget:
      CapacityReservationId: string
      CapacityReservationResourceGroupArn: string
    CapacityType: string
    AllocationStrategy: string
    JobExclusiveAllocation: boolean
    CustomSlurmSettings: dict
    Tags:
      - Key: string
        Value: string
    HealthChecks:
      Gpu:
        Enabled: boolean
    Networking:
      SubnetIds:
        - string
      AssignPublicIp: boolean
      SecurityGroups:
        - string
      AdditionalSecurityGroups:
        - string
      PlacementGroup:
        Enabled: boolean
        Id: string
        Name: string
      Proxy:
        HttpProxyAddress: string
    ComputeResources:
      - Name: string
        InstanceType: string
        Instances:
          - InstanceType: string
        MinCount: integer
        MaxCount: integer
        DynamicNodePriority: integer
        StaticNodePriority: integer
```

```
        SpotPrice: float
        DisableSimultaneousMultithreading: boolean
        SchedulableMemory: integer
        HealthChecks:
          Gpu:
            Enabled: boolean
        Efa:
          Enabled: boolean
          GdrSupport: boolean
        CapacityReservationTarget:
          CapacityReservationId: string
          CapacityReservationResourceGroupArn: string
        Networking:
          PlacementGroup:
            Enabled: boolean
            Name: string
        CustomSlurmSettings: dict
        Tags:
          - Key: string
            Value: string
    CustomActions:
      OnNodeStart:
        Sequence:
          - Script: string
            Args:
              - string
        Script: string
        Args:
          - string
      OnNodeConfigured:
        Sequence:
          - Script: string
            Args:
              - string
        Script: string
        Args:
          - string
    Iam:
      InstanceProfile: string
      InstanceRole: string
      S3Access:
        - BucketName: string
          EnableWriteAccess: boolean
          KeyName: string
```

```
      AdditionalIamPolicies:
        - Policy: string
    Image:
      CustomAmi: string
```

更新ポリシー:このリスト値の設定では、更新中に新しい値を追加することができ、既存の値を削除する場合はコンピューティングフリートを停止する必要があります。

SlurmQueues のプロパティ

Name (必須)、**String**)

Slurm キューの名前。

> **Note**
> クラスターのサイズは、更新中に変わる場合があります。詳細については、「クラスター容量のサイズと更新」を参照してください。

更新ポリシー: この設定が変更された場合、更新は許可されません。

CapacityReservationTarget

> **Note**
> `CapacityReservationTarget` が AWS ParallelCluster バージョン 3.3.0 で追加されました。

```
CapacityReservationTarget:
   CapacityReservationId: string
   CapacityReservationResourceGroupArn: string
```

キューのコンピューティングリソースのオンデマンドキャパシティ予約を指定します。

CapacityReservationId (オプション、**String**)

キューのコンピューティングリソースを対象とする既存のキャパシティ予約の ID。ID は、ODCR または 機械学習用のキャパシティブロックを参照できます。

予約は、インスタンスが使用するのと同じプラットフォームを使用する必要があります。例えば、インスタンスを rhel8 で実行する場合、キャパシティ予約は Red Hat Enterprise Linux プラットフォームで実行する必要があります。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「サポートされているプラットフォーム」を参照してください。

> **Note**
>
> クラスター設定に Instances を含める場合、このキューレベルの CapacityReservationId 設定を設定から除外する必要があります。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

CapacityReservationResourceGroupArn (オプション、**String**)

キューのコンピューティングリソースのキャパシティ予約のサービスリンクグループとして機能するリソースグループの Amazon リソースネーム (ARN)。 AWS ParallelCluster は、次の条件に基づいて、リソースグループから最も適切なキャパシティ予約を識別して使用します。

- PlacementGroup が SlurmQueues/ Networkingまたは SlurmQueues / ComputeResources / で有効になっている場合Networking、コンピューティングリソースが存在する場合、 はコンピューティングリソースPlacementGroupのインスタンスタイプと をターゲットとするリソースグループ AWS ParallelCluster を選択します。

  PlacementGroup は、ComputeResources で定義されているインスタンスタイプの 1 つをターゲットにしている必要があります。
- PlacementGroup / Networkingまたは SlurmQueues / SlurmQueues / ComputeResources で が有効になっていない場合Networking、コンピューティングリソースが存在する場合、 はコンピューティングリソースのインスタンスタイプのみをターゲットとするリソースグループ AWS ParallelCluster を選択します。

リソースグループには、キューのすべてのコンピューティングリソースとアベイラビリティーゾーン間のアベイラビリティーゾーンに予約されているインスタンスタイプごとに少なくとも 1 つの ODCR があることが必要です。詳細については、「オンデマンドキャパシティ予約 (ODCR) を使用してインスタンスを起動する」を参照してください。

複数のサブネットの設定要件の詳細については、「Networking」/「SubnetIds」を参照してください。

> **Note**
>
> AWS ParallelCluster バージョン 3.4.0 では、複数のアベイラビリティーゾーンが追加されています。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

CapacityType (オプション、**String**)

Slurm キューが使用するコンピューティングリソースのタイプ。サポートされている値は ONDEMAND、SPOT、または CAPACITY_BLOCK です。デフォルト値は ONDEMAND です。

> **Note**
>
> CapacityType を SPOT に設定した場合、アカウントに AWSServiceRoleForEC2Spot サービスにリンクされたロールがある必要があります。次の AWS CLI コマンドを使用して、このロールを作成できます。
>
> ```
> $ aws iam create-service-linked-role --aws-service-name spot.amazonaws.com
> ```
>
> 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「スポットインスタンスリクエスト向けのサービスにリンクされたロール」を参照してください。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

AllocationStrategy (オプション、**String**)

Instances で定義されているすべてのコンピューティングリソースの配分戦略を指定します。

有効な値: lowest-price | capacity-optimized | price-capacity-optimized

デフォルト: lowest-price

lowest-price

- CapacityType = ONDEMAND を使用すると、Amazon EC2 Fleet は料金に従って順序を決定し、最低価格のインスタンスを最初に起動します。

- `CapacityType = SPOT` を使用すると、Amazon EC2 Fleet は、利用可能な容量のある最低価格のスポットインスタンスプールからインスタンスを起動します。プールが必要な容量を満たす前に容量が不足した場合、Amazon EC2 フリート はインスタンスを起動してリクエストを実行します。特に、Amazon EC2 Fleet は、利用可能な容量のある最低価格のスポットインスタンスプールからインスタンスを起動します。Amazon EC2 Fleet はいくつかの異なるプールからスポットインスタンスを起動することがあります。
- `CapacityType = CAPACITY_BLOCK` を設定すると、割り当て戦略がないため、`AllocationStrategy` パラメータを設定できません。

`capacity-optimized`

- `CapacityType = ONDEMAND` を設定した場合、`capacity-optimized` は使用できません。
- `CapacityType = SPOT` を設定すると、Amazon EC2 Fleet は、起動するインスタンス数に最適な容量を持つスポットインスタンスプールからインスタンスを起動します。

`price-capacity-optimized`

- `CapacityType = ONDEMAND` を設定した場合、`capacity-optimized` は使用できません。
- `CapacityType = SPOT` を設定すると、Amazon EC2 Fleet は、起動中のインスタンスの数に最適な容量を利用できるプールを識別します。つまり、短期的に中断の可能性が最も低いと思われるプールからスポットインスタンスをリクエストすることになります。次に Amazon EC2 Fleet は、これらのプールのうち最も価格の低いスポットインスタンスをリクエストします。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、`QueueUpdateStrategy` が設定されている必要があります。

> **Note**
>
> `AllocationStrategy` は AWS ParallelCluster バージョン 3.3.0 からサポートされています。

`JobExclusiveAllocation` (オプション、**`String`**)

`true` に設定した場合、Slurm パーティションの `OverSubscribe` フラグは `EXCLUSIVE` に設定されます。`OverSubscribe=EXCLUSIVE` の場合、パーティションのジョブは割り当てられ

たすべてのノードに排他的にアクセスできます。詳細については、「Slurm ドキュメント」の「EXCLUSIVE」を参照してください。

有効な値: `true` | `false`

デフォルト: `false`

更新ポリシー: この設定は、更新中に変更できます。

> **Note**
>
> `JobExclusiveAllocation` は AWS ParallelCluster バージョン 3.7.0 からサポートされています。

`CustomSlurmSettings` (オプション、**`Dict`**)

カスタム Slurm パーティション (キュー) 構成設定を定義します。

キュー (パーティション) に適用されるカスタム Slurm 設定パラメータのキーと値のペアのディクショナリを指定します。

`Param1: Value1` などの各個別のキーと値のペアは、`Param1=Value1` の形式で Slurm パーティション設定行の最後に個別に追加されます。

`CustomSlurmSettings` の拒否リストに記載されていない Slurm 設定パラメータのみ指定できます。拒否リストの Slurm 設定パラメータの詳細については、「`CustomSlurmSettings` で拒否リストに記載されている Slurm 設定パラメータ」を参照してください。

AWS ParallelCluster は、パラメータが拒否リストにあるかどうかのみをチェックします。 AWS ParallelCluster はカスタムSlurm設定パラメータの構文またはセマンティクスを検証しません。カスタムSlurm設定パラメータを検証するのはユーザーの責任です。無効なカスタム Slurm 設定パラメータは、クラスターの作成や更新の失敗につながる Slurm デーモンの障害を引き起こす可能性があります。

でカスタムSlurm設定パラメータを指定する方法の詳細については AWS ParallelCluster、「」を参照してくださいSlurm 設定のカスタマイズ。

Slurm 設定パラメータに関する詳細については、「Slurm ドキュメント」の「slurm.conf」を参照してください。

更新ポリシー: この設定は、更新中に変更できます。

> **Note**
>
> `CustomSlurmSettings` は AWS ParallelCluster バージョン 3.6.0 からサポートされています。

Tags (オプション、[文字列])

タグのキーと値のペアのリスト。`ComputeResource` タグは、Tags セクション または `SlurmQueues/Tags` で指定された重複するタグを上書きします。

Key (オプション、**String**)

タグキー。

Value (オプション、**String**)

タグ値。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、`QueueUpdateStrategy` が設定されている必要があります。

HealthChecks (オプション)

キューのすべてのコンピューティングリソースのコンピューティングノードのヘルスチェックを指定します。

Gpu (オプション)

キューのすべてのコンピューティングリソースの GPU のヘルスチェックを指定します。

> **Note**
>
> AWS ParallelCluster は`HealthChecks`、ARM オペレーティングシステムを使用するノード`Gpu`で / `alinux2` をサポートしていません。これらのプラットフォームは NVIDIA データセンター GPUマネージャー (DCGM) をサポートしていません。

Enabled (オプション、**Boolean**)

がコンピューティングノードで GPU ヘルスチェック AWS ParallelCluster を実行するかどうか。デフォルトは `false` です。

Gpu ヘルスチェックの動作

- Gpu/Enabled が true に設定されている場合、 AWS ParallelCluster はキューのコンピューティングリソースで GPU ヘルスチェックを実行します。
- Gpu ヘルスチェックはコンピューティングリソースで GPU ヘルスチェックを実行して、GPU のパフォーマンスが低下したノードでのジョブ送信を防止します。
- コンピューティングノードが Gpu ヘルスチェックに失敗すると、コンピューティングノードのステータスは DRAIN に変わります。このノードで新しいジョブは開始されません。既存のジョブは完了まで実行されます。実行中のジョブがすべて完了すると、コンピューティングノードが動的ノードの場合は終了し、静的ノードの場合は置き換えられます。
- Gpu ヘルスチェックの所要時間は、選択したインスタンスタイプ、インスタンスの GPU の数、および Gpu ヘルスチェックのターゲットの数 (ジョブ GPU ターゲットの数と同じ) によって異なります。8 つの GPU を持つインスタンスの場合、一般的な所要時間は 3 分未満です。
- サポートされていないインスタンスで Gpu ヘルスチェックを実行すると終了し、ジョブはコンピューティングノードで実行されます。例えば、インスタンスが GPU を使用していない場合や、インスタンスが GPU を使用していても NVIDIA GPU ではない場合、ヘルスチェックは終了し、ジョブはコンピューティングノードで実行されます。NVIDIA GPU のみがサポートされています。
- Gpu ヘルスチェックは、dcgmi ツールを使用してノードのヘルスチェックを実行し、次の手順を実行します。

  ノードで Gpu ヘルスチェックが開始される場合。

  1. nvidia-dcgm および nvidia-fabricmanager サービスが実行中かどうかを検出します。
  2. これらのサービスが実行されていない場合、Gpu ヘルスチェックによって開始されます。
  3. 永続化モードが有効になっているかどうかを検出します。
  4. 永続化モードが有効になっていない場合は、Gpu ヘルスチェックによって有効になります。

  ヘルスチェックの終了時に、Gpu ヘルスチェックはこれらのサービスとリソースの初期状態を復元します。
- ジョブが特定のノード GPU セットに割り当てられている場合、Gpu ヘルスチェックはその特定のセットでのみ実行されます。それ以外の場合は、ノードのすべての GPU で Gpu ヘルスチェックが実行されます。

- コンピューティングノードが同時に 2 つ以上の Gpu ヘルスチェックのリクエストを受け取った場合、最初のヘルスチェックのみが実行され、他はスキップされます。これは、ノード GPUs をターゲットとするヘルスチェックにも当てはまります。この状況に関する追加情報については、ログファイルを確認できます。
- 特定のコンピューティングノードのヘルスチェックログは `/var/log/parallelcluster/slurm_health_check.log` ファイルで使用可能です。このファイルは クラスター CloudWatch ロググループの Amazon CloudWatch で使用可能で、次があります。
  - サービスおよび永続化モードの有効化と無効化を含め、Gpu ヘルスチェックによって実行されたアクションの詳細。
  - GPU の識別子、シリアル ID、および UUID。
  - ヘルスチェックの出力。

更新ポリシー: この設定は、更新中に変更できます。

> **Note**
>
> `HealthChecks` は AWS ParallelCluster バージョン 3.6.0 以降でサポートされています。

Networking

(必須) Slurm キューのネットワーク設定を定義します。

```
Networking:
  SubnetIds:
    - string
  AssignPublicIp: boolean
  SecurityGroups:
    - string
  AdditionalSecurityGroups:
    - string
  PlacementGroup:
    Enabled: boolean
    Id: string
    Name: string
  Proxy:
    HttpProxyAddress: string
```

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

Networking のプロパティ

SubnetIds (必須)、**[String]**)

Slurm キューをプロビジョニングする既存のサブネットの ID。

SlurmQueues/ComputeResources/InstanceType のインスタンスタイプを設定する場合、1 つのサブネットのみを定義できます。

SlurmQueues/ComputeResources/Instances のインスタンスタイプを設定する場合、単一のサブネットまたは複数のサブネットを定義できます。

複数のサブネットを使用する場合、キューに定義されるすべてのサブネットは同じ VPC 内にあり、各サブネットは個別のアベイラビリティーゾーン (AZ) にある必要があります。

例えば、キューに subnet-1 と subnet-2 を定義するとします。

`subnet-1` と `subnet-2` の両方を AZ-1 にすることはできません。

`subnet-1` は AZ-1 に、`subnet-2` は AZ-2 にすることができます。

1 つのインスタンスタイプのみを設定し、また複数のサブネットを使用する場合、`InstanceType` ではなく `Instances` でインスタンスタイプを定義します。

例えば、`ComputeResources/InstanceType = instance.type` の代わりに `ComputeResources/Instances/InstanceType = instance.type` を定義します。

> **Note**
>
> Elastic Fabric Adapter (EFA) は、異なるアベイラビリティーゾーン間ではサポートされていません。

複数のアベイラビリティーゾーンを使用すると、ストレージネットワーキングのレイテンシーが増加し、また inter-AZ のデータ転送コストが増加する可能性があります。例えば、インスタンスが異なる AZ に配置されているファイルストレージにアクセスする場合に発生することがあります。詳細については、「同一の AWS リージョンでのデータ転送」を参照してください。

クラスターを更新して、単一のサブネットから複数のサブネットを使用するように変更する。

- クラスターのサブネット定義が、単一のサブネットと AWS ParallelCluster マネージド FSx for Lustre ファイルシステムで定義されているとします。続いて、更新されたサブネット ID 定義を使用してこのクラスターを直接更新することはできません。クラスターを更新するには、まずマネージドファイルシステムを外部ファイルシステムに変更する必要があります。詳細については、「AWS ParallelCluster マネージドストレージを外部ストレージに変換する」を参照してください。
- 追加するように定義された複数のサブネットのすべての AZ に EFS マウントターゲットが存在しない場合、クラスターのサブネット定義が単一のサブネットと外部の Amazon EFS ファイルシステムで定義されているとします。続いて、更新されたサブネット ID 定義を使用してこのクラスターを直接更新することはできません。クラスターを更新するか、クラスターを作成するには、まず定義した複数のサブネットのすべての AZ のマウントターゲットをすべて作成する必要があります。

CapacityReservationResourceGroupArn で定義されているアベイラビリティーゾーンとクラスターのキャパシティ予約。

- 定義済みのキャパシティ予約のリソースグループおよびキューに定義されている一連のインスタンスタイプとアベイラビリティーゾーンの対象となる一連のインスタンスタイプとアベイラビリティーゾーンが重複していない場合、クラスターを作成することはできません。
- 定義されたキャパシティ予約リソースグループの対象となるインスタンスタイプとアベイラビリティーゾーンのセットと、キューに定義されたインスタンスタイプとアベイラビリティーゾーンのセットの間に部分的な重複がある場合は、クラスターを作成できます。 AWS ParallelCluster は、このケースの部分的な重複に関する警告メッセージを送信します。
- 詳細については、「オンデマンドキャパシティ予約 (ODCR) を使用してインスタンスを起動する」を参照してください。

> **Note**
>
> AWS ParallelCluster バージョン 3.4.0 では、複数のアベイラビリティーゾーンが追加されています。

> **Warning**
>
> この警告は、 AWS ParallelCluster このパラメータを変更しても、バージョン 3.3.1.version 3.3.1 より前のすべての 3.x.y AWS ParallelCluster バージョンには影響しません。
> バージョン AWS ParallelCluster 3.3.1 より前の 3 バージョンの場合：
> このパラメータを変更してクラスターを更新すると、新しいマネージド FSx for Lustre ファイルシステムが作成され、既存のデータを保存せずに既存のマネージド FSx for Lustre ファイルシステムが削除されます。その結果、データの損失が発生します。データを保持したい場合は、先に進む前に、必ず既存の FSx for Lustre ファイルシステムからデータをバックアップしてください。詳細については、「FSx for Lustre ユーザーガイド」の「Working with backups」を参照してください。

新しいサブネット値を追加する場合は、更新ポリシー: この設定は、更新中に変更できます。

サブネット値を削除する場合は、更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、`QueueUpdateStrategy` が設定されている必要があります。

`AssignPublicIp` (オプション、**`String`**)

Slurm キューのノードにパブリック IP アドレスを作成または割り当てます。サポートされている値は`true` および `false` です。指定したサブネットによってデフォルト値が決定されます。パブリック IP を持つサブネットは、デフォルトでパブリック IP アドレスが割り当てられます。

p4d または hpc6idインスタンスタイプ、または複数のネットワークインターフェイスまたはネットワークインターフェイスカードを持つ別のインスタンスタイプを定義する場合は、パブリックアクセスを提供する`true`ように `Networking` `HeadNode`// `ElasticIp` を に設定する必要があります。 AWS パブリック IPsは、単一のネットワークインターフェイスで起動されたインスタンスにのみ割り当てることができます。この場合、クラスターコンピューティングノードへのパブリックアクセスを提供するのに、NAT ゲートウェイを使用することをお勧めします。この場合は、`AssignPublicIp` を `false` に設定します。IP アドレスに関する詳細については、「Amazon EC2 Linux インスタンス用ユーザーガイド」の「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。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`SecurityGroups` (オプション、**`[String]`**)

Slurm キューに使用するセキュリティグループのリスト。セキュリティグループが指定されていない場合、 はセキュリティグループ AWS ParallelCluster を作成します。

セキュリティグループが SharedStorage システム用に正しく設定されていることを確認します。

> **Warning**
>
> この警告は、このパラメータを変更しても、バージョン 3.3.0. AWS ParallelCluster version 3.3.0 より前のすべての 3.*x*.*y* AWS ParallelCluster バージョンには影響しません。
> バージョン AWS ParallelCluster 3.3.0 より前の 3 バージョンの場合：
> このパラメータを変更してクラスターを更新すると、新しいマネージド FSx for Lustre ファイルシステムが作成され、既存のデータを保存せずに既存のマネージド FSx for Lustre ファイルシステムが削除されます。その結果、データの損失が発生します。データを保持したい場合は、必ず既存の FSx for Lustre ファイルシステムからデータをバックアップしてください。詳細については、「FSx for Lustre ユーザーガイド」の「Working with backups」を参照してください。

> **Warning**
>
> コンピューティングインスタンスに対して Efa を有効にする場合、EFA 対応のインスタンスが、インバウンドおよびアウトバウンドのトラフィックをすべて許可するセキュリティグループのメンバーであることを確認してください。

更新ポリシー: この設定は、更新中に変更できます。

`AdditionalSecurityGroups` (オプション、**`[String]`**)

Slurm キューに使用する追加のセキュリティグループのリスト。

更新ポリシー: この設定は、更新中に変更できます。

`PlacementGroup` (オプション)

Slurm キューの配置グループ設定を指定します。

```
PlacementGroup:
```

```
  Enabled: boolean
  Id: string
  Name: string
```

更新ポリシー: マネージドプレースメントグループを削除するには、すべてのコンピューティングノードを停止する必要があります。この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

Enabled (オプション、**Boolean**)

Slurm キューに配置グループを使用するかどうかを示します。デフォルトは `false` です。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

Id (オプション、**String**)

Slurm キューが使用する既存のクラスタープレイスメントグループのプレイスメントグループ ID。名前ではなく、プレイスメントグループ ID を必ず指定してください。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

Name (オプション、**String**)

Slurm キューが使用する既存のクラスタープレイスメントグループのプレースメントグループ名です。ID ではなくプレイスメントグループの名前を必ず指定してください。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

> **Note**
>
> - Name または Id を定義せずに PlacementGroup/Enabled を true に設定した場合、ComputeResources/Networking/PlacementGroup がこの設定を上書きするように定義されていない限り、各コンピューティングリソースには独自のマネージドプレイスメントグループが割り当てられます。
> - AWS ParallelCluster バージョン 3.3.0 以降、//NetworkingPlacementGroup/SlurmQueuesは //Networking/ SlurmQueues の推奨代替手段として追加NameされましたplacementGroupId。
>
> PlacementGroup/Id と PlacementGroup/Name は同等です。いずれかを使用できます。

PlacementGroup / Idと PlacementGroup / の両方を含めるとName、 AWS ParallelCluster 失敗します。どちらかを選択することができます。

PlacementGroup/Name を使用するのにクラスターを更新する必要はありません。

Proxy (オプション)

Slurm キューのプロキシ設定を指定します。

```
Proxy:
  HttpProxyAddress: string
```

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

HttpProxyAddress (オプション、**String**)

Slurm キューの HTTP または HTTPS のプロキシサーバーを定義します。通常は `https://x.x.x.x:8080` です。

デフォルト値はありません。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

Image

(オプション) Slurm キューに使用するイメージを指定します。すべてのノードで同じ AMI を使用するには、Image セクションの CustomAmi 設定を使用します。

```
Image:
  CustomAmi: string
```

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

Image プロパティ

CustomAmi (オプション、**String**)

デフォルトの AMI ではなく、Slurm キューに使用する AMI です。pcluster CLI コマンドを使用して、デフォルトの AMI のリストを表示できます。

> **Note**
>
> AMI は、ヘッドノードで使用しているのと同じオペレーティングシステムに基づいている必要があります。

```
pcluster list-official-images
```

カスタム AMI の起動に追加のアクセス許可が必要な場合は、ヘッドノードポリシーにそれらのアクセス許可を追加する必要があります。

例えば、カスタム AMI に暗号化されたスナップショットが関連付けられている場合、ヘッドノードポリシーに次の追加のポリシーが必要です。

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
           "Effect": "Allow",
           "Action": [
               "kms:DescribeKey",
               "kms:ReEncrypt*",
               "kms:CreateGrant",
               "kms:Decrypt"
           ],
           "Resource": [
               "arn:aws:kms:<AWS_REGION>:<AWS_ACCOUNT_ID>:key/<AWS_KMS_KEY_ID>"
           ]
       }
   ]
}
```

カスタム AMI 検証の警告のトラブルシューティングについては、「カスタム AMI の問題のトラブルシューティング」を参照してください。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

ComputeResources

(必須) Slurm キューの ComputeResources 設定を定義します。

> **Note**
>
> - クラスターのサイズは、更新中に変わる場合があります。詳細については、「クラスター容量のサイズと更新」を参照してください。
> - 新しいコンピューティングリソースは、クラスターの作成時に存在する CIDR ブロックに属するサブネットにデプロイされている場合にのみ、クラスターに追加できます。

```
ComputeResources:
  - Name: string
    InstanceType: string
    Instances:
      - InstanceType: string
    MinCount: integer
    MaxCount: integer
    DynamicNodePriority: integer
    StaticNodePriority: integer
    SpotPrice: float
    DisableSimultaneousMultithreading: boolean
    SchedulableMemory: integer
    HealthChecks:
      Gpu:
        Enabled: boolean
    Efa:
      Enabled: boolean
      GdrSupport: boolean
    CapacityReservationTarget:
      CapacityReservationId: string
      CapacityReservationResourceGroupArn: string
    Networking:
      PlacementGroup:
        Enabled: boolean
        Name: string
```

```
    CustomSlurmSettings: dict
    Tags:
      - Key: string
        Value: string
```

更新ポリシー:このリスト値の設定では、更新中に新しい値を追加することができ、既存の値を削除する場合はコンピューティングフリートを停止する必要があります。

ComputeResources のプロパティ

Name (必須)、**String**)

Slurm キューのコンピューティング環境の名前。名前の最大長は 25 文字です。

更新ポリシー: この設定が変更された場合、更新は許可されません。

InstanceType (必須)、**String**)

この Slurm コンピューティングリソースで使用されるインスタンスタイプ。クラスターのすべてのインスタンスタイプは、同じプロセッサアーキテクチャを使用している必要があります。インスタンスは `x86_64` または `arm64` のアーキテクチャのいずれかを使用できます。

クラスター設定で、InstanceType または Instances のいずれかを定義する必要があります。両方が定義されている場合、 AWS ParallelCluster 失敗します。

`InstanceType` を定義する場合、複数のサブネットを定義することはできません。1 つのインスタンスタイプのみを設定し、また複数のサブネットを使用する場合、`InstanceType` ではなく `Instances` のインスタンスタイプを定義します。詳細については、「`Networking`」/「`SubnetIds`」を参照してください。

p4d または hpc6idインスタンスタイプ、または複数のネットワークインターフェイスまたはネットワークインターフェイスカードを持つ別のインスタンスタイプを定義する場合は、「」で説明されているように、プライベートサブネットでコンピューティングインスタンスを起動する必要がありますAWS ParallelCluster 2 つのサブネットの使用。 AWS パブリック IP は、単一のネットワークインターフェイスで起動されたインスタンスにのみ割り当てることができます。 IPs 詳細については、「Amazon EC2 Linux インスタンス用ユーザーガイド」の「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。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

Instances (必須)

コンピューティングリソースのインスタンスタイプのリストを指定します。インスタンスタイプのリストの配分戦略を指定するには、「AllocationStrategy」を参照してください。

クラスター設定で、InstanceType または Instances のいずれかを定義する必要があります。両方が定義されていると、 AWS ParallelCluster は失敗します。

詳細については、「Slurm による複数のインスタンスタイプの割り当て」を参照してください。

```
Instances:
   - InstanceType: string
```

> **Note**
>
> AWS ParallelCluster バージョン 3.7.0 以降では、インスタンスで複数のインスタンスタイプを設定すると、 ???を有効にEnableMemoryBasedSchedulingできます。
> AWS ParallelCluster バージョン 3.2.0 から 3.6.*x* では、インスタンスで複数のインスタンスタイプを設定した場合、 を有効にするEnableMemoryBasedSchedulingことはできません。

更新ポリシー:このリスト値の設定では、更新中に新しい値を追加することができ、既存の値を削除する場合はコンピューティングフリートを停止する必要があります。

InstanceType (必須)、**String**)

この Slurm コンピューティングリソースで使用するインスタンスタイプ。クラスター内のすべてのインスタンスタイプは、x86_64 または arm64 のいずれかの同じプロセッサアーキテクチャを使用する必要があります。

Instances に一覧表示されているインスタンスタイプには、次が必要です。

- 同じ数の vCPUs、または DisableSimultaneousMultithreading を true に設定している場合は同じ数のコア。
- 同じ製造元で同じ数のアクセラレーター。
- Efa/Enabled が true に設定されている場合、EFA がサポートされます。

Instances に一覧表示されているインスタンスタイプには、次があります。

- さまざまな量のメモリ。

  この場合、最小メモリが使用可能な Slurm リソースとして設定されます。

  > **Note**
  >
  > AWS ParallelCluster バージョン 3.7.0 以降では、インスタンスで複数のインスタンスタイプを設定すると、 を有効にEnableMemoryBasedSchedulingできます。AWS ParallelCluster バージョン 3.2.0 から 3.6.*x* では、インスタンスで複数のインスタンスタイプを設定した場合、 を有効にするEnableMemoryBasedSchedulingことはできません。

- さまざまなネットワークカード。

  この場合、コンピューティングリソースに設定されるネットワークインターフェイスの数は、ネットワークカードの数が最小のインスタンスタイプによって定義されます。

- さまざまなネットワーク帯域幅。
- さまざまなインスタンスストアのサイズ。

p4d または hpc6idインスタンスタイプ、または複数のネットワークインターフェイスまたはネットワークインターフェイスカードを持つ別のインスタンスタイプを定義する場合は、「」で説明されているように、プライベートサブネットでコンピューティングインスタンスを起動する必要がありますAWS ParallelCluster 2 つのサブネットの使用。 AWS パブリック IP は、単一のネットワークインターフェイスで起動されたインスタンスにのみ割り当てることができます。 IPs 詳細については、「Amazon EC2 Linux インスタンス用ユーザーガイド」の「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。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

> **Note**
>
> `Instances` は AWS ParallelCluster バージョン 3.3.0 以降でサポートされています。

`MinCount` (オプション、**`Integer`**)

Slurm コンピューティングリソースが使用するインスタンス数の最小値。デフォルトは 0 です。

> **Note**
> クラスターのサイズは、更新中に変わる場合があります。詳細については、「クラスター容量のサイズと更新」を参照してください。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

MaxCount (オプション、**Integer**)

Slurm コンピューティングリソースが使用する最大インスタンス数。デフォルトは 10 です。

を使用する場合`CapacityType = CAPACITY_BLOCK`、キャパシティブロック予約のすべてのインスタンスは静的ノードとして管理されるため、 は 0 `MinCount`以上`MaxCount`である必要があります。

クラスターの作成時に、ヘッドノードはすべての静的ノードの準備が整うのを待ってから、クラスターの作成の成功を通知します。ただし、 を使用する場合`CapacityType = CAPACITY_BLOCK`、キャパシティブロックに関連付けられたコンピューティングリソースのノード部分は、このチェックでは考慮されません。クラスターは、設定したキャパシティブロックの一部がアクティブでない場合でも作成されます。

> **Note**
> クラスターのサイズは、更新中に変わる場合があります。詳細については、「クラスター容量のサイズと更新」を参照してください。

DynamicNodePriority (オプション、**Integer**)

キューコンピューティングリソースの動的ノードの優先度。優先度は、コンピューティングリソースの動的ノードの Slurm ノード `Weight` 設定パラメータにマッピングされます。デフォルト値は `1000` です。

Slurm は、最初に `Weight` 値が最も低いノードを優先します。

> **Warning**
> Slurm パーティション (キュー) でさまざまな `Weight` 値を多く使用すると、キューのジョブスケジューリングのレートが遅くなる可能性があります。

AWS ParallelCluster バージョン 3.7.0 より前のバージョンでは、静的ノードと動的ノードの両方に同じデフォルトの重み が割り当てられていました1。この場合、静的および動的ノードの命名スキーマにより、Slurm はアイドル状態の静的ノードよりもアイドル状態の動的ノードを優先することがあります。他のすべてが等しい場合は、Slurm は名前のアルファベット順にノードをスケジュールします。

> **Note**
>
> `DynamicNodePriority` が AWS ParallelCluster バージョン 3.7.0 に追加されました。

更新ポリシー: この設定は、更新中に変更できます。

`StaticNodePriority` (オプション、**`Integer`**)

キューコンピューティングリソースの静的ノードの優先度。優先度は、コンピューティングリソースの静的ノードの Slurm ノード `Weight` 設定パラメータにマッピングされます。デフォルト値は 1 です。

Slurm は、最初に `Weight` 値が最も低いノードを優先します。

> **Warning**
>
> Slurm パーティション (キュー) でさまざまな `Weight` 値を多く使用すると、キューのジョブスケジューリングのレートが遅くなる可能性があります。

> **Note**
>
> `StaticNodePriority` バージョン 3.7.0 AWS ParallelCluster で が追加されました。

更新ポリシー: この設定は、更新中に変更できます。

`SpotPrice` (オプション、**`Float`**)

任意のインスタンスの起動前に、EC2 スポットインスタンスに支払われる上限価格です。デフォルトの値はオンデマンドの料金です。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

DisableSimultaneousMultithreading (オプション、**Boolean**)

true の場合、Slurm キューのノードでのマルチスレッドは無効になります。デフォルト値は false です。

すべてのインスタンスタイプのマルチスレッドを無効にできるわけではありません。マルチスレッディングの無効化をサポートするインスタンスタイプの一覧については、「Amazon EC2 ユーザーガイド」のインスタンスタイプ別の CPU コア数と CPU コアごとのスレッド数」を参照してください。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

SchedulableMemory (オプション、**Integer**)

コンピューティングリソースのコンピューティングノードの Slurm パラメータ RealMemory で設定されるメモリの量 (MiB)。SlurmSettings/EnableMemoryBasedScheduling が有効になっている場合、この値はジョブに使用できるノードのメモリの上限です。デフォルト値は、Amazon EC2 インスタンスタイプに一覧表示され、Amazon EC2 API DescribeInstanceTypes によって返されるメモリの 95% です。GiB で指定された値が MiB に変換されていることを確認してください。

サポートされる値: 1-EC2Memory

EC2Memory は、Amazon EC2 インスタンスタイプに一覧表示され、Amazon EC2 API DescribeInstanceTypes によって返されるメモリ (MiB 単位) です。GiB で指定された値が MiB に変換されていることを確認してください。

このオプションは、SlurmSettings/EnableMemoryBasedScheduling が有効になっている場合、最も関連性が高くなります。詳細については、「Slurm メモリベースのスケジューリング」を参照してください。

> **Note**
>
> SchedulableMemory は AWS ParallelCluster バージョン 3.2.0 からサポートされています。
> バージョン 3.2.0 以降、デフォルトで、 AWS ParallelCluster はSlurmコンピューティングノードRealMemoryを Amazon EC2 API によって返されるメモリの 95% に設定しま

すDescribeInstanceTypes。この設定は EnableMemoryBasedScheduling の値とは無関係です。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

HealthChecks (オプション)

コンピューティングリソースのヘルスチェックを指定します。

Gpu (オプション)

コンピューティングリソースの GPU ヘルスチェックを指定します。

Enabled (オプション、**Boolean**)

がキュー内のリソースを計算するために GPU ヘルスチェック AWS ParallelCluster を実行するかどうか。デフォルトは false です。

> **Note**
>
> AWS ParallelCluster はHealthChecks、ARM オペレーティングシステムを使用するノードGpuで / alinux2 をサポートしていません。これらのプラットフォームは NVIDIA データセンター GPUマネージャー (DCGM) をサポートしていません。

Gpu ヘルスチェックの動作

- / Gpu Enabledが に設定されている場合true、 はコンピューティングリソースに対してヘルス GPU ヘルスチェック AWS ParallelCluster を実行します。
- Gpu ヘルスチェックはコンピューティングリソースでヘルスチェックを実行して、GPU のパフォーマンスが低下したノードでのジョブ送信を防止します。
- コンピューティングノードが Gpu ヘルスチェックに失敗すると、コンピューティングノードのステータスは DRAIN に変わります。このノードで新しいジョブは開始されません。既存のジョブは完了まで実行されます。実行中のジョブがすべて完了すると、コンピューティングノードが動的ノードの場合は終了し、静的ノードの場合は置き換えられます。
- Gpu ヘルスチェックの所要時間は、選択したインスタンスタイプ、インスタンスの GPU の数、および Gpu ヘルスチェックのターゲットの数 (ジョブ GPU ターゲットの数と同じ) に

よって異なります。8 つの GPU を持つインスタンスの場合、一般的な所要時間は 3 分未満です。

- サポートされていないインスタンスで Gpu ヘルスチェックを実行すると終了し、ジョブはコンピューティングノードで実行されます。例えば、インスタンスが GPU を使用していない場合や、インスタンスが GPU を使用していても NVIDIA GPU ではない場合、ヘルスチェックは終了し、ジョブはコンピューティングノードで実行されます。NVIDIA GPU のみがサポートされています。
- Gpu ヘルスチェックは、dcgmi ツールを使用してノードのヘルスチェックを実行し、次の手順を実行します。

  ノードで Gpu ヘルスチェックが開始される場合。

  1. nvidia-dcgm および nvidia-fabricmanager サービスが実行中かどうかを検出します。
  2. これらのサービスが実行されていない場合、Gpu ヘルスチェックによって開始されます。
  3. 永続化モードが有効になっているかどうかを検出します。
  4. 永続化モードが有効になっていない場合は、Gpu ヘルスチェックによって有効になります。

  ヘルスチェックの終了時に、Gpu ヘルスチェックはこれらのサービスとリソースの初期状態を復元します。
- ジョブが特定のノード GPU セットに割り当てられている場合、Gpu ヘルスチェックはその特定のセットでのみ実行されます。それ以外の場合は、ノードのすべての GPU で Gpu ヘルスチェックが実行されます。
- コンピューティングノードが同時に 2 つ以上の Gpu ヘルスチェックのリクエストを受け取った場合、最初のヘルスチェックのみが実行され、他はスキップされます。これは、ノード GPU をターゲットとするヘルスチェックにも当てはまります。この状況に関する追加情報については、ログファイルを確認できます。
- 特定のコンピューティングノードのヘルスチェックログは /var/log/parallelcluster/slurm_health_check.log ファイルで使用可能です。このファイルは クラスター CloudWatch ロググループの Amazon CloudWatch で使用可能で、次があります。
  - サービスおよび永続化モードの有効化と無効化を含め、Gpu ヘルスチェックによって実行されたアクションの詳細。
  - GPU の識別子、シリアル ID、および UUID。

- ヘルスチェックの出力。

更新ポリシー: この設定は、更新中に変更できます。

> **Note**
>
> `HealthChecks` は AWS ParallelCluster バージョン 3.6.0 以降でサポートされています。

`Efa` (オプション)

Slurm キュー内のノードの EFA (Elastic Fabric Adapter) 設定を指定します。

```
Efa:
  Enabled: boolean
  GdrSupport: boolean
```

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、`QueueUpdateStrategy` が設定されている必要があります。

`Enabled` (オプション、**`Boolean`**)

EFA (Elastic Fabric Adapter) が有効であることを指定します。EFA をサポートする Amazon EC2 インスタンスのリストを確認するには、「Linux インスタンス用 Amazon EC2 ユーザーガイド」の「サポートされるインスタンスタイプ」を参照してください。詳細については、「Elastic Fabric Adapter」を参照してください。クラスター `SlurmQueues`/`Networking`/`PlacementGroup` を使用して、インスタンス間のレイテンシーを最小限に抑えることをお勧めします。

デフォルト値は `false` です。

> **Note**
>
> Elastic Fabric Adapter (EFA) は、異なるアベイラビリティーゾーン間ではサポートされていません。詳細については、「SubnetIds」を参照してください。

> **Warning**
>
> SecurityGroups でカスタムセキュリティグループを定義する場合、EFA 対応のインスタンスが、インバウンドおよびアウトバウンドのトラフィックをすべて許可するセキュリティグループのメンバーであることを確認してください。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、`QueueUpdateStrategy` が設定されている必要があります。

`GdrSupport` (オプション、**`Boolean`**)

(オプション) AWS ParallelCluster バージョン 3.0.2 より、この設定は無効となりました。Elastic Fabric Adapter (EFA)による GPUDirect RDMA (リモートディレクトメモリアクセス) のサポートは、Slurm コンピューティングリソースのインスタンスタイプとオペレーティングシステムでサポートされていれば、常に有効です。

> **Note**
>
> AWS ParallelCluster バージョン 3.0.0 から 3.0.1: GPUDirect RDMA のサポートが Slurmコンピューティングリソースに対して有効になっています。GPUDirect RDMA のサポートは、特定のオペレーティングシステム (`Os` は `alinux2`、`ubuntu1804`、または `ubuntu2004`) における特定のインスタンスタイプ (`p4d.24xlarge`) でサポートされています 。デフォルト値は false です。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、`QueueUpdateStrategy` が設定されている必要があります。

`CapacityReservationTarget`

```
CapacityReservationTarget:
  CapacityReservationId: string
  CapacityReservationResourceGroupArn: string
```

コンピューティングリソースに使用するオンデマンドキャパシティ予約を指定します。

`CapacityReservationId` (オプション、**`String`**)

キューのコンピューティングリソースを対象とする既存のキャパシティ予約の ID。ID は、ODCR または 機械学習用のキャパシティブロックを参照できます。

このパラメータをコンピューティングリソースレベルで指定する場合、InstanceType はオプションであり、予約から自動的に取得されます。

CapacityReservationResourceGroupArn (オプション、**String**)

コンピューティングリソースのキャパシティ予約のサービスリンクグループとして機能するリソースグループの Amazon リソースネーム (ARN) を示します。 AWS ParallelCluster は、グループから最も適切なキャパシティ予約を識別して使用します。リソースグループには、コンピューティングリソースに一覧表示されているインスタンスタイプごとに少なくとも 1 つの ODCR があることが必要です。詳細については、「オンデマンドキャパシティ予約 (ODCR) を使用してインスタンスを起動する」を参照してください。

- `PlacementGroup` が `SlurmQueues`/ `Networking`または `SlurmQueues` / `ComputeResources` / で有効になっている場合`Networking`、 はインスタンスタイプをターゲットとするリソースグループ AWS ParallelCluster を選択し、存在する場合はコンピューティングリソース`PlacementGroup`を選択します。

  `PlacementGroup` は、`ComputeResources` で定義されているインスタンスタイプの 1 つをターゲットにしている必要があります。

- `PlacementGroup` / `Networking`または `SlurmQueues` / `SlurmQueues` / `ComputeResources` で が有効になっていない場合`Networking`、 AWS ParallelCluster はコンピューティングリソースのインスタンスタイプのみを対象とするリソースグループを選択します。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、`QueueUpdateStrategy` が設定されている必要があります。

> **Note**
>
> `CapacityReservationTarget` バージョン 3.3.0 で AWS ParallelCluster が追加されます。

Networking

```
Networking:
  PlacementGroup:
    Enabled: boolean
    Name: string
```

更新ポリシー: マネージドプレースメントグループを削除するには、すべてのコンピューティングノードを停止する必要があります。この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

PlacementGroup (オプション)

コンピューティングリソースの配置グループ設定を指定します。

Enabled (オプション、**Boolean**)

コンピューティングリソースに配置グループを使用するかどうかを示します。

- Name を定義せずに true に設定すると、そのコンピューティングリソースには SlurmQueues/Networking/PlacementGroup 設定に関係なく、独自のマネージドプレイスメントグループが割り当てられます。
- Name を定義して true に設定すると、そのコンピューティングリソースには SlurmQueues/Networking/PlacementGroup 設定に関係なく、名前付きのプレイスメントグループが割り当てられます。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

Name (オプション、**String**)

コンピューティングリソースに使用される既存のクラスタープレイスメントグループのプレースメントグループ名です。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

> **Note**
>
> - PlacementGroup/Enabled の両方と Name が設定されていない場合、それぞれの値はデフォルトで SlurmQueues/Networking/PlacementGroup 設定になります。
> - ComputeResources /Networking/ PlacementGroupが AWS ParallelCluster バージョン 3.3.0 で追加されました。

CustomSlurmSettings (オプション、**Dict**)

(オプション) カスタム Slurm ノード (コンピューティングリソース) の設定を定義します。

Slurm ノード (コンピューティングリソース) に適用されるカスタム Slurm 設定パラメータのキーと値のペアのディクショナリを指定します。

`Param1: Value1` などの各個別のキーと値のペアは、`Param1=Value1` の形式で Slurm ノードの設定行の最後に個別に追加されます。

`CustomSlurmSettings` の拒否リストに記載されていない Slurm 設定パラメータのみ指定できます。拒否リストの Slurm 設定パラメータの詳細については、「`CustomSlurmSettings` で拒否リストに記載されている Slurm 設定パラメータ」を参照してください。

AWS ParallelCluster は、パラメータが拒否リストにあるかどうかのみをチェックします。 AWS ParallelCluster はカスタムSlurm設定パラメータの構文またはセマンティクスを検証しません。カスタムSlurm設定パラメータを検証するのはユーザーの責任です。無効なカスタム Slurm 設定パラメータは、クラスターの作成や更新の失敗につながる Slurm デーモンの障害を引き起こす可能性があります。

でカスタムSlurm設定パラメータを指定する方法の詳細については AWS ParallelCluster、「」を参照してくださいSlurm 設定のカスタマイズ。

Slurm 設定パラメータに関する詳細については、「Slurm ドキュメント」の「slurm.conf」を参照してください。

更新ポリシー: この設定は、更新中に変更できます。

> **Note**
>
> `CustomSlurmSettings` は AWS ParallelCluster バージョン 3.6.0 からサポートされています。

`Tags` (オプション、[文字列])

タグのキーと値のペアのリスト。`ComputeResource` タグは、`Tags` セクション または `SlurmQueues`/`Tags` で指定された重複するタグを上書きします。

`Key` (オプション、**`String`**)

タグキー。

`Value` (オプション、**`String`**)

タグ値。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

ComputeSettings

(必須) Slurm キューの ComputeSettings 設定を定義します。

ComputeSettings のプロパティ

Slurm キュー内のノードの ComputeSettings のプロパティを指定します。

```
ComputeSettings:
  LocalStorage:
    RootVolume:
      Size: integer
      Encrypted: boolean
      VolumeType: string
      Iops: integer
      Throughput: integer
     EphemeralVolume:
      MountDir: string
```

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

LocalStorage (オプション)

Slurm キュー内のノードの LocalStorage のプロパティを指定します。

```
LocalStorage:
  RootVolume:
    Size: integer
    Encrypted: boolean
    VolumeType: string
    Iops: integer
    Throughput: integer
  EphemeralVolume:
    MountDir: string
```

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

RootVolume (オプション)

Slurm キューのノードのルートボリュームの詳細を指定します。

```
RootVolume:
  Size: integer
  Encrypted: boolean
  VolumeType: string
  Iops: integer
  Throughput: integer
```

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

Size (オプション、**Integer**)

Slurm キューのノードのルートボリュームサイズをギビバイト (GiB) で指定します。デフォルトのサイズは AMI から取得されます。異なるサイズを使用するには、AMI が growroot をサポートしている必要があります。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

Encrypted (オプション、**Boolean**)

true の場合、Slurm キューのノードのルートボリュームが暗号化されます。デフォルト値は false です。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

VolumeType (オプション、**String**)

Slurm キュー内のノードの Amazon EBS ボリュームタイプを指定します。サポートされている値は gp2、gp3、io1、io2、sc1、st1 および standard です。デフォルト値は gp3 です。

詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EBS ボリュームの種類」を参照してください。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

Iops (オプション、**Boolean**)

io1、io2 および gp3 タイプボリュームの IOPS の数を定義します。

デフォルト値、対応値、volume_iops 対 volume_size の比率は、VolumeType と Size で異なります。

VolumeType = io1

デフォルト Iops = 100

サポートする値 Iops = 100 — 64000 †

最大 volume_iops/volume_size 比率 = 50 IOPS/GiB。5000 IOPS には、少なくとも 100 GiB の volume_size が必要です。

VolumeType = io2

デフォルト Iops = 100

サポートする値 Iops = 100 — 64000 (io2 Block Express ボリュームの場合は 256000) †

最大 Iops/Size 比率 = 500 IOPS/GiB。5000 IOPS には、少なくとも 10 GiB の Size が必要です。

VolumeType = gp3

デフォルト Iops = 3000

サポートする値 Iops = 3,000–16,000 †

IOPS が 3,000 を超えるボリュームの場合、Size に対する Iops の最大レート = GiB あたり 500 IOPS。

† 最大 IOPS は、32,000 IOPS 以上でプロビジョニングされ、Nitro System で構築されたインスタンスにのみ保証されます。他のインスタンスは、最大 32,000 IOPS までです。ボリュームを変更しない限り、以前の io1 ボリュームはパフォーマンスが完全にはならないことがあります。io2Block Express ボリュームは、R5b インスタンスタイプで最大 256000 までの volume_iops 値をサポートします。詳細については、「Amazon EC2 ユーザーガイド」の「io2 Block Express ボリューム」を参照してください。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

Throughput (オプション、**Integer**)

gp3 ボリュームタイプのスループットを MiB/秒で定義します。この設定は、VolumeType が gp3 の場合のみ有効です。デフォルト値は 125 です。サポートされる値: 125 — 1000 MIB/秒

Throughput と Iops の比率は 0.25 以下にします。最大のスループットである 1000 MiB/秒を実現するためには、Iops の設定を 4000 以上にする必要があります。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

EphemeralVolume (オプション、**Boolean**)

エフェメラルボリュームの設定を指定します。エフェメラルボリュームは、すべてのインスタンスストアボリュームを、ext4 ファイルシステムでフォーマットされた 1 つの論理ボリュームにまとめることで作成されます。デフォルトは /scratch です。インスタンスタイプにインスタンスストアボリュームがない場合、エフェメラルボリュームは作成されません。詳細については、「Amazon EC2 Linux インスタンス用ユーザーガイド」の「インスタンスストアボリューム」を参照してください。

```
EphemeralVolume:
  MountDir: string
```

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

MountDir (オプション、**String**)

Slurm キューの各ノードのエフェメラルボリュームのマウントディレクトリ。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

CustomActions

(Optional) Slurm キュー内のノードで実行するカスタムスクリプトを指定します。

```
CustomActions:
  OnNodeStart:
    Sequence:
```

```
      - Script: string
        Args:
          - string
    Script: string
    Args:
      - string
  OnNodeConfigured:
    Sequence:
      - Script: string
        Args:
          - string
    Script: string
    Args:
      - string
```

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

CustomActions プロパティ

OnNodeStart (オプション、**String**)

ノードデプロイのブートストラップアクションが開始される前に Slurm キューのノードで実行する一連のスクリプトまたは単一のスクリプトを指定します。 AWS ParallelCluster は、単一のスクリプトと同じカスタムアクション用の Sequence の両方を含めることはサポートしていません。詳細については、「カスタムブートストラップアクション」を参照してください。

Sequence (オプション)

実行するスクリプトのリスト。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

Script (必須)、**String**)

使用するファイル。ファイルパスは `https://` または `s3://` で始まる必要があります。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

Args (オプション、**[String]**)

スクリプトに渡す引数のリスト。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

Script (必須)、**String**)

単一のスクリプトに使用するファイル。ファイルパスは `https://` または `s3://` で始まる必要があります。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

Args (オプション、**[String]**)

単一のスクリプトに渡す引数のリスト。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

OnNodeConfigured (オプション、**String**)

ノードのブートストラップアクションが完了した後に Slurm キューのノードで実行する一連のスクリプトまたは単一のスクリプトを指定します。 AWS ParallelCluster は、単一のスクリプトと同じカスタムアクション用の Sequence の両方を含めることはサポートしていません。詳細については、「カスタムブートストラップアクション」を参照してください。

Sequence (オプション)

実行するスクリプトのリスト。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

Script (必須)、**String**)

使用するファイル。ファイルパスは `https://` または `s3://` で始まる必要があります。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

Args (オプション、**[String]**)

スクリプトに渡す引数のリスト。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

Script (必須)、**String**)

単一のスクリプトに使用するファイル。ファイルパスは `https://` または `s3://` で始まる必要があります。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

Args (オプション、**[String]**)

単一のスクリプトに渡す引数のリスト。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

> **Note**
>
> Sequence バージョン 3.6.0 AWS ParallelCluster 以降で が追加されました。を指定するとSequence、カスタムアクションの複数のスクリプトを一覧表示できます。 は、を含めずに 1 つのスクリプトでカスタムアクションの設定をサポートするように AWS ParallelCluster 続行しますSequence。
> AWS ParallelCluster は、1 つのスクリプトと の両方を同じカスタムアクションSequenceに含めることをサポートしていません。

Iam

(オプション) Slurm キューのオプションの IAM 設定を定義します。

```
Iam:
  S3Access:
    - BucketName: string
      EnableWriteAccess: boolean
      KeyName: string
  AdditionalIamPolicies:
```

```
    - Policy: string
  InstanceProfile: string
  InstanceRole: string
```

更新ポリシー: この設定は、更新中に変更できます。

Iam プロパティ

InstanceProfile (オプション、**String**)

Slurm キューのデフォルトのインスタンスロールまたはインスタンスプロファイルをオーバーライドするインスタンスプロファイルを指定します。`InstanceProfile` および `InstanceRole` の両方を指定することはできません。形式は `arn:${Partition}:iam::${Account}:instance-profile/${InstanceProfileName}` です。

これを指定すると、`S3Access` と `AdditionalIamPolicies` の設定を指定することはできません。

AWS ParallelCluster に追加された機能は新しいアクセス許可を必要とすることが多いため、`S3Access` と `AdditionalIamPolicies` の設定のいずれかまたは両方を指定することをお勧めします。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

InstanceRole (オプション、**String**)

Slurm キューのデフォルトのインスタンスロールやインスタンスプロファイルを上書きするインスタンスロールを指定します。`InstanceProfile` および `InstanceRole` の両方を指定することはできません。形式は `arn:${Partition}:iam::${Account}:role/${RoleName}` です。

これを指定すると、`S3Access` と `AdditionalIamPolicies` の設定を指定することはできません。

AWS ParallelCluster に追加された機能は新しいアクセス許可を必要とすることが多いため、`S3Access` と `AdditionalIamPolicies` の設定のいずれかまたは両方を指定することをお勧めします。

更新ポリシー: この設定は、更新中に変更できます。

S3Access (オプション)

Slurm キューのバケットを指定します。これは、Slurm キューのバケットに指定されたアクセスを許可するポリシーを生成するために使用されます。

これを指定すると、`InstanceProfile` と `InstanceRole` の設定を指定することはできません。

AWS ParallelCluster に追加された機能は新しいアクセス許可を必要とすることが多いため、`S3Access` と `AdditionalIamPolicies` の設定のいずれかまたは両方を指定することをお勧めします。

```
S3Access:
  - BucketName: string
    EnableWriteAccess: boolean
    KeyName: string
```

更新ポリシー: この設定は、更新中に変更できます。

BucketName (必須)、**String**)

バケットの名前。

更新ポリシー: この設定は、更新中に変更できます。

KeyName (オプション、**String**)

バケットのキーです。デフォルト値は `*` です。

更新ポリシー: この設定は、更新中に変更できます。

EnableWriteAccess (オプション、**Boolean**)

バケットに対して書き込みアクセスが可能かどうかを示す。

更新ポリシー: この設定は、更新中に変更できます。

AdditionalIamPolicies (オプション)

Amazon EC2 の IAM ポリシーの Amazon リソースネーム (ARN) のリストを指定します。このリストは、 に必要なアクセス許可に加えて、Slurmキューに使用されるルートロールにアタッチされます AWS ParallelCluster。

IAM ポリシー名とその ARN が異なります。名前を使用することはできません。

これを指定すると、`InstanceProfile` と `InstanceRole` の設定を指定することはできません。

AWS ParallelCluster が必要とするパーミッションに `AdditionalIamPolicies` が追加され、`InstanceRole` には必要なパーミッションがすべて含まれている必要があるため、`AdditionalIamPolicies` を使用することをお勧めします。必要な権限は、機能が追加されるにつれ、リリースごとに変更されることがよくあります。

デフォルト値はありません。

```
AdditionalIamPolicies:
  - Policy: string
```

更新ポリシー: この設定は、更新中に変更できます。

Policy (必須)、**[String]**)

IAM ポリシーの一覧。

更新ポリシー: この設定は、更新中に変更できます。

SlurmSettings

(オプション) クラスター全体に適用される Slurm の設定を定義します。

```
SlurmSettings:
  ScaledownIdletime: integer
  QueueUpdateStrategy: string
  EnableMemoryBasedScheduling: boolean
  CustomSlurmSettings: [dict]
  CustomSlurmSettingsIncludeFile: string
  Database:
    Uri: string
    UserName: string
    PasswordSecretArn: string
  ExternalSlurmdbd:
    Host: string
    Port: integer
  Dns:
```

```
    DisableManagedDns: boolean
    HostedZoneId: string
    UseEc2Hostnames: boolean
```

SlurmSettings プロパティ

ScaledownIdletime (オプション、**Integer**)

ジョブがなく、Slurm ノードが終了する時間 (分単位) を定義します。

デフォルト値は 10 です。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

MungeKeySecretArn (オプション、**String**)

Slurm クラスターで使用する base64 エンコードされた munge キーを含むプレーンテキスト Secrets Manager シー AWS クレットの Amazon リソースネーム (ARN)。この munge キーは、Slurm クライアントコマンドとリモートサーバーとして機能する Slurm デーモンとの間の RPC コールを認証するために使用します。MungeKeySecretArn が指定されていない場合、 AWS ParallelCluster はクラスターのランダムな munge キーを生成します。

> **Note**
>
> `MungeKeySecretArn` は AWS ParallelCluster バージョン 3.8.0 以降でサポートされています。

> **Warning**
>
> MungeKeySecretArn を既存のクラスターに新しく追加した場合、ロールバックしたり、後で MungeKeySecretArn を削除したりしたときに、ParallelCluster は以前の munge キーを復元しません。代わりに、新しいランダムな munge キーを生成します。

AWS ParallelCluster ユーザーにその特定のシークレットリソースに対する DescribeSecret のアクセス許可がある場合、MungeKeySecretArn が検証されます。MungeKeySecretArn は、以下の場合に有効です。

- 指定したシークレットが存在する。
- シークレットはプレーンテキストで、base64 でエンコードされた有効な文字列を含んでいる。
- デコードされたバイナリ munge キーのサイズが 256 ~ 8192 ビットである。

pcluster ユーザーの IAM ポリシーに DescribeSecret が含まれていない場合、MungeKeySecretArn は検証されず、警告メッセージが表示されます。詳細については、「AWS ParallelCluster `pcluster` 基本ユーザーポリシー」を参照してください。

MungeKeySecretArn を更新するときは、コンピューティングフリートとすべてのログインノードを停止する必要があります。

シークレット ARN のシークレット値が変更されても ARN が同じままである場合、クラスターは新しい munge キーで自動的に更新されません。シークレット ARN の新しい munge キーを使用するには、コンピューティングフリートとログインノードを停止してから、次のコマンドをヘッドノードから実行する必要があります。

```
sudo /opt/parallelcluster/scripts/slurm/update_munge_key.sh
```

コマンドを実行すると、コンピューティングフリートとログインノードの両方を再開できます。新しくプロビジョニングされたコンピューティングノードとログインノードは自動的に新しい munge キーの使用を開始します。

base64 でエンコードされたカスタム munge キーを生成するには、munge ソフトウェアと共に配布された mungekey ユーティリティを使用し、OS で一般的に利用可能な base64 ユーティリティを使用してエンコードできます。または、bash を使用します (bs パラメータを 32 ~ 1024 に設定してください)。

```
dd if=/dev/random bs=128 count=1 2>/dev/null | base64 -w 0
```

あるいは、次のように Python を使用します。

```
import random
import os
import base64

# key length in bytes
key_length=128

base64.b64encode(os.urandom(key_length)).decode("utf-8")
```

更新ポリシー: NEW UPDATE POLICY WITH COMPUTE FLEET AND LOGIN NODES STOPPED (3.7.0 では誤って追加されていませんでした)

QueueUpdateStrategy (オプション、**String**)

次の更新ポリシーを持つ SlurmQueues セクションパラメータの置換戦略を指定します。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートが停止しているか、QueueUpdateStrategy が設定されている必要があります。

QueueUpdateStrategy 値は、クラスター更新プロセスが開始するときにのみ使用されます。

有効な値: COMPUTE_FLEET_STOP | DRAIN | TERMINATE

デフォルト値: COMPUTE_FLEET_STOP

DRAIN

パラメータ値が変更されたキューのノードは DRAINING に設定されます。このステータスのノードは新しいジョブを受け入れず、実行中のジョブは継続し、完了します。

ノードが idle (DRAINED) になった後、ノードが静的な場合はノードが置き換えられ、動的な場合は終了します。パラメータ値が変更されていない他のキューの他のノードは影響を受けません。

この戦略でパラメータ値が変更されたキューノードをすべて置き換えるのに必要な時間は、実行中のワークロードによって異なります。

COMPUTE_FLEET_STOP

QueueUpdateStrategy パラメータのデフォルト値。この設定では、 SlurmQueuesセクションのパラメータを更新するには、クラスターの更新を実行する前にコンピューティングフリートを停止する必要があります。

```
$ pcluster update-compute-fleet --status STOP_REQUESTED
```

TERMINATE

パラメータ値が変更されたキューでは、実行中のジョブは終了し、ノードの電源はすぐにオフになります。

静的ノードは置き換えられ、動的ノードは終了します。

パラメータ値が変更されていない他のキューの他のノードは影響を受けません。

更新ポリシー: この設定は、更新中には分析されません。

> **Note**
>
> `QueueUpdateStrategy` は AWS ParallelCluster バージョン 3.2.0 からサポートされています。

`EnableMemoryBasedScheduling` (オプション、**`Boolean`**)

`true` の場合、Slurm でメモリベースのスケジューリングが有効になります。詳細については、「`SlurmQueues`」/「`ComputeResources`」/「`SchedulableMemory`」を参照してください。

デフォルト値は `false` です。

> **Warning**
>
> メモリベースのスケジューリングを有効にすると、Slurm スケジューラがジョブとノードの割り当てを処理する方法に影響します。
> 詳細については、「Slurm メモリベースのスケジューリング」を参照してください。

> **Note**
>
> `EnableMemoryBasedScheduling` は AWS ParallelCluster バージョン 3.2.0 からサポートされています。

> **Note**
>
> AWS ParallelCluster バージョン 3.7.0 以降では、インスタンスで複数のインスタンスタイプを設定すると、 を有効に`EnableMemoryBasedScheduling`できます。
> AWS ParallelCluster バージョン 3.2.0 から 3.6.*x* では、インスタンスで複数のインスタンスタイプを設定した場合、 を有効にする`EnableMemoryBasedScheduling`ことはできません。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

CustomSlurmSettings (オプション、**[Dict]**)

クラスター全体に適用されるカスタム Slurm 設定を定義します。

AWS ParallelCluster が生成する `slurm.conf` ファイルの末尾に追加するキーと値のペアの Slurm 設定ディクショナリのリストを指定します。

リスト内の各ディクショナリは、Slurm 設定ファイルに追加される個別の行として表示されます。指定するパラメータはシンプルにすることも複雑にすることもできます

次の例に示すように、シンプルなパラメータは 1 つのキーペアで構成されます。

```
- Param1: 100
- Param2: "SubParam1,SubParam2=SubValue2"
```

Slurm 設定でレンダリングされた例。

```
Param1=100
Param2=SubParam1,SubParam2=SubValue2
```

複雑な Slurm 設定パラメータは、次の例に示すように、スペースで区切られた複数のキーと値のペアで構成されます。

```
- NodeName: test-nodes[1-10]
  CPUs: 4
  RealMemory: 4196
  ... # other node settings
- NodeSet: test-nodeset
  Nodes: test-nodes[1-10]
  ... # other nodeset settings
- PartitionName: test-partition
  Nodes: test-nodeset
  ... # other partition settings
```

Slurm 設定でレンダリングされた例。

```
NodeName=test-nodes[1-10] CPUs=4 RealMemory=4196 ... # other node settings
NodeSet=test-nodeset Nodes=test-nodes[1-10] ... # other nodeset settings
```

```
PartitionName=test-partition Nodes=test-nodeset ... # other partition settings
```

> **Note**
>
> カスタム Slurm ノードの名前には、`-st-` または `-dy-` のパターンを含めないでください。これらのパターンは AWS ParallelClusterが管理するノードのために予約されています。

`CustomSlurmSettings` のカスタム Slurm 設定パラメータを指定する場合、`CustomSlurmSettingsIncludeFile` のカスタム Slurm 設定パラメータは指定できません。

`CustomSlurmSettings` の拒否リストに記載されていない Slurm 設定パラメータのみ指定できます。拒否リストの Slurm 設定パラメータの詳細については、「`CustomSlurmSettings` で拒否リストに記載されている Slurm 設定パラメータ」を参照してください。

AWS ParallelCluster は、パラメータが拒否リストにあるかどうかのみをチェックします。 AWS ParallelCluster はカスタムSlurm設定パラメータの構文またはセマンティクスを検証しません。カスタムSlurm設定パラメータを検証するのはユーザーの責任です。無効なカスタム Slurm 設定パラメータは、クラスターの作成や更新の失敗につながる Slurm デーモンの障害を引き起こす可能性があります。

でカスタムSlurm設定パラメータを指定する方法の詳細については AWS ParallelCluster、「」を参照してくださいSlurm 設定のカスタマイズ。

Slurm 設定パラメータに関する詳細については、「Slurm ドキュメント」の「slurm.conf」を参照してください。

更新ポリシー: この設定は、更新中に変更できます。

> **Note**
>
> `CustomSlurmSettings` は AWS ParallelCluster バージョン 3.6.0 からサポートされています。

`CustomSlurmSettingsIncludeFile` (オプション、**`String`**)

クラスター全体に適用されるカスタム Slurm 設定を定義します。

AWS ParallelCluster が生成する `slurm.conf` ファイルの末尾に追加するカスタム Slurm 設定パラメータで構成されるカスタム Slurm ファイルを指定します。

ファイルへのパスを含める必要があります。パスは `https://` または `s3://` で始まる必要があります。

`CustomSlurmSettingsIncludeFile` のカスタム Slurm 設定パラメータを指定する場合、`CustomSlurmSettings` のカスタム Slurm 設定パラメータは指定できません。

> **Note**
>
> カスタム Slurm ノードの名前には、`-st-` または `-dy-` のパターンを含めないでください。これらのパターンは AWS ParallelClusterが管理するノードのために予約されています。

`CustomSlurmSettingsIncludeFile` の拒否リストに記載されていない Slurm 設定パラメータのみ指定できます。拒否リストの Slurm 設定パラメータの詳細については、「`CustomSlurmSettings` で拒否リストに記載されている Slurm 設定パラメータ」を参照してください。

AWS ParallelCluster は、パラメータが拒否リストにあるかどうかのみをチェックします。 AWS ParallelCluster はカスタムSlurm設定パラメータの構文またはセマンティクスを検証しません。カスタムSlurm設定パラメータを検証するのはユーザーの責任です。無効なカスタム Slurm 設定パラメータは、クラスターの作成や更新の失敗につながる Slurm デーモンの障害を引き起こす可能性があります。

でカスタムSlurm設定パラメータを指定する方法の詳細については AWS ParallelCluster、「」を参照してくださいSlurm 設定のカスタマイズ。

Slurm 設定パラメータに関する詳細については、「Slurm ドキュメント」の「slurm.conf」を参照してください。

更新ポリシー: この設定は、更新中に変更できます。

> **Note**
>
> `CustomSlurmSettings` は AWS ParallelCluster バージョン 3.6.0 からサポートされています。

Database

(オプション) クラスターの Slurm アカウンティングを有効にする設定を定義します。詳細については、「Slurm による アカウンティング AWS ParallelCluster」を参照してください。

```
Database:
   Uri: string
   UserName: string
   PasswordSecretArn: string
```

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

Database のプロパティ

Uri (必須)、**String**)

Slurm アカウンティングのバックエンドとして使用されるデータベースサーバーのアドレス。この URI は `host:port` の形式である必要があり、`mysql://` などのスキームを含めないでください。ホストは IP アドレス、またはヘッドノードにより解決される DNS 名のいずれかです。ポートが指定されていない場合は、 AWS ParallelCluster は MySQL のデフォルトのポート 3306 を使用します。

AWS ParallelCluster はアSlurmカウンティングデータベースをクラスターにブートストラップし、データベースにアクセスする必要があります。

データベースは、次が発生する前にアクセスできる必要があります。

- クラスターが作成される。
- クラスターの更新により Slurm アカウンティングが有効になる。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

UserName (必須)、**String**)

データベースへの接続、アカウンティングログの書き込み、およびクエリの実行に Slurm が使用する ID。ユーザーは、データベースに対する読み込みと書き込みの両方のアクセス許可を持っている必要があります。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

`PasswordSecretArn` (必須)、**`String`**)

プレーンテキストのパスワードを含む AWS Secrets Manager シークレットの Amazon `UserName` リソースネーム (ARN)。このパスワードは、データベースサーバーでの認証のために `UserName` および Slurm アカウンティングと共に使用されます。

> **Note**
>
> - AWS Secrets Manager コンソールを使用してシークレットを作成する場合は、「その他のタイプのシークレット」を選択し、プレーンテキストを選択し、パスワードテキストのみをシークレットに含めます。
> - Slurm は slurmdbd.conf でサポートしていないため、データベースパスワードに「#」文字を使用することはできません。
> - AWS Secrets Manager を使用してシークレットを作成する方法の詳細については、「 シーAWS Secrets Manager クレットの作成」を参照してください。

ユーザーに DescribeSecret のアクセス許可がある場合、`PasswordSecretArn` が検証されます。指定したシークレットが存在する場合、`PasswordSecretArn` は有効です。ユーザーの IAM ポリシーに `DescribeSecret` が含まれていない場合、`PasswordSecretArn` は検証されず、警告メッセージが表示されます。詳細については、「AWS ParallelCluster `pcluster` 基本ユーザーポリシー」を参照してください。

`PasswordSecretArn` を更新するとき、コンピューティングフリートを停止する必要があります。シークレットの値が変更され、シークレット ARN が変更されていない場合、自動的に新しいデータベースパスワードでクラスターが更新されることはありません。クラスターを新しいシークレット値に更新するには、コンピューティングフリートを停止した後、ヘッドノードから次のコマンドを実行します。

```
$ sudo /opt/parallelcluster/scripts/slurm/update_slurm_database_password.sh
```

> **Warning**
>
> アカウンティングデータの損失を回避するため、コンピューティングフリートが停止している場合にのみデータベースパスワードを変更することをお勧めします。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

DatabaseName (オプション、**String**)

Slurm アカウンティングに使用するデータベースサーバー上のデータベースの名前 (Uri パラメータで定義)。

データベース名には小文字、数字、アンダースコアを使用できます。名前は 64 文字以下にする必要があります。

このパラメータは、slurmdbd.conf の `StorageLoc` パラメータにマッピングされます。

`DatabaseName` を指定しない場合、ParallelCluster はクラスター名を使用して `StorageLoc` の値を定義します。

`DatabaseName` の更新は、以下の考慮事項に基づいて許可されます。

- DatabaseName という名前のデータベースがデータベースサーバーにまだ存在しない場合は、slurmdbd によって作成されます。必要に応じて新しいデータベースを再設定する (例: クラスター、アカウント、ユーザー、関連付け、QOS などのアカウンティングエンティティを追加する) のはユーザーの責任です。
- DatabaseName という名前のデータベースがデータベースサーバーに既に存在する場合、slurmdbd はそれを Slurm アカウンティング機能に使用します。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

> **Note**
>
> リリース 3.3.0 以降、`Database` が追加されます。

ExternalSlurmdbd

(オプション) 外部 slurmdbd サーバーで Slurm アカウンティングを有効にするための設定を定義します。詳細については、「 でのSlurmアカウンティング AWS ParallelCluster」を参照してください。

```
ExternalSlurmdbd:
  Host: string
  Port: integer
```

ExternalSlurmdbd のプロパティ

Host (必須、**String**)

Slurm アカウンティング用の外部 slurmdbd サーバーへのアドレス。ホストは IP アドレス、またはヘッドノードにより解決される DNS 名のいずれかです。

更新ポリシー: この設定は、更新中に変更できます。

Port (オプション、**Integer**)

slurmdbd サービスがリッスンするポート。デフォルト値は 6819 です。

更新ポリシー: この設定は、更新中に変更できます。

Dns

(オプション) クラスター全体に適用される Slurm の設定を定義します。

```
Dns:
  DisableManagedDns: boolean
  HostedZoneId: string
  UseEc2Hostnames: boolean
```

Dns のプロパティ

DisableManagedDns (オプション、**Boolean**)

`true` の場合、クラスターの DNS エントリは作成されず、Slurm ノード名は解決できません。

デフォルトでは、 は起動時にノードが登録される Route 53 ホストゾーン AWS ParallelCluster を作成します。デフォルト値は `false` です。`DisableManagedDns` が に設定されている場合`true`、ホストゾーンは によって作成されません AWS ParallelCluster。

この設定を使用してインターネットにアクセスできないサブネットにクラスターをデプロイする方法については、「AWS ParallelCluster インターネットアクセスのない単一のサブネット内」を参照してください。

> **Warning**
> クラスターが正常に動作するためには、名前解決システムが必要です。`DisableManagedDns` を `true` に設定した場合、名前解決システムを指定する必

要があります。Amazon EC2 のデフォルト DNS を使用するには、`UseEc2Hostnames` を `true` に設定します。または、独自の DNS リゾルバーを設定し、インスタンスの起動時に確実にノード名が登録されるようにしてください。例えば、`CustomActions`/`OnNodeStart` を設定することにより、これを行うことができます。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`HostedZoneId` (オプション、**`String`**)

クラスターの DNS 名前解決に使用するカスタム Route 53 ホストゾーン ID を定義します。指定した場合、 は指定されたホストゾーンにクラスターノード AWS ParallelCluster を登録し、マネージドホストゾーンを作成しません。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`UseEc2Hostnames` (オプション、**`Boolean`**)

`true` の場合、クラスターコンピューティングノードはデフォルトの EC2 ホスト名を使用して設定されます。Slurm `NodeHostName` もこの情報で更新されます。デフォルトは `false` です。

この設定を使用してインターネットにアクセスできないサブネットにクラスターをデプロイする方法については、「AWS ParallelCluster インターネットアクセスのない単一のサブネット内」を参照してください。

> **Note**
>
> この注記は、 AWS ParallelCluster バージョン 3.3.0 以降は関係ありません。
> 3.3.0 より前の AWS ParallelCluster サポートされているバージョンの場合：
> `UseEc2Hostnames` が に設定されている場合`true`、Slurm 設定ファイルは AWS ParallelCluster `prolog`および `epilog`スクリプトで設定されます。
>
> - `prolog` が実行されると、各ジョブが割り当てられるときにコンピューティングノードの `/etc/hosts` にノード情報が追加されます。
> - `epilog` が実行されると、`prolog` によって書き込まれた内容がクリーンアップされます。
>
> カスタム `prolog` または `epilog` スクリプトを追加するには、`/opt/slurm/etc/pcluster/prolog.d/` または `/opt/slurm/etc/pcluster/epilog.d/` フォルダにそれぞれ追加します。

更新ポリシー: この設定が変更された場合、更新は許可されません。

SharedStorage セクション

(オプション) クラスターの共有ストレージの設定。

AWS ParallelCluster は、Amazon EBS、FSx for ONTAP、FSx for OpenZFS 共有ストレージボリューム、Amazon EFS および FSx for Lustre 共有ストレージファイルシステム、またはファイルキャッシュの使用をサポートしています。

SharedStorage セクションで、外部ストレージまたはマネージドストレージのいずれかを定義できます。

- 外部ストレージとは、ユーザーが管理する既存のボリュームまたはファイルシステムを指します。AWS ParallelCluster は作成も削除もしません。
- AWS ParallelCluster マネージドストレージとは、 が AWS ParallelCluster 作成し、削除できるボリュームまたはファイルシステムを指します。

共有ストレージクォータおよび共有ストレージの設定方法の詳細については、共有ストレージ「 の使用 AWS ParallelCluster」の「」を参照してください。

> **Note**
>
> AWS Batch がスケジューラとして使用されている場合、FSx for Lustre はクラスターヘッドノードでのみ使用できます。

```
SharedStorage:
  - MountDir: string
    Name: string
    StorageType: Ebs
    EbsSettings:
      VolumeType: string
      Iops: integer
      Size: integer
      Encrypted: boolean
      KmsKeyId: string
      SnapshotId: string
      Throughput: integer
```

```
      VolumeId: string
      DeletionPolicy: string
      Raid:
        Type: string
        NumberOfVolumes: integer
  - MountDir: string
    Name: string
    StorageType: Efs
    EfsSettings:
      Encrypted: boolean
      KmsKeyId: string
      EncryptionInTransit: boolean
      IamAuthorization: boolean
      PerformanceMode: string
      ThroughputMode: string
      ProvisionedThroughput: integer
      FileSystemId: string
      DeletionPolicy: string
      AccessPointId: string
  - MountDir: string
    Name: string
    StorageType: FsxLustre
    FsxLustreSettings:
      StorageCapacity: integer
      DeploymentType: string
      ImportedFileChunkSize: integer
      DataCompressionType: string
      ExportPath: string
      ImportPath: string
      WeeklyMaintenanceStartTime: string
      AutomaticBackupRetentionDays: integer
      CopyTagsToBackups: boolean
      DailyAutomaticBackupStartTime: string
      PerUnitStorageThroughput: integer
      BackupId: string
      KmsKeyId: string
      FileSystemId: string
      AutoImportPolicy: string
      DriveCacheType: string
      StorageType: string
      DeletionPolicy: string
      DataRepositoryAssociations:
      - Name: string
        BatchImportMetaDataOnCreate: boolean
```

```
        DataRepositoryPath: string
        FileSystemPath: string
        ImportedFileChunkSize: integer
        AutoExportPolicy: string
        AutoImportPolicy: string
  - MountDir: string
    Name: string
    StorageType: FsxOntap
    FsxOntapSettings:
      VolumeId: string
  - MountDir: string
    Name: string
    StorageType: FsxOpenZfs
    FsxOpenZfsSettings:
      VolumeId: string
  - MountDir: string
    Name: string
    StorageType: FileCache
    FileCacheSettings:
      FileCacheId: string
```

SharedStorage 更新ポリシー

- マネージド/外部 EBS、マネージド EFS、マネージド FSx Lustre の場合、更新ポリシーは 更新ポリシー: このリスト値の設定では、コンピューティングフリートを停止するか、QueueUpdateStrategy が設定されている必要があります。既存の値を削除する場合は、コンピューティングフリートを停止する必要があります。 です。
- 外部 EFS、FSx Lustre、FSx ONTAP、FSx OpenZfs、File Cache の場合、更新ポリシーは 更新ポリシー: この設定は、更新中に変更できます。 です。

SharedStorage のプロパティ

`MountDir` (必須)、`String`)

共有ストレージがマウントされているパス。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`Name` (必須)、`String`)

共有ストレージの名前。この名前は設定を更新するときに使用します。

> **Warning**
>
> AWS ParallelCluster マネージド共有ストレージを指定し、 の値を変更するとName、既存のマネージド共有ストレージとデータは削除され、新しいマネージド共有ストレージが作成されます。クラスターの更新で Name の値を変更することは、既存のマネージド共有ストレージを新しいマネージド共有ストレージに置き換えることと同等です。既存の共有ストレージからデータを保持する必要がある場合Nameは、変更する前に必ずデータをバックアップしてください。

更新ポリシー: このリスト値の設定では、コンピューティングフリートを停止するか、QueueUpdateStrategy が設定されている必要があります。既存の値を削除する場合は、コンピューティングフリートを停止する必要があります。

StorageType (必須)、String)

共有ストレージのタイプ。サポートされている値は、Ebs、Efs、FsxLustre、FsxOntap、および FsxOpenZfs です。

詳細については、「FsxLustreSettings」、「FsxOntapSettings」、および「FsxOpenZfsSettings」を参照してください。

> **Note**
>
> スケジューラ AWS Batch として を使用する場合、FSx for Lustre はクラスターヘッドノードでのみ使用できます。

更新ポリシー: この設定が変更された場合、更新は許可されません。

EbsSettings

(オプション) Amazon EBS ボリュームの設定。

```
EbsSettings:
  VolumeType: string
  Iops: integer
  Size: integer
  Encrypted: boolean
  KmsKeyId: string
```

```
  SnapshotId: string
  VolumeId: string
  Throughput: integer
  DeletionPolicy: string
  Raid:
    Type: string
    NumberOfVolumes: integer
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

EbsSettings のプロパティ

DeletionPolicy が Delete に設定されている場合、クラスターが削除されたり、クラスターの更新によってボリュームが削除されたりした場合、マネージドボリュームはそのデータと共に削除されます。

詳細については、「 AWS ParallelClusterの使用」の「共有ストレージ」を参照してください。

VolumeType (オプション、String)

Amazon EBS ボリュームタイプを指定します。サポートされている値は gp2、gp3、io1、io2、sc1、st1 および standard です。デフォルト値は gp3 です。

詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EBS ボリュームの種類」を参照してください。

更新ポリシー: この設定が変更された場合、更新は許可されません。

Iops (オプション、Integer)

io1、io2 および gp3 タイプボリュームの IOPS の数を定義します。

デフォルト値、対応値、volume_iops 対 volume_size の比率は、VolumeType と Size で異なります。

VolumeType = io1

デフォルト Iops = 100

サポートする値 Iops = 100 — 64000 †

最大 volume_iops/volume_size 比率 = 50 IOPS/GiB。5000 IOPS には、少なくとも 100 GiB の volume_size が必要です。

VolumeType = io2

デフォルト Iops = 100

サポートする値 Iops = 100 — 64000 (io2 Block Express ボリュームの場合は 256000) †

最大 Iops/Size 比率 = 500 IOPS/GiB。5000 IOPS には、少なくとも 10 GiB の Size が必要です。

VolumeType = gp3

デフォルト Iops = 3000

サポートされる値 Iops = 3000 — 16000

最大 Iops/Size 比率 = 500 IOPS/GiB。5000 IOPS には、少なくとも 10 GiB の Size が必要です。

† 最大 IOPS は、32,000 IOPS 以上でプロビジョニングされた Nitro System で構築されたインスタンスにのみ保証されます。他のインスタンスは、最大 32,000 IOPS を保証します。ボリュームを変更しない限り、以前の io1 ボリュームはパフォーマンスが完全にはならないことがあります。io2Block Express ボリュームは、R5b インスタンスタイプで最大 256000 までの volume_iops 値をサポートします。詳細については、「Amazon EC2 ユーザーガイド」の「io2 Block Express ボリューム」を参照してください。

更新ポリシー: この設定は、更新中に変更できます。

Size (オプション、Integer)

ボリュームのサイズをギビバイト (GiB) で指定します。デフォルト値は 35 です。

更新ポリシー: この設定が変更された場合、更新は許可されません。

Encrypted (オプション、Boolean)

ボリュームを暗号化するかどうかを指定します。デフォルト値は true です。

更新ポリシー: この設定が変更された場合、更新は許可されません。

KmsKeyId (オプション、String)

暗号化に使用するカスタム AWS KMS キーを指定します。この設定は、Encrypted の設定を true にすることが必要です。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`SnapshotId` (オプション、`String`)

ボリュームのソースとしてスナップショットを使用する場合は、Amazon EBS スナップショット ID を指定します。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`VolumeId` (オプション、`String`)

Amazon EBS ボリューム ID を指定します。`EbsSettings` インスタンスに指定した場合、`MountDir` パラメータのみを指定することも可能です。

`HeadNode` として同じアベイラビリティーゾーンにボリュームを作成する必要があります。

> **Note**
>
> AWS ParallelCluster バージョン 3.4.0 では、複数のアベイラビリティーゾーンが追加されています。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`Throughput` (オプション、`Integer`)

ボリュームにプロビジョニングするスループットを MiB/秒で表したもので、最大 1,000 MiB/秒です。

この設定は、`VolumeType` が `gp3` の場合のみ有効です。サポートされている範囲は 125 ~ 1000 で、デフォルト値は 125 です。

更新ポリシー: この設定は、更新中に変更できます。

`DeletionPolicy` (オプション、`String`)

クラスターの削除またはボリュームの削除時にボリュームを保持、削除、またはスナップショットするかどうかを指定します。サポートされる値は `Delete`、`Retain` および `Snapshot` です。デフォルト値は `Delete` です。

DeletionPolicy が `Delete` に設定されている場合、クラスターが削除されたり、クラスターの更新によってボリュームが削除されたりした場合、マネージドボリュームはそのデータと共に削除されます。

詳細については、「共有ストレージ」を参照してください。

更新ポリシー: この設定は、更新中に変更できます。

> **Note**
>
> DeletionPolicy は AWS ParallelCluster バージョン 3.2.0 以降でサポートされています。

Raid

(オプション) RAID ボリュームの設定を定義します。

```
Raid:
  Type: string
  NumberOfVolumes: integer
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

Raid のプロパティ

Type (必須)、String)

RAID 配列のタイプを定義します。サポートされている値は「0」(ストライピング) および「1」(ミラーリング) です。

更新ポリシー: この設定が変更された場合、更新は許可されません。

NumberOfVolumes (オプション、Integer)

RAID　配列の作成に使用する Amazon EBS ボリュームの数を定義します。設定可能な値の範囲は 2~5 です。デフォルト値 (Raid設定が定義されている場合) は 2 です。

更新ポリシー: この設定が変更された場合、更新は許可されません。

EfsSettings

(オプション) Amazon EFS ファイルシステムの設定です。

```
EfsSettings:
  Encrypted: boolean
  KmsKeyId: string
```

```
  EncryptionInTransit: boolean
  IamAuthorization: boolean
  PerformanceMode: string
  ThroughputMode: string
  ProvisionedThroughput: integer
  FileSystemId: string
  DeletionPolicy: string
  AccessPointId: string
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

EfsSettings のプロパティ

DeletionPolicy が Delete に設定されている場合、クラスターが削除されたり、クラスターの更新によってファイルシステムが削除されたりした場合、マネージドボリュームはそのデータと共に削除されます。

詳細については、「 AWS ParallelClusterの使用」の「共有ストレージ」を参照してください。

Encrypted (オプション、Boolean)

Amazon EFS ファイルシステムを暗号化するかどうかを指定します。デフォルト値は false です。

更新ポリシー: この設定が変更された場合、更新は許可されません。

KmsKeyId (オプション、String)

暗号化に使用するカスタム AWS KMS キーを指定します。この設定は、Encrypted の設定を true にすることが必要です。

更新ポリシー: この設定が変更された場合、更新は許可されません。

EncryptionInTransit (オプション、Boolean)

true に設定した場合、Amazon EFS ファイルシステムは Transport Layer Security (TLS) を使用してマウントされます。デフォルトで、これは false に設定されます。

> **Note**
>
> AWS Batch がスケジューラとして使用されている場合、 EncryptionInTransitはサポートされていません。

> **Note**
>
> AWS ParallelCluster バージョン 3.4.0 以降、`EncryptionInTransit` が追加されます。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`IamAuthorization` (オプション、`Boolean`)

`IamAuthorization` バージョン 3.4.0 AWS ParallelCluster 以降で が追加されました。

`true` に設定すると、Amazon EFS はシステムの IAM ID を使用して認証されます。デフォルトで、これは `false` に設定されます。

> **Note**
>
> `IamAuthorization` が `true` に設定されている場合は `EncryptionInTransit` も `true` に設定する必要があります。

> **Note**
>
> AWS Batch がスケジューラとして使用されている場合、 `IamAuthorization`はサポートされていません。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`PerformanceMode` (オプション、`String`)

Amazon EFS ファイルシステムのパフォーマンスモードを指定します。サポートされている値は`generalPurpose` および `maxIO` です。デフォルト値は `generalPurpose` です。詳細については、「Amazon Elastic File System ユーザーガイド」の「パフォーマンスモード」を参照してください。

ほとんどのファイルシステムに `generalPurpose` パフォーマンスモードをお勧めします。

`maxIO` パフォーマンスモードを使用するファイルシステムでは、集計スループットと 1 秒あたりのオペレーション数をより高いレベルにスケーリングできます。ただし、ほとんどのファイルオペレーションでは、レイテンシーがわずかに長くなるというトレードオフがあります。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`ThroughputMode` (オプション、`String`)

Amazon EFS ファイルシステムのスループットモードを指定します。サポートされている値は`bursting` および `provisioned` です。デフォルト値は `bursting` です。`provisioned` を使用する場合は、`ProvisionedThroughput` を指定する必要があります。

更新ポリシー: この設定は、更新中に変更できます。

`ProvisionedThroughput` (`ThroughputMode` が `provisioned` または `Integer` の場合、必須)

Amazon EFS ファイルシステムのプロビジョニングされたスループット (MiB/秒) を定義します (測定単位は MiB/秒)。これは、「Amazon EFS API Reference」(Amazon EFS API リファレンス) の ProvisionedThroughputInMibps パラメータに対応しています。

このパラメータを使用する場合は、`ThroughputMode` を `provisioned` に設定する必要があります。

対応する範囲は `1 ~ 1024` です。制限の引き上げをリクエストするには、 サポートにお問い合わせください。

更新ポリシー: この設定は、更新中に変更できます。

`FileSystemId` (オプション、`String`)

既存のファイルシステムの Amazon EFS ファイルシステム ID を定義します。

クラスターが複数のアベイラビリティーゾーンにまたがるように設定されている場合、クラスターが使用するアベイラビリティーゾーンごとにファイルシステムのマウントターゲットを定義する必要があります。

これが指定されている場合、`MountDir` のみ指定できます。それ以外の `EfsSettings` は指定できません。

このオプションを設定する場合、定義するファイルシステムについて次の条件に一致している必要があります。

- ファイルシステムには、クラスターのアベイラビリティーゾーンに既存のマウントターゲットがあり、`HeadNode` と `ComputeNodes` からのインバウンドおよびアウトバウンドの NFS トラフィックが許可されています。Scheduling/SlurmQueues/Networking/SubnetIds で複数のアベイラビリティーゾーンが設定されます。

クラスターとファイルシステムの間でトラフィックが許可されていることを確認するために、次のいずれかを実行できます。

- クラスターサブネットの CIDR またはプレフィックスリストとの間のトラフィックを許可するように、マウントターゲットのセキュリティグループを設定します。

  > **Note**
  >
  > AWS ParallelCluster は、ポートが開いていること、および CIDR またはプレフィックスリストが設定されていることを検証します。CIDR AWS ParallelCluster ブロックまたはプレフィックスリストの内容を検証しません。

- `SlurmQueues`/`Networking`/`SecurityGroups` と `HeadNode`/`Networking`/`SecurityGroups` を使用して、クラスターノードのカスタムセキュリティグループを設定します。カスタムセキュリティグループを設定して、クラスターとファイルシステムの間のトラフィックを許可する必要があります。

  > **Note**
  >
  > すべてのクラスターノードがカスタムセキュリティグループを使用している場合、はポートが開いていること AWS ParallelCluster のみを検証します。送信元と送信先が正しく設定されていることは検証 AWS ParallelCluster しません。

> **Warning**
>
> EFS OneZone は、すべてのコンピューティングノードとヘッドノードが同じアベイラビリティーゾーンにある場合にのみサポートされます。EFS OneZone にはマウントターゲットを 1 つだけ設定できます。

> **Note**
>
> AWS ParallelCluster バージョン 3.4.0 では、複数のアベイラビリティーゾーンが追加されています。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`DeletionPolicy` (オプション、`String`)

ファイルシステムがクラスターから削除されるか、クラスターが削除されたときに、ファイルシステムを保持または削除するかを指定します。サポートされる値は `Delete` と `Retain` です。デフォルト値は `Delete` です。

DeletionPolicy が `Delete` に設定されている場合、クラスターが削除されたり、クラスターの更新によってファイルシステムが削除されたりした場合、マネージドボリュームはそのデータと共に削除されます。

詳細については、「共有ストレージ」を参照してください。

更新ポリシー: この設定は、更新中に変更できます。

> **Note**
>
> `DeletionPolicy` は AWS ParallelCluster バージョン 3.3.0 以降でサポートされています。

`AccessPointId` (オプション、`String`)

このオプションを指定すると、ファイルシステムのルートではなく、`access point ID` で定義したファイルシステムのエントリポイントがマウントされます。

詳細については、「共有ストレージ」を参照してください。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`FsxLustreSettings`

> **Note**
>
> `FsxLustre` が StorageType に指定されている場合、`FsxLustreSettings` を定義する必要があります。

(オプション) FSx for Lustre ファイルシステムの設定。

```
FsxLustreSettings:
  StorageCapacity: integer
```

```
  DeploymentType: string
  ImportedFileChunkSize: integer
  DataCompressionType: string
  ExportPath: string
  ImportPath: string
  WeeklyMaintenanceStartTime: string
  AutomaticBackupRetentionDays: integer
  CopyTagsToBackups: boolean
  DailyAutomaticBackupStartTime: string
  PerUnitStorageThroughput: integer
  BackupId: string # BackupId cannot coexist with some of the fields
  KmsKeyId: string
  FileSystemId: string # FileSystemId cannot coexist with other fields
  AutoImportPolicy: string
  DriveCacheType: string
  StorageType: string
  DeletionPolicy: string
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

> **Note**
>
> AWS Batch がスケジューラとして使用されている場合、FSx for Lustre はクラスターヘッドノードでのみ使用できます。

FsxLustreSettings のプロパティ

DeletionPolicy が Delete に設定されている場合、クラスターが削除されたり、クラスターの更新によってファイルシステムが削除されたりした場合、マネージドボリュームはそのデータと共に削除されます。

詳細については、「共有ストレージ」を参照してください。

StorageCapacity (必須)、Integer)

FSx for Lustre のストレージ容量を GiB 単位で設定します。StorageCapacity は、新しいファイルシステムを作成する場合に必要です。BackupId または FileSystemId が指定されている場合、StorageCapacity は含めてはいけません。

- SCRATCH_2、PERSISTENT_1、および PERSISTENT_2 デプロイタイプの場合、有効な値は 1,200 GiB、2400 GiB、それ以降は 2,400 GiB の増分です。

- SCRATCH_1 デプロイタイプの場合、有効な値は 1,200 GiB、2,400 GiB、それ以降 3,600 GiB の増分です。

更新ポリシー: この設定が変更された場合、更新は許可されません。

DeploymentType (オプション、String)

FSx for Lustre ファイルシステムのデプロイタイプを指定します。サポートされている値は SCRATCH_1、SCRATCH_2、PERSISTENT_1、PERSISTENT_2 です。デフォルト値は SCRATCH_2 です。

一時的なストレージと短期間のデータ処理が必要な場合、SCRATCH_1 および SCRATCH_2 デプロイタイプを選択します。SCRATCH_2 デプロイタイプでは、データの転送中の暗号化と SCRATCH_1 よりも高いバーストスループットキャパシティが提供されます。

長期ストレージと、レイテンシーの影響を受けないスループット重視のワークロードには、PERSISTENT_1デプロイタイプを選択します。 は、転送中のデータの暗号化PERSISTENT_1をサポートしています。FSx for Lustre が利用可能なすべての AWS リージョン で使用できます。

長期ストレージ、および最高レベルの IOPS とスループットを必要とする、レイテンシーの影響を受けやすいワークロードの場合は、PERSISTENT_2 デプロイタイプを選択します。PERSISTENT_2 は SSD ストレージをサポートし、より高い PerUnitStorageThroughput (最大1,000 MB/s/TiB) を提供します。PERSISTENT_2 は限られた数の AWS リージョンで使用可能です。デプロイタイプとPERSISTENT_2利用可能な AWS リージョン のリストの詳細については、「Amazon FSx for Lustre ユーザーガイド」の「FSx for Lustre のファイルシステムデプロイオプション」を参照してください。 FSx

この機能をサポートする Amazon EC2 インスタンスから SCRATCH_2、PERSISTENT_1、または PERSISTENT_2 デプロイタイプのファイルシステムにアクセスするとき、転送中のデータの暗号化が自動的に有効になります。

SCRATCH_2、PERSISTENT_1、および PERSISTENT_2 デプロイタイプの転送中のデータの暗号化は、サポートされる AWS リージョンでサポートされるインスタンスタイプからアクセスした場合にサポートされます。詳細については、「Amazon FSx for Lustre ユーザーガイド」の「転送中のデータの暗号化」を参照してください。

> **Note**
>
> AWS ParallelCluster バージョン 3.2.0 で、PERSISTENT_2 デプロイタイプのサポートが追加されました。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`ImportedFileChunkSize` (オプション、`Integer`)

データリポジトリからインポートされたファイルの場合、この値がストライプカウントと 1 つの物理ディスクに保存された各ファイルごとの最大データ量 (MiB 単位) を決定します。1 つのファイルにストライピングできるディスクの最大数は、ファイルシステムを構成するディスクの合計数によって制限されます。

デフォルトのチャンクサイズは 1,024 MiB (1 GiB) であり、512,000 MiB (500 GiB) まで高くすることができます。Amazon S3 オブジェクトの最大サイズは 5 TB です。

> **Note**
>
> このパラメータは、 PERSISTENT_2デプロイタイプを使用するファイルシステムではサポートされていません。データリポジトリの関連付けを設定する方法については、「Amazon FSx for Lustre ユーザーガイド」の「S3 バケットにファイルシステムをリンクする」を参照してください。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`DataCompressionType` (オプション、`String`)

FSx for Lustre ファイルシステムのデータ圧縮構成を設定します。サポートされている値は `LZ4` です。`LZ4` は LZ4 アルゴリズムによるデータ圧縮がオンになっていることを示しています。`DataCompressionType` が指定されていない場合は、ファイルシステムの作成時にデータ圧縮がオフになります。

詳細については、「Lustre データ圧縮」を参照してください。

更新ポリシー: この設定は、更新中に変更できます。

`ExportPath` (オプション、`String`)

FSx for Lustre ファイルシステムのルートがエクスポートされる Amazon S3 のパス。この設定は、`ImportPath` パラメータ が設定されている場合にのみサポートされます。パスでは、`ImportPath` で指定されている同じ Amazon S3 バケットを使用する必要があります。FSx for Lustre ファイルシステムからエクスポートされる新しいデータおよび変更されたデータのオプションのプレフィックスを指定できます。`ExportPath` 値を指定しない場合、FSx for Lustre でデフォルトのエクスポートパス (`s3://amzn-s3-demo-bucket/FSxLustre[creation-timestamp]`) が設定されます。タイムスタンプは UTC 形式です (例えば、`s3://amzn-s3-demo-bucket/FSxLustre20181105T222312Z`)。

Amazon S3 エクスポートバケットは、`ImportPath` で指定されたインポートバケットと同じである必要があります。`s3://amzn-s3-demo-bucket` など、バケット名のみを指定した場合、Amazon S3 バケットオブジェクト へのファイルシステムオブジェクトの 1 対 1 のマッピングを取得します。このマッピングは、Amazon S3 の入力データがエクスポートで上書きされることを意味します。`s3://amzn-s3-demo-bucket/[custom-optional-prefix]` など、エクスポートパスでカスタムプレフィックスを指定した場合、FSx for Lustre は、ファイルシステムの内容を Amazon S3 バケット内のそのエクスポートプレフィックスにエクスポートします。

> **Note**
>
> このパラメータは、 `PERSISTENT_2`デプロイタイプを使用するファイルシステムではサポートされていません。「Amazon FSx for Lustre ユーザーガイド」の「S3 バケットにファイルシステムをリンクする」で説明されているように、データリポジトリの関連付けを設定します。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`ImportPath` (オプション、`String`)

FSx for Lustre ファイルシステムのデータリポジトリとして使用している Amazon S3 バケットへのパス (オプションのプレフィックスを含む)。FSx for Lustre ファイルシステムのルートは、選択した Amazon S3 バケットのルートにマップされます。例は `s3://amzn-s3-demo-bucket/optional-prefix` です。Amazon S3 バケット名の後にプレフィックスを指定した場合、そのプレフィックスが付いたオブジェクトキーのみがファイルシステムにロードされます。

> **Note**
>
> このパラメータは、 PERSISTENT_2デプロイタイプを使用するファイルシステムではサポートされていません。「Amazon FSx for Lustre ユーザーガイド」の「S3 バケットにファイルシステムをリンクする」で説明されているように、データリポジトリの関連付けを設定します。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`WeeklyMaintenanceStartTime` (オプション、`String`)

週次メンテナンスを実行するための優先開始時間。UTC+0 タイムゾーンの `"d:HH:MM"` の形式でフォーマットします。このフォーマットの場合、d は月曜日で始まり日曜日で終わる 1～7 の曜日の番号です。このフィールドには引用符が必要です。

更新ポリシー: この設定は、更新中に変更できます。

`AutomaticBackupRetentionDays` (オプション、`Integer`)

自動バックアップの保持日数です。この値を 0 に設定すると、自動バックアップが無効になります。サポートされる範囲は 0～90 です。デフォルトは 0 です。この設定は、`PERSISTENT_1` と `PERSISTENT_2` デプロイタイプでの使用でのみ有効です。詳細については、「Amazon FSx for Lustre ユーザーガイド」の「バックアップの使用」を参照してください。

更新ポリシー: この設定は、更新中に変更できます。

`CopyTagsToBackups` (オプション、`Boolean`)

`true` の場合、FSx for Lustre ファイルシステムのタグをバックアップにコピーする必要があります。この値のデフォルト値は `false` です。この値を `true` に設定すると、ファイルシステムのすべてのタグが、ユーザーがタグを指定しないすべての自動バックアップおよびユーザー始動型バックアップにコピーされます。この値が `true` で 1 つ以上のタグを指定すると、指定したタグのみがバックアップにコピーされます。ユーザー主導のバックアップの作成時に 1 つ以上のタグを指定した場合、この値に関係なく、ファイルシステムからタグはコピーされません。この設定は、`PERSISTENT_1` と `PERSISTENT_2` デプロイタイプでの使用でのみ有効です。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`DailyAutomaticBackupStartTime` (オプション、`String`)

毎日繰り返される時刻 (形式 `HH:MM`)。`HH` はその日のゼロ詰めの時間 (00 ~ 23) です。`MM` はその時間のゼロ詰めの分 (00 ~ 59) です。例えば、`05:00` は毎日午前 5 時を指定します。この設定は、`PERSISTENT_1` と `PERSISTENT_2` デプロイタイプでの使用でのみ有効です。

更新ポリシー: この設定は、更新中に変更できます。

`PerUnitStorageThroughput` (**`PERSISTENT_1`** と **`PERSISTENT_2`** のデプロイタイプに必要、`Integer`)

1 テビバイト (TiB) のストレージごとの読み取りおよび書き込みスループットの量を MB/s/TiB 単位で表示します。ファイルシステムのスループット容量は、ファイルシステムのストレージ容量 (TiB) に `PerUnitStorageThroughput` (MB/s/TiB) を掛けて計算されます。2.4 TiB のファイルシステムの場合、50 MB/s/TiB の `PerUnitStorageThroughput` をプロビジョニングすると、ファイルシステムのスループットは 120 MB/s になります。プロビジョニングしたスループットに対して支払いが発生します。これは、PerUnitStorageThroughput プロパティに対応しています。

有効な値:

PERSISTENT_1 SSD ストレージ: 50、100、200 MB/s/TiB。

PERSISTENT_1 HDD ストレージ: 12、40 MB/s/TiB。

PERSISTENT_2 SSD ストレージ: 125、250、500、1,000 MB/s/TiB。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`BackupId` (オプション、`String`)

既存のバックアップから FSx for Lustre ファイルシステムを復元するために使用するバックアップの ID を指定します。`BackupId` を指定した場合は、`AutoImportPolicy`、`DeploymentType`、`ExportPath`、`KmsKeyId`、`ImportPath`、`ImportedFileChunkSize` および `PerUnitStorageThroughput` を指定してはいけません。これらの設定はバックアップから読み取られます。また、`AutoImportPolicy`、`ExportPath`、`ImportPath` および `ImportedFileChunkSize` の設定は指定できません。これは、BackupId プロパティに対応しています。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`KmsKeyId` (オプション、`String`)

保管中の永続的な FSx for Lustre ファイルシステムの FSx for Lustre ファイルシステムのデータを暗号化するために使用される AWS Key Management Service（AWS KMS) キー ID の

ID。指定しない場合は、FSx for Lustre のマネージドキーが使用されます。SCRATCH_1 および SCRATCH_2 の FSx for Lustre ファイルシステムは、FSx for Lustre が管理する鍵を用いて、静止状態で常に暗号化されています。詳細については、「AWS Key Management Service API リファレンス」の「暗号化」を参照してください。

更新ポリシー: この設定が変更された場合、更新は許可されません。

FileSystemId (オプション、String)

既存の FSx for Lustre ファイルシステムの ID を指定します。

このオプションを指定すると、FsxLustreSettings の MountDir と FileSystemId の設定のみが使用されます。FsxLustreSettings のその他のすべての設定は無視されます。

> **Note**
>
> ス AWS Batch ケジューラを使用する場合、FSx for Lustre はヘッドノードでのみ使用できます。

> **Note**
>
> ファイルシステムは、ポート 988、1021、1022、および 1023 経由のインバウンドおよびアウトバウンドの TCP トラフィックを許可するセキュリティグループに関連付けられている必要があります。

次のいずれかを実行して、クラスターとファイルシステムの間でトラフィックが許可されていることを確認してください。

- クラスターサブネットの CIDR またはプレフィックスリストとの間のトラフィックを許可するように、ファイルシステムのセキュリティグループを設定します。

  > **Note**
  >
  > AWS ParallelCluster は、ポートが開いていること、および CIDR またはプレフィックスリストが設定されていることを検証します。CIDR AWS ParallelCluster ブロックまたはプレフィックスリストの内容を検証しません。

- SlurmQueues/Networking/SecurityGroups と HeadNode/Networking/SecurityGroups を使用して、クラスターノードのカスタムセキュ

リティグループを設定します。カスタムセキュリティグループを設定して、クラスターとファイルシステムの間のトラフィックを許可する必要があります。

> **Note**
>
> すべてのクラスターノードがカスタムセキュリティグループを使用している場合、 はポートが開いていること AWS ParallelCluster のみを検証します。送信元と送信先が正しく設定されていることは検証 AWS ParallelCluster しません。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`AutoImportPolicy` (オプション、`String`)

FSx for Lustre ファイルシステムを作成すると、既存の Amazon S3 オブジェクトがファイルとディレクトリのリストとして表示されます。このプロパティを使用して、リンクされた Amazon S3 バケットのオブジェクトを追加または変更したときに、FSx for Lustre がファイルとディレクトリのリストを最新の状態に保つ方法を選択します。`AutoImportPolicy` は、次の値を持つことができます。

- `NEW` - 自動インポートがオンになっています。FSx for Lustre は、リンクされた Amazon S3 バケットに追加された新しいオブジェクトのうち、現在 FSx for Lustre ファイルシステム内に存在しないオブジェクトのディレクトリリストを自動的にインポートします。
- `NEW_CHANGED` - 自動インポートがオンになっています。このオプションの選択後は、Amazon S3 バケットに追加された新しいオブジェクトや Amazon S3 バケットで変更された既存のオブジェクトのファイルとディレクトリのリストが、FSx for Lustre によって自動的にインポートされます。
- `NEW_CHANGED_DELETED` - 自動インポートがオンになっています。このオプションの選択後は、Amazon S3 バケットに追加された新しいオブジェクト、Amazon S3 バケットで変更された既存のオブジェクト、Amazon S3 バケットで削除されたオブジェクトのファイルとディレクトリのリストが、FSx for Lustre によって自動的にインポートされます。

> **Note**
>
> AWS ParallelCluster バージョン 3.1.1 で `NEW_CHANGED_DELETED` のサポートが追加されました。

`AutoImportPolicy` が指定されていない場合、自動インポートはオフになります。FSx for Lustre は、ファイルシステムの作成時に、リンクされた Amazon S3 バケットのファイルとディ

レクトリのリストのみを更新します。このオプションの選択後は、新しいオブジェクトや変更されたオブジェクトのファイルとディレクトリのリストは FSx for Lustre で更新されません。

詳細については、「Amazon FSx for Lustre ユーザーガイド」の「S3 バケットから更新を自動的にインポートする」を参照してください。

> **Note**
>
> このパラメータは、PERSISTENT_2 デプロイタイプを使用するファイルシステムではサポートされていません。データリポジトリの関連付けを設定する方法については、「Amazon FSx for Lustre ユーザーガイド」の「S3 バケットにファイルシステムをリンクする」を参照してください。

更新ポリシー: この設定が変更された場合、更新は許可されません。

DriveCacheType (オプション、String)

ファイルシステムに SSD ドライブキャッシュを搭載することを指定します。StorageType 設定が HDD、DeploymentType 設定が PERSISTENT_1 に設定されている場合のみ設定可能です。これは、DriveCacheType プロパティに対応しています。詳細については、「Amazon FSx for Lustre ユーザーガイド」の「FSx for Lustre デプロイオプション」を参照してください。

唯一の有効な値は READ です。SSD ドライブのキャッシュを無効にするには、DriveCacheType の設定を指定しないでください。

更新ポリシー: この設定が変更された場合、更新は許可されません。

StorageType (オプション、String)

作成する FSx for Lustre ファイルシステムのストレージタイプを設定します。有効な値は、SSD および HDD です。

- ソリッドステートドライブストレージを使用する場合は、SSD に設定します。
- ハードディスクドライブのストレージを使用する場合は HDD に設定します。HDD は PERSISTENT のデプロイタイプでサポートされています。

デフォルト値は SSD です。詳細については、「Amazon FSx for Windows ユーザーガイド」の「ストレージタイプのオプション」および「Amazon FSx for Lustre ユーザーガイド」の「複数のストレージオプション」を参照してください。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`DeletionPolicy` (オプション、`String`)

ファイルシステムがクラスターから削除されるか、クラスターが削除されたときに、ファイルシステムを保持または削除するかを指定します。サポートされる値は `Delete` と `Retain` です。デフォルト値は `Delete` です。

DeletionPolicy が `Delete` に設定されている場合、クラスターが削除されたり、クラスターの更新によってファイルシステムが削除されたりした場合、マネージドボリュームはそのデータと共に削除されます。

詳細については、「共有ストレージ」を参照してください。

更新ポリシー: この設定は、更新中に変更できます。

> **Note**
>
> `DeletionPolicy` は AWS ParallelCluster バージョン 3.3.0 以降でサポートされています。

`DataRepositoryAssociations` (オプション、`String`)

DRA のリスト (ファイルシステムあたり最大 8)

それぞれのデータリポジトリの関連付けには、一意の Amazon FSx ファイルシステムディレクトリと、一意の S3 バケットまたはプレフィックスが関連付けられている必要があります。

DRA を使用すると同時に ExportPath と ImportPath を FsxLustreSettings で使用することはできません。

更新ポリシー: この設定は、更新中に変更できます。

`Name` (必須)、`String`)

DRA の名前。この名前は設定を更新するときに使用します。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`BatchImportMetaDataOnCreate` (オプション、`Boolean`)

データリポジトリの関連付けが作成された後に、メタデータをインポートするインポートデータリポジトリタスクを実行する必要があるかどうかを示すブールフラグ。このフラグが `true` に設定されている場合、タスクは実行されます。

デフォルト値: false

更新ポリシー: この設定が変更された場合、更新は許可されません。

DataRepositoryPath (必須)、String)

ファイルシステムにリンクされる Amazon S3 データリポジトリへのパス。パスには、次の S3 バケットまたは s3://*amzn-s3-demo-bucket*/myPrefix/ 形式のプレフィックスを使用できます。このパスは、S3 データリポジトリのファイルのインポート先またはエクスポート先を指定します。

他の DRA との重複は不可

パターン: ^[^\u0000\u0085\u2028\u2029\r\n]{3,4357}$

最小: 3

最大: 4357

更新ポリシー: この設定が変更された場合、更新は許可されません。

FileSystemPath (必須)、String)

Amazon FSx for Lustre ファイルシステム上のパスで、DataRepositoryPath により 1 対 1 でマッピングされる上位ディレクトリ (/ns1/ など) またはサブディレクトリ (/ns1/subdir/ など) を指します。名前の先頭のスラッシュは必須です。2 つのデータリポジトリの関連付けは、重複するファイルシステムパスを持つことはできません。例えば、データリポジトリがファイルシステムパス /ns1/ に関連付けられている場合、ファイルシステムパス /ns1/ns2 に別のデータリポジトリをリンクすることはできません。

このパスは、ファイルシステム上でファイルをエクスポートまたはインポートする場所を指定します。このファイルシステムディレクトリは 1 つの Amazon S3 バケットにのみリンクでき、他の S3 バケットをディレクトリにリンクすることはできません。

他の DRA との重複は不可

> **Note**
>
> ファイルシステムパスとしてスラッシュ (/) のみを指定した場合、ファイルシステムにリンクできるデータリポジトリは 1 つだけです。「/」は、ファイルシステムに関連付けられた最初のデータリポジトリのファイルシステムパスとしてのみ指定できます。

パターン: ^[^\u0000\u0085\u2028\u2029\r\n]{1,4096}$

最小: 1

最大: 4096

更新ポリシー: この設定が変更された場合、更新は許可されません。

ImportedFileChunkSize (オプション、Integer)

データリポジトリからインポートされたファイルの場合、この値がストライプカウントと 1 つの物理ディスクに保存されたファイルごとの最大データ量 (MiB 単位) を決定します。1 つのファイルにストライピングできるディスクの最大数は、ファイルシステムまたはキャッシュを構成するディスクの合計数によって制限されます。

デフォルトのチャンクサイズは 1,024 MiB (1 GiB) であり、512,000 MiB (500 GiB) まで高くすることができます。Amazon S3 オブジェクトの最大サイズは 5 TB です。

最小: 1

最大: 4096

更新ポリシー: この設定は、更新中に変更できます。

AutoExportPolicy (オプション、Array of strings)

リストには、以下の値の 1 つ以上を含めることができます。

- NEW – 新しいファイルとディレクトリは、それらがファイルシステムに追加された時点で、自動的にデータリポジトリにエクスポートされます。
- CHANGED – ファイルシステム上のファイルおよびディレクトリが変更された場合は、その内容が自動的にデータリポジトリにエクスポートされます。
- DELETED – ファイルシステム上で削除されたファイルとディレクトリは、データリポジトリからも自動的に削除されます。

AutoExportPolicy の定義では、イベントタイプを任意に組み合わせて使用できます。

最大: 3

更新ポリシー: この設定は、更新中に変更できます。

AutoImportPolicy (オプション、Array of strings)

リストには、以下の値の 1 つ以上を含めることができます。

- NEW – Amazon FSx は、FSx ファイルシステムにリンクされた S3 バケットに追加済みで、現在ファイルシステム内に存在していない、ファイルのメタデータを自動的にインポートします。
- CHANGED – データリポジトリ内でファイルが変更された際、Amazon FSx は、そのファイルのメタデータを自動的に更新し、ファイルシステム上に既に存在するファイルの内容を無効にします。
- DELETED –ファイルシステム上のファイルに対応するファイルがデータリポジトリで削除された場合、 Amazon FSx はファイルシステム上のファイルも自動的に削除します。

AutoImportPolicy の定義では、イベントタイプを任意に組み合わせて使用できます。

最大: 3

更新ポリシー: この設定は、更新中に変更できます。

FsxOntapSettings

> **Note**
>
> FsxOntap が StorageType に指定されている場合、FsxOntapSettings を定義する必要があります。

(オプション) FSx for ONTAP ファイルシステムの設定。

```
FsxOntapSettings:
  VolumeId: string
```

FsxOntapSettings のプロパティ

VolumeId (必須)、String)

既存の FSx for ONTAP システムのボリューム ID を指定します。

> **Note**
>
> - ス AWS Batch ケジューラを使用する場合、FSx for ONTAP はヘッドノードでのみ使用できます。

- FSx for ONTAP のデプロイタイプが `Multi-AZ` の場合、ヘッドノードサブネットのルートテーブルが適切に設定されていることを確認してください。
- FSx for ONTAP のサポートが AWS ParallelCluster バージョン 3.2.0 で追加されました。
- ファイルシステムは、ポート 111、635、2049、および 4046 経由のインバウンドおよびアウトバウンドの TCP および UDP トラフィックを許可するセキュリティグループに関連付けられている必要があります。

次のいずれかのアクションを実行して、クラスターとファイルシステムの間でトラフィックが許可されていることを確認してください。

- クラスターサブネットの CIDR またはプレフィックスリストとの間のトラフィックを許可するように、ファイルシステムのセキュリティグループを設定します。

  > **Note**
  >
  > AWS ParallelCluster は、ポートが開いていること、および CIDR またはプレフィックスリストが設定されていることを検証します。CIDR AWS ParallelCluster ブロックまたはプレフィックスリストの内容を検証しません。

- `SlurmQueues`/`Networking`/`SecurityGroups` と `HeadNode`/`Networking`/`SecurityGroups` を使用して、クラスターノードのカスタムセキュリティグループを設定します。カスタムセキュリティグループを設定して、クラスターとファイルシステムの間のトラフィックを許可する必要があります。

  > **Note**
  >
  > すべてのクラスターノードがカスタムセキュリティグループを使用している場合、 はポートが開いていること AWS ParallelCluster のみを検証します。送信元と送信先が正しく設定されていることは検証 AWS ParallelCluster しません。

更新ポリシー: この設定が変更された場合、更新は許可されません。

FsxOpenZfsSettings

> **Note**
>
> FsxOpenZfs が StorageType に指定されている場合、FsxOpenZfsSettings を定義する必要があります。

(オプション) FSx for OpenZFS ファイルシステムの設定。

```
FsxOpenZfsSettings:
  VolumeId: string
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

FsxOpenZfsSettings のプロパティ

VolumeId (必須)、String)

既存の FSx for OpenZFS システムのボリューム ID を指定します。

> **Note**
>
> - ス AWS Batch ケジューラを使用する場合、FSx for OpenZFS はヘッドノードでのみ使用できます。
> - FSx for OpenZFS のサポートが AWS ParallelCluster バージョン 3.2.0 に追加されました。
> - ファイルシステムは、ポート 111、2049、20001、20002、および 20003 経由のインバウンドおよびアウトバウンドの TCP および UDP トラフィックを許可するセキュリティグループに関連付けられている必要があります。

次のいずれかを実行して、クラスターとファイルシステムの間でトラフィックが許可されていることを確認してください。

- クラスターサブネットの CIDR またはプレフィックスリストとの間のトラフィックを許可するように、ファイルシステムのセキュリティグループを設定します。

> **Note**
>
> AWS ParallelCluster は、ポートが開いていること、および CIDR またはプレフィックスリストが設定されていることを検証します。CIDR AWS ParallelCluster ブロックまたはプレフィックスリストの内容を検証しません。

- `SlurmQueues`/`Networking`/`SecurityGroups` と `HeadNode`/`Networking`/`SecurityGroups` を使用して、クラスターノードのカスタムセキュリティグループを設定します。カスタムセキュリティグループを設定して、クラスターとファイルシステムの間のトラフィックを許可する必要があります。

> **Note**
>
> すべてのクラスターノードがカスタムセキュリティグループを使用している場合、 はポートが開いていること AWS ParallelCluster のみを検証します。送信元と送信先が正しく設定されていることは検証 AWS ParallelCluster しません。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`FileCacheSettings`

> **Note**
>
> `FileCache` が `StorageType` に指定されている場合、`FileCacheSettings` を定義する必要があります。

(オプション) ファイルキャッシュの設定。

```
FileCacheSettings:
  FileCacheId: string
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

FileCacheSettings のプロパティ

`FileCacheId` (必須)、`String`)

既存のファイルキャッシュのファイルキャッシュ ID を指定します。

> **Note**
>
> - ファイルキャッシュはス AWS Batch ケジューラをサポートしていません。
> - ファイルキャッシュのサポートが AWS ParallelCluster バージョン 3.7.0 に追加されました。
> - ファイルシステムは、ポート 988 経由のインバウンドおよびアウトバウンドの TCP トラフィックを許可するセキュリティグループに関連付けられている必要があります。

次のいずれかを実行して、クラスターとファイルシステムの間でトラフィックが許可されていることを確認してください。

- クラスターサブネットの CIDR またはプレフィックスリストとの間のトラフィックを許可するように、ファイルキャッシュのセキュリティグループを設定します。

  > **Note**
  >
  > AWS ParallelCluster は、ポートが開いていること、および CIDR またはプレフィックスリストが設定されていることを検証します。CIDR AWS ParallelCluster ブロックまたはプレフィックスリストの内容を検証しません。

- `SlurmQueues`/`Networking`/`SecurityGroups` と `HeadNode`/`Networking`/`SecurityGroups` を使用して、クラスターノードのカスタムセキュリティグループを設定します。カスタムセキュリティグループを設定して、クラスターとファイルシステムの間のトラフィックを許可する必要があります。

  > **Note**
  >
  > すべてのクラスターノードがカスタムセキュリティグループを使用している場合、 はポートが開いていること AWS ParallelCluster のみを検証します。送信元と送信先が正しく設定されていることは検証 AWS ParallelCluster しません。

更新ポリシー: この設定が変更された場合、更新は許可されません。

Iam セクション

(オプション) クラスターの IAM プロパティを指定します。

```
Iam:
  Roles:
    LambdaFunctionsRole: string
  PermissionsBoundary: string
  ResourcePrefix: string
```

更新ポリシー: この設定は、更新中に変更できます。

Iam のプロパティ

`PermissionsBoundary` (オプション、`String`)

AWS ParallelClusterで作成されたすべてのロールのパーミッションバウンダリとして使用する IAM ポリシーの ARN です。詳細については、「IAM ユーザーガイド」の「IAM エンティティのアクセス許可境界」を参照してください。形式は `arn:${Partition}:iam::${Account}:policy/${PolicyName}` です。

更新ポリシー: この設定は、更新中に変更できます。

`Roles` (オプション)

クラスターが使用する IAM ロールの設定を指定します。

更新ポリシー: この設定は、更新中に変更できます。

`LambdaFunctionsRole` (オプション、`String`)

使用する IAM ロールの ARN AWS Lambda。これにより、 AWS CloudFormation カスタムリソースをサポートするすべての Lambda 関数にアタッチされたデフォルトのロールが上書きされます。Lambda は、その役割を担うことを許されたプリンシパルとして設定する必要があります。これにより、使用されている Lambda 関数のロールが上書きされることはありません AWS Batch。形式は `arn:${Partition}:iam::${Account}:role/${RoleName}` です。

更新ポリシー: この設定は、更新中に変更できます。

`ResourcePrefix` (オプション)

によって作成される IAM リソースのパスまたは名前プレフィックスを指定します AWS ParallelCluster。

リソースプレフィックスは IAM が指定する命名規則に従う必要があります。

- 名前の長さは最大 30 文字です。
- 名前には、スラッシュ (/) 文字を含まない文字列のみを使用できます。
- パスは最大 512 文字です。
- パスはスラッシュ (/) で始まり、終わる必要があります。始めのスラッシュと終わりのスラッシュ (/) の間に複数のスラッシュ (/) を含むことができます。
- パスと名前 `/path/name` は組み合わせることができます。

名前を指定します。

```
Iam:
  ResourcePrefix: my-prefix
```

パスを指定します。

```
Iam:
  ResourcePrefix: /org/dept/team/project/user/
```

パスと名前を指定します。

```
Iam:
  ResourcePrefix: /org/dept/team/project/user/my-prefix
```

`/my-prefix` を指定すると、エラーが返されます。

```
Iam:
  ResourcePrefix: /my-prefix
```

設定エラーが返されます。1 つのパスには 2 つの / が必要です。プレフィックス単独で / を付けることはできません。

更新ポリシー: この設定が変更された場合、更新は許可されません。

LoginNodes セクション

> **Note**
>
> のサポートLoginNodesが AWS ParallelCluster バージョン 3.7.0 に追加されました。

(オプション) ログインノードプールの設定を指定します。

```
LoginNodes:
  Pools:
    - Name: string
      Count: integer
      InstanceType: string
      GracetimePeriod: integer
      Image:
        CustomAmi: string
      Ssh:
        KeyName: string
        AllowedIps: string
      Networking:
        SubnetIds:
          - string
        SecurityGroups:
          - string
        AdditionalSecurityGroups:
          - string
      Dcv:
        Enabled: boolean
        Port: integer
        AllowedIps: string
      CustomActions:
        OnNodeStart:
          Sequence:
            - Script: string
              Args:
                - string
          Script: string
          Args:
            - string
        OnNodeConfigured:
          Sequence:
            - Script: string
```

```
            Args:
              - string
          Script: string
          Args:
            - string
        OnNodeUpdated:
          Sequence:
            - Script: string
              Args:
                - string
          Script: string
          Args:
            - string
      Iam:
        InstanceRole: string
        InstanceProfile: string
        AdditionalIamPolicies:
          - Policy: string
```

更新ポリシー: この設定を変更して更新するには、クラスターのログインノードを停止する必要があります。

LoginNodes のプロパティ

Pools のプロパティ

同じリソース設定のログインノードのグループを定義します。3.11.0 以降では AWS ParallelCluster 、最大 10 個のプールを指定できます。

```
Pools:
  - Name: string
    Count: integer
    InstanceType: string
    GracetimePeriod: integer
    Image:
      CustomAmi: string
    Ssh:
      KeyName: string
      AllowedIps: string
    Networking:
      SubnetIds:
        - string
      SecurityGroups:
```

```
        - string
      AdditionalSecurityGroups:
        - string
    Dcv:
      Enabled: boolean
      Port: integer
      AllowedIps: string
    CustomActions:
      OnNodeStart:
        Sequence:
          - Script: string
            Args:
              - string
        Script: string
        Args:
          - string
      OnNodeConfigured:
        Sequence:
          - Script: string
            Args:
              - string
        Script: string
        Args:
          - string
      OnNodeUpdated:
        Sequence:
          - Script: string
            Args:
              - string
        Script: string
        Args:
          - string
    Iam:
      InstanceRole: string
      InstanceProfile: string
      AdditionalIamPolicies:
        - Policy: string
```

更新ポリシー: ログインノードプールを追加できますが、プールを削除するには、クラスター内のすべてのログインノードを停止する必要があります。

`Name` (必須 `String`)

`LoginNodes` プールの名前を指定します。これは、`LoginNodes` リソースのタグ付けに使用されます。

更新ポリシー: この設定が変更された場合、更新は許可されません。

> **Note**
> AWS ParallelCluster バージョン 3.11.0 以降、更新ポリシーは次のとおりです。更新のためにこの設定を変更するには、プール内のログインノードを停止する必要があります。

`Count` (必須 `Integer`)

アクティブを維持するログインノードの数を指定します。

更新ポリシー: この設定は、更新中に変更できます。

`InstanceType` (必須 `String`)

ログインノードに使用される Amazon EC2 インスタンスタイプを定義します。インスタンスタイプのアーキテクチャは、Slurm `InstanceType` 設定に使用されるアーキテクチャと同じでなければなりません。

更新ポリシー: この設定は、ログインノードプールが停止している場合に変更できます。

> **Note**
> AWS ParallelCluster バージョン 3.11.0 以降、更新ポリシーは次のとおりです。更新のためにこの設定を変更するには、プール内のログインノードを停止する必要があります。

`GracetimePeriod` (オプション `Integer`)

ログインノードが廃止されることをログインユーザーに通知してから実際の停止イベントまでに経過する最小時間を分単位で指定します。`GracetimePeriod` の有効な値は 3 分から 120 分までです。デフォルトは 10 分です。

> **Note**
>
> トリガーイベントには、複数の AWS サービス間のインタラクションが含まれます。場合によっては、ネットワークレイテンシーや情報の伝達に時間がかかることがあります。そのため、 AWS サービスの内部遅延により、猶予期間が予想よりも長くかかることがあります。

更新ポリシー: この設定は、更新中に変更できます。

`Image` (オプション)

ログインノードのイメージ設定を定義します。

```
Image:
  CustomAmi: String
```

`CustomAmi` (オプション `String`)

ログインノードのプロビジョニングに使用するカスタム AMI を指定します。指定しない場合、値はデフォルトで HeadNode セクション で指定された値に設定されます。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`Ssh` (オプション)

ログインノードの ssh 設定を定義します。

```
Ssh:
  KeyName: string
  AllowedIps: string
```

> **Note**
>
> AWS ParallelCluster バージョン 3.11.0 以降、更新ポリシーは次のとおりです。更新のためにこの設定を変更するには、プール内のログインノードを停止する必要があります。

`KeyName` (オプション `String`)

ログインノードへのログインに使用する ssh キーを指定します。指定しない場合、値はデフォルトで HeadNode セクション で指定された値に設定されます。

更新ポリシー: この設定を変更して更新するには、プール内のログインノードを停止する必要があります。

`AllowedIps` (オプション `String`)

プール内のログインノードへの SSH 接続の CIDR 形式の IP 範囲またはプレフィックスリスト ID を指定します。デフォルトは、ヘッドノード設定で定義されている AllowedIps です。HeadNode セクション で指定されていない場合は `0.0.0.0/0` です。

更新ポリシー: この設定を変更して更新するには、プール内のログインノードを停止する必要があります。

> **Note**
>
> ログインノードの AllowedIps のサポートが AWS ParallelCluster バージョン 3.11.0 に追加されました。

`Networking` (必須)

```
Networking:
  SubnetIds:
    - string
  SecurityGroups:
    - string
  AdditionalSecurityGroups:
    - string
```

> **Note**
>
> AWS ParallelCluster バージョン 3.11.0 以降、更新ポリシーは次のとおりです。更新のためにこの設定を変更するには、プール内のログインノードを停止する必要があります。

`SubnetIds` (必須 `[String]`)

ログインノードプールをプロビジョニングする既存のサブネットの ID。定義できるサブネットは 1 つだけです。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`SecurityGroups` (オプション `[String]`)

ログインノードプールに使用するセキュリティグループのリスト。セキュリティグループが指定されていない場合、 はセキュリティグループ AWS ParallelCluster を作成します。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`AdditionalSecurityGroups` (オプション `[String]`)

ログインノードプールに使用する追加のセキュリティグループのリスト。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`Dcv` (オプション)

ログインノードで実行される NICE DCV サーバーの設定を定義します。詳細については、Amazon DCV 経由でヘッドノードとログインノードに接続するを参照してください。

```
Dcv:
  Enabled: boolean
  Port: integer
  AllowedIps: string
```

> **Important**
>
> デフォルトでは、 によって設定された NICE DCV ポート AWS ParallelCluster はすべての IPv4 アドレスに対して開かれています。NICE DCV ポートに接続できるのは、NICE DCV セッションの URL があり、その URL が pcluster dcv-connect から返されてから 30 秒以内に NICE DCV セッションに接続する場合のみです。`AllowedIps` 設定を使用して、CIDR 形式の IP 範囲で NICE DCV ポートへのアクセスをさらに制限し、ポート設定を使用して非標準ポートを設定します。

更新ポリシー: この設定が変更された場合、更新は許可されません。

> **Note**
>
> ログインノードでの DCV のサポートが AWS ParallelCluster バージョン 3.11.0 に追加されました。

`Enabled` (必須 `Boolean`)

プール内のログインノードで NICE DCV を有効にするかどうかを指定します。デフォルト値は `false` です。

更新ポリシー: この設定が変更された場合、更新は許可されません。

> **Note**
>
> NICE DCV は、ログインノードで実行される NICE DCV クライアントと NICE DCV サーバー間のトラフィックを保護するために使用される自己署名証明書を自動的に生成します。独自の証明書を設定するには、「Amazon DCV の HTTPS 証明書」を参照してください。

`Port` (オプション `Integer`)

NICE DCV のポートを指定します。デフォルト値は 8443 です。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`AllowedIps` (オプション `String`)

NICE DCV への接続の CIDR 形式の IP 範囲を指定します。この設定は、 がセキュリティグループ AWS ParallelCluster を作成する場合にのみ使用されます。デフォルト値は `0.0.0.0/0` で、すべてのインターネットアドレスからのアクセスを許可します。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`CustomActions` (オプション)

ログインノードで実行するカスタムスクリプトを指定します。

```
CustomActions:
  OnNodeStart:
    Sequence:
      - Script: string
        Args:
          - string
    Script: string
    Args:
      - string
  OnNodeConfigured:
```

```
    Sequence:
      - Script: string
        Args:
          - string
    Script: string
    Args:
      - string
  OnNodeUpdated:
    Sequence:
      - Script: string
        Args:
          - string
    Script: string
    Args:
      - string
```

> **Note**
>
> ログインノードでのカスタムアクションのサポートが AWS ParallelCluster バージョン 3.11.0 に追加されました。

`OnNodeStart` (オプション)

ノードデプロイのブートストラップアクションを開始する前にログインノードで実行する単一のスクリプトまたは一連のスクリプトを指定します。詳細については、「カスタムブートストラップアクション」を参照してください。

`Sequence` (オプション)

run. AWS ParallelCluster run するスクリプトのリスト。 は、設定ファイルに記載されているのと同じ順序でスクリプトを実行します。最初から開始します。

`Script` (必須 `String`)

使用するファイルを指定します。ファイルパスは `https://` または `s3://` で始まる必要があります。

`Args` (オプション `[String]`)

スクリプトに渡す引数のリスト。

更新ポリシー: この設定が変更された場合、更新は許可されません。

Script (必須 String)

単一のスクリプトに使用するファイルを指定します。ファイルパスは https:// または s3:// で始まる必要があります。

Args (オプション [String])

単一のスクリプトに渡す引数のリスト。

OnNodeConfigured (オプション)

ノードブートストラッププロセスの完了後にログインノードで実行する 1 つのスクリプトまたは一連のスクリプトを指定します。詳細については、「カスタムブートストラップアクション」を参照してください。

Sequence (オプション)

run. AWS ParallelCluster run するスクリプトのリスト。 は、設定ファイルに記載されているのと同じ順序でスクリプトを実行します。最初から開始します。

Script (必須 String)

使用するファイルを指定します。ファイルパスは https:// または s3:// で始まる必要があります。

Args (オプション [String])

スクリプトに渡す引数のリスト。

更新ポリシー: この設定が変更された場合、更新は許可されません。

Script (必須 String)

単一のスクリプトに使用するファイルを指定します。ファイルパスは https:// または s3:// で始まる必要があります。

Args (オプション [String])

単一のスクリプトに渡す引数のリスト。

更新ポリシー: この設定が変更された場合、更新は許可されません。

OnNodeUpdated (オプション)

ヘッドノードの更新が完了し、スケジューラと共有ストレージが最新のクラスター設定変更と一致した後に実行する単一のスクリプトまたは一連のスクリプトを指定します。詳細については、「カスタムブートストラップアクション」を参照してください。

`Sequence` (オプション)

run. AWS ParallelCluster run するスクリプトのリスト。 は、設定ファイルに記載されているのと同じ順序でスクリプトを実行します。最初から開始します。

`Script` (必須 `String`)

使用するファイルを指定します。ファイルパスは `https://` または `s3://` で始まる必要があります。

`Args` (オプション `[String]`)

スクリプトに渡す引数のリスト。

`Script` (必須 `String`)

単一のスクリプトに使用するファイルを指定します。ファイルパスは `https://` または `s3://` で始まる必要があります。

`Args` (オプション `[String]`)

単一のスクリプトに渡す引数のリスト。

更新ポリシー: この設定が変更された場合、更新は許可されません。

> **Note**
>
> AWS ParallelCluster は、1 つのスクリプトと の両方を同じカスタムアクションSequenceに含めることをサポートしていません。

`Iam` (オプション)

クラスターのデフォルトのインスタンスロールまたはインスタンスプロファイルをオーバーライドするために、ログインノードで使用するインスタンスロールまたはインスタンスプロファイルのいずれかを指定します。

```
Iam:
  InstanceRole: string
  InstanceProfile: string
  AdditionalIamPolicies:
    - Policy: string
```

> **Note**
>
> AWS ParallelCluster バージョン 3.11.0 以降、更新ポリシーは次のとおりです。更新のためにこの設定を変更するには、プール内のログインノードを停止する必要があります。

`InstanceProfile` (オプション `String`)

デフォルトのログインノードのインスタンスプロファイルをオーバーライドするインスタンスプロファイルを指定します。`InstanceProfile` と `InstanceRole` の両方を指定することはできません。形式は `arn:Partition:iam::Account:instance-profile/`*`InstanceProfileName`* です。これを指定すると、`InstanceRole` と `AdditionalIamPolicies` の設定を指定することはできません。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`InstanceRole` (オプション `String`)

デフォルトのログインノードのインスタンスロールをオーバーライドするインスタンスロールを指定します。`InstanceProfile` と `InstanceRole` の両方を指定することはできません。形式は `arn:Partition:iam::Account:role/RoleName` です。これを指定すると、`InstanceProfile` と `AdditionalIamPolicies` の設定を指定することはできません。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`AdditionalIamPolicies` (オプション)

```
AdditionalIamPolicies:
  - Policy: string
```

IAM ポリシーの Amazon リソースネーム (ARN)。

Amazon EC2 の IAM ポリシーの Amazon リソースネーム (ARN) のリストを指定します。このリストは、 に必要なアクセス許可に加えて、ログインノードに使用されるルートロールにアタッチされます AWS ParallelCluster。

IAM ポリシー名とその ARN が異なります。名前を使用することはできません。

これを指定すると、`InstanceProfile` と `InstanceRole` の設定を指定することはできません。`AdditionalIamPolicies` は AWS ParallelCluster に必要なアクセス許可

に追加され、 には必要なすべてのアクセス許可が含まれているInstanceRole必要があるAdditionalIamPoliciesため、 を使用することをお勧めします。必要な権限は、機能が追加されるにつれ、リリースごとに変更されることがよくあります。

デフォルト値はありません。

更新ポリシー: この設定が変更された場合、更新は許可されません。

Policy (必須 [String])

更新ポリシー: この設定が変更された場合、更新は許可されません。

Monitoring セクション

(オプション) クラスターのモニタリング設定を指定します。

```
Monitoring:
  Logs:
    CloudWatch:
      Enabled: boolean
      RetentionInDays: integer
      DeletionPolicy: string
    Rotation:
      Enabled: boolean
  Dashboards:
    CloudWatch:
      Enabled: boolean
  DetailedMonitoring: boolean
  Alarms:
   Enabled: boolean
```

更新ポリシー: この設定は、更新中には分析されません。

Monitoring のプロパティ

Logs (オプション)

クラスターのログ設定。

更新ポリシー: この設定が変更された場合、更新は許可されません。

CloudWatch (オプション)

クラスターの CloudWatch Logs の設定。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`Enabled` (必須)、`Boolean`)

`true` の場合、クラスターのログは CloudWatch Logs にストリームされます。デフォルト値は `true` です。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`RetentionInDays` (オプション、`Integer`)

CloudWatch Logs にログイベントを保持する日数。デフォルト値は 180 です。サポートされている値は
0、1、3、5、7、14、30、60、90、120、150、180、365、400、545、731、1827、3653
です。値 0 では、デフォルトの CloudWatch ログ保持設定を使用します。つまり、有効期限はありません。

更新ポリシー: この設定は、更新中に変更できます。

`DeletionPolicy` (オプション、`String`)

クラスターの削除時に CloudWatch Logs のログイベントを削除するかどうかを示します。指定できる値は `Delete` および `Retain` です。デフォルト値は `Retain` です。

更新ポリシー: この設定は、更新中に変更できます。

`Rotation` (オプション)

クラスターログのローテーション設定。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`Enabled` (必須)、`Boolean`)

`true` の場合、ログローテーションは有効になっています。デフォルトは `true` です。AWS ParallelCluster 設定されたログファイルが特定のサイズに達すると、ログファイルはローテーションされ、1 つのバックアップが維持されます。詳細については、「AWS ParallelCluster ログローテーションの設定」を参照してください。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`Dashboards` (オプション)

クラスターのダッシュボード設定。

更新ポリシー: この設定は、更新中に変更できます。

`CloudWatch` (オプション)

クラスターの CloudWatch ダッシュボード設定。

更新ポリシー: この設定は、更新中に変更できます。

`Enabled` (必須)、`Boolean`)

`true` の場合、CloudWatch ダッシュボードが有効になります。デフォルト値は `true` です。

更新ポリシー: この設定は、更新中に変更できます。

`DetailedMonitoring` (オプション、`Boolean`)

`true` に設定すると、コンピューティングフリートの Amazon EC2 インスタンスに対して詳細モニタリングが有効になります。有効な場合、Amazon EC2 コンソールに 1 分間隔でインスタンスをモニタリングするグラフが表示されます。この機能を有効にすると追加コストが発生します。デフォルトは `false` です。

詳細については、「Amazon EC2 – Linux インスタンス用ユーザーガイド」の「インスタンスの詳細モニタリングを有効または無効にする」を参照してください。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

> **Note**
>
> `DetailedMonitoring` バージョン 3.6.0 AWS ParallelCluster 以降で が追加されました。

`Alarms` (オプション)

クラスターの CloudWatch アラーム。

更新ポリシー: この設定は、更新中に変更できます。

`Enabled` (オプション)

`true` の場合、クラスターの CloudWatch アラームが作成されます。デフォルト値は `true` です。

更新ポリシー: この設定は、更新中に変更できます。

> **Note**
>
> AWS ParallelCluster バージョン 3.8.0 以降、ヘッドノードに対して次のアラームが作成されます。Amazon EC2 ヘルスチェック、CPU/Memory/Disk使用率、その他すべてを含む複合アラーム。

Tags セクション

（オプション）、配列 によって使用され、すべてのクラスターリソースに AWS CloudFormation 伝達されるタグを定義します。詳細については、「AWS CloudFormation ユーザーガイド」の「AWS CloudFormation リソースタグ」を参照してください。

```
Tags:
  - Key: string
    Value: string
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

Tags のプロパティ

Key (必須)、String)

タグの名前を定義します。

更新ポリシー: この設定が変更された場合、更新は許可されません。

Value (必須)、String)

タグの値を定義します。

更新ポリシー: この設定が変更された場合、更新は許可されません。

AdditionalPackages セクション

(オプション) インストールする追加パッケージを識別するために使用されます。

```
AdditionalPackages:
  IntelSoftware:
    IntelHpcPlatform: boolean
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

IntelSoftware

(オプション) インテル Select ソリューションの設定を定義します。

```
IntelSoftware:
  IntelHpcPlatform: boolean
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

IntelSoftware のプロパティ

`IntelHpcPlatform` (オプション、Boolean)

の場合`true`、Intel Parallel Studio のエンドユーザーライセンス契約が承諾されたことを示します。これにより、インテル Parallel Studio がマスターノードにインストールされ、コンピューティングノードと共有されます。これにより、ヘッドノードのブートストラップに要する時間が数分長くなります。

更新ポリシー: この設定が変更された場合、更新は許可されません。

DirectoryService セクション

> **Note**
>
> AWS ParallelCluster バージョン 3.1.1 で のサポートが追加され`DirectoryService`ました。

(オプション) 複数のユーザーアクセスをサポートするクラスターのディレクトリサービス設定。

AWS ParallelCluster は、System Security Services Daemon (SSSD) でサポートされている Lightweight Directory Access Protocol (LDAP) を介した Active Directory (AD) を持つクラスターへの複数のユーザーアクセスをサポートするアクセス許可を管理します。詳細については、「 AWS Directory Service 管理ガイド」の「What is AWS Directory Service？」を参照してください。

潜在的に機密性の高い情報が暗号化されたチャネルを介して送信されるように LDAP over TLS/SSL (略して LDAPS) を使用することをお勧めします。

```
DirectoryService:
```

```
  DomainName: string
  DomainAddr: string
  PasswordSecretArn: string
  DomainReadOnlyUser: string
  LdapTlsCaCert: string
  LdapTlsReqCert: string
  LdapAccessFilter: string
  GenerateSshKeysForUsers: boolean
  AdditionalSssdConfigs: dict
```

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

DirectoryService のプロパティ

> **Note**
>
> インターネットアクセスのない単一のサブネット AWS ParallelCluster で を使用する予定がある場合は、追加の要件AWS ParallelCluster インターネットアクセスのない単一のサブネット内について「」を参照してください。

`DomainName` (必須)、`String`)

ID 情報に使用するアクティブディレクトリ (AD) ドメイン。

`DomainName` は、完全修飾ドメイン名 (FQDN) と LDAP 識別名 (DN) の両方の形式を受け入れます。

- FQDN の例: `corp.example.com`
- LDAP DN の例: `DC=corp,DC=example,DC=com`

このプロパティは、`ldap_search_base` で呼び出される sssd-ldap パラメータに対応します。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

`DomainAddr` (必須)、`String`)

LDAP サーバーとして使用される AD ドメインコントローラーを指す URI。`ldap_uri` で呼び出される SSSD-LDAP パラメータに対応する URI。値はカンマ区切りの URI の文字列になることもあります。LDAP を使用するには、各 URI の先頭に `ldap://` を追加する必要があります。

値の例:

```
ldap://192.0.2.0,ldap://203.0.113.0          # LDAP
ldaps://192.0.2.0,ldaps://203.0.113.0        # LDAPS without support for certificate
 verification
ldaps://abcdef01234567890.corp.example.com  # LDAPS with support for certificate
 verification
192.0.2.0,203.0.113.0                        # AWS ParallelCluster uses LDAPS by
 default
```

LDAPS を証明書検証と共に使用する場合、URI はホスト名である必要があります。

証明書検証なしで LDAPS または LDAP を使用する場合、URI はホスト名または IP アドレスのいずれかになります。

パスワードやその他の機密情報が暗号化されていないチャネルを介して送信されるのを回避するには、LDAP over TLS/SSL (LDAPS) を使用します。 AWS ParallelCluster でプロトコルが見つからない場合は、各 URI またはホスト名の先頭に `ldaps://` が追加されます。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

`PasswordSecretArn` (必須)、`String`)

プレーンテキストのパスワードを含む AWS Secrets Manager シークレットの Amazon `DomainReadOnlyUser` リソースネーム (ARN)。シークレットの内容は、`ldap_default_authtok` で呼び出される SSSD-LDAP パラメータに対応します。

> **Note**
>
> AWS Secrets Manager コンソールを使用してシークレットを作成する場合は、「その他のタイプのシークレット」を選択し、プレーンテキストを選択し、パスワードテキストのみをシークレットに含めます。
> AWS Secrets Manager を使用してシークレットを作成する方法の詳細については、「 シークAWS Secrets Manager レットの作成」を参照してください。

LDAP クライアントは、ID 情報をリクエスト`DomainReadOnlyUser`するときに、パスワードを使用して AD ドメインを として認証します。

ユーザーに DescribeSecret のアクセス許可がある場合、PasswordSecretArn が検証されます。指定されたシークレットが存在する場合、PasswordSecretArn は有効です。ユーザーの IAM ポリシーに DescribeSecret が含まれていない場合、PasswordSecretArn は検証されず、警告メッセージが表示されます。詳細については、「AWS ParallelCluster pcluster 基本ユーザーポリシー」を参照してください。

シークレットの値が変更された場合、クラスターは自動的に更新されません。クラスターを新しいシークレット値に更新するには、the section called "pcluster update-compute-fleet" コマンドを使用してコンピューティングフリートを停止し、ヘッドノード内から次のコマンドを実行します。

```
$ sudo /opt/parallelcluster/scripts/directory_service/
update_directory_service_password.sh
```

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

DomainReadOnlyUser (必須)、String)

クラスターユーザーのログインを認証するとき、AD ドメインに ID 情報をクエリするために使用される ID。ldap_default_bind_dn で呼び出される SSSD-LDAP パラメータに対応します。この値には AD ID 情報を使用してください。

ノードの特定の LDAP クライアントが必要とする形式で ID を指定します。

- MicrosoftAD:

```
cn=ReadOnlyUser,ou=Users,ou=CORP,dc=corp,dc=example,dc=com
```

- SimpleAD:

```
cn=ReadOnlyUser,cn=Users,dc=corp,dc=example,dc=com
```

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

LdapTlsCaCert (オプション、String)

ドメインコントローラーの証明書を発行した証明書チェーン内のすべての認証機関の証明書を含む証明書バンドルへの絶対パス。ldap_tls_cacert で呼び出される SSSD-LDAP パラメータに対応します。

証明書バンドルは Windows では DER Base64 形式とも呼ばれる PEM 形式 の個別の証明書を連結して構成されたファイルです。これは、LDAP サーバーとして機能する AD ドメインコントローラーの ID を検証するために使用されます。

AWS ParallelCluster は、ノードへの証明書の初期配置には責任を負いません。クラスター管理者として、クラスターの作成後にヘッドノードの証明書を手動で設定することも、ブートストラップスクリプトを使用することもできます。または、ヘッドノードで設定された証明書を含む Amazon マシンイメージ (AMI) を使用することもできます。

Simple AD は LDAPS をサポートしていません。Simple AD ディレクトリを と統合する方法については AWS ParallelCluster、 AWS セキュリティブログの「How to configure an LDAPS endpoint for Simple AD」を参照してください。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

`LdapTlsReqCert` (オプション、`String`)

TLS セッションでサーバー証明書に対して実行する確認内容を指定します。`ldap_tls_reqcert` で呼び出される SSSD-LDAP パラメータに対応します。

有効な値: `never`、`allow`、`try`、`demand`、および `hard`。

`never`、`allow`、および `try` は、証明書の問題が見つかった場合でも接続を続行できるようにします。

`demand` および `hard` は、証明書に問題がない場合でも通信を継続できるようにします。

クラスター管理者が証明書の検証の成功を必要としない値を使用した場合、警告メッセージが管理者に返されます。セキュリティ上の理由から、証明書の検証を無効にしないことをお勧めします。

デフォルト値は `hard` です。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

`LdapAccessFilter` (オプション、`String`)

ディレクトリアクセスを一部のユーザーに制限するフィルターを指定します。このプロパティは、`ldap_access_filter` で呼び出される SSSD-LDAP パラメータに対応します。これを使用して、多数のユーザーをサポートする AD へのクエリに制限できます。

このフィルターは、クラスターへのユーザーアクセスをブロックできます。ただし、ブロックされたユーザーの検出性には影響しません。

このプロパティが設定されている場合、SSSD パラメータ `access_provider` は AWS ParallelCluster によって内部で `ldap` に設定され、`DirectoryService`/`AdditionalSssdConfigs` 設定では変更できません。

このプロパティを省略し、カスタマイズされたユーザーアクセスを `DirectoryService`/`AdditionalSssdConfigs` で指定していない場合、ディレクトリ内のすべてのユーザーがクラスターにアクセスできます。

例:

```
"!(cn=SomeUser*)"  # denies access to every user with an alias that starts with
 "SomeUser"
"(cn=SomeUser*)"   # allows access to every user with alias that starts with
 "SomeUser"
"memberOf=cn=TeamOne,ou=Users,ou=CORP,dc=corp,dc=example,dc=com" # allows access
 only to users in group "TeamOne".
```

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

`GenerateSshKeysForUsers` (オプション、Boolean)

がヘッドノードでの最初の認証の直後にクラスターユーザーに SSH キー AWS ParallelCluster を生成するかどうかを定義します。

`true` に設定すると、ヘッドノードでの最初の認証後、すべてのユーザーについて SSH キーが存在しない場合に生成されて *`USER_HOME_DIRECTORY`*`/.ssh/id_rsa` に保存されます。

ヘッドノードでまだ認証されていないユーザーの場合、次のような場合に初回認証が行われます。

- ユーザーが自身のパスワードで初めてヘッドノードにログインします。
- ヘッドノードで、sudoer が初めてそのユーザーに切り替えます。`su` *`USERNAME`*
- ヘッドノードで、sudoer が初めてそのユーザーとしてコマンドを実行します。`su -u` *`USERNAME COMMAND`*

それ以降は、ユーザーは SSH キーを使用して、クラスターヘッドノードとコンピューティングノードにログインできます。では AWS ParallelCluster、クラスターコンピューティングノードへのパスワードログインは設計上無効になっています。ユーザーがヘッドノードにログインしたこ

とがない場合、SSH キーは生成されず、ユーザーはコンピューティングノードにログインすることはできません。

デフォルトは true です。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

AdditionalSssdConfigs (オプション、Dict)

クラスターインスタンスの SSSD 設定ファイルに書き込む SSSD パラメータと値を含むキーと値のペアのディクショナリ。SSSD 設定ファイルの説明については、SSSD および関連設定ファイルのインスタンスのマニュアルページを参照してください。

SSSD パラメータと値は、次のリストで説明されているように AWS ParallelCluster、 の SSSD 設定と互換性がある必要があります。

- id_provider は によってldap内部的に に設定 AWS ParallelCluster されるため、変更しないでください。
- access_provider は、/ DirectoryService LdapAccessFilterが指定され AWS ParallelCluster ている場合、 によってldap内部的に に設定されます。この設定は変更しないでください。

  DirectoryService/LdapAccessFilter を省略すると、その access_provider の指定も省略されます。例えば、AdditionalSssdConfigs で access_provider を simple に設定した場合、DirectoryService/LdapAccessFilter は指定しないでください。

次の設定スニペットは、AdditionalSssdConfigs の有効な設定の例です。

この例では、SSSD ログのデバッグレベルを有効にし、検索ベースを特定の組織単位に制限し、認証情報のキャッシュを無効にしています。

```
DirectoryService:
  ...
  AdditionalSssdConfigs:
    debug_level: "0xFFF0"
    ldap_search_base: OU=Users,OU=CORP,DC=corp,DC=example,DC=com
    cache_credentials: False
```

この例では SSSD simple access_provider の設定を指定しています。EngineeringTeam からのユーザーにはディレクトリへのアクセス許可が与えられます。この場合、DirectoryService/LdapAccessFilter は設定できません。

```
DirectoryService:
  ...
  AdditionalSssdConfigs:
    access_provider: simple
    simple_allow_groups: EngineeringTeam
```

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

DeploymentSettings セクション

> **Note**
>
> DeploymentSettings バージョン 3.4.0 AWS ParallelCluster 以降で が追加されました。

(オプション) デプロイ設定を指定します。

```
DeploymentSettings:
  LambdaFunctionsVpcConfig:
    SecurityGroupIds:
      - string
    SubnetIds:
      - string
  DisableSudoAccessForDefaultUser: Boolean
  DefaultUserHome: string # 'Shared' or 'Local'
```

DeploymentSettings のプロパティ

LambdaFunctionsVpcConfig

（オプション）AWS Lambda 関数 VPC 設定を指定します。詳細については、「AWS Lambda での VPC 設定 AWS ParallelCluster」を参照してください。

```
LambdaFunctionsVpcConfig:
  SecurityGroupIds:
    - string
  SubnetIds:
    - string
```

LambdaFunctionsVpcConfig properties

`SecurityGroupIds` (必須、`[String]`)

Lambda 関数にアタッチされている Amazon VPC セキュリティグループ ID のリスト。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`SubnetIds` (必須、`[String]`)

Lambda 関数にアタッチされているサブネット ID のリスト。

更新ポリシー: この設定が変更された場合、更新は許可されません。

> **Note**
> サブネットとセキュリティグループは、同じ VPC にある必要があります。

DisableSudoAccessForDefaultUser プロパティ

> **Note**
> この設定オプションは、Slurm クラスターでのみサポートされています。

(オプション) `True` の場合、デフォルトユーザーの sudo 権限は無効になります。これは、クラスター内のすべてのノードに適用されます。

```
# Main DeploymentSettings section in config yaml(applies to HN, CF and LN)
DeploymentSettings:
  DisableSudoAccessForDefaultUser: True
```

`DisableSudoAccessForDefaultUser` の値を更新するには、コンピューティングフリートとすべてのログインノードを停止する必要があります。

更新ポリシー: この設定を変更して更新するには、コンピューティングフリートとログインノードを停止する必要があります。

DefaultUserHome プロパティ

`Shared` に設定すると、クラスターはデフォルト設定を使用し、`/home/<default user>` により、クラスター全体でデフォルトユーザーのディレクトリを共有します。

`Local` に設定すると、ヘッドノード、ログインノード、およびコンピューティングノードごとに、個別のローカルデフォルトユーザーディレクトリが `local/home/<default user>` に保存されます。

ビルドイメージの設定ファイル

AWS ParallelCluster バージョン 3 では、ビルドイメージ設定パラメータに YAML 1.1 ファイルを使用します。設定エラーを減らすために、インデントが正しいことを確認してください。詳細については、https://yaml.org/spec/1.1/ の「YAML 1.1 仕様」を参照してください。

これらの設定ファイルは、EC2 Image Builder を使用して AWS ParallelCluster AMIsを構築する方法を定義するために使用されます。カスタム AMI の構築プロセスは、`pcluster build-image` コマンドを使って起動します。設定ファイルの例については、https://github.com/aws/aws-parallelcluster/tree/release-3.0/cli/tests/pcluster/schemas/test_imagebuilder_schema/test_imagebuilder_schema を参照してください。

トピック

- ビルドイメージの設定ファイルプロパティ
- Build セクション
- Image セクション
- DeploymentSettings セクション

ビルドイメージの設定ファイルプロパティ

`Region` (オプション、`String`)

`build-image` オペレーション AWS リージョン の を指定します。例えば、`us-east-2`。

`CustomS3Bucket` (オプション、`String`)

カスタム AMI ビルドプロセスで使用されるリソースを保存し、ログをエクスポートするために AWS アカウントで作成される Amazon S3 バケットの名前を指定します。イメージで使用される情報は、イメージ config のカスタムバケットにあります。クラスターを作成する各 AWS リー

ジョンに 1 つの Amazon S3 バケットが AWS ParallelCluster 維持されます。デフォルトでは、これらの Amazon S3 バケットには `parallelcluster-hash-v1-DO-NOT-DELETE` という名前が付けられます。

Build セクション

(必須) イメージが構築される構成を指定します。

```
Build:
  Imds:
    ImdsSupport: string
  InstanceType: string
  SubnetId: string
  ParentImage: string
  Iam:
    InstanceRole: string
    InstanceProfile: string
    CleanupLambdaRole: string
    AdditionalIamPolicies:
      - Policy: string
    PermissionsBoundary: string
  Components:
    - Type: string
      Value: string
  Tags:
    - Key: string
      Value: string
  SecurityGroupIds:
    - string
  UpdateOsPackages:
    Enabled: boolean
  Installation:
    NvidiaSoftware:
      Enabled: boolean
    LustreClient:
      Enabled: boolean
```

Build のプロパティ

`InstanceType` (必須)、`String`)

イメージのビルドに使用するインスタンスのインスタンスタイプを指定します。

`SubnetId` (オプション、`String`)

イメージを構築するためにインスタンスをプロビジョニングする既存のサブネットの ID を指定します。提供されるサブネットには、インターネットへのアクセスが必要です。ビルドが失敗した場合、サブネットの IP アドレス属性を変更する必要がある場合があります。

> **Warning**
>
> `pcluster build-image` はデフォルトの VPC を使用します。 AWS Control Tower または AWS Landing Zone を使用してデフォルトの VPC が削除された場合は、サブネット ID を指定する必要があります。

SubnetId を指定する場合は、SecurityGroupIds プロパティも指定することをお勧めします。SecurityGroupIds を除外すると、 AWS ParallelCluster はデフォルトのセキュリティグループを使用するか、指定されたサブネット内のデフォルトの動作に依存します。両方を使用すると、次の利点があります。

- きめ細かな制御: 両方を明示的に定義すると、イメージビルドプロセス中に起動されたインスタンスが正しいサブネットに配置され、ビルドコンポーネントと必要なサービス (ビルドスクリプトの S3 へのアクセスなど) に必要な正確なネットワークアクセスが確保されます。
- セキュリティのベストプラクティス: 適切なセキュリティグループを定義すると、必要なポートとサービスのみにネットワークアクセスを制限できるため、ビルド環境のセキュリティが向上します。
- 潜在的な問題の回避: デフォルトのみに依存すると、セキュリティグループが開かれすぎたり制限が厳しくなり、ビルドプロセス中に問題が発生する可能性があります。

`ParentImage` (必須)、`String`)

ベースイメージを指定します。親イメージは、非 AWS ParallelCluster AMI でも、同じバージョンの公式 AMI AWS ParallelCluster でもかまいません。別のバージョンの AWS ParallelCluster 公式またはカスタム AMI を使用することはできません AWS ParallelCluster。フォーマットは、イメージ `arn:`*`Partition`*`:imagebuilder:`*`Region`*`:`*`Account`*`:image/`*`ImageName`*`/`*`ImageVersion` の ARN または AMI ID `ami-12345678` のいずれかでなければなりません。

`SecurityGroupIds` (オプション、`[String]`)

イメージのセキュリティグループ ID のリストを指定します。

Imds

Imds のプロパティ

(オプション) Amazon EC2 ImageBuilder ビルドおよびテストインスタンスメタデータサービス (IMDS) の設定を指定します。

```
Imds:
  ImdsSupport: string
```

`ImdsSupport` (オプション、`String`)

EC2 ImageBuilder ビルドとテストインスタンスで、どの IMDS バージョンをサポートするかを指定します。サポートされている値は`v2.0` および `v1.0` です。デフォルト値は `v2.0` です。

`ImdsSupport` が `v1.0` に設定されている場合、IMDSv1 と IMDSv2 の両方がサポートされます。

`ImdsSupport` が `v2.0` に設定されている場合、IMDSv2 のみがサポートされます。

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「IMDSv2 の使用」を参照してください。

更新ポリシー: この設定が変更された場合、更新は許可されません。

> **Note**
>
> AWS ParallelCluster バージョン 3.7.0 以降、`ImdsSupport`デフォルト値は ですv2.0。`ImdsSupport` を `v2.0` に設定し、カスタムアクション呼び出しにおいて、IMDSv1 を IMDSv2 に置き換えることをお勧めします。
> AWS ParallelCluster バージョン `Imds` 3.3.0 で / のサポートが追加され`ImdsSupport`ました。

Iam

Iam のプロパティ

(オプション) イメージビルドの IAM リソースを指定します。

```
Iam:
  InstanceRole: string
```

```
  InstanceProfile: string
  CleanupLambdaRole: string
  AdditionalIamPolicies:
    - Policy: string
  PermissionsBoundary: string
```

`InstanceProfile` (オプション、`String`)

EC2 Image Builder インスタンスのデフォルトのインスタンスプロファイルをオーバーライドするインスタンスプロファイルを指定します。`InstanceProfile`、`InstanceRole` および `AdditionalIamPolicies` を一緒に指定することはできません。形式は `arn:`*`Partition`*`:iam::`*`Account`*`:instance-profile/`*`InstanceProfileName` です。

`InstanceRole` (オプション、`String`)

EC2 Image Builder インスタンスのデフォルトのインスタンスロールをオーバーライドするインスタンスロールを指定します。 `InstanceProfile`、`InstanceRole` および `AdditionalIamPolicies` を一緒に指定することはできません。形式は `arn:`*`Partition`*`:iam::`*`Account`*`:role/`*`RoleName` です。

`CleanupLambdaRole` (オプション、`String`)

ビルド完了時にビルドアーティファクトを削除する AWS CloudFormation カスタムリソースをバックアップする AWS Lambda 関数に使用する IAM ロールの ARN。Lambda は、その役割を担うことを許されたプリンシパルとして設定する必要があります。形式は `arn:`*`Partition`*`:iam::`*`Account`*`:role/`*`RoleName` です。

`AdditionalIamPolicies` (オプション)

カスタム AMI の生成に使用する EC2 Image Builder インスタンスにアタッチする追加の IAM ポリシーを指定します。

```
AdditionalIamPolicies:
  - Policy: string
```

`Policy` (オプション、`[String]`)

IAM ポリシーの一覧。形式は `arn:`*`Partition`*`:iam::`*`Account`*`:policy/`*`PolicyName` です。

`PermissionsBoundary` (オプション、`String`)

AWS ParallelClusterで作成されたすべてのロールのパーミッションバウンダリとして使用する IAM ポリシーの ARN です。許可の境界の詳細については、「IAM ユー

ザーガイド」の「IAM エンティティの許可の境界」を参照してください。形式は `arn:Partition:iam::Account:policy/PolicyName` です。

Components

Components のプロパティ

(オプション) AMI 構築プロセスで使用する Amazon EC2 ImageBuilder コンポーネントを、 AWS ParallelClusterがデフォルトで提供するものに加えて指定します。このようなコンポーネントは、AMI のビルドプロセスをカスタマイズするために使用できます。詳細については、「AWS ParallelCluster AMI のカスタマイズ」を参照してください。

```
Components:
  - Type: string
    Value: string
```

Type (オプション、String)

コンポーネントのタイプ-値のペアのタイプを指定します。タイプは `arn` または `script` となります。

Value (オプション、String)

コンポーネントのタイプ-値のペアの値を指定します。タイプが `arn` の場合は、EC2 Image Builder コンポーネントの ARN です。type が の場合`script`、これは EC2 Image Builder コンポーネントの作成時に使用するスクリプトを指す https または s3 リンクです。

Tags

Tags のプロパティ

(オプション) AMI を構築するために使用されるリソースに設定するタグのリストを指定します。

```
Tags:
  - Key: string
    Value: string
```

Key (オプション、String)

タグの名前を定義します。

`Value` (オプション、`String`)

タグの値を定義します。

UpdateOsPackages

UpdateOsPackages のプロパティ

（オプション) AWS ParallelCluster ソフトウェアスタックをインストールする前にオペレーティングシステムを更新するかどうかを指定します。

```
UpdateOsPackages:
  Enabled: boolean
```

`Enabled` (オプション、`Boolean`)

の場合`true`、 AWS ParallelCluster ソフトウェアをインストールする前に OS が更新され、再起動されます。デフォルトは `false` です。

> **Note**
>
> `UpdateOsPackages` を有効にすると、カーネルを含め、使用可能なすべての OS パッケージが更新されます。お客様は、更新が更新に含まれていない AMI 依存関係と互換性があることを確認する責任があります。
> たとえば、カーネル AWS ParallelCluster バージョン Y.0 と一部のコンポーネントバージョン Z.0 に同梱されているバージョン X.0 用の AMI を構築するとします。利用可能な更新に、更新済みのカーネルバージョン Y.1 が含まれており、コンポーネント Z.0 への更新は行われていないとします。`UpdateOsPackages` を有効にする前に、コンポーネント Z.0 がカーネル Y.1 をサポートしていることを確認する責任があります。

Installation

Installation のプロパティ

（オプション）イメージにインストールする追加のソフトウェアを指定します。

```
Installation:
  NvidiaSoftware:
```

```
    Enabled: boolean
  LustreClient:
    Enabled: boolean
```

`NvidiaSoftware` プロパティ (オプション)

インストールする Nvidia ソフトウェアを指定します。

```
NvidiaSoftware:
    Enabled: boolean
```

`Enabled` (オプション、`boolean`)

の場合`true`、Nvidia GPU ドライバーと CUDA がインストールされます。デフォルトは `false` です。

`LustreClient` プロパティ (オプション)

Amazon FSx Lustre クライアントをインストールするように指定します。

```
LustreClient:
    Enabled: boolean
```

`Enabled` (オプション、`boolean`)

の場合`true`、Lustre クライアントがインストールされます。デフォルトは `true` です。

`Image` セクション

(オプション) イメージビルドのイメージプロパティを定義します。

```
Image:
  Name: string
  RootVolume:
    Size: integer
    Encrypted: boolean
    KmsKeyId: string
  Tags:
    - Key: string
      Value: string
```

Image のプロパティ

Name (オプション、String)

AMI の名前を指定します。指定されていない場合は、pcluster build-image コマンドの呼び出し時の名前が使用されます。

Tags

Tags のプロパティ

(オプション) イメージのキーバリューのペアを指定します。

```
Tags:
  - Key: string
    Value: string
```

Key (オプション、String)

タグの名前を定義します。

Value (オプション、String)

タグの値を定義します。

RootVolume

RootVolume のプロパティ

(オプション) イメージのルートボリュームのプロパティを指定します。

```
RootVolume:
  Size: integer
  Encrypted: boolean
  KmsKeyId: string
```

Size (オプション、Integer)

イメージのルートボリュームのサイズを GiB 単位で指定します。デフォルトのサイズは、ParentImage のサイズに 27 GiB を加えたものです。

`Encrypted` (オプション、`Boolean`)

ボリュームを暗号化するかどうかを指定します。デフォルト値は `false` です。

`KmsKeyId` (オプション、`String`)

ボリュームの暗号化に使用される AWS KMS キーの ARN を指定します。形式は `arn:Partition:kms:Region:Account:key/KeyId` です。

`DeploymentSettings` セクション

> **Note**
>
> `DeploymentSettings` バージョン 3.4.0 AWS ParallelCluster 以降で が追加されました。

(オプション) デプロイ設定を指定します。

```
DeploymentSettings:
  LambdaFunctionsVpcConfig:
    SecurityGroupIds:
      - string
    SubnetIds:
      - string
```

`DeploymentSettings` のプロパティ

`LambdaFunctionsVpcConfig`

（オプション）AWS Lambda 関数 VPC 設定を指定します。詳細については、「AWS Lambda での VPC 設定 AWS ParallelCluster」を参照してください。

```
LambdaFunctionsVpcConfig:
  SecurityGroupIds:
    - string
  SubnetIds:
    - string
```

LambdaFunctionsVpcConfig properties

`SecurityGroupIds` (必須、`[String]`)

Lambda 関数にアタッチされている Amazon VPC セキュリティグループ ID のリスト。

更新ポリシー: この設定が変更された場合、更新は許可されません。

`SubnetIds` (必須、`[String]`)

Lambda 関数にアタッチされているサブネット ID のリスト。

更新ポリシー: この設定が変更された場合、更新は許可されません。

> **Note**
>
> サブネットとセキュリティグループは、同じ VPC にある必要があります。

AWS ParallelCluster API リファレンス

このセクションでは、各 AWS ParallelCluster API アクションの説明、構文、使用例について説明します。

トピック

- buildImage
- createCluster
- deleteCluster
- deleteClusterInstances
- deleteImage
- describeCluster
- describeClusterInstances
- describeComputeFleet
- describeImage
- getClusterLogEvents

- getClusterStackEvents
- getImageLogEvents
- getImageStackEvents
- listClusters
- listClusterLogStreams
- listImageLogStreams
- listImages
- listOfficialImages
- updateCluster
- updateComputeFleet

buildImage

でカスタム AWS ParallelCluster イメージを作成します AWS リージョン。

トピック

- リクエスト構文
- リクエスト本文
- レスポンスの構文
- レスポンス本文
- 例

リクエスト構文

```
POST /v3/images/custom
{
  "imageConfiguration": "string",
  "imageId": "string",
  "dryrun": boolean,
  "region": "string",
  "rollbackOnFailure": boolean,
  "supressValidators": [ "string" ],
  "validationFailureLevel": "string"
}
```

リクエスト本文

imageConfiguration

YAML ドキュメントとしてのイメージ設定。

タイプ: 文字列

必須: はい

imageId

ビルドするイメージの ID。

タイプ: 文字列

必須: はい

dryrun

`true` に設定すると、リソースを作成することなく、リクエストの検証のみを行います。このパラメータを使用して、イメージ設定を検証します。デフォルトは `false` です。

型: ブール値

必須: いいえ

region

コマンド AWS リージョン を実行してイメージを構築する 。

型: 文字列

必須: いいえ

rollbackOnFailure

`true` に設定すると、イメージの作成に失敗した場合にイメージスタックのロールバックが行われます。デフォルトは `false` です。

型: ブール値

必須: いいえ

suppressValidators

抑制する 1 つまたは複数の設定バリデータを指定します。

タイプ: 文字列のリスト

形式: `(ALL|type:[A-Za-z0-9]+)`

必須: いいえ

validationFailureLevel

イメージビルドが失敗する最小の検証レベル。デフォルトは `ERROR` です。

タイプ: 文字列

有効な値: `INFO | WARNING | ERROR`

必須 : いいえ

レスポンスの構文

```
{
  "image": {
    "imageId": "string",
    "ec2AmiInfo": {
      "amiId": "string"
    },
    "region": "string",
    "version": "string",
    "cloudformationStackArn": "string",
    "imageBuildStatus": "BUILD_IN_PROGRESS",
    "cloudformationStackStatus": "CREATE_IN_PROGRESS"
  },
  "validationMessages": [
    {
      "id": "string",
      "type": "string",
      "level": "INFO",
      "message": "string"
    }
  ]
```

```
}
```

レスポンス本文

画像

imageId

イメージの ID。

タイプ: 文字列

cloudformationStackArn

メイン CloudFormation スタックの Amazon リソースネーム (ARN)。

タイプ: 文字列

cloudformationStackStatus

CloudFormation スタックの状態。

タイプ: 文字列

有効な値: `CREATE_IN_PROGRESS | CREATE_FAILED | CREATE_COMPLETE | ROLLBACK_IN_PROGRESS | ROLLBACK_FAILED | ROLLBACK_COMPLETE | DELETE_IN_PROGRESS | DELETE_FAILED | DELETE_COMPLETE | UPDATE_IN_PROGRESS | UPDATE_COMPLETE_CLEANUP_IN_PROGRESS | UPDATE_COMPLETE | UPDATE_ROLLBACK_IN_PROGRESS | UPDATE_ROLLBACK_FAILED | UPDATE_ROLLBACK_COMPLETE_CLEANUP_IN_PROGRESS | UPDATE_ROLLBACK_COMPLETE`

ec2AmiInfo

ami_id

Amazon EC2 AMI ID。

タイプ: 文字列

imageBuildStatus

イメージビルドのステータス。

タイプ: 文字列

有効な値: `BUILD_IN_PROGRESS | BUILD_FAILED | BUILD_COMPLETE | DELETE_IN_PROGRESS | DELETE_FAILED | DELETE_COMPLETE`

region

イメージが構築される AWS リージョン 。

タイプ: 文字列

version

イメージの構築に使用される AWS ParallelCluster バージョン。

タイプ: 文字列

validationMessages

検証レベルが `validationFailureLevel` 以下のメッセージのリスト。メッセージのリストは設定の検証中に収集されます。

id

バリデータの ID。

タイプ: 文字列

level

検証レベル。

タイプ: 文字列

有効な値: `INFO | WARNING | ERROR`

message

検証メッセージ。

タイプ: 文字列

type

バリデータのタイプ。

タイプ: 文字列

例

Python

リクエスト

```
$ build_image(custom-image-id, custom-image-config.yaml)
```

200 レスポンス

```
{
  "image": {
    "cloudformation_stack_arn": "arn:aws:cloudformation:us-
east-1:123456789012:stack/custom-image-id/711b76b0-af81-11ec-a29f-0ee549109f1f",
    "cloudformation_stack_status": "CREATE_IN_PROGRESS",
    "image_build_status": "BUILD_IN_PROGRESS",
    "image_id": "custom-image-id",
    "region": "us-east-1",
    "version": "3.2.1"
  }
}
```

createCluster

AWS リージョンにマネージドクラスターを作成します。

トピック

- リクエスト構文
- リクエスト本文
- レスポンスの構文
- レスポンス本文
- 例

リクエスト構文

```
POST /v3/clusters
{
```

```
  "clusterName": "string",
  "clusterConfiguration": "string",
  "dryrun": boolean,
  "region": "string",
  "rollbackOnFailure", boolean,
  "suppressValidators": [ "string" ],
  "validationFailureLevel": "string"
}
```

リクエスト本文

clusterConfiguration

YAML ドキュメントとしてのクラスター設定。

タイプ: 文字列

必須: はい

clusterName

作成するクラスターの名前。

名前はアルファベット文字で始まる必要があります。名前は最大 60 文字です。Slurm アカウンティングが有効になっている場合、名前は最大 40 文字です。

タイプ: 文字列

必須: はい

dryrun

に設定すると`true`、リクエストの検証のみが実行され、リソースは作成されません。このパラメータを使用してクラスター設定を検証します。デフォルトは `false` です。

型: ブール値

必須: いいえ

region

クラスター AWS リージョン がある 。

型: 文字列

必須: いいえ

rollbackOnFailure

true に設定した場合、クラスターの作成に失敗するとクラスタースタックのロールバックが行われます。デフォルトは true です。

型: ブール値

必須: いいえ

suppressValidators

抑制する 1 つまたは複数の設定バリデータを指定します。

タイプ: 文字列のリスト

形式: (ALL|type:[A-Za-z0-9]+)

必須: いいえ

validationFailureLevel

クラスター作成が失敗する最小の検証レベル。デフォルトは ERROR です。

タイプ: 文字列

有効な値: INFO | WARNING | ERROR

必須：いいえ

レスポンスの構文

```
{
  "cluster": {
    "clusterName": "string",
    "region": "string",
    "version": "string",
    "cloudformationStackArn": "string",
    "cloudformationStackStatus": "CREATE_IN_PROGRESS",
    "clusterStatus": "CREATE_IN_PROGRESS",
    "scheduler": {
      "type": "string",
```

```
      "metadata": {
        "name": "string",
        "version": "string"
      }
    }
  },
  "validationMessages": [
    {
      "id": "string",
      "type": "string",
      "level": "INFO",
      "message": "string"
    }
  ]
}
```

レスポンス本文

clusterName

クラスターの名前。

タイプ: 文字列

cloudformationStackArn

メイン CloudFormation スタックの Amazon リソースネーム (ARN)。

タイプ: 文字列

cloudformationStackStatus

タイプ: 文字列

有効な値: `CREATE_IN_PROGRESS | CREATE_FAILED | CREATE_COMPLETE | ROLLBACK_IN_PROGRESS | ROLLBACK_FAILED | ROLLBACK_COMPLETE | DELETE_IN_PROGRESS | DELETE_FAILED | DELETE_COMPLETE | UPDATE_IN_PROGRESS | UPDATE_COMPLETE_CLEANUP_IN_PROGRESS | UPDATE_COMPLETE | UPDATE_ROLLBACK_IN_PROGRESS | UPDATE_ROLLBACK_FAILED | UPDATE_ROLLBACK_COMPLETE_CLEANUP_IN_PROGRESS | UPDATE_ROLLBACK_COMPLETE`

clusterStatus

タイプ: 文字列

有効な値: `CREATE_IN_PROGRESS | CREATE_FAILED | CREATE_COMPLETE | DELETE_IN_PROGRESS | DELETE_FAILED | DELETE_COMPLETE | UPDATE_IN_PROGRESS | UPDATE_COMPLETE | UPDATE_FAILED`

region

クラスター AWS リージョン が作成される 。

タイプ: 文字列

スケジューラ

metadata

スケジューラのメタデータ

名前

スケジューラの名前。

タイプ: 文字列

version

スケジューラのバージョン。

タイプ: 文字列

type

スケジューラタイプ。

タイプ: 文字列

version

クラスターの作成に使用される AWS ParallelCluster バージョン。

タイプ: 文字列

validation_messages

検証レベルが `validationFailureLevel` 以下のメッセージのリスト。メッセージのリストは設定の検証中に収集されます。

id

バリデータの ID。

タイプ: 文字列

level

タイプ: 文字列

有効な値: `INFO | WARNING | ERROR`

message

検証メッセージ。

タイプ: 文字列

type

バリデータのタイプ。

タイプ: 文字列

例

Python

リクエスト

```
$ create_cluster(cluster_name_3x, cluster-config.yaml)
```

200 レスポンス

```
{
  "cluster": {
    "cloudformation_stack_arn": "arn:aws:cloudformation:us-
east-1:123456789012:stack/cluster-3x/e0462730-50b5-11ed-99a3-0a5ddc4a34c7",
    "cloudformation_stack_status": "CREATE_IN_PROGRESS",
    "cluster_name": "cluster-3x",
    "cluster_status": "CREATE_IN_PROGRESS",
    "region": "us-east-1",
    "scheduler": {
      "type": "slurm"
    },
    "version": "3.2.1"
  }
}
```

deleteCluster

クラスターの削除を開始します。

トピック

- リクエスト構文
- リクエスト本文
- レスポンスの構文
- レスポンス本文
- 例

リクエスト構文

```
DELETE /v3/clusters/{clusterName}
{
  "region": "string"
}
```

リクエスト本文

clusterName

クラスターの名前。

タイプ: 文字列

必須: はい

region

クラスターが削除される AWS リージョン 。

型: 文字列

必須: いいえ

レスポンスの構文

```
{
```

```
    "cluster": {
        "clusterName": "string",
        "region": "string",
        "version": "string",
        "cloudformationStackArn": "string",
        "cloudformationStackStatus": "DELETE_IN_PROGRESS",
        "clusterStatus": "DELETE_IN_PROGRESS",
        "scheduler": {
            "type": "string",
            "metadata": {
                "name": "string",
                "version": "string"
            }
         }
     }
}
```

レスポンス本文

クラスター

クラスターインスタンスのリスト。

clusterName

クラスターの名前。

タイプ: 文字列

cloudformationStackArn

メイン CloudFormation スタックの Amazon リソースネーム (ARN)。

タイプ: 文字列

cloudformationStackStatus

タイプ: 文字列

有効な値: `CREATE_IN_PROGRESS | CREATE_FAILED | CREATE_COMPLETE | ROLLBACK_IN_PROGRESS | ROLLBACK_FAILED | ROLLBACK_COMPLETE | DELETE_IN_PROGRESS | DELETE_FAILED | DELETE_COMPLETE | UPDATE_IN_PROGRESS | UPDATE_COMPLETE_CLEANUP_IN_PROGRESS | UPDATE_COMPLETE | UPDATE_ROLLBACK_IN_PROGRESS |`

UPDATE_ROLLBACK_FAILED | UPDATE_ROLLBACK_COMPLETE_CLEANUP_IN_PROGRESS | UPDATE_ROLLBACK_COMPLETE

clusterStatus

タイプ: 文字列

有効な値: CREATE_IN_PROGRESS | CREATE_FAILED | CREATE_COMPLETE | DELETE_IN_PROGRESS | DELETE_FAILED | DELETE_COMPLETE | UPDATE_IN_PROGRESS | UPDATE_COMPLETE | UPDATE_FAILED

region

クラスターが作成される AWS リージョン 。

タイプ: 文字列

スケジューラ

metadata

スケジューラのメタデータ。

名前

スケジューラの名前。

タイプ: 文字列

version

スケジューラのバージョン。

タイプ: 文字列

type

スケジューラタイプ。

タイプ: 文字列

version

クラスターの作成に使用される AWS ParallelCluster バージョン。

タイプ: 文字列

例

Python

リクエスト

```
$ delete_cluster(cluster_name_3x)
```

200 レスポンス

```
{
  "cluster": {
    "cloudformation_stack_arn": "arn:aws:cloudformation:us-
east-1:123456789012:stack/cluster_name_3x/16b49540-aee5-11ec-8e18-0ac1d712b241",
    "cloudformation_stack_status": "DELETE_IN_PROGRESS",
    "cluster_name": "cluster_name_3x",
    "cluster_status": "DELETE_IN_PROGRESS",
    "region": "us-east-1",
    "version": "3.2.1"
  }
}
```

deleteClusterInstances

すべてのクラスターコンピューティングノードの強制終了を開始します。このアクションは AWS Batch クラスターをサポートしていません。

トピック

- リクエスト構文
- リクエスト本文
- レスポンス本文
- 例

リクエスト構文

```
DELETE /v3/clusters/{clusterName}/instances
{
  "force": boolean,
```

```
  "region": "string"
}
```

リクエスト本文

clusterName

クラスターの名前。

タイプ: 文字列

必須: はい

force

`true` に設定すると、指定した名前のクラスターが見つからなかったときに強制的に削除します。デフォルトは `false` です。

型: ブール値

必須: いいえ

region

クラスター AWS リージョン がある 。

型: 文字列

必須: いいえ

レスポンス本文

なし

例

Python

リクエスト

```
$ delete_cluster_instances(cluster_name_3x)
```

200 レスポンス

なし

deleteImage

カスタム AWS ParallelCluster イメージの削除を開始します。

トピック

- リクエスト構文
- リクエスト本文
- レスポンスの構文
- レスポンス本文
- 例

リクエスト構文

```
DELETE /v3/images/custom/{imageId}
{
  "force": boolean,
  "region": "string"
}
```

リクエスト本文

imageId

イメージの ID。

タイプ: 文字列

必須: はい

force

`true` に設定すると、AMI を強制的に削除します。AMI を使用するインスタンスがある場合、または AMI が共有されている場合は、このパラメータを使用します。デフォルトは `false` です。

型: ブール値

必須: いいえ

region

イメージが作成された AWS リージョン 。

型: 文字列

必須: いいえ

レスポンスの構文

```
{
  "image": {
    "imageId": "string",
    "ec2AmiInfo": {
      "amiId": "string"
    },
    "region": "string",
    "version": "string",
    "cloudformationStackArn": "string",
    "imageBuildStatus": "DELETE_IN_PROGRESS",
    "cloudformationStackStatus": "DELETE_IN_PROGRESS"
  }
}
```

レスポンス本文

画像

cloudformationStackArn

メイン CloudFormation スタックの Amazon リソースネーム (ARN)。

タイプ: 文字列

cloudformationStackStatus

CloudFormation スタックの状態。

タイプ: 文字列

有効な値: `CREATE_IN_PROGRESS | CREATE_FAILED | CREATE_COMPLETE | ROLLBACK_IN_PROGRESS | ROLLBACK_FAILED | ROLLBACK_COMPLETE | DELETE_IN_PROGRESS | DELETE_FAILED | DELETE_COMPLETE |`

`UPDATE_IN_PROGRESS | UPDATE_COMPLETE_CLEANUP_IN_PROGRESS | UPDATE_COMPLETE | UPDATE_ROLLBACK_IN_PROGRESS | UPDATE_ROLLBACK_FAILED | UPDATE_ROLLBACK_COMPLETE_CLEANUP_IN_PROGRESS | UPDATE_ROLLBACK_COMPLETE`

ec2AmiInfo

amiId

Amazon EC2 AMI ID。

タイプ: 文字列

imageBuildStatus

イメージビルドのステータス。

タイプ: 文字列

有効な値: `BUILD_IN_PROGRESS | BUILD_FAILED | BUILD_COMPLETE | DELETE_IN_PROGRESS | DELETE_FAILED | DELETE_COMPLETE`

imageId

イメージの ID。

タイプ: 文字列

region

イメージが作成される AWS リージョン 。

タイプ: 文字列

version

イメージの構築に使用される AWS ParallelCluster バージョン。

タイプ: 文字列

例

Python

リクエスト

```
$ delete_image(custom-image-id)
```

200 レスポンス

```
{
  "image": {
    "image_build_status": "DELETE_IN_PROGRESS",
    "image_id": "custom-image-id",
    "region": "us-east-1",
    "version": "3.2.1"
  }
}
```

describeCluster

既存のクラスターの詳細情報を取得します。

トピック

- リクエスト構文
- リクエスト本文
- レスポンスの構文
- レスポンス本文
- 例

リクエスト構文

```
GET /v3/clusters/{clusterName}
{
  "region": "string"
}
```

リクエスト本文

clusterName

クラスターの名前。

タイプ: 文字列

必須: はい

region

クラスター AWS リージョン がある 。

型: 文字列

必須: いいえ

レスポンスの構文

> **Note**
>
> `failureReason` が AWS ParallelCluster バージョン 3.5.0 `failures`以降に変更されました。

```
{
  "clusterName": "string",
  "region": "string",
  "version": "string",
  "cloudFormationStackStatus": "CREATE_IN_PROGRESS",
  "clusterStatus": "CREATE_IN_PROGRESS",
  "scheduler": {
    "type": "string",
    "metadata": {
      "name": "string",
      "version": "string"
    }
  },
  "cloudformationStackArn": "string",
  "creationTime": "2019-08-24T14:15:22Z",
  "lastUpdatedTime": "2019-08-24T14:15:22Z",
  "clusterConfiguration": {
    "url": "string"
  },
  "computeFleetStatus": "START_REQUESTED",
  "tags": [
    {
```

```
      "key": "string",
      "value": "string"
    }
  ],
  "headNode": {
    "instanceId": "string",
    "instanceType": "string",
    "launchTime": "2019-08-24T14:15:22Z",
    "privateIpAddress": "string",
    "publicIpAddress": "string",
    "state": "pending"
  },
  "failures": [
    {
      "failureCode": "string",
      "failureReason": "string"
    }
  ],
  "loginNodes": {
    "status": "string",
    "address": "string",
    "poolName": "string",
    "scheme": "string",
    "healthyNodes": integer,
    "unhealthyNodes": integer
  }
}
```

レスポンス本文

clusterName

クラスターの名前。

タイプ: 文字列

cloudformationStackArn

メイン CloudFormation スタックの Amazon リソースネーム (ARN)。

タイプ: 文字列

cloudformationStackStatus

CloudFormation スタックの状態。

タイプ: 文字列

有効な値: `CREATE_IN_PROGRESS | CREATE_FAILED | CREATE_COMPLETE | ROLLBACK_IN_PROGRESS | ROLLBACK_FAILED | ROLLBACK_COMPLETE | DELETE_IN_PROGRESS | DELETE_FAILED | DELETE_COMPLETE | UPDATE_IN_PROGRESS | UPDATE_COMPLETE_CLEANUP_IN_PROGRESS | UPDATE_COMPLETE | UPDATE_ROLLBACK_IN_PROGRESS | UPDATE_ROLLBACK_FAILED | UPDATE_ROLLBACK_COMPLETE_CLEANUP_IN_PROGRESS | UPDATE_ROLLBACK_COMPLETE`

clusterConfiguration

url

クラスター設定ファイルの URL。

タイプ: 文字列

clusterStatus

クラスターのステータス。

タイプ: 文字列

有効な値: `CREATE_IN_PROGRESS | CREATE_FAILED | CREATE_COMPLETE | DELETE_IN_PROGRESS | DELETE_FAILED | DELETE_COMPLETE | UPDATE_IN_PROGRESS | UPDATE_COMPLETE | UPDATE_FAILED`

computeFleetStatus

コンピューティングフリートのステータス。

タイプ: 文字列

有効な値: `START_REQUESTED | STARTING | RUNNING | PROTECTED | STOP_REQUESTED | STOPPING | STOPPED | UNKNOWN | ENABLED | DISABLED`

creationTime

クラスターが作成された時刻のタイムスタンプ。

型: 日時

lastUpdatedTime

クラスターが最後に更新された時刻のタイムスタンプ。

型: 日時

region

クラスターが作成される AWS リージョン 。

タイプ: 文字列

tags

クラスターに関連付けられているタグのリスト。

キー

タグ名。

タイプ: 文字列

タグ

タグ値。

タイプ: 文字列

version

クラスターの作成に使用される AWS ParallelCluster バージョン。

タイプ: 文字列

エラー

クラスタースタックが `CREATE_FAILED` ステータスのときの障害リスト。

failureCode

クラスタースタックが `CREATE_FAILED` ステータスのときの失敗コード。

タイプ: 文字列

failureReason

クラスタースタックが `CREATE_FAILED` ステータスになったときの失敗の理由。

タイプ: 文字列

head_node

クラスターヘッドノード。

instanceId

Amazon EC2 インスタンス ID。

タイプ: 文字列

instanceType

Amazon EC2 インスタンスタイプ

タイプ: 文字列

launchTime

EC2 インスタンスの起動時刻。

型: 日時

privateIpAddress

クラスターのプライベート IP アドレス。

タイプ: 文字列

publicIpAddress

クラスターのパブリック IP アドレス。

タイプ: 文字列

state

ヘッドノードのインスタンスステータス。

タイプ: 文字列

有効な値: `pending | running | shutting-down | terminated | stopping | stopped`

スケジューラ

metadata

スケジューラのメタデータ。

名前

スケジューラの名前。

タイプ: 文字列

version

スケジューラのバージョン。

タイプ: 文字列

loginNodes

ステータス

ログインノードのステータス。

タイプ: 文字列

有効な値: `PENDING | FAILED | ACTIVE`

アドレス

ログインノードのアドレス。

タイプ: 文字列

poolName

ログインノードのプール名。

タイプ: 文字列

scheme

ログインノードのスキーム。

タイプ: 文字列

healthyNodes

正常なノードの数。

タイプ: 整数

unhealthyNodes

異常なノードの数。

タイプ: 整数

type

スケジューラタイプ。

タイプ: 文字列

例

Python

リクエスト

```
$ describe_cluster(cluster_name_3x)
```

200 レスポンス

```
{
  "cloud_formation_stack_status": "CREATE_COMPLETE",
  "cloudformation_stack_arn": "arn:aws:cloudformation:us-east-1:123456789012:stack/
cluster_name_3x/16b49540-aee5-11ec-8e18-0ac1d712b241",
  "cluster_configuration": {
    "url": "https://parallelcluster-...."
  },
  "cluster_name": "cluster_name_3x",
  "cluster_status": "CREATE_COMPLETE",
  "compute_fleet_status": "RUNNING",
  "creation_time": datetime.datetime(2022, 3, 28, 22, 19, 9, 661000,
 tzinfo=tzlocal()),
  "head_node": {
    "instance_id": "i-abcdef01234567890",
    "instance_type": "t2.micro",
    "launch_time": datetime.datetime(2022, 3, 28, 22, 21, 56, tzinfo=tzlocal()),
    "private_ip_address": "172.31.56.3",
    "public_ip_address": "107.23.100.164",
    "state": "running"
  },
  "last_updated_time": datetime.datetime(2022, 3, 28, 22, 19, 9, 661000,
 tzinfo=tzlocal()),
  "region": "us-east-1",
  "tags": [
    {
      "key": "parallelcluster:version", "value": "3.2.1"
```

```
    }
  ],
  "version": "3.2.1"
}
```

describeClusterInstances

クラスターに属するインスタンスについて説明します。

トピック

- リクエスト構文
- リクエスト本文
- レスポンスの構文
- レスポンス本文
- 例

リクエスト構文

```
GET /v3/clusters/{clusterName}/instances
{
  "nextToken": "string",
  "nodeType": "string",
  "queueName": "string",
  "region": "string"
}
```

リクエスト本文

clusterName

クラスターの名前。

タイプ: 文字列

必須: はい

nextToken

次の結果セットのトークン。

型: 文字列

必須: いいえ

nodeType

ノードタイプ別にインスタンスをフィルタリングします。

タイプ: 文字列

有効な値: `HeadNode`、`ComputeNode`、`LoginNode`

必須: いいえ

queueName

キュー名でインスタンスをフィルタリングします。

型: 文字列

必須: いいえ

region

クラスター AWS リージョン がある 。

型: 文字列

必須: いいえ

レスポンスの構文

```
{
  "nextToken": "string",
  "instances": [
    {
      "instanceId": "string",
      "instanceType": "string",
      "launchTime": "2019-08-24T14:15:22Z",
      "privateIpAddress": "string",
      "publicIpAddress": "string",
      "state": "pending",
      "nodeType": "HeadNode",
      "queueName": "string",
      "poolName": "string"
```

```
    }
  ]
}
```

レスポンス本文

インスタンス

クラスターインスタンスのリスト。

instanceId

Amazon EC2 インスタンス ID。

タイプ: 文字列

instanceType

Amazon EC2 インスタンスタイプ

タイプ: 文字列

launchTime

EC2 インスタンスの起動時刻。

型: 日時

nodeType

ノードタイプ。

タイプ: 文字列

有効な値: `HeadNode`、`ComputeNode`、`LoginNode`

publicIpAddress

クラスターのパブリック IP アドレス。

タイプ: 文字列

queueName

Amazon EC2 インスタンスがノードをバックアップしているキューの名前。

タイプ: 文字列

state

ノードの Amazon EC2 インスタンスのステータス。

タイプ: 文字列

有効な値: `pending | running | shutting-down | terminated | stopping | stopped`

nextToken

次の結果セットの取得に使用できるトークン、または追加の結果がない場合は `null` です。

タイプ: 文字列

例

Python

リクエスト

```
$ describe_cluster_instances(cluster_name_3x)
```

200 レスポンス

```
{
  "instances": [
    {
      "instance_id": "i-abcdef01234567890",
      "instance_type": "t2.micro",
      "launch_time": datetime.datetime(2022, 3, 30, 14, 2, 7, tzinfo=tzlocal()),
      "node_type": "HeadNode",
      "private_ip_address": "192.0.2.5",
      "public_ip_address": "198.51.100.180",
      "state": "running"
    }
  ]
}
```

describeComputeFleet

コンピューティングフリートの状況について説明してください。

トピック

- リクエスト構文
- リクエスト本文
- レスポンスの構文
- レスポンス本文
- 例

リクエスト構文

```
GET /v3/clusters/{clusterName}/computefleet
{
  "region": "string"
}
```

リクエスト本文

clusterName

クラスターの名前。

タイプ: 文字列

必須: はい

region

クラスター AWS リージョン がある 。

型: 文字列

必須: いいえ

レスポンスの構文

```
{
  "status": "START_REQUESTED",
  "lastStatusUpdatedTime": "2019-08-24T14:15:22Z"
}
```

レスポンス本文

ステータス

タイプ: 文字列

有効な値: `START_REQUESTED | STARTING | RUNNING | PROTECTED | STOP_REQUESTED | STOPPING | STOPPED | UNKNOWN | ENABLED | DISABLED`

lastStatusUpdatedTime

ステータスの最終更新時刻を表すタイムスタンプ。

型: 日時

例

Python

リクエスト

```
$ describe_compute_fleet(cluster_name_3x)
```

200 レスポンス

```
{
  "last_status_updated_time": datetime.datetime(2022, 3, 28, 22, 27, 14,
 tzinfo=tzlocal()),
  "status": "RUNNING"
}
```

describeImage

既存のイメージの詳細情報を取得します。

トピック

- リクエスト構文
- リクエスト本文
- レスポンスの構文

- レスポンス本文
- 例

リクエスト構文

```
GET /v3/images/custom/{imageId}
{
  "region": "string"
}
```

リクエスト本文

imageId

イメージの ID。

タイプ: 文字列

必須: はい

region

イメージが作成された AWS リージョン 。

型: 文字列

必須: いいえ

レスポンスの構文

```
{
  "imageId": "string",
  "region": "string",
  "version": "string",
  "imageBuildStatus": "BUILD_IN_PROGRESS",
  "imageBuildLogsArn": "string",
  "cloudformationStackStatus": "CREATE_IN_PROGRESS",
  "cloudformationStackStatusReason": "string",
  "cloudformationStackArn": "string",
  "creationTime": "2019-08-24T14:15:22Z",
```

```
  "cloudformationStackCreationTime": "2019-08-24T14:15:22Z",
  "cloudformationStackTags": [
    {
      "key": "string",
      "value": "string"
    }
  ],
  "imageConfiguration": {
    "url": "string"
  },
  "imagebuilderImageStatus": "PENDING",
  "imagebuilderImageStatusReason": "string",
  "ec2AmiInfo": {
    "amiId": "string",
    "tags": [
      {
        "key": "string",
        "value": "string"
      }
    ],
    "amiName": "string",
    "architecture": "string",
    "state": "PENDING",
    "description": "string"
  }
}
```

レスポンス本文

imageId

詳細情報を取得するイメージの ID。

タイプ: 文字列

imageBuildStatus

イメージビルドのステータス。

タイプ: 文字列

有効な値: `BUILD_IN_PROGRESS | BUILD_FAILED | BUILD_COMPLETE | DELETE_IN_PROGRESS | DELETE_FAILED | DELETE_COMPLETE`

imageConfiguration

url

イメージ設定ファイルの URL。

タイプ: 文字列

region

イメージが作成される AWS リージョン 。

タイプ: 文字列

version

イメージの構築に使用される AWS ParallelCluster バージョン。

タイプ: 文字列

cloudformationStackArn

メイン CloudFormation スタックの Amazon リソースネーム (ARN)。

タイプ: 文字列

cloudformationStackCreationTime

CloudFormation スタックが作成された時刻。

型: 日時

cloudformationStackStatus

CloudFormation スタックの状態。

タイプ: 文字列

有効な値: `CREATE_IN_PROGRESS | CREATE_FAILED | CREATE_COMPLETE | ROLLBACK_IN_PROGRESS | ROLLBACK_FAILED | ROLLBACK_COMPLETE | DELETE_IN_PROGRESS | DELETE_FAILED | DELETE_COMPLETE | UPDATE_IN_PROGRESS | UPDATE_COMPLETE_CLEANUP_IN_PROGRESS | UPDATE_COMPLETE | UPDATE_ROLLBACK_IN_PROGRESS | UPDATE_ROLLBACK_FAILED | UPDATE_ROLLBACK_COMPLETE_CLEANUP_IN_PROGRESS | UPDATE_ROLLBACK_COMPLETE`

cloudformationStackStatusReason

CloudFormation スタックのステータスの理由。

タイプ: 文字列

cloudformationStackTags

CloudFormation スタックのタグのリスト。

キー

タグ名。

タイプ: 文字列

値

タグ値。

タイプ: 文字列

creationTime

イメージが作成された時刻。

型: 日時

ec2AmiInfo

amiId

Amazon EC2 AMI ID。

タイプ: 文字列

amiName

Amazon EC2 AMI 名。

タイプ: 文字列

architecture

Amazon EC2 AMI アーキテクチャ。

タイプ: 文字列

state

Amazon EC2 AMI の状態。

タイプ: 文字列

有効な値: `PENDING | AVAILABLE | INVALID | DEREGISTERED | TRANSIENT | FAILED | ERROR`

tags

Amazon EC2 AMI タグのリスト。

キー

タグ名。

タイプ: 文字列

値

タグ値。

タイプ: 文字列

imagebuilderImageStatus

ImageBuilder イメージのステータス。

タイプ: 文字列

有効な値: `PENDING | CREATING | BUILDING | TESTING | DISTRIBUTING | INTEGRATING | AVAILABLE | CANCELLED | FAILED | DEPRECATED | DELETED`

imagebuilderImageStatusReason

ImageBuilder イメージにそのステータスがある理由。

タイプ: 文字列

imageBuildLogsArn

イメージビルドプロセスのログの Amazon リソースネーム (ARN)。

タイプ: 文字列

例

Python

リクエスト

```
$ describe_image(custom-image-id)
```

200 レスポンス

```
{
  "cloudformation_stack_arn": "arn:aws:cloudformation:us-east-1:123456789012:stack/
custom-image-id/6accc570-b080-11ec-845e-0e2dc6386985",
  "cloudformation_stack_creation_time": datetime.datetime(2022, 3, 30, 23, 23, 33,
 731000, tzinfo=tzlocal()),
  "cloudformation_stack_status": "CREATE_IN_PROGRESS",
  "cloudformation_stack_tags": [
    {
      "key": "parallelcluster:version", "value": "3.2.1"
    },
    {
      "key": "parallelcluster:image_name",
      "value": 'custom-image-id"
    },
    {
      "key": "parallelcluster:custom-image-id",
      "value": "custom-image-id"
    },
    {
      "key": 'parallelcluster:amzn-s3-demo-bucket",
      "value": 'amzn-s3-demo-bucket"
    },
    {
      "key": "parallelcluster:s3_image_dir",
      "value": "parallelcluster/3.2.1/images/custom-image-id-1234567890abcdef0"
    },
    {
      "key": "parallelcluster:build_log",
      "value": "arn:aws:logs:us-east-1:123456789012:log-group:/aws/imagebuilder/
ParallelClusterImage-custom-image-id"
    },
    {
      "key": "parallelcluster:build_config",
      "value": "s3://amzn-s3-demo-bucket/parallelcluster/3.2.1/images/custom-image-
id-1234567890abcdef0/configs/image-config.yaml"
    }
  ],
  "image_build_logs_arn": "arn:aws:logs:us-east-1:123456789012:log-group:/aws/
imagebuilder/ParallelClusterImage-alinux2-image",
```

```
  "image_build_status": "BUILD_IN_PROGRESS",
  "image_configuration": {
    "url": "https://amzn-s3-demo-bucket.s3.amazonaws.com/parallelcluster/3.2.1/
images/custom-image-id-1234567890abcdef0/configs/image-config.yaml?..."
  },
  "image_id": 'custom-image-id',
  "imagebuilder_image_status": "PENDING",
  "region": "us-east-1",
  "version": "3.2.1"
}
```

getClusterLogEvents

ログストリームに関連するイベントを取得します。

トピック

- リクエスト構文
- リクエスト本文
- レスポンスの構文
- レスポンス本文
- 例

リクエスト構文

```
GET /v3/clusters/{clusterName}/logstreams/{logStreamName}
{
  "endTime": datetime,
  "limit": float,
  "nextToken": "string",
  "region": "string",
  "startFromHead": boolean,
  "startTime": datetime
}
```

リクエスト本文

clusterName

クラスターの名前。

タイプ: 文字列

必須: はい

logStreamName

ログストリームの名前。

タイプ: 文字列

必須: はい

endTime

ISO 8601 形式で表される時間範囲の終了時刻。この時間と同じかそれ以降のタイムスタンプを持つイベントは含まれません。

型: 日時

形式: `2021-01-01T20:00:00Z`

必須: いいえ

制限

返されるログイベントの最大数数。値を指定しない場合、最大値は 1 MB のレスポンスサイズに収まるログイベントの数、または最大 10,000 個のログイベントです。

タイプ: 浮動小数点

必須: いいえ

nextToken

次の結果セットのトークン。

型: 文字列

必須: いいえ

region

クラスター AWS リージョン がある 。

型: 文字列

必須: いいえ

startFromHead

`true` に設定すると、最も古いログイベントが最初に返されます。値が `false` の場合、最新のログイベントが最初に返されます。デフォルトは `false` です。

型: ブール値

必須: いいえ

startTime

ISO 8601 形式で表される時間範囲の開始時刻。タイムスタンプがこの時間と同じか、この時間よりも遅いイベントが含まれます。

型: 日時

形式: `2021-01-01T20:00:00Z`

必須: いいえ

レスポンスの構文

```
{
  "nextToken": "string",
  "prevToken": "string",
  "events": [
    {
      "timestamp": "2019-08-24T14:15:22Z",
      "message": "string"
    }
  ]
}
```

レスポンス本文

events

フィルタリングされたイベントのリスト。

message

イベントメッセージ。

タイプ: 文字列

timestamp

イベントタイムスタンプ。

型: 日時

nextToken

次の結果セットの取得に使用できるトークン、または追加の結果がない場合は null です。

タイプ: 文字列

prevToken

前の結果セットを取得するために使用できるトークン、または追加の結果nullがない場合。

タイプ: 文字列

例

Python

リクエスト

```
$ get_cluster_log_events(cluster_name_3x, log_stream_name=ip-192-0-2-26.i-abcdef01234567890.cfn-init)
```

200 レスポンス

```
"events": [
  {
    "message": "2022-09-22 16:40:15,127 [DEBUG] CloudFormation client initialized
 with endpoint https://cloudformation.us-east-1.amazonaws.com",
    "timestamp": "2022-09-22T16:40:15.127Z"
  },
  {
    "message": "2022-09-22 16:40:15,127 [DEBUG] Describing resource
 HeadNodeLaunchTemplate in stack cluster_name_3x",
    "timestamp": "2022-09-22T16:40:15.127Z"
  },
  ...
]
```

getClusterStackEvents

クラスターのスタックに関連するイベントを取得します。

> **Note**
>
> バージョン 3.6.0 以降、 はネストされたスタック AWS ParallelCluster を使用して、キューとコンピューティングリソースに関連付けられたリソースを作成します。`GetClusterStackEvents` API と `pcluster get-cluster-stack-events` コマンドはクラスターのメインスタックイベントのみを返します。CloudFormation コンソールでは、キューやコンピューティングリソースに関連するイベントを含むクラスタースタックイベントを表示できます。

トピック

- リクエスト構文
- リクエスト本文
- レスポンスの構文
- レスポンス本文
- 例

リクエスト構文

```
GET /v3/clusters/{clusterName}/stackevents
{
  "nextToken": "string",
  "region": "string"
}
```

リクエスト本文

clusterName

クラスターの名前。

タイプ: 文字列

必須: はい

nextToken

次の結果セットのトークン。

型: 文字列

必須: いいえ

region

クラスター AWS リージョン がある 。

型: 文字列

必須: いいえ

レスポンスの構文

```
{
  "nextToken": "string",
  "events": [
    {
      "stackId": "string",
      "eventId": "string",
      "stackName": "string",
      "logicalResourceId": "string",
      "physicalResourceId": "string",
      "resourceType": "string",
      "timestamp": "2019-08-24T14:15:22Z",
      "resourceStatus": "CREATE_IN_PROGRESS",
      "resourceStatusReason": "string",
      "resourceProperties": "string",
      "clientRequestToken": "string"
    }
  ]
}
```

レスポンス本文

events

フィルタリングされたイベントのリスト。

clientRequestToken

このイベントを生成したアクションに渡されたトークン。

タイプ: 文字列

eventId

このイベントの一意の ID。

タイプ: 文字列

logicalResourceId

テンプレートに定義されたリソースの論理名。

タイプ: 文字列

physicalResourceId

リソースの物理インスタンスに関連付けられている名前または一意の識別子。

タイプ: 文字列

resourceProperties

リソースの作成に使用されるプロパティの BLOB。

タイプ: 文字列

resourceStatus

リソースステータス。

タイプ: 文字列

有効な値: `CREATE_IN_PROGRESS | CREATE_FAILED | CREATE_COMPLETE | DELETE_IN_PROGRESS | DELETE_FAILED | DELETE_COMPLETE | DELETE_SKIPPED | UPDATE_IN_PROGRESS | UPDATE_FAILED | UPDATE_COMPLETE | IMPORT_FAILED | IMPORT_COMPLETE | IMPORT_IN_PROGRESS | IMPORT_ROLLBACK_IN_PROGRESS | IMPORT_ROLLBACK_FAILED | IMPORT_ROLLBACK_COMPLETE`

resourceStatusReason

リソースに関連する成功または失敗のメッセージ。

タイプ: 文字列

resourceType

リソースのタイプ。

タイプ: 文字列

stackId

スタックのインスタンスの一意の ID 名。

タイプ: 文字列

stackName

スタックに関連付けられた名前。

タイプ: 文字列

timestamp

ステータスが更新された時刻。

型: 日時

nextToken

次の結果セットの取得に使用できるトークン、または追加の結果がない場合は `null` です。

タイプ: 文字列

例

Python

リクエスト

```
$ get_cluster_stack_events(cluster_name_3x)
```

200 レスポンス

```
{
```

```
  "events": [
    {
      "event_id": "590b3820-b081-11ec-985e-0a7af5751497",
      "logical_resource_id": "cluster_name_3x",
      "physical_resource_id": "arn:aws:cloudformation:us-east-1:123456789012:stack/
cluster_name_3x/11a59710-b080-11ec-b8bd-129def1380e9",
      "resource_status": "CREATE_COMPLETE",
      "resource_type": "AWS::CloudFormation::Stack",
      "stack_id": "arn:aws:cloudformation:us-east-1:123456789012:stack/
cluster_name_3x/11a59710-b080-11ec-b8bd-129def1380e9",
      "stack_name": "cluster_name_3x",
      "timestamp": datetime.datetime(2022, 3, 30, 23, 30, 13, 268000,
 tzinfo=tzlocal())
    },
    ...
  ]
}
```

getImageLogEvents

イメージビルドに関連するイベントを取得します。

トピック

- リクエスト構文
- リクエスト本文
- レスポンスの構文
- レスポンス本文
- 例

リクエスト構文

```
GET /v3/images/custom/{imageId}/logstreams/{logStreamName}
{
  "endTime": datetime,
  "limit": float,
  "nextToken": "string",
  "region": "string",
  "startFromHead": boolean,
  "startTime": datetime
```

```
}
```

リクエスト本文

imageId

イメージの ID。

タイプ: 文字列

必須: はい

logStreamName

ログストリームの名前。

タイプ: 文字列

必須: はい

endTime

ISO 8601 形式で表される時間範囲の終了時刻。この時間と同じかそれ以降のタイムスタンプを持つイベントは含まれません。

型: 日時

形式: `2021-01-01T20:00:00Z`

必須: いいえ

制限

返されるログイベントの最大数数。値を指定しない場合、最大値は 1 MB の応答サイズに収まるログイベントの数で、最大 10,000 ログイベントまでとなります。

タイプ: 浮動小数点

必須: いいえ

nextToken

次の結果セットのトークン。

型: 文字列

必須: いいえ

region

イメージ AWS リージョン がある 。

型: 文字列

必須: いいえ

startFromHead

`true` に設定すると、最も古いログイベントを最初に返します。false に設定すると、最新のログイベントを最初に返します。デフォルトは `false` です。

型: ブール値

必須: いいえ

startTime

ISO 8601 形式で表される時間範囲の開始時刻。タイムスタンプがこの時間と同じか、この時間よりも遅いイベントが含まれます。

型: 日時

形式: `2021-01-01T20:00:00Z`

必須: いいえ

レスポンスの構文

```
{
  "nextToken": "string",
  "prevToken": "string",
  "events": [
    {
      "timestamp": "2019-08-24T14:15:22Z",
      "message": "string"
    }
  ]
```

```
}
```

レスポンス本文

events

フィルタリングされたイベントのリスト。

message

イベントメッセージ。

タイプ: 文字列

timestamp

イベントタイムスタンプ。

型: 日時

nextToken

次の結果セットの取得に使用できるトークン、または追加の結果がない場合は null です。

タイプ: 文字列

prevToken

前の結果セットを取得するために使用できるトークン、または追加の結果nullがない場合。

タイプ: 文字列

例

Python

リクエスト

```
$ get_image_log_events(image_id, log_stream_name=3.2.1/1)
```

200 レスポンス

```
"events": [
  {
```

```
    "message": "ExecuteBash: STARTED EXECUTION",
    "timestamp": 2022-04-05T15:51:20.228Z"
  },
  {
    "message": "ExecuteBash: Created temporary directory: /tmp/1234567890abcdef0",
    "timestamp": "2022-04-05T15:51:20.228Z"
  },
  ...
]
```

getImageStackEvents

イメージビルドのスタックに関連するイベントを取得します。

トピック

- リクエスト構文
- リクエスト本文
- レスポンスの構文
- レスポンス本文
- 例

リクエスト構文

```
GET /v3/images/custom/{imageId}/stackevents
{
  "nextToken": "string",
  "region": "string"
}
```

リクエスト本文

imageId

イメージの ID。

タイプ: 文字列

必須: はい

nextToken

次の結果セットのトークン。

型: 文字列

必須: いいえ

region

イメージ AWS リージョン がある 。

型: 文字列

必須: いいえ

レスポンスの構文

```
{
  "nextToken": "string",
  "events": [
    {
      "stackId": "string",
      "eventId": "string",
      "stackName": "string",
      "logicalResourceId": "string",
      "physicalResourceId": "string",
      "resourceType": "string",
      "timestamp": "2019-08-24T14:15:22Z",
      "resourceStatus": "CREATE_IN_PROGRESS",
      "resourceStatusReason": "string",
      "resourceProperties": "string",
      "clientRequestToken": "string"
    }
  ]
}
```

レスポンス本文

events

フィルタリングされたイベントのリスト。

clientRequestToken

このイベントを生成したアクションに渡されたトークン。

タイプ: 文字列

eventId

このイベントの一意の ID。

タイプ: 文字列

logicalResourceId

テンプレートに定義されたリソースの論理名。

タイプ: 文字列

physicalResourceId

リソースの物理インスタンスに関連付けられている名前または一意の識別子。

タイプ: 文字列

resourceProperties

リソースの作成に使用されるプロパティの BLOB。

タイプ: 文字列

resourceStatus

リソースステータス。

タイプ: 文字列

有効な値: `CREATE_IN_PROGRESS | CREATE_FAILED | CREATE_COMPLETE | DELETE_IN_PROGRESS | DELETE_FAILED | DELETE_COMPLETE | DELETE_SKIPPED | UPDATE_IN_PROGRESS | UPDATE_FAILED | UPDATE_COMPLETE | IMPORT_FAILED | IMPORT_COMPLETE | IMPORT_IN_PROGRESS | IMPORT_ROLLBACK_IN_PROGRESS | IMPORT_ROLLBACK_FAILED | IMPORT_ROLLBACK_COMPLETE`

resourceStatusReason

リソースに関連する成功または失敗のメッセージ。

タイプ: 文字列

resourceType

リソースのタイプ。

タイプ: 文字列

stackId

スタックのインスタンスの一意の ID 名。

タイプ: 文字列

stackName

スタックに関連付けられた名前。

タイプ: 文字列

timestamp

ステータスが更新された時刻。

型: 日時

nextToken

次の結果セットの取得に使用できるトークン、または追加の結果がない場合は `null` です。

タイプ: 文字列

例

Python

リクエスト

```
$ get_image_stack_events(image_id)
```

200 レスポンス

```
{
  'events': [
    {
```

```
      'event_id': 'ParallelClusterImage-
CREATE_IN_PROGRESS-2022-03-30T23:26:33.499Z',
      'logical_resource_id': 'ParallelClusterImage',
      'physical_resource_id': 'arn:aws:imagebuilder:us-east-1:123456789012:image/
parallelclusterimage-alinux2-image/3.2.1/1',
      'resource_properties': {
        "InfrastructureConfigurationArn":"arn:aws:imagebuilder:us-
east-1:123456789012:infrastructure-configuration/parallelclusterimage-6accc570-
b080-11ec-845e-0e2dc6386985",
        "ImageRecipeArn":"arn:aws:imagebuilder:us-east-1:123456789012:image-recipe/
parallelclusterimage-alinux2-image/3.2.1",
        "DistributionConfigurationArn":"arn:aws:imagebuilder:us-
east-1:123456789012:distribution-configuration/parallelclusterimage-6accc570-
b080-11ec-845e-0e2dc6386985",
        "EnhancedImageMetadataEnabled":"false",
        "Tags": {
          "parallelcluster:image_name":"alinux2-
image","parallelcluster:image_id":"alinux2-image"
        }
      },
      'resource_status': 'CREATE_IN_PROGRESS',
      'resource_status_reason': 'Resource creation Initiated',
      'resource_type': 'AWS::ImageBuilder::Image',
      'stack_id': 'arn:aws:cloudformation:us-east-1:123456789012:stack/alinux2-
image/6accc570-b080-11ec-845e-0e2dc6386985',
      'stack_name': 'alinux2-image',
      'timestamp': datetime.datetime(2022, 3, 30, 23, 26, 33, 499000,
 tzinfo=tzlocal())
    },
    ...
  ]
}
```

listClusters

既存のクラスターのリストを取得します。

トピック

- リクエスト構文
- リクエスト本文
- レスポンスの構文

- レスポンス本文
- 例

リクエスト構文

```
GET /v3/clusters
{
  "clusterStatus": "string",
  "nextToken": "string",
  "region": "string"
}
```

リクエスト本文

clusterStatus

クラスターステータスによるフィルタリング。デフォルトはすべてのクラスターです。

タイプ: 文字列

有効な値: `CREATE_IN_PROGRESS | CREATE_FAILED | CREATE_COMPLETE | DELETE_IN_PROGRESS | DELETE_FAILED | UPDATE_IN_PROGRESS | UPDATE_COMPLETE | UPDATE_FAILED`

必須: いいえ

nextToken

次の結果セットのトークン。

型: 文字列

必須: いいえ

region

クラスター AWS リージョン の 。

型: 文字列

必須: いいえ

レスポンスの構文

```
{
  "nextToken": "string",
  "clusters": [
    {
      "clusterName": "string",
      "region": "string",
      "version": "string",
      "cloudformationStackArn": "string",
      "cloudformationStackStatus": "CREATE_IN_PROGRESS",
      "clusterStatus": "CREATE_IN_PROGRESS",
      "scheduler": {
        "type": "string",
        "metadata": {
          "name": "string",
          "version": "string"
        }
      }
    }
  ]
}
```

レスポンス本文

クラスター

cloudformationStackArn

メイン CloudFormation スタックの Amazon リソースネーム (ARN)。

タイプ: 文字列

cloudformationStackStatus

CloudFormation スタックの状態。

タイプ: 文字列

有効な値: `CREATE_IN_PROGRESS | CREATE_FAILED | CREATE_COMPLETE | ROLLBACK_IN_PROGRESS | ROLLBACK_FAILED | ROLLBACK_COMPLETE | DELETE_IN_PROGRESS | DELETE_FAILED | DELETE_COMPLETE | UPDATE_IN_PROGRESS | UPDATE_COMPLETE_CLEANUP_IN_PROGRESS`

| UPDATE_COMPLETE | UPDATE_ROLLBACK_IN_PROGRESS | UPDATE_ROLLBACK_FAILED | UPDATE_ROLLBACK_COMPLETE_CLEANUP_IN_PROGRESS | UPDATE_ROLLBACK_COMPLETE

clusterName

クラスターの名前。

タイプ: 文字列

clusterStatus

クラスターのステータス。

タイプ: 文字列

有効な値: CREATE_IN_PROGRESS | CREATE_FAILED | CREATE_COMPLETE | DELETE_IN_PROGRESS | DELETE_FAILED | DELETE_COMPLETE | UPDATE_IN_PROGRESS | UPDATE_COMPLETE | UPDATE_FAILED

スケジューラ

metadata

スケジューラのメタデータ。

名前

スケジューラの名前。

タイプ: 文字列

version

スケジューラのバージョン。

タイプ: 文字列

type

スケジューラのタイプ。

タイプ: 文字列

region

クラスターが作成される AWS リージョン 。

タイプ: 文字列

version

クラスターの作成に使用される AWS ParallelCluster バージョン。

タイプ: 文字列

nextToken

次の結果セットの取得に使用できるトークン、または追加の結果がない場合は `null` です。

タイプ: 文字列

例

Python

リクエスト

```
$ list_clusters()
```

200 レスポンス

```
{
  'clusters':
   [
      {
        'cloudformation_stack_arn': 'arn:aws:cloudformation:us-
east-1:123456789012:stack/cluster_name_3x/16b49540-aee5-11ec-8e18-0ac1d712b241',
        'cloudformation_stack_status': 'CREATE_COMPLETE',
        'cluster_name': 'cluster_name_3x',
        'cluster_status': 'CREATE_COMPLETE',
        'region': 'us-east-1',
        'version': '3.2.1'
      },
      ...
   ]
}
```

listClusterLogStreams

クラスターに関連付けられているログストリームのリストを取得します。

トピック

- リクエスト構文
- リクエスト本文
- レスポンスの構文
- レスポンス本文
- 例

リクエスト構文

```
GET /v3/clusters/{clusterName}/logstreams
{
  "filters": [ "string" ],
  "nextToken": "string",
  "region": "string"
}
```

リクエスト本文

clusterName

クラスターの名前。

タイプ: 文字列

必須: はい

フィルター

ログストリームをフィルタリングします。

次のフィルターが使用できます。

- private-dns-name: インスタンスのプライベート DNS 名の短縮形 (例: ip-10-0-0-101)。
- node-type: 有効値: `HeadNode`。

型: 一意の文字列の配列

形式: `Name=a,Values=1 Name=b,Values=2,3`

必須: いいえ

nextToken

次の結果セットのトークン。

型: 文字列

必須: いいえ

region

クラスター AWS リージョン がある 。

型: 文字列

必須: いいえ

レスポンスの構文

```
{
  "nextToken": "string",
  "logStreams": [
    {
      "logStreamName": "string",
      "creationTime": "2019-08-24T14:15:22Z",
      "firstEventTimestamp": "2019-08-24T14:15:22Z",
      "lastEventTimestamp": "2019-08-24T14:15:22Z",
      "lastIngestionTime": "2019-08-24T14:15:22Z",
      "uploadSequenceToken": "string",
      "logStreamArn": "string"
    }
  ]
}
```

レスポンス本文

logStreams

ログストリームのリスト。

creationTime

ストリームが作成された時刻。

型: 日時

firstEventTimestamp

ストリームの最初のイベントの時刻。

型: 日時

lastEventTimestamp

ストリームの最後のイベントの時刻。lastEventTime 値は、最終的な整合性に基づいて更新されます。通常、取り込まれてから 1 時間以内に更新されますが、まれにそれ以上かかる場合もあります。

型: 日時

lastIngestionTime

最終取り込み時間。

型: 日時

logStreamArn

ログストリームの Amazon リソースネーム (ARN)。

タイプ: 文字列

logStreamName

ログストリームの名前。

タイプ: 文字列

uploadSequenceToken

シーケンストークン。

タイプ: 文字列

nextToken

次の結果セットの取得に使用できるトークン、または追加の結果がない場合は `null` です。

タイプ: 文字列

例

Python

リクエスト

```
$ list_cluster_log_streams(cluster_name_3x)
```

200 レスポンス

```
{
  'log_streams': [
      {
        'creation_time': datetime.datetime(2022, 3, 30, 14, 7, 34, 354000,
 tzinfo=tzlocal()),
        'first_event_timestamp': datetime.datetime(2022, 3, 30, 14, 6, 41, 444000,
 tzinfo=tzlocal()),
        'last_event_timestamp': datetime.datetime(2022, 3, 30, 14, 25, 55, 462000,
 tzinfo=tzlocal()),
        'last_ingestion_time': datetime.datetime(2022, 3, 30, 14, 49, 50, 62000,
 tzinfo=tzlocal()),
        'log_stream_arn': 'arn:aws:logs:us-east-1:123456789012:log-group:/aws/
parallelcluster/cluster_name_3x:log-stream:ip-192-0-2-26.i-abcdef01234567890.cfn-
init',
        'log_stream_name': 'ip-192-0-2-26.i-abcdef01234567890.cfn-init',
         ...
        'upload_sequence_token': '####'
      },
      ...
   ]
}
```

listImageLogStreams

イメージに関連付けられたログストリームのリストを取得します。

トピック

- リクエスト構文
- リクエスト本文
- レスポンスの構文

- レスポンス本文
- 例

リクエスト構文

```
GET /v3/images/custom/{imageId}/logstreams
{
  "nextToken": "string",
  "region": "string"
}
```

リクエスト本文

imageId

イメージの ID。

タイプ: 文字列

必須: はい

nextToken

次の結果セットのトークン。

型: 文字列

必須: いいえ

region

イメージ AWS リージョン がある 。

型: 文字列

必須: いいえ

レスポンスの構文

```
{
  "nextToken": "string",
```

```
  "logStreams": [
    {
      "logStreamName": "string",
      "creationTime": "2019-08-24T14:15:22Z",
      "firstEventTimestamp": "2019-08-24T14:15:22Z",
      "lastEventTimestamp": "2019-08-24T14:15:22Z",
      "lastIngestionTime": "2019-08-24T14:15:22Z",
      "uploadSequenceToken": "string",
      "logStreamArn": "string"
    }
  ]
}
```

レスポンス本文

logStreams

ログストリームのリスト。

creationTime

ストリームが作成された時刻。

型: 日時

firstEventTimestamp

ストリーム内の最初のイベントの時刻。

型: 日時

lastEventTimestamp

ストリームの最後のイベントの時刻。lastEventTime 値は、最終的な整合性に基づいて更新されます。通常、取り込まれてから 1 時間以内に更新されますが、まれにそれ以上かかる場合もあります。

型: 日時

lastIngestionTime

最終取り込み時間。

型: 日時

logStreamArn

ログストリームの Amazon リソースネーム (ARN)。

タイプ: 文字列

logStreamName

ログストリームの名前。

タイプ: 文字列

uploadSequenceToken

シーケンストークン。

タイプ: 文字列

next_token

次の結果セットの取得に使用できるトークン、または追加の結果がない場合は `null` です。

タイプ: 文字列

例

Python

リクエスト

```
$ list_image_log_streams(custom-image-id)
```

200 レスポンス

```
{
  'log_streams': [
    {
      'creation_time': datetime.datetime(2022, 3, 29, 20, 29, 24, 875000,
 tzinfo=tzlocal()),
      'first_event_timestamp': datetime.datetime(2022, 3, 29, 20, 29, 24, 775000,
 tzinfo=tzlocal()),
      'last_event_timestamp': datetime.datetime(2022, 3, 29, 20, 38, 23, 944000,
 tzinfo=tzlocal()),
```

```
      'last_ingestion_time': datetime.datetime(2022, 3, 29, 20, 51, 56, 26000,
 tzinfo=tzlocal()),
      'log_stream_arn': 'arn:aws:logs:us-east-1:123456789012:log-group:/aws/
imagebuilder/ParallelClusterImage-alinux2-image:log-stream:3.2.1/1',
      'log_stream_name': '3.2.1/1',
      'upload_sequence_token': '####'
    },
    ...
  ]
}
```

listImages

既存のカスタムイメージのリストを取得します。

トピック

- リクエスト構文
- リクエスト本文
- レスポンスの構文
- レスポンス本文
- 例

リクエスト構文

```
GET /images/custom
{
  "imageStatus": "string",
  "nextToken": "string",
  "region": "string"
}
```

リクエスト本文

imageStatus

指定されたステータスでイメージを絞り込みます。

タイプ: 文字列

有効な値: `AVAILABLE | PENDING | FAILED`

必須: はい

nextToken

次の結果セットのトークン。

型: 文字列

必須: いいえ

region

イメージ AWS リージョン がある 。

型: 文字列

必須: いいえ

レスポンスの構文

```
{
  "nextToken": "string",
  "images": [
    {
      "imageId": "string",
      "ec2AmiInfo": {
        "amiId": "string"
      },
      "region": "string",
      "version": "string",
      "cloudformationStackArn": "string",
      "imageBuildStatus": "BUILD_IN_PROGRESS",
      "cloudformationStackStatus": "CREATE_IN_PROGRESS"
    }
  ]
}
```

レスポンス本文

images

イメージのリスト。

cloudformationStackArn

メイン CloudFormation スタックの Amazon リソースネーム (ARN)。

タイプ: 文字列

cloudformationStackStatus

CloudFormation スタックの状態。

タイプ: 文字列

有効な値: `CREATE_IN_PROGRESS | CREATE_FAILED | CREATE_COMPLETE | ROLLBACK_IN_PROGRESS | ROLLBACK_FAILED | ROLLBACK_COMPLETE | DELETE_IN_PROGRESS | DELETE_FAILED | DELETE_COMPLETE | UPDATE_IN_PROGRESS | UPDATE_COMPLETE_CLEANUP_IN_PROGRESS | UPDATE_COMPLETE | UPDATE_ROLLBACK_IN_PROGRESS | UPDATE_ROLLBACK_FAILED | UPDATE_ROLLBACK_COMPLETE_CLEANUP_IN_PROGRESS | UPDATE_ROLLBACK_COMPLETE`

ec2AmiInfo

ami_id

Amazon EC2 AMI ID。

タイプ: 文字列

imageBuildStatus

イメージビルドのステータス。

有効な値: `BUILD_IN_PROGRESS | BUILD_FAILED | BUILD_COMPLETE | DELETE_IN_PROGRESS | DELETE_FAILED | DELETE_COMPLETE`

タイプ: 文字列

imageId

イメージの ID。

タイプ: 文字列

region

イメージが作成される AWS リージョン 。

タイプ: 文字列

version

イメージの構築に使用される AWS ParallelCluster バージョン。

タイプ: 文字列

nextToken

次の結果セットの取得に使用できるトークン、または追加の結果がない場合は `null` です。

タイプ: 文字列

例

Python

リクエスト

```
$ list_images("AVAILABLE")
```

200 レスポンス

```
{
  'images': [
    {
      'ec2_ami_info': {
        'ami_id': 'ami-abcdef01234567890'
      },
      'image_build_status': 'BUILD_COMPLETE',
      'image_id': 'custom-image',
      'region': 'us-east-1',
      'version': '3.2.1'
    }
  ]
}
```

listOfficialImages

AWS ParallelCluster 公式イメージのリストを取得します。

トピック

- リクエスト構文
- リクエスト本文
- レスポンスの構文
- レスポンス本文
- 例

リクエスト構文

```
GET /v3/images/official
{
  "architecture": "string",
  "os": "string",
  "region": "string"
}
```

リクエスト本文

architecture

アーキテクチャによるフィルタリング。デフォルトはフィルタリングなしです。

タイプ: 文字列

有効な値: `x86_64 | arm64`

必須：いいえ

os

OS ディストリビューションによるフィルタリング。デフォルトはフィルタリングなしです。

タイプ: 文字列

有効な値: `alinux2 | alinux2023 | ubuntu2404 | ubuntu2204 | ubuntu2004 | rhel8 | rhel9`

必須：いいえ

region

公式イメージ AWS リージョン がリストされている 。

型: 文字列

必須: いいえ

レスポンスの構文

```
{
  "images": [
    {
      "architecture": "string",
      "amiId": "string",
      "name": "string",
      "os": "string",
      "version": "string"
    }
  ]
}
```

レスポンス本文

images

amiId

AMI の ID。

タイプ: 文字列

architecture

AMI アーキテクチャ。

タイプ: 文字列

名前

AMI の名前。

タイプ: 文字列

os

AMI オペレーティングシステム。

タイプ: 文字列

version

AWS ParallelCluster バージョン。

タイプ: 文字列

例

Python

リクエスト

```
$ list_official_images()
```

200 レスポンス

```
{
  'images': [
    {
      'ami_id': 'ami-015cfeb4e0d6306b2',
      'architecture': 'x86_64',
      'name': 'aws-parallelcluster-3.2.1-ubuntu-2004-lts-hvm-x86_64-202202261505 '
      '2022-02-26T15-08-34.759Z',
      'os': 'ubuntu2004',
      'version': '3.2.1'
    },
    ...
  ]
}
```

updateCluster

クラスターを更新します。

トピック

- リクエスト構文
- リクエスト本文
- レスポンスの構文
- レスポンス本文

- 例

リクエスト構文

```
PUT /v3/clusters/{clusterName}
{
  "clusterConfiguration": "string",
  "dryrun": boolean,
  "forceUpdate": boolean,
  "region": "string",
  "suppressValidators": "string",
  "validationFailureLevel": "string"
}
```

リクエスト本文

clusterConfiguration

YAML ドキュメントとしてのクラスター設定。

必須: はい

clusterName

クラスターの名前。

タイプ: 文字列

必須: はい

dryrun

`true` に設定すると、リソースを作成することなく、リクエストの検証のみを行います。このパラメータを使用して、クラスター設定と更新の要件を検証します。デフォルトは `false` です。

型: ブール値

必須: いいえ

forceUpdate

`true` に設定すると、更新検証のエラーを無視して、強制的に更新を行います。デフォルトは `false` です。

型: ブール値

必須: いいえ

region

クラスター AWS リージョン がある 。

型: 文字列

必須: いいえ

suppressValidators

抑制する 1 つまたは複数の設定バリデータを指定します。

タイプ: 文字列

形式: `(ALL|type:[A-Za-z0-9]+)`

必須: いいえ

有効な値の例: `currentValue`、`requestedValue`、`message`

validationFailureLevel

更新が失敗する最小の検証レベル。

タイプ: 文字列

有効な値: `INFO | WARNING | ERROR`

必須 : いいえ

レスポンスの構文

```
{
  "cluster": {
    "clusterName": "string",
    "region": "string",
    "version": "string",
    "cloudformationStackArn": "string",
    "cloudformationStackStatus": "UPDATE_IN_PROGRESS",
    "clusterStatus": "UPDATE_IN_PROGRESS",
    "scheduler": {
```

```
      "type": "string",
      "metadata": {
        "name": "string",
        "version": "string"
      }
    }
  },
  "validationMessages": [
    {
      "id": "string",
      "type": "string",
      "level": "INFO",
      "message": "string"
    }
  ],
  "changeSet": [
    {
      "parameter": "string",
      "currentValue": "string",
      "requestedValue": "string"
    }
  ]
}
```

レスポンス本文

changeSet

クラスターアップデート用の変更セット。

currentValue

更新するパラメータの現在の値。

タイプ: 文字列

パラメータ

更新するパラメータ。

タイプ: 文字列

requestedValue

更新するパラメータの要求された値。

タイプ: 文字列

クラスター

cloudformationStackArn

メイン CloudFormation スタックの Amazon リソースネーム (ARN)。

タイプ: 文字列

cloudformationStackStatus

CloudFormation スタックの状態。

タイプ: 文字列

有効な値: `CREATE_IN_PROGRESS | CREATE_FAILED | CREATE_COMPLETE | ROLLBACK_IN_PROGRESS | ROLLBACK_FAILED | ROLLBACK_COMPLETE | DELETE_IN_PROGRESS | DELETE_FAILED | DELETE_COMPLETE | UPDATE_IN_PROGRESS | UPDATE_COMPLETE_CLEANUP_IN_PROGRESS | UPDATE_COMPLETE | UPDATE_ROLLBACK_IN_PROGRESS | UPDATE_ROLLBACK_FAILED | UPDATE_ROLLBACK_COMPLETE_CLEANUP_IN_PROGRESS | UPDATE_ROLLBACK_COMPLETE`

clusterName

クラスターの名前。

タイプ: 文字列

clusterStatus

クラスターのステータス。

タイプ: 文字列

有効な値: `CREATE_IN_PROGRESS | CREATE_FAILED | CREATE_COMPLETE | DELETE_IN_PROGRESS | DELETE_FAILED | DELETE_COMPLETE | UPDATE_IN_PROGRESS | UPDATE_COMPLETE | UPDATE_FAILED`

region

クラスターが作成される AWS リージョン 。

タイプ: 文字列

スケジューラ

metadata

スケジューラのメタデータ。

名前

スケジューラの名前。

タイプ: 文字列

version

スケジューラのバージョン。

タイプ: 文字列

type

スケジューラタイプ。

タイプ: 文字列

version

AWS ParallelCluster クラスターの作成に使用される バージョン。

タイプ: 文字列

validationMessages

検証レベルが `validationFailureLevel` 以下のメッセージのリスト。メッセージのリストは設定の検証中に収集されます。

id

バリデータの ID。

タイプ: 文字列

level

検証レベル。

タイプ: 文字列

有効な値: `INFO | WARNING | ERROR`

message

検証メッセージ。

タイプ: 文字列

type

バリデータのタイプ。

タイプ: 文字列

例

Python

リクエスト

```
$ update_cluster(cluster_name_3x, path/config-file.yaml)
```

200 レスポンス

```
{
  'change_set': [
    {
      'current_value': '10',
      'parameter':
 'Scheduling.SlurmQueues[queue1].ComputeResources[t2micro].MaxCount',
      'requested_value': '15'
    }
  ],
  'cluster': {
    'cloudformation_stack_arn': 'arn:aws:cloudformation:us-
east-1:123456789012:stack/test-api-cluster/e0462730-50b5-11ed-99a3-0a5ddc4a34c7',
    'cloudformation_stack_status': 'UPDATE_IN_PROGRESS',
    'cluster_name': 'cluster-3x',
    'cluster_status': 'UPDATE_IN_PROGRESS',
    'region': 'us-east-1',
    'scheduler': {
      'type': 'slurm'
    },
    'version': '3.2.1'
  }
```

```
}
```

updateComputeFleet

クラスターコンピューティングフリートの状態を更新します。

トピック

- リクエスト構文
- リクエスト本文
- レスポンスの構文
- レスポンス本文
- 例

リクエスト構文

```
PATCH /v3/clusters/{clusterName}/computefleet
{
  "status": "string",
  "region": "string"
}
```

リクエスト本文

clusterName

クラスターの名前。

タイプ: 文字列

必須: はい

ステータス

コンピューティングフリートのステータス。

タイプ: 文字列

有効な値: `START_REQUESTED | STOP_REQUESTED | ENABLED | DISABLED`

必須: はい

region

クラスター AWS リージョン がある 。

型: 文字列

必須: いいえ

レスポンスの構文

```
{
  "status": "START_REQUESTED",
  "lastStatusUpdatedTime": "2019-08-24T14:15:22Z"
}
```

レスポンス本文

ステータス

コンピューティングフリートのステータス。

タイプ: 文字列

有効な値: `START_REQUESTED | STARTING | RUNNING | PROTECTED | STOP_REQUESTED | STOPPING | STOPPED | UNKNOWN | ENABLED | DISABLED`

lastStatusUpdatedTime

ステータスの最終更新時刻を表すタイムスタンプ。

型: 日時

例

Python

リクエスト

```
$ update_compute_fleet(cluster_name_3x,  "START_REQUESTED")
```

200 レスポンス

```
{
  'last_status_updated_time': datetime.datetime(2022, 3, 28, 22, 27, 14,
 tzinfo=tzlocal()),
  'status': 'START_REQUESTED'
}
```

AWS ParallelCluster Python ライブラリ API

AWS ParallelCluster バージョン 3.5.0 以降では、Python AWS ParallelCluster ライブラリ AWS ParallelCluster を使用して にアクセスできます。`pcluster` 環境内または AWS Lambda ランタイム内から AWS ParallelCluster ライブラリにアクセスできます。Python ライブラリを使用して API AWS ParallelCluster にアクセスする AWS ParallelCluster 方法について説明します。Python ライブラリは、 AWS ParallelCluster API AWS ParallelCluster が提供するのと同じ機能を提供します。

AWS ParallelCluster Python ライブラリオペレーションとパラメータは、大文字なしで に変換すると、API パラメータのオペレーションとパラメータを反映し`snake_case`ます。

トピック

- AWS ParallelCluster Python ライブラリの認可
- Python AWS ParallelCluster ライブラリをインストールする
- クラスター API オペレーション
- コンピューティングフリートの API オペレーション
- クラスターとスタックのログ操作
- イメージの API オペレーション
- イメージとスタックのログ操作
- 例
- AWS Lambda Python AWS ParallelCluster ライブラリ用の

AWS ParallelCluster Python ライブラリの認可

boto3 で有効な標準的な方法のいずれかを使用して認証情報を指定します。詳細については、「Boto3 ドキュメント」を参照してください。

Python AWS ParallelCluster ライブラリをインストールする

1. セットアップ AWS ParallelCluster に記載されている手順に従って `pcluster` CLI バージョン 3.5.0 以降をインストールします。
2. 次の例に示すように、`pcluster` モジュールをインポートし、ライブラリの使用を開始します。

```
import pcluster.lib as pc
pc.create_cluster(cluster_name="mycluster", cluster_configuration="config.yaml"
```

クラスター API オペレーション

トピック

- list_clusters
- create_cluster
- delete_cluster
- describe_cluster
- update_cluster

`list_clusters`

```
list_clusters(region, next_token, cluster_status)
```

既存のクラスターのリストを取得します。

パラメータ：

`region`

特定の にデプロイされたクラスターを一覧表示します AWS リージョン。

`next_token`

次の結果セットのトークン。

`cluster_status`

クラスターのステータスでフィルタリングします。デフォルトではすべてのクラスターが一覧表示されます。

有効な値: CREATE_IN_PROGRESS | CREATE_FAILED | CREATE_COMPLETE | DELETE_IN_PROGRESS | DELETE_FAILED | UPDATE_IN_PROGRESS | UPDATE_COMPLETE | UPDATE_FAILED

create_cluster

```
create_cluster(cluster_name, cluster_configuration, region, suppress_validators,
 validation_failure_level, dry_run, rollback_on_failure, wait)
```

特定のリージョンにクラスターを作成します。

パラメータ :

cluster_name (必須)

クラスター名。

cluster_configuration (必須)

Python データ型としてのクラスター設定。

region

クラスター AWS リージョン。

suppress_validators

抑制する 1 つまたは複数のクラスター設定バリデータを指定します。

形式: (ALL | type:[A-Za-z0-9]+)

validation_failure_level

クラスター作成が失敗する最小の検証レベル。デフォルトは ERROR です。

有効な値: INFO | WARNING | ERROR

dry_run

リソースを作成することなく、リクエストの検証を行います。これを使用して、クラスターの設定を検証できます。デフォルトは False です。

rollback_on_failure

に設定するとTrue、障害発生時にクラスタースタックのロールバック AWS ParallelCluster が自動的に開始されます。デフォルトは True です。

wait

True に設定すると、 AWS ParallelCluster はオペレーションが完了するまで待機します。デフォルトは False です。

delete_cluster

```
delete_cluster(cluster_name, region, wait)
```

特定のリージョンのクラスターを削除します。

パラメータ :

cluster_name (必須)

クラスター名。

region

クラスター AWS リージョン。

wait

True に設定すると、オペレーションが完了するまで待機します。デフォルトは False です。

describe_cluster

```
describe_cluster(cluster_name, region)
```

既存のクラスターの詳細情報を取得します。

パラメータ :

cluster_name (必須)

クラスター名。

region

クラスター AWS リージョン。

update_cluster

```
update_cluster(cluster_name, cluster_configuration, suppress_validators,
 validation_failure_level, region, force_update, dry_run, wait)
```

特定のリージョンのクラスターを更新します。

パラメータ :

cluster_name (必須)

クラスター名。

cluster_configuration (必須)

Python データ型としてのクラスター設定。

suppress_validators

抑制する 1 つまたは複数のクラスター設定バリデータを指定します。

形式: `(ALL | type:[A-Za-z0-9]+)`

validation_failure_level

クラスターの更新が失敗する最小の検証レベル。デフォルトは `ERROR` です。

有効な値: `INFO` | `WARNING` | `ERROR`

region

クラスター AWS リージョン。

dry_run

リソースの作成や更新を行わずにリクエストの検証のみを行います。これを使用して、クラスターの設定を検証できます。デフォルトは `False` です。

force_update

`True` に設定すると、更新検証のエラーを無視して、強制的に更新を行います。デフォルトは `False` です。

wait

`True` に設定すると、オペレーションが完了するまで待機します。デフォルトは `False` です。

コンピューティングフリートの API オペレーション

トピック

- describe_compute_fleet
- update_compute_fleet
- delete_cluster_instances
- describe_cluster_instances

describe_compute_fleet

```
describe_compute_fleet(cluster_name, region)
```

特定のクラスターのクラスターコンピューティングフリートのステータスを説明します。

パラメータ：

cluster_name (必須)

クラスター名。

region

特定の にデプロイされたクラスターのコンピューティングフリートステータスについて説明します AWS リージョン。

update_compute_fleet

```
update_compute_fleet(cluster_name, status, region)
```

クラスターコンピューティングフリートの状態を更新します。

パラメータ：

cluster_name (必須)

クラスター名。

status (必須)

更新先のステータス。

有効な値: START_REQUESTED | STOP_REQUESTED | ENABLED | DISABLED

region

クラスター AWS リージョン。

delete_cluster_instances

```
delete_cluster_instances(cluster_name, region, force)
```

すべてのクラスターコンピューティングノードの強制終了を開始します。このアクションは AWS Batch クラスターをサポートしていません。

パラメータ :

cluster_name (必須)

クラスター名。

region

クラスター AWS リージョン。

force

True に設定すると、指定した cluster_name を持つクラスターが見つからない場合に強制的に削除します。デフォルトは False です。

describe_cluster_instances

```
describe_cluster_instances(cluster_name, region, next_token, node_type, queue_name)
```

クラスターのインスタンスについて説明します。

パラメータ :

cluster_name (必須)

クラスター名。

region

クラスター AWS リージョン。

next_token

次の結果セットのトークン。

node_type

node_type でインスタンスをフィルタリングします。

有効な値: HeadNode | ComputeNode

queue_name

インスタンスをキュー名でフィルタリングします。

クラスターとスタックのログ操作

トピック

- list_cluster_log_streams
- get_cluster_log_events
- get_cluster_stack_events

list_cluster_log_streams

```
list_cluster_log_streams(cluster_name, region, filters, next_token)
```

特定のクラスターのログストリームを一覧表示します。

パラメータ :

cluster_name (必須)

クラスター名。

region

クラスター AWS リージョン。

filters

クラスターログストリームをフィルタリングします。

形式: 'Name=a,Values=1 Name=b,Values=2,3'

使用できるフィルター:

code-dns-name

インスタンスのプライベート DNS 名の短縮形 (例: `ip-10-0-0-101`)。

node-type

ノードタイプ。

有効な値: `HeadNode`

next_token

次の結果セットのトークン。

get_cluster_log_events

```
get_cluster_log_events(cluster_name, log_stream_name, region, next_token,
 start_from_head, limit, start_time, end_time)
```

特定のクラスターとログストリームのログイベントを取得します。

パラメータ :

cluster_name (必須)

クラスター名。

log_stream_name (必須)

ログストリーム名。

region

クラスター AWS リージョン。

next_token

次の結果セットのトークン。

start_from_head

に設定すると`True`、 は最も古いログイベントを最初に AWS ParallelCluster 返します。`False` に設定すると、最新のログイベントが最初に返されます。デフォルトは `False` です。

limit

返されるログイベントの最大数。値を指定しない場合、最大値は 1 MB の応答サイズに収まるログの数で、最大 10,000 ログイベントまでとなります。

start_time

ISO 8601 形式で表したログイベントの時間範囲の開始時刻 (例: `'2021-01-01T20:00:00Z'`)。この時間と同じかそれ以降のタイムスタンプを持つイベントが含まれます。

end_time

ISO 8601 形式で表したログイベントの時間範囲の終了時刻 (例: `'2021-01-01T20:00:00Z'`)。この時間と同じかそれ以降のタイムスタンプを持つイベントは含まれません。

get_cluster_stack_events

```
get_cluster_stack_events(cluster_name, region, next_token)
```

特定のクラスターのスタックイベントを取得します。

パラメータ :

cluster_name (必須)

クラスター名。

region

クラスター AWS リージョン。

next_token

次の結果セットのトークン。

イメージの API オペレーション

トピック

- list_images
- build_image

- delete_image
- describe_image

list_images

```
list_images(image_status, region, next_token)
```

既存のイメージのリストを取得します。

パラメータ :

image_status (必須)

イメージのステータスでフィルタリングします。

有効な値: AVAILABLE | PENDING | FAILED

region

特定の AWS リージョンにビルドされているイメージを一覧表示します。

next_token

次の結果セットのトークン。

build_image

```
build_image(image_configuration, image_id, suppress_validators,
 validation_failure_level, dry_run, rollback_on_failure, region)
```

特定のリージョンにカスタム AWS ParallelCluster イメージを作成します。

パラメータ :

image_configuration (必須)

Python データとしてのイメージ設定。

image_id (必須)

イメージ ID。

suppress_validators

抑制する 1 つまたは複数のイメージ設定バリデータを指定します。

形式: `(ALL | type:[A-Za-z0-9]+)`

validation_failure_level

イメージ作成が失敗する最小の検証レベル。デフォルトは `ERROR` です。

有効な値: `INFO` | `WARNING` | `ERROR`

dry_run

に設定すると`True`、 はリソースを作成せずにリクエストの検証 AWS ParallelCluster を実行します。イメージの設定を検証するために使用することができます。デフォルトは `False` です。

rollback_on_failure

に設定すると`True`、障害発生時にイメージスタックのロールバック AWS ParallelCluster が自動的に開始されます。デフォルトは `False` です。

region

AWS リージョンイメージ。

delete_image

```
delete_image(image_id, region, force)
```

特定のリージョンのイメージを削除します。

パラメータ :

image_id (必須)

イメージ ID。

region

AWS リージョンイメージ。

force

に設定すると`True`、インスタンスが AMI を使用している場合、または AMI が共有されている場合、 は AWS ParallelCluster 強制的に削除します。デフォルトは `False` です。

describe_image

```
describe_image(image_id, region)
```

既存のイメージの詳細情報を取得します。

パラメータ :

image_id (必須)

イメージ ID。

region

AWS リージョンイメージ。

イメージとスタックのログ操作

トピック

- list_image_log_streams
- get_image_log_events
- get_image_stack_events
- list_official_images

list_image_log_streams

```
list_image_log_streams(image_id, region, next_token)
```

イメージのログストリームを一覧表示します。

パラメータ :

image_id (必須)

イメージ ID。

region

AWS リージョンイメージ。

next_token

次の結果セットのトークン。

get_image_log_events

```
get_image_log_events(image_id, log_stream_name, region, next_token, start_from_head,
 limit, start_time, end_time)
```

特定のイメージとログストリームのログイベントを取得します。

パラメータ :

image_id (必須)

イメージ ID。

log_stream_name (必須)

ログストリーム名。

region

AWS リージョンイメージ。

next_token

次の結果セットのトークン。

start_from_head

に設定すると`True`、 は最も古いログイベントを最初に AWS ParallelCluster 返します。`False` に設定すると、最新のログイベントが最初に返されます。デフォルトは `False` です。

limit

返されるログイベントの最大数。値を指定しない場合、最大値は 1 MB の応答サイズに収まるログの数で、最大 10,000 ログイベントまでとなります。

start_time

ISO 8601 形式で表したログイベントの時間範囲の開始時刻 (例: `'2021-01-01T20:00:00Z'`)。この時間と同じかそれ以降のタイムスタンプを持つイベントが含まれます。

end_time

ISO 8601 形式で表したログイベントの時間範囲の終了時刻 (例: `'2021-01-01T20:00:00Z'`)。この時間と同じかそれ以降のタイムスタンプを持つイベントは含まれません。

get_image_stack_events

```
get_image_stack_events(image_id, region, next_token)
```

特定のイメージのスタックイベントを取得します。

パラメータ :

image_id (必須)

イメージ ID。

region

AWS リージョンイメージ。

next_token

次の結果セットのトークン。

list_official_images

```
list_official_images(region,os, architecture)
```

公式 AWS ParallelCluster イメージのリストを取得します。

パラメータ :

region

AWS リージョンイメージ。

os

オペレーティングシステムのディストリビューションでフィルタリングします。デフォルトはフィルタリングなしです。

architecture

アーキテクチャでフィルタリングします。デフォルトはフィルタリングなしです。

例

トピック

- クラスターを作成する

クラスターを作成する

環境に保存されている特定の入力を使用して次のサンプルスクリプトを実行すると、クラスターが作成されます。クラスター設定は、クラスター設定ドキュメントに基づいて Python データ型として作成されます。

```
import os
import pprint
import pcluster.lib as pc
pp = pprint.PrettyPrinter()

HEAD_NODE_SUBNET = os.environ["HEAD_NODE_SUBNET"]
COMPUTE_NODE_SUBNET = os.environ["HEAD_NODE_SUBNET"]
KEY_NAME = os.environ["KEY_NAME"]
CONFIG = {'Image': {'Os': 'alinux2'},
          'HeadNode': {'InstanceType': 't2.large',
                       'Networking': {'SubnetId': HEAD_NODE_SUBNET},
                       'Ssh': {'KeyName': KEY_NAME}},

          'Scheduling': {'Scheduler': 'slurm',
                         'SlurmQueues':
                         [{'Name': 'queue0',
                           'ComputeResources':
                           [{'Name': 'queue0-i0', 'InstanceType': 't2.micro',
                             'MinCount': 0, 'MaxCount': 10}],
                           'Networking': {'SubnetIds': [COMPUTE_NODE_SUBNET]}}]}}


pp.pprint(pc.create_cluster(cluster_name="mycluster", cluster_configuration=CONFIG))
```

出力:

```
{'cluster': {'cloudformationStackArn': 'arn:aws:cloudformation:us-
east-2:123456789012:stack/mycluster/00000000-aaaa-1111-999-000000000000',
             'cloudformationStackStatus': 'CREATE_IN_PROGRESS',
             'clusterName': 'mycluster',
             'clusterStatus': 'CREATE_IN_PROGRESS',
             'region': 'us-east-2',
             'scheduler': {'type': 'slurm'},
             'version': '3.13.2'}}
```

AWS Lambda Python AWS ParallelCluster ライブラリ用の

Lambda レイヤーとランタイムをデプロイして Python AWS ParallelCluster ライブラリにアクセスできます。以下のステップで説明するように、 AWS ParallelCluster zip ファイルへのリンクを入力して使用できる zip ファイルをホストします。Lambda は zip ファイルを使用して Python ライブラリへのアクセスをサポートするランタイム環境を準備します。Python AWS ParallelCluster ライブラリが AWS ParallelCluster バージョン 3.5.0 で追加されました。このライブラリはバージョン 3.5.0 以降でのみ使用できます。

ホストされた zip ファイル URL の形式は ですs3:*//aws-region-id*-aws-parallelcluster/parallelcluster/*3.13.2*/layers/aws-parallelcluster/lambda-layer.zip。（*3.13.2* を次のステップで使用する AWS ParallelCluster バージョンに置き換えます）。

で Python AWS ParallelCluster ライブラリへのアクセスを開始する AWS Lambda

Lambda レイヤーを作成する

1. にログイン AWS Management Console し、 AWS Lambda コンソールに移動します。
2. ナビゲーションペインで [レイヤー] を選択し、[レイヤーの作成] を選択します。
3. レイヤーの名前を入力し、[Amazon S3 からファイルをアップロードする] を選択します。
4. zip ファイルへの URL を入力します。s3:*//aws-region-id*-aws-parallelcluster/parallelcluster/*3.13.2*/layers/aws-parallelcluster/lambda-layer.zip。
5. [互換性のあるアーキテクチャ] で、[x86_64] アーキテクチャを選択します。
6. 互換性のあるランタイムの場合は、Python 3.12 ランタイムを選択します。
7. [作成] を選択します。

Lambda レイヤーを使用する

1. Lambda コンソールのナビゲーションペインで、[関数] を選択し、[関数の作成] を選択します。

2. 関数の名前を入力します。
3. Runtime で、Python 3.12 ランタイムを選択します。
4. [アーキテクチャ] で、[x86_64] アーキテクチャを選択します。
5. [関数の作成] を選択してください。
6. 関数が作成されたら、[レイヤー] を選択し、[レイヤーの追加] を選択します。
7. [カスタムレイヤー] を選択し、前のステップで作成したレイヤーを選択します。
8. レイヤーバージョンを選択します。
9. [Add] (追加) を選択します。
10. Lambda には、 で作成されたクラスターを管理するためのアクセス許可が必要です AWS ParallelCluster。AWS ParallelCluster `pcluster` 基本ユーザーポリシー に一覧表示されているアクセス許可を使用して Lambda ロールを作成します。

「」で説明されているように、Python ライブラリ AWS ParallelCluster から にアクセスできるようになりましたAWS ParallelCluster Python ライブラリ API。

の使用方法に関するチュートリアル AWS ParallelCluster

以下のチュートリアルでは、 AWS ParallelCluster バージョン 3 の使用を開始する方法を示し、いくつかの一般的なタスクのベストプラクティスガイダンスを提供します。

AWS ParallelCluster コマンドラインインターフェイス (CLI) または API を使用する場合、 AWS ParallelCluster イメージとクラスターを作成または更新したときに作成された AWS リソースに対してのみ料金が発生します。詳細については、「AWS が使用する サービス AWS ParallelCluster」を参照してください。

トピック

- で最初のジョブを実行する AWS ParallelCluster
- カスタム AMI AWS ParallelCluster の構築
- Active Directory の統合
- AWS KMS キーを使用した共有ストレージ暗号化の設定
- マルチキューモードのクラスターでジョブを実行する
- AWS ParallelCluster API の使用
- Slurm アカウンティングによるクラスターの作成
- 外部 Slurmdbd アカウンティングによるクラスターの作成
- 以前の AWS Systems Manager ドキュメントバージョンに戻す
- を使用したクラスターの作成 AWS CloudFormation
- Terraform を使用して ParallelCluster API をデプロイする
- Terraform を使用したクラスターの作成
- Terraform を使用したカスタム AMI の作成
- AWS ParallelCluster UI と Identity Center の統合
- Pyxis でコンテナ化されたジョブを実行する
- EFA 対応 FSx Lustre を使用したクラスターの作成

で最初のジョブを実行する AWS ParallelCluster

このチュートリアルでは、 で最初の Hello World ジョブを実行する方法について説明します。 AWS ParallelCluster

AWS ParallelCluster コマンドラインインターフェイス (CLI) または API を使用する場合、 AWS ParallelCluster イメージとクラスターを作成または更新したときに作成された AWS リソースに対してのみ料金が発生します。詳細については、「AWS が使用する サービス AWS ParallelCluster」を参照してください。

前提条件

- AWS ParallelCluster がインストールされます。
- AWS CLI がインストールされ、設定されています。
- Amazon EC2 のキーペアがある。
- `pcluster` CLI の実行に必要なアクセス許可を持つ IAM ロールがある。

インストールを確認する

まず、Node.js の依存関係を含む AWS ParallelCluster が正しくインストールおよび設定されていることを確認します。

```
$ node --version
v16.8.0
$ pcluster version
{
  "version": "3.13.2"
}
```

これにより、実行中のバージョンが返されます AWS ParallelCluster。

初めてクラスターを作成する

では、最初のクラスターを作成していきましょう。このチュートリアルのワークロードのパフォーマンス負荷は高くないため、デフォルトのインスタンスサイズ `t2.micro` を使います。(本稼働ワークロードの場合は、ニーズに最適なインスタンスサイズを選択します) クラスター `hello-world` を呼び出してみましょう。

```
$ pcluster create-cluster \
    --cluster-name hello-world \
    --cluster-configuration hello-world.yaml
```

> **Note**
>
> AWS リージョン 使用する は、ほとんどのpclusterコマンドで指定する必要があります。AWS_DEFAULT_REGION 環境変数、~/.aws/config ファイルの [default] セクションにある region 設定で指定されていない場合は、pcluster コマンドラインで --region パラメータを指定する必要があります。

設定に関するメッセージが出力に表示されたら、 AWS ParallelClusterを設定するために次のコマンドを実行する必要があります。

```
$ pcluster configure --config hello-world.yaml
```

pcluster create-cluster コマンドが正常に完了した場合は、次のような出力が表示されます。

```
{
  "cluster": {
    "clusterName": "hello-world",
    "cloudformationStackStatus": "CREATE_IN_PROGRESS",
    "cloudformationStackArn": "arn:aws:cloudformation:xxx:stack/xxx",
    "region": "...",
    "version": "...",
    "clusterStatus": "CREATE_IN_PROGRESS"
  }
}
```

以下の方法でクラスターの作成をモニタリングします。

```
$ pcluster describe-cluster --cluster-name hello-world
```

クラスターの作成中に clusterStatus が「CREATE_IN_PROGRESS」とレポートします。クラスターの作成に成功すると、clusterStatus は「CREATE_COMPLETE」に遷移します。また、出力には、ヘッドノードの publicIpAddress と privateIpAddress が表示されます。

ヘッドノードにログインする

OpenSSH pem ファイルを使用してヘッドノードにログインします。

```
$ pcluster ssh --cluster-name hello-world -i /path/to/keyfile.pem
```

ログインしたら、sinfo コマンドを実行して、コンピューティングノードがセットアップおよび設定されていることを確認します。

```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
queue1*      up   infinite     10  idle~ queue1-dy-queue1t2micro-[1-10]
```

出力には、クラスターに 1 つのキューがあり、最大 10 のノードがあることが示されます。

Slurm を使用して最初のジョブを実行する

次に、しばらくの間スリープしてから、独自のホスト名を出力するジョブを作成します。hellojob.sh というファイルを次の内容で作成します。

```
#!/bin/bash
sleep 30
echo "Hello World from $(hostname)"
```

次に、sbatch を使用してジョブを送信し、実行されることを確認します。

```
$ sbatch hellojob.sh
Submitted batch job 2
```

これで、キューを表示してジョブのステータスを確認できます。新しい Amazon EC2 インスタンスのプロビジョニングがバックグランドで開始されます。クラスターインスタンスのステータスは、sinfo コマンドでモニタリングできます。

```
$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                 2    queue1 hellojob ec2-user CF       3:30      1 queue1-dy-
queue1t2micro-1
```

出力には、ジョブが queue1 に送信されたことが示されます。ジョブが終了するまで 30 秒間待ってから、もう一度 squeue を実行します。

```
$ squeue
```

```
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
```

キューにはジョブがないため、現在のディレクトリで出力を確認できます。

```
$ ls -l
total 8
-rw-rw-r-- 1 ec2-user ec2-user 57 Sep  1 14:25 hellojob.sh
-rw-rw-r-- 1 ec2-user ec2-user 43 Sep  1 14:30 slurm-2.out
```

出力には、「out」ファイルが示されます。ジョブからの出力を確認できます。

```
$ cat slurm-2.out
Hello World from queue1-dy-queue1t2micro-1
```

また、出力には、ジョブがインスタンス queue1-dy-queue1t2micro-1 上で正常に実行されていることも示されています。

作成したばかりのクラスターでは、ホームディレクトリのみがクラスターの全ノード間で共有されます。

クラスターの作成と使用の詳細については、「ベストプラクティス」を参照してください。

アプリケーションで共有ソフトウェア、ライブラリ、またはデータが必要な場合は、以下のオプションを検討してください。

- 「」で説明されているように、ソフトウェアを含む AWS ParallelCluster 有効なカスタム AMI を構築しますカスタム AMI AWS ParallelCluster の構築。
- AWS ParallelCluster 設定ファイルの StorageSettings オプションを使用して共有ファイルシステムを指定し、インストールしたソフトウェアを指定されたマウント場所に保存します。
- カスタムブートストラップアクション を使用して、クラスターの各ノードのブートストラップ手順を自動化します。

カスタム AMI AWS ParallelCluster の構築

AWS ParallelCluster コマンドラインインターフェイス (CLI) または API を使用する場合、 AWS ParallelCluster イメージとクラスターを作成または更新したときに作成された AWS リソースに対してのみ料金が発生します。詳細については、「AWS が使用する サービス AWS ParallelCluster」を参照してください。

> **Important**
>
> カスタム AMI を構築した場合は、新しい AWS ParallelCluster のリリースごとに、カスタム AMI を作成したときの手順を繰り返す必要があります。

この先をお読みになる前に、まず「カスタムブートストラップアクション」のセクションを確認することをお勧めします。希望する変更がスクリプト化され、今後の AWS ParallelCluster リリースでサポートされるかどうかを確認してください。

一般的にカスタム AMI の構築は理想的ではありませんが、 用のカスタム AMI の構築 AWS ParallelCluster が必要な特定のシナリオがあります。このチュートリアルでは、これらのシナリオに対応するカスタム AMI を構築する方法について説明します。

前提条件

- AWS ParallelCluster がインストールされます。
- AWS CLI がインストールされ、設定されています。
- Amazon EC2 のキーペアがある。
- `pcluster` CLI を実行してイメージをビルドするのに必要なアクセス許可を持つ IAM ロールがある。

AMI AWS ParallelCluster をカスタマイズする方法

カスタム AMI を構築するには、2 AWS ParallelCluster つの方法があります。これら 2 つの方法の 1 つは、 CLI を使用して新しい AMI AWS ParallelCluster を構築することです。もう一つの方法では、手動で変更を加えて AWS アカウントで使用できる新しい AMI を構築する必要があります。

カスタム AMI AWS ParallelCluster を構築する

カスタマイズされた AMI とソフトウェアがある場合は、その AWS ParallelCluster 上に に必要な変更を適用できます。 AWS ParallelCluster は EC2 Image Builder サービスを使用してカスタマイズされた AMIs。詳細については、「Image Builder User Guide」を参照してください。

キーポイント:

- このプロセスには約 1 時間かかります。この時間は、ビルド時に追加で `Build`/`Components` をインストールする場合には変動します。

- AMI には主なコンポーネントのバージョンがタグ付けされています。これらには、カーネル、スケジューラー、EFA ドライバーが含まれます。コンポーネントバージョンのサブセットも AMI の説明にレポートされます。
- AWS ParallelCluster 3.0.0 以降では、新しい CLI コマンドセットを使用してイメージのライフサイクルを管理できます。これには `build-image`、`list-images`、`describe-image`、および `delete-image` があります。
- この方法は繰り返し実行可能です。再実行して AMI を最新の状態に保ち (OS の更新など)、既存のクラスターを更新するときに使用できます。

> **Note**
>
> AWS 中国パーティションでこの方法を使用すると、ネットワークエラーが発生する可能性があります。例えば、GitHub または OS リポジトリからパッケージをダウンロードする際に、`pcluster build-image` コマンドでこれらのエラーが表示される場合があります。このような場合には、次の代替方法のいずれかを使用することをお勧めします。
>
> 1. このコマンドをバイパスする「AMI AWS ParallelCluster の変更」アプローチに従ってください。
> 2. イメージを `us-east-1` などの別のパーティションとリージョンに構築し、保存/復元して中国リージョンに移動します。詳細については、「Amazon EC2 ユーザーガイド」の「S3 を使用して AMI を保存および復元する」を参照してください。

手順:

1. AWS ParallelCluster クライアントがユーザーに代わって AWS API オペレーションを呼び出すことができるように AWS アカウント 認証情報を設定します。必要なアクセス権限のリストについては、「AWS Identity and Access Management での アクセス許可 AWS ParallelCluster」を参照してください。
2. 基本的なビルドイメージ設定ファイルを作成します。これを行うには、イメージの構築に使用する `InstanceType` と、`ParentImage` を指定します。これらは AMI を作成するための開始点として使用されます。オプションのビルドパラメータの詳細については、「Image Configuration」を参照してください。

```
Build:
 InstanceType: <BUILD_INSTANCE_TYPE>
 ParentImage: <BASE_AMI_ID>
```

3. CLI コマンドを使用して`pcluster build-image`、ベースとして指定した AWS ParallelCluster AMI から AMI を構築します。

```
$ pcluster build-image --image-id IMAGE_ID --image-configuration IMAGE_CONFIG.yaml --
region REGION
    {
 "image": {
   "imageId": "IMAGE_ID",
   "imageBuildStatus": "BUILD_IN_PROGRESS",
   "cloudformationStackStatus": "CREATE_IN_PROGRESS",
   "cloudformationStackArn": "arn:aws:cloudformation:us-east-1:123456789012:stack/
IMAGE_ID/abcd1234-ef56-gh78-ij90-1234abcd5678",
   "region": "us-east-1",
   "version": "3.13.2"
 }
}
```

> **Warning**
>
> `pcluster build-image` はデフォルトの VPC を使用します。 AWS Control Tower または AWS Landing Zone を使用してデフォルトの VPC を削除する場合は、イメージ設定ファイルでサブネット ID を指定する必要があります。詳細については、「`SubnetId`」を参照してください。

その他のパラメータのリストについては、「`pcluster build-image`」コマンドリファレンスページを参照してください。上記のコマンドの結果は次のとおりです。

- イメージの構成に基づいて、CloudFormation スタックが作成されます。このスタックには、ビルドに必要なすべての EC2 Image Builder リソースが含まれています。
- 作成されたリソースには、カスタム Image Builder AWS ParallelCluster コンポーネントを追加できる公式の Image Builder コンポーネントが含まれています。詳細については、EC2 Image Builder ユーザーガイドの「Image Builder を使用してカスタムコンポーネントを作成する」を参照してください。 EC2
- EC2 Image Builder はビルドインスタンスを起動し、 AWS ParallelCluster クックブックを適用して AWS ParallelCluster ソフトウェアスタックをインストールし、必要な設定タスクを実行します。 AWS ParallelCluster クックブックは、ビルドとブートストラップに使用されます AWS ParallelCluster。
- インスタンスが停止され、そこから新しい AMI が作成されます。

- 新たに作成した AMI から別のインスタンスが起動します。テストフェーズでは、EC2 Image Builder は Image Builder コンポーネントで定義されたテストを実行します。
- ビルドが成功すると、スタックは削除されます。ビルドに失敗した場合、スタックはインスペクション可能な状態で保持されます。

4. 次のコマンドを実行すると、ビルドプロセスのステータスをモニタリングできます。ビルドが完了したら、ビルドを実行してレスポンスで指定された AMI ID を取得できます。

```
$ pcluster describe-image --image-id IMAGE_ID --region REGION

# BEFORE COMPLETE
{
 "imageConfiguration": {
   "url": "https://parallelcluster-1234abcd5678efgh-v1-do-not-
delete.s3.amazonaws.com/parallelcluster/3.13.2/images/IMAGE_ID-abcd1234efgh5678/
configs/image-config.yaml?...",
 },
 "imageId": "IMAGE_ID",
 "imagebuilderImageStatus": "BUILDING",
 "imageBuildStatus": "BUILD_IN_PROGRESS",
 "cloudformationStackStatus": "CREATE_IN_PROGRESS",
 "cloudformationStackArn": "arn:aws:cloudformation:us-east-1:123456789012:stack/
IMAGE_ID/abcd1234-ef56-gh78-ij90-1234abcd5678",
 "region": "us-east-1",
 "version": "3.13.2",
 "cloudformationStackTags": [
   {
     "value": "3.13.2",
     "key": "parallelcluster:version"
   },
   {
     "value": "IMAGE_ID",
     "key": "parallelcluster:image_name"
   },
   ...
 ],
 "imageBuildLogsArn": "arn:aws:logs:us-east-1:123456789012:log-group:/aws/
imagebuilder/ParallelClusterImage-IMAGE_ID",
 "cloudformationStackCreationTime": "2022-04-05T21:36:26.176Z"
}

# AFTER COMPLETE
{
```

```
 "imageConfiguration": {
   "url": "https://parallelcluster-1234abcd5678efgh-v1-do-not-delete.s3.us-
east-1.amazonaws.com/parallelcluster/3.13.2/images/IMAGE_ID-abcd1234efgh5678/configs/
image-config.yaml?Signature=..."
 },
 "imageId": "IMAGE_ID",
 "imageBuildStatus": "BUILD_COMPLETE",
 "region": "us-east-1",
 "ec2AmiInfo": {
     "amiName": "IMAGE_ID 2022-04-05T21-39-24.020Z",
     "amiId": "ami-1234stuv5678wxyz",
     "description": "AWS ParallelCluster AMI for alinux2,
 kernel-4.14.238-182.422.amzn2.x86_64, lustre-2.10.8-5.amzn2.x86_64,
 efa-1.13.0-1.amzn2.x86_64, dcv-2021.1.10598-1.el7.x86_64, slurm-20-11-8-1",
     "state": "AVAILABLE",
     "tags": [
      {
        "value": "2021.3.11591-1.el7.x86_64",
        "key": "parallelcluster:dcv_version"
      },
      ...
     ],
   "architecture": "x86_64"
 },
 "version": "3.13.2"
}
```

5. クラスターを作成するには、クラスター設定内の `CustomAmi` フィールドに AMI ID を入力します。

AMI 作成プロセスのトラブルシューティングとモニタリング

イメージの作成は約 1 時間で完了します。`pcluster describe-image` コマンドまたはログ検索コマンドを実行してプロセスをモニタリングできます。

```
$ pcluster describe-image --image-id IMAGE_ID --region REGION
```

`build-image` コマンドは、イメージの構築に必要なすべての Amazon EC2 リソースを含む CloudFormation スタックを作成し、EC2 Image Builder プロセスを起動します。

`build-image` コマンドを実行した後、`pcluster get-image-stack-events` を使用して CloudFormation スタックイベントを取得できます。`--query` パラメータで結果をフィルタリングし

て、最新のイベントを表示できます。詳細については、「 AWS Command Line Interface ユーザーガイド」の AWS CLI 「出力のフィルタリング」を参照してください。

```
$ pcluster get-image-stack-events --image-id IMAGE_ID --region REGION --query
 "events[0]"
{
 "eventId": "ParallelClusterImage-CREATE_IN_PROGRESS-2022-04-05T21:39:24.725Z",
 "physicalResourceId": "arn:aws:imagebuilder:us-east-1:123456789012:image/
parallelclusterimage-IMAGE_ID/3.13.2/1",
 "resourceStatus": "CREATE_IN_PROGRESS",
 "resourceStatusReason": "Resource creation Initiated",
 "resourceProperties": "{\"InfrastructureConfigurationArn\":
\"arn:aws:imagebuilder:us-east-1:123456789012:infrastructure-configuration/
parallelclusterimage-abcd1234-ef56-gh78-ij90-1234abcd5678\",\"ImageRecipeArn\":
\"arn:aws:imagebuilder:us-east-1:123456789012:image-recipe/parallelclusterimage-
IMAGE_ID/3.13.2\",\"DistributionConfigurationArn\":\"arn:aws:imagebuilder:us-
east-1:123456789012:distribution-configuration/parallelclusterimage-abcd1234-ef56-
gh78-ij90-1234abcd5678\",\"Tags\":{\"parallelcluster:image_name\":\"IMAGE_ID\",
\"parallelcluster:image_id\":\"IMAGE_ID\"}}",
 "stackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/IMAGE_ID/abcd1234-
ef56-gh78-ij90-1234abcd5678",
 "stackName": "IMAGE_ID",
 "logicalResourceId": "ParallelClusterImage",
 "resourceType": "AWS::ImageBuilder::Image",
 "timestamp": "2022-04-05T21:39:24.725Z"
}
```

約 15 分後、Image Builder の作成に関連するログイベントエントリにスタックイベントが表示されます。pcluster list-image-log-streams コマンドや pcluster get-image-log-events コマンドを使用して、イメージログストリームの一覧表示および Image Builder のステップのモニタリングができるようになりました。

```
$ pcluster list-image-log-streams --image-id IMAGE_ID --region REGION \
    --query 'logStreams[*].logStreamName'

 "3.13.2/1"
]

$ pcluster get-image-log-events --image-id IMAGE_ID --region REGION \
--log-stream-name 3.13.2/1 --limit 3
{
 "nextToken": "f/36295977202298886557255241372854078762600452615936671762",
```

```
 "prevToken": "b/36295977196879805474012299949460899222346900769983430672",
 "events": [
   {
     "message": "ExecuteBash: FINISHED EXECUTION",
     "timestamp": "2022-04-05T22:13:26.633Z"
   },
   {
     "message": "Document arn:aws:imagebuilder:us-east-1:123456789012:component/
parallelclusterimage-test-abcd1234-ef56-gh78-ij90-1234abcd5678/3.13.2/1",
     "timestamp": "2022-04-05T22:13:26.741Z"
   },
   {
     "message": "TOE has completed execution successfully",
     "timestamp": "2022-04-05T22:13:26.819Z"
   }
 ]
}
```

BUILD_COMPLETE のステータスが表示されるまで、describe-image コマンドで確認を続けます。

```
$ pcluster describe-image --image-id IMAGE_ID --region REGION
{
 "imageConfiguration": {
   "url": "https://parallelcluster-1234abcd5678efgh-v1-do-not-delete.s3.us-
east-1.amazonaws.com/parallelcluster/3.13.2/images/IMAGE_ID-abcd1234efgh5678/configs/
image-config.yaml?Signature=..."
 },
 "imageId": "IMAGE_ID",
 "imageBuildStatus": "BUILD_COMPLETE",
 "region": "us-east-1",
 "ec2AmiInfo": {
     "amiName": "IMAGE_ID 2022-04-05T21-39-24.020Z",
     "amiId": "ami-1234stuv5678wxyz",
     "description": "AWS ParallelCluster AMI for alinux2,
 kernel-4.14.238-182.422.amzn2.x86_64, lustre-2.10.8-5.amzn2.x86_64,
 efa-1.13.0-1.amzn2.x86_64, dcv-2021.1.10598-1.el7.x86_64, slurm-20-11-8-1",
     "state": "AVAILABLE",
     "tags": [
      {
        "value": "2021.3.11591-1.el7.x86_64",
        "key": "parallelcluster:dcv_version"
      },
```

```
      ...
     ],
   "architecture": "x86_64"
 },
 "version": "3.13.2"
}
```

カスタム AMI 作成の問題をトラブルシューティングする必要がある場合は、次の手順で説明されているようにイメージログのアーカイブを作成します。

`--output` パラメータに応じて、ログを Amazon S3 バケットまたはローカルファイルにアーカイブすることが可能です。

```
$ pcluster export-image-logs --image-id IMAGE_ID --region REGION \
--bucket BUCKET_NAME --bucket-prefix BUCKET_FOLDER
{
 "url": "https://BUCKET_NAME.s3.us-east-1.amazonaws.com/BUCKET-FOLDER/IMAGE_ID-
logs-202209071136.tar.gz?AWSAccessKeyId=..."
}

$ pcluster export-image-logs --image-id IMAGE_ID \
--region REGION --bucket BUCKET_NAME --bucket-prefix BUCKET_FOLDER --output-file /tmp/
archive.tar.gz
{
 "path": "/tmp/archive.tar.gz"
}
```

アーカイブには、Image Builder プロセスと AWS CloudFormation スタックイベントに関連する CloudWatch Logs Streams が含まれています。このコマンドの実行には数分かかることがあります。

カスタム AMI の管理

AWS ParallelCluster 3.0.0 以降、イメージのライフサイクルを構築、モニタリング、管理するための新しいコマンドセットが CLI に追加されました。コマンドの詳細については、「pcluster commands」を参照してください。

AMI AWS ParallelCluster の変更

この方法は、公式 AMI AWS ParallelCluster にカスタマイズを追加して変更することで構成されます。 AWS ParallelCluster AMIs新しいリリースで更新されます。これらの AMIs には、インストール

および設定時に が機能 AWS ParallelCluster するために必要なすべてのコンポーネントがあります。これらのいずれかをベースとして開始できます。

キーポイント:

- この方法は、`build-image` コマンドよりも高速です。ただし、これは手動のプロセスであり、自動的に繰り返すことはできません。
- この方法では、CLI で利用できるログ検索やイメージライフサイクル管理コマンドにはアクセスできません。

手順:

New Amazon EC2 console

1. 使用する特定の に対応する AMI を見つけ AWS リージョン ます。これを見つけるには、 `--region`パラメータを指定して `pcluster list-official-images` コマンドを使用し、使用する OS AWS リージョン `--os`とアーキテクチャを使用して目的の AMI をフィルタリングする特定の パラメータと `--architecture`パラメータを選択します。出力から、Amazon EC2 イメージ ID を取得します。
2. にサインイン AWS Management Console し、https://console.aws.amazon.com/ec2/ で Amazon EC2 コンソールを開きます。
3. ナビゲーションペインで、[イメージ]、[AMI] の順に選択します。取得した EC2 イメージ ID を検索し、[AMI] を選択して、[AMI からインスタンスを起動] を選択します。
4. 下にスクロールして、使用する [インスタンスタイプ] を選択します。
5. 使用する [キーペア]、[インスタンスの作成] を選択します。
6. OS ユーザーと SSH キーを使用してインスタンスにログインします。
7. 要件に合わせてインスタンスを手動でカスタマイズします。
8. 次のコマンドを実行して、インスタンスを AMI 作成用に準備します。

   ```
   sudo /usr/local/sbin/ami_cleanup.sh
   ```

9. コンソールから [インスタンスの状態] および [インスタンスの停止] を選択します。

   [インスタンス] に移動して、新しいインスタンスを選択し、[インスタンスの状態]、[インスタンスの停止] の順に選択します。

10. Amazon EC2 コンソールまたは AWS CLI create-image を使用して、インスタンスから新しい AMI を作成します。

Amazon EC2 コンソールから

a. ナビゲーションペインで、[Instances (インスタンス)] を選択します。

b. 作成および変更したインスタンスを選択します。

c. [アクション]で、[イメージ]、[イメージの作成] の順に選択します。

d. [Create Image] を選択します。

11. クラスター設定内の `CustomAmi` フィールドに新しい AMI ID を入力して、クラスターを作成します。

Old Amazon EC2 console

1. 使用する特定の に対応する AWS ParallelCluster AMI を見つけ AWS リージョン ます。これを見つけるには、 `pcluster list-official-images` コマンドと `--region`パラメータを使用して、使用する OS AWS リージョン とアーキテクチャで目的の AMI をフィルタリングする特定の `--os`パラメータと `--architecture`パラメータを選択できます。出力から、Amazon EC2 イメージ ID を取得できます。
2. にサインイン AWS Management Console し、https://console.aws.amazon.com/ec2/ で Amazon EC2 コンソールを開きます。
3. ナビゲーションペインで、[イメージ]、[AMI] の順に選択します。[パブリックイメージ] のフィルターを設定し、取得した EC2 イメージ ID を検索し、AMI を選択して [起動] を選択します。
4. インスタンスタイプを選択し、[次へ:インスタンスの詳細の設定] または [確認と作成] を選択してインスタンスを起動します。
5. [起動] を選択し、[キーペア]、[インスタンスの作成] を選択します。
6. OS ユーザーと SSH キーを使用してインスタンスにログインします。詳細については、[インスタンス] に移動し、新しいインスタンスを選択して [接続] を選択します。
7. 要件を満たすようにインスタンスを手動でカスタマイズします。
8. 次のコマンドを実行して、インスタンスを AMI 作成用に準備します。

```
sudo /usr/local/sbin/ami_cleanup.sh
```

9. Amazon EC2 コンソールから、ナビゲーションペインで [インスタンス] を選択し、新しいインスタンスを選択して [アクション]、[インスタンスの状態]、[停止] の順に選択します。
10. Amazon EC2 コンソールまたは AWS CLI create-image を使用して、インスタンスから新しい AMI を作成します。

Amazon EC2 コンソールから

a. ナビゲーションペインで、[Instances (インスタンス)] を選択します。

b. 作成および変更したインスタンスを選択します。

c. [アクション] で、[イメージ]、[イメージの作成] の順に選択します。

d. [Create Image] を選択します。

11. クラスター設定内の `CustomAmi` フィールドに新しい AMI ID を入力して、クラスターを作成します。

Active Directory の統合

このチュートリアルでは、マルチユーザー環境を作成します。この環境には、 の (Active Directory) と AWS ParallelCluster AWS Managed Microsoft AD 統合された が含まれています`corp.example.com`。ディレクトリを管理する `Admin` ユーザー、ディレクトリを読み取る `ReadOnly` ユーザー、クラスターにログインする `user000` ユーザーを設定します。自動パスまたは手動パスのいずれかを使用して、AD の設定に使用するネットワークリソース、Active Directory (AD)、Amazon EC2 インスタンスを作成できます。パスに関係なく、作成するインフラストラクチャは、次のいずれかの方法 AWS ParallelCluster を使用して統合するように事前設定されています。

- 証明書の検証付き LDAPS (最も安全なオプションとして推奨)
- 証明書の検証なしの LDAPS
- LDAP

LDAP 自体は暗号化を提供しません。機密である可能性の高い情報を安全に送信するために、AD と統合されたクラスターには LDAPS (TLS/SSL 経由の LDAP) を使用することを強くお勧めします。詳細については、「 管理ガイド」の「 を使用してサーバー側の LDAPS を有効にする AWS Managed Microsoft AD」を参照してください。 AWS Directory Service

これらのリソースを作成したら、Active Directory (AD) と統合されたクラスターの設定と作成に進みます。クラスターが作成されたら、作成したユーザーでログインします。このチュートリアルで作成する設定の詳細については、「クラスターへの複数のユーザーアクセス」と「`DirectoryService` 設定」セクションを参照してください。

このチュートリアルでは、クラスターへの複数のユーザーアクセスをサポートする環境を作成する方法について説明します。このチュートリアルでは、 AWS Directory Service AD の作成方法と使用方

法について説明していません。このチュートリアル AWS Managed Microsoft AD で をセットアップするために実行する手順は、テストのみを目的としています。これらは、「AWS Directory Service 管理ガイド」の「AWS Managed Microsoft AD」および「Simple Active Directory」に記載されている公式ドキュメントやベストプラクティスに代わるものではありません。

> **Note**
>
> ディレクトリユーザーのパスワードは、ディレクトリパスワードポリシーのプロパティ定義に従って有効期限が切れます。でディレクトリパスワードをリセットするには AWS ParallelCluster、「」を参照してくださいユーザーパスワードと失効しているパスワードをリセットする方法。

> **Note**
>
> ディレクトリのドメインコントローラーの IP アドレスは、ドメインコントローラーの変更やディレクトリのメンテナンスにより変わる可能性があります。自動クイック作成方法を選択して、ディレクトリインフラストラクチャを作成した場合、ディレクトリ IP アドレスが変更されると、ロードバランサーをディレクトリコントローラーの前に手動で配置する必要があります。クイック作成方法を使用する場合、ディレクトリ IP アドレスはロードバランサーと自動的に調整されません。

AWS ParallelCluster コマンドラインインターフェイス (CLI) または API を使用する場合、 AWS ParallelCluster イメージとクラスターを作成または更新したときに作成された AWS リソースに対してのみ料金が発生します。詳細については、「AWS が使用する サービス AWS ParallelCluster」を参照してください。

前提条件

- AWS ParallelCluster がインストールされます。
- AWS CLI がインストールされ、設定されています。
- Amazon EC2 のキーペアがある。
- `pcluster` CLI の実行に必要なアクセス許可を持つ IAM ロールがある。

チュートリアルを進めながら、*region-id* や *d-abcdef01234567890* など、*inputs highlighted in red* を自分の名前と ID に置き換えてます。を自分の AWS アカウント 番号*0123456789012*に置き換えます。

AD インフラストラクチャを作成する

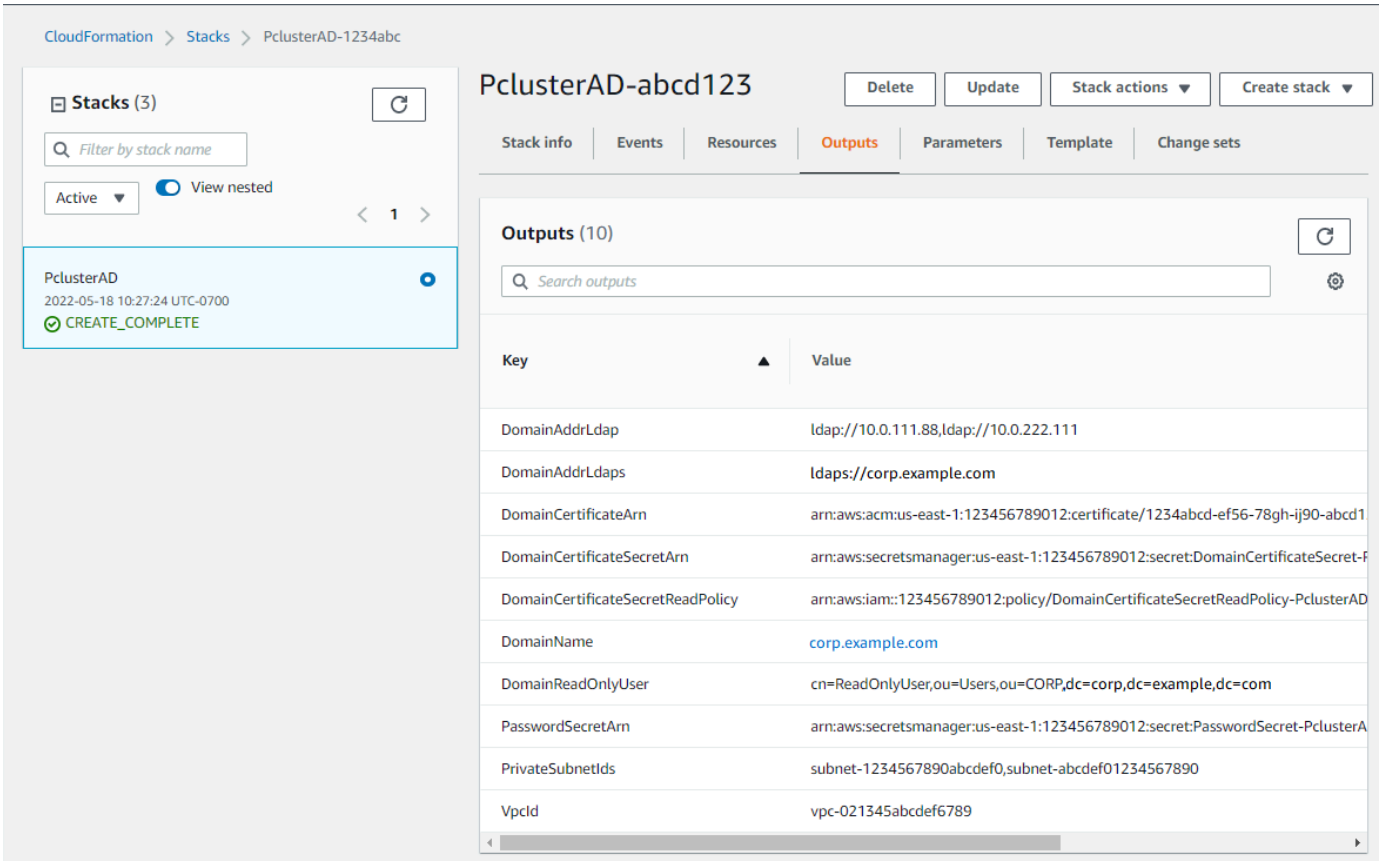
自動タブを選択して、 AWS CloudFormation クイック作成テンプレートを使用して Active Directory (AD) インフラストラクチャを作成します。

[手動] タブを選択し、AD インフラストラクチャを手動で作成します。

自動

1. AWS Management Consoleにサインインします。
2. CloudFormation クイック作成 (リージョン us-east-1) を開いて CloudFormation コンソールに以下のリソースを作成します。
   - 2 つのサブネットとパブリックアクセス用のルーティングを持つ VPC (VPC が指定されていない場合)。
   - AWS Managed Microsoft AD。
   - ディレクトリの管理に使用できる、AD に参加している Amazon EC2 インスタンス。
3. [スタックのクイック作成] ページの [パラメータ] セクションで、以下のパラメータのパスワードを入力します。
   - AdminPassword
   - ReadOnlyPassword
   - UserPassword

   パスワードを書き留めます。これらは、このチュートリアルで後ほど使用します。
4. [DomainName] に「**`corp.example.com`**」と入力します。
5. [Keypair] に、Amazon EC2 キーペアの名前を入力します。
6. ページの下部で、チェックボックスを選択し、各アクセス機能を確認します。
7. [スタックの作成] を選択してください。
8. CloudFormation スタックが `CREATE_COMPLETE` 状態になったら、スタックの[出力] タブを選択します。出力リソース名と ID は後のステップで使用する必要があるため、書き留めます。出力には、クラスターの作成に必要な情報が用意されています。



9. (オプション) AD のユーザーおよびグループの管理 の演習を完了するには、ディレクトリ ID が必要です。[リソース] を選択し、下にスクロールして、ディレクトリ ID を書き留めます。
10. (オプション) AD のユーザーおよびグループの管理 または クラスターを作成する に進みます。

手動

異なるアベイラビリティゾーンにある 2 つのサブネットと、 AWS Managed Microsoft ADを使用して、ディレクトリサービス用の VPC を作成します。

AD の作成

> **Note**
>
> - ディレクトリとドメイン名は `corp.example.com` です。省略名は `CORP` です。
> - スクリプト内の `Admin` パスワードを変更します。
> - Active Directory (AD) の作成には少なくとも 15 分かかります。

次の Python スクリプトを使用して、ローカルに VPC、サブネット、AD リソースを作成します AWS リージョン。このファイルを ad.py として保存し、実行します。

```
import boto3
import time
from pprint import pprint

vpc_name = "PclusterVPC"
ad_domain = "corp.example.com"
admin_password = "asdfASDF1234"

ec2 = boto3.client("ec2")
ds = boto3.client("ds")
region = boto3.Session().region_name

# Create the VPC, Subnets, IGW, Routes
vpc = ec2.create_vpc(CidrBlock="10.0.0.0/16")["Vpc"]
vpc_id = vpc["VpcId"]
time.sleep(30)
ec2.create_tags(Resources=[vpc_id], Tags=[{"Key": "Name", "Value": vpc_name}])
subnet1 = ec2.create_subnet(VpcId=vpc_id, CidrBlock="10.0.0.0/17",
 AvailabilityZone=f"{region}a")["Subnet"]
subnet1_id = subnet1["SubnetId"]
time.sleep(30)
ec2.create_tags(Resources=[subnet1_id], Tags=[{"Key": "Name", "Value": f"{vpc_name}/
subnet1"}])
ec2.modify_subnet_attribute(SubnetId=subnet1_id, MapPublicIpOnLaunch={"Value": True})
subnet2 = ec2.create_subnet(VpcId=vpc_id, CidrBlock="10.0.128.0/17",
 AvailabilityZone=f"{region}b")["Subnet"]
subnet2_id = subnet2["SubnetId"]
time.sleep(30)
ec2.create_tags(Resources=[subnet2_id], Tags=[{"Key": "Name", "Value": f"{vpc_name}/
subnet2"}])
ec2.modify_subnet_attribute(SubnetId=subnet2_id, MapPublicIpOnLaunch={"Value": True})
igw = ec2.create_internet_gateway()["InternetGateway"]
ec2.attach_internet_gateway(InternetGatewayId=igw["InternetGatewayId"], VpcId=vpc_id)
route_table = ec2.describe_route_tables(Filters=[{"Name": "vpc-id", "Values":
 [vpc_id]}])["RouteTables"][0]
ec2.create_route(RouteTableId=route_table["RouteTableId"],
 DestinationCidrBlock="0.0.0.0/0", GatewayId=igw["InternetGatewayId"])
ec2.modify_vpc_attribute(VpcId=vpc_id, EnableDnsSupport={"Value": True})
ec2.modify_vpc_attribute(VpcId=vpc_id, EnableDnsHostnames={"Value": True})
```

```
# Create the Active Directory
ad = ds.create_microsoft_ad(
    Name=ad_domain,
    Password=admin_password,
    Description="ParallelCluster AD",
    VpcSettings={"VpcId": vpc_id, "SubnetIds": [subnet1_id, subnet2_id]},
    Edition="Standard",
)
directory_id = ad["DirectoryId"]

# Wait for completion
print("Waiting for the directory to be created...")
directories = ds.describe_directories(DirectoryIds=[directory_id])
["DirectoryDescriptions"]
directory = directories[0]
while directory["Stage"] in {"Requested", "Creating"}:
    time.sleep(3)
    directories = ds.describe_directories(DirectoryIds=[directory_id])
["DirectoryDescriptions"]
    directory = directories[0]

dns_ip_addrs = directory["DnsIpAddrs"]

pprint({"directory_id": directory_id,
        "vpc_id": vpc_id,
        "subnet1_id": subnet1_id,
        "subnet2_id": subnet2_id,
        "dns_ip_addrs": dns_ip_addrs})
```

Python スクリプトからの出力例を次に示します。

```
{
  "directory_id": "d-abcdef01234567890",
  "dns_ip_addrs": ["192.0.2.254", "203.0.113.237"],
  "subnet1_id": "subnet-021345abcdef6789",
  "subnet2_id": "subnet-1234567890abcdef0",
  "vpc_id": "vpc-021345abcdef6789"
}
```

出力リソース名と ID を書き留めます。これらは、後のステップで使用します。

スクリプトが完了したら、次のステップに進みます。

Amazon EC2 インスタンスを作成する

New Amazon EC2 console

1. AWS Management Consoleにサインインします。
2. ステップ 4 で記載されたポリシーがアタッチされているロールがない場合は、https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。それ以外の場合は、ステップ 5 に進みます。
3. `ResetUserPassword` ポリシーを作成し、赤で強調表示されたコンテンツを、AD を作成するために実行したスクリプトの出力の AWS リージョン ID、アカウント ID、ディレクトリ ID に置き換えます。

   ResetUserPassword

   ```
   {
          "Statement": [
           {
               "Action": [
                   "ds:ResetUserPassword"
               ],
               "Resource": "arn:aws:ds:region-id:123456789012:directory/d-abcdef01234567890",
               "Effect": "Allow"
           }
       ]
   }
   ```

4. 以下のポリシーがアタッチされた IAM ロールを作成します。
   - AWS マネージドポリシー AmazonSSMManagedInstanceCore
   - AWS マネージドポリシー AmazonSSMDirectoryServiceAccess
   - ResetUserPassword ポリシー
5. Amazon EC2 コンソールの https://console.aws.amazon.com/ec2/ を開いてください。
6. [Amazon EC2 ダッシュボード] で、[インスタンスを起動] を選択します。
7. [アプリケーションイメージと OS イメージ] で、最近の Amazon Linux 2 AMI を選択します。
8. [インスタンスタイプ] で [t2.micro] を選択します。
9. [キーペア] で、キーペアを選択します。

10. [ネットワーク設定] で、[編集] を選択してください。
11. [VPC] で、ディレクトリ VPC を選択します。
12. 下にスクロールして [高度な詳細] を選択します。
13. [高度な詳細] の [ドメイン結合ディレクトリ] で、**`corp.example.com`** を選択します。
14. [IAM インスタンスプロファイル] で、ステップ 1 で作成したロール、またはステップ 4 でリストしたポリシーがアタッチされたロールを選択します。
15. [概要] で [インスタンスを起動] を選択します。
16. インスタンス ID (例:i-1234567890abcdef0) を書き留め、インスタンスの起動が完了するまで待ちます。
17. インスタンスが起動したら、次のステップに進みます。

Old Amazon EC2 console

1. AWS Management Consoleにサインインします。
2. ステップ 4 で記載されたポリシーがアタッチされているロールがない場合は、https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。それ以外の場合は、ステップ 5 に進みます。
3. `ResetUserPassword` ポリシーを作成します。赤色で強調表示されたコンテンツを、Active Directory (AD) を作成するために実行したスクリプトの出力の AWS リージョン ID、 AWS アカウント ID、ディレクトリ ID に置き換えます。

   ResetUserPassword

```
{
        "Statement": [
            {
                "Action": [
                    "ds:ResetUserPassword"
                ],
                "Resource": "arn:aws:ds:region-id:123456789012:directory/d-abcdef01234567890",
                "Effect": "Allow"
            }
        ]
    }
```

4. 以下のポリシーがアタッチされた IAM ロールを作成します。

- AWS マネージドポリシー AmazonSSMManagedInstanceCore
- AWS マネージドポリシー AmazonSSMDirectoryServiceAccess
- ResetUserPassword ポリシー

5. Amazon EC2 コンソールの https://console.aws.amazon.com/ec2/ を開いてください。
6. [Amazon EC2 ダッシュボード] で、[インスタンスを起動] を選択します。
7. [アプリケーションイメージと OS イメージ] で、最近の Amazon Linux 2 AMI を選択します。
8. [Instance type (インスタンスタイプ)] として [t2.micro] を選択します。
9. [キーペア] で、キーペアを選択します。
10. [ネットワーク設定] で、[編集] を選択します。
11. [ネットワーク設定] の [VPC] で、ディレクトリ VPC を選択します。
12. 下にスクロールして [高度な詳細] を選択します。
13. [高度な詳細] の [ドメイン結合ディレクトリ] で、**corp.example.com** を選択します。
14. [高度な詳細] の [インスタンスプロファイル] で、ステップ 1 で作成したロール、またはステップ 4 で記載されたポリシーがアタッチされているロールを選択します。
15. [概要] で [インスタンスを起動] を選択します。
16. インスタンス ID (例: i-1234567890abcdef0) を書き留め、インスタンスの起動が完了するまで待ちます。
17. インスタンスが起動したら、次のステップに進みます。

インスタンスを AD に結合

1. インスタンスに接続し、**admin** として AD 領域に結合します。

   次のコマンドを実行して、インスタンスに接続します。

```
$ INSTANCE_ID="i-1234567890abcdef0"
```

```
$ PUBLIC_IP=$(aws ec2 describe-instances \
--instance-ids $INSTANCE_ID \
--query "Reservations[0].Instances[0].PublicIpAddress" \
--output text)
```

```
$ ssh -i ~/.ssh/keys/keypair.pem ec2-user@$PUBLIC_IP
```

2. 必要なソフトウェアをインストールし、領域に結合します。

```
$ sudo yum -y install sssd realmd oddjob oddjob-mkhomedir adcli samba-common samba-
common-tools krb5-workstation openldap-clients policycoreutils-python
```

3. 管理者パスワードを **admin** パスワードに置き換えます。

```
$ ADMIN_PW="asdfASDF1234"
```

```
$ echo $ADMIN_PW | sudo realm join -U Admin corp.example.com
Password for Admin:
```

上記が成功した場合は、領域に結合したので、次のステップに進むことができます。

アカウントにユーザーを追加

1. ReadOnlyUser と追加のユーザーを作成します。

   このステップでは、前のステップでインストールした adcli ツールと openldap-clients ツールを使用します。

```
$ echo $ADMIN_PW | adcli create-user -x -U Admin --domain=corp.example.com --
display-name=ReadOnlyUser ReadOnlyUser
```

```
$ echo $ADMIN_PW | adcli create-user -x -U Admin --domain=corp.example.com --
display-name=user000 user000
```

2. ユーザーが作成されていることを確認します。

   ディレクトリの DNS IP アドレスは Python スクリプトの出力です。

```
$ DIRECTORY_IP="192.0.2.254"
```

```
$ ldapsearch -x -h $DIRECTORY_IP -D Admin -w $ADMIN_PW -b
 "cn=ReadOnlyUser,ou=Users,ou=CORP,dc=corp,dc=example,dc=com"
```

```
$ ldapsearch -x -h $DIRECTORY_IP -D Admin -w $ADMIN_PW -b
 "cn=user000,ou=Users,ou=CORP,dc=corp,dc=example,dc=com"
```

デフォルトでは、`ad-cli` を使用してユーザーを作成すると、そのユーザーは無効になります。

3. ローカルマシンからユーザーパスワードをリセットおよび有効化します。

   Amazon EC2 インスタンスからログアウトします。

   > **Note**
   >
   > - `ro-p@ssw0rd` は`ReadOnlyUser`、 から取得した のパスワードです AWS Secrets Manager。
   > - `user-p@ssw0rd` はクラスターに接続 (ssh) するときに指定されるクラスターユーザーのパスワードです。

   `directory-id` は Python スクリプトの出力です。

```
$ DIRECTORY_ID="d-abcdef01234567890"
```

```
$ aws ds reset-user-password \
--directory-id $DIRECTORY_ID \
--user-name "ReadOnlyUser" \
--new-password "ro-p@ssw0rd" \
--region "region-id"
```

```
$ aws ds reset-user-password \
--directory-id $DIRECTORY_ID \
--user-name "user000" \
--new-password "user-p@ssw0rd" \
--region "region-id"
```

4. Secrets Manager シークレットにパスワードを追加します。

を作成してパスワードReadOnlyUserを設定したら、 がログインの検証 AWS ParallelCluster に使用するシークレットに保存します。

Secrets Manager を使用して、ReadOnlyUser のパスワードを値として保持する新しいシークレットを作成します。シークレット値の形式はプレーンテキストのみである必要があります (JSON 形式ではない)。今後のステップに備えて、シークレットの ARN を書き留めておきます。

```
$ aws secretsmanager create-secret --name "ADSecretPassword" \
--region region_id \
--secret-string "ro-p@ssw0rd" \
--query ARN \
--output text
arn:aws:secretsmanager:region-id:123456789012:secret:ADSecretPassword-1234
```

証明書の検証付き LDAPS (推奨) の設定

リソース ID を書き留めます。これらは、後のステップで使用します。

1. ドメイン証明書をローカルで生成します。

```
$ PRIVATE_KEY="corp-example-com.key"
CERTIFICATE="corp-example-com.crt"
printf ".\n.\n.\n.\n.\ncorp.example.com\n.\n" | openssl req -x509 -sha256 -nodes -
newkey rsa:2048 -keyout $PRIVATE_KEY -days 365 -out $CERTIFICATE
```

2. 証明書を Secrets Manager に保存して、後でクラスター内から取得できるようにします。

```
$ aws secretsmanager create-secret --name example-cert \
  --secret-string file://$CERTIFICATE \
  --region region-id
{
  "ARN": "arn:aws:secretsmanager:region-id:123456789012:secret:example-
cert-123abc",
  "Name": "example-cert",
  "VersionId": "14866070-092a-4d5a-bcdd-9219d0566b9c"
}
```

3. Amazon EC2 インスタンスを AD ドメインに結合するために作成した IAM ロールに、次のポリシーを追加します。

PutDomainCertificateSecrets

```
{
    "Statement": [
        {
            "Action": [
                "secretsmanager:PutSecretValue"
            ],
            "Resource": [
                "arn:aws:secretsmanager:region-id:123456789012:secret:example-
cert-123abc"
            ],
            "Effect": "Allow"
        }
    ]
}
```

4. 証明書を AWS Certificate Manager (ACM) にインポートします。

```
$ aws acm import-certificate --certificate fileb://$CERTIFICATE \
  --private-key fileb://$PRIVATE_KEY \
  --region region-id
{
  "CertificateArn": "arn:aws:acm:region-
id:123456789012:certificate/343db133-490f-4077-b8d4-3da5bfd89e72"
}
```

5. Active Directory エンドポイントの前に配置するロードバランサーを作成します。

```
$ aws elbv2 create-load-balancer --name CorpExampleCom-NLB \
  --type network \
  --scheme internal \
  --subnets subnet-1234567890abcdef0 subnet-021345abcdef6789 \
  --region region-id
{
  "LoadBalancers": [
    {
      "LoadBalancerArn": "arn:aws:elasticloadbalancing:region-
id:123456789012:loadbalancer/net/CorpExampleCom-NLB/3afe296bf4ba80d4",
      "DNSName": "CorpExampleCom-NLB-3afe296bf4ba80d4.elb.region-id.amazonaws.com",
      "CanonicalHostedZoneId": "Z2IFOLAFXWLO4F",
      "CreatedTime": "2022-05-05T12:56:55.988000+00:00",
```

```
      "LoadBalancerName": "CorpExampleCom-NLB",
      "Scheme": "internal",
      "VpcId": "vpc-021345abcdef6789",
      "State": {
        "Code": "provisioning"
       },
       "Type": "network",
       "AvailabilityZones": [
         {
           "ZoneName": "region-idb",
           "SubnetId": "subnet-021345abcdef6789",
           "LoadBalancerAddresses": []
         },
         {
           "ZoneName": "region-ida",
           "SubnetId": "subnet-1234567890abcdef0",
           "LoadBalancerAddresses": []
         }
       ],
       "IpAddressType": "ipv4"
    }
  ]
}
```

6. Active Directory エンドポイントをターゲットとするターゲットグループを作成します。

```
$ aws elbv2 create-target-group --name CorpExampleCom-Targets --protocol TCP \
  --port 389 \
  --target-type ip \
  --vpc-id vpc-021345abcdef6789 \
  --region region-id
{
  "TargetGroups": [
    {
      "TargetGroupArn": "arn:aws:elasticloadbalancing:region-
id:123456789012:targetgroup/CorpExampleCom-Targets/44577c583b695e81",
      "TargetGroupName": "CorpExampleCom-Targets",
      "Protocol": "TCP",
      "Port": 389,
      "VpcId": "vpc-021345abcdef6789",
      "HealthCheckProtocol": "TCP",
      "HealthCheckPort": "traffic-port",
      "HealthCheckEnabled": true,
      "HealthCheckIntervalSeconds": 30,
```

```
      "HealthCheckTimeoutSeconds": 10,
      "HealthyThresholdCount": 3,
      "UnhealthyThresholdCount": 3,
      "TargetType": "ip",
      "IpAddressType": "ipv4"
    }
  ]
}
```

7. Active Directory (AD) エンドポイントをターゲットグループに登録します。

```
$ aws elbv2 register-targets --target-group-arn
 arn:aws:elasticloadbalancing:region-id:123456789012:targetgroup/CorpExampleCom-
Targets/44577c583b695e81 \
  --targets Id=192.0.2.254,Port=389 Id=203.0.113.237,Port=389 \
  --region region-id
```

8. 証明書を使用して LB リスナーを作成します。

```
$ aws elbv2 create-listener --load-balancer-arn
 arn:aws:elasticloadbalancing:region-id:123456789012:loadbalancer/net/
CorpExampleCom-NLB/3afe296bf4ba80d4 \
  --protocol TLS \
  --port 636 \
  --default-actions
 Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:region-
id:123456789012:targetgroup/CorpExampleCom-Targets/44577c583b695e81 \
  --ssl-policy ELBSecurityPolicy-TLS-1-2-2017-01 \
  --certificates CertificateArn=arn:aws:acm:region-
id:123456789012:certificate/343db133-490f-4077-b8d4-3da5bfd89e72 \
  --region region-id
  "Listeners": [
  {
    "ListenerArn": "arn:aws:elasticloadbalancing:region-id:123456789012:listener/
net/CorpExampleCom-NLB/3afe296bf4ba80d4/a8f9d97318743d4b",
    "LoadBalancerArn": "arn:aws:elasticloadbalancing:region-
id:123456789012:loadbalancer/net/CorpExampleCom-NLB/3afe296bf4ba80d4",
    "Port": 636,
    "Protocol": "TLS",
    "Certificates": [
      {
        "CertificateArn": "arn:aws:acm:region-
id:123456789012:certificate/343db133-490f-4077-b8d4-3da5bfd89e72"
```

```
        }
      ],
      "SslPolicy": "ELBSecurityPolicy-TLS-1-2-2017-01",
      "DefaultActions": [
        {
          "Type": "forward",
          "TargetGroupArn": "arn:aws:elasticloadbalancing:region-
id:123456789012:targetgroup/CorpExampleCom-Targets/44577c583b695e81",
          "ForwardConfig": {
            "TargetGroups": [
              {
                 "TargetGroupArn": "arn:aws:elasticloadbalancing:region-
id:123456789012:targetgroup/CorpExampleCom-Targets/44577c583b695e81"
               }
             ]
           }
         }
       ]
     }
   ]
}
```

9. ホストゾーンを作成して、クラスター VPC 内でドメインを検出できるようにします。

```
$ aws route53 create-hosted-zone --name corp.example.com \
  --vpc VPCRegion=region-id,VPCId=vpc-021345abcdef6789 \
  --caller-reference "ParallelCluster AD Tutorial"
{
  "Location": "https://route53.amazonaws.com/2013-04-01/hostedzone/
Z09020002B5MZQNXMSJUB",
  "HostedZone": {
    "Id": "/hostedzone/Z09020002B5MZQNXMSJUB",
    "Name": "corp.example.com.",
    "CallerReference": "ParallelCluster AD Tutorial",
    "Config": {
         "PrivateZone": true
    },
    "ResourceRecordSetCount": 2
  },
  "ChangeInfo": {
    "Id": "/change/C05533343BF3IKSORW1TQ",
    "Status": "PENDING",
    "SubmittedAt": "2022-05-05T13:21:53.863000+00:00"
  },
```

```
  "VPC": {
    "VPCRegion": "region-id",
    "VPCId": "vpc-021345abcdef6789"
  }
}
```

10. 次のコンテンツを使用して、**recordset-change.json** という名前のファイルを作成します。**HostedZoneId** はロードバランサーの正規のホストゾーン ID です。

```
{
  "Changes": [
    {
      "Action": "CREATE",
      "ResourceRecordSet": {
        "Name": "corp.example.com",
        "Type": "A",
        "Region": "region-id",
        "SetIdentifier": "example-active-directory",
        "AliasTarget": {
          "HostedZoneId": "Z2IFOLAFXWLO4F",
          "DNSName": "CorpExampleCom-NLB-3afe296bf4ba80d4.elb.region-
id.amazonaws.com",
          "EvaluateTargetHealth": true
        }
      }
    }
  ]
}
```

11. 今度はホストゾーン ID を使用して、レコードセットの変更をホストゾーンに送信します。

```
$ aws route53 change-resource-record-sets --hosted-zone-id Z09020002B5MZQNXMSJUB \
  --change-batch file://recordset-change.json
{
  "ChangeInfo": {
    "Id": "/change/C0137926I56R3GC7XW2Y",
    "Status": "PENDING",
    "SubmittedAt": "2022-05-05T13:40:36.553000+00:00"
  }
}
```

12. 次の内容でポリシードキュメント **policy.json** を作成します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Resource": [
        "arn:aws:secretsmanager:region-id:123456789012:secret:example-cert-abc123"
      ],
      "Effect": "Allow"
    }
  ]
}
```

13. 次の内容で **policy.json** という名前のポリシードキュメントを作成します。

```
$ aws iam create-policy --policy-name ReadCertExample \
  --policy-document file://policy.json
{
  "Policy": {
    "PolicyName": "ReadCertExample",
    "PolicyId": "ANPAUUXUVBC42VZSI4LDY",
    "Arn": "arn:aws:iam::123456789012:policy/ReadCertExample-efg456",
    "Path": "/",
    "DefaultVersionId": "v1",
    "AttachmentCount": 0,
    "PermissionsBoundaryUsageCount": 0,
    "IsAttachable": true,
    "CreateDate": "2022-05-05T13:42:18+00:00",
    "UpdateDate": "2022-05-05T13:42:18+00:00"
  }
}
```

14. 引き続き、(オプション) AD のユーザーおよびグループの管理 または クラスターを作成する のステップを実行します。

(オプション) AD のユーザーおよびグループの管理

このステップでは、Active Delivery (AD) ドメインに参加している Amazon EC2 Amazon Linux 2 インスタンスのユーザーとグループを管理します。

自動パスに従った場合は、オートメーションの一部として作成した AD に結合されたインスタンスを再起動してログインします。

手動パスに従った場合は、前のステップで作成して AD に結合したインスタンスを再起動してログインします。

これらのステップでは、前のステップの一部としてインスタンスにインストールした adcli ツールと openldap-clients ツールを使用します。

AD ドメインに参加している Amazon EC2 インスタンスへのログイン

1. Amazon EC2 コンソールから、前のステップで作成したタイトルのない Amazon EC2 インスタンスを選択します。インスタンスの状態は [停止] になっている可能性があります。
2. インスタンスの状態が [停止] の場合は、[インスタンスの状態] を選択し、[インスタンスを開始] を選択します。
3. ステータスチェックに合格したら、インスタンスを選択して [接続] を選択し、インスタンスに SSH 接続します。

AD に参加している Amazon EC2 Amazon Linux 2 インスタンスにログインしているときのユーザーとグループの管理

`-U "Admin"` オプションで `adcli` コマンドを実行すると、AD `Admin` パスワードの入力を求められます。`ldapsearch` コマンドの一部として AD `Admin` パスワードを含めます。

1. ユーザーを作成します。

```
$ adcli create-user "clusteruser" --domain "corp.example.com" -U "Admin"
```

2. ユーザーパスワードを設定します。

```
$ aws --region "region-id" ds reset-user-password --directory-id "d-
abcdef01234567890" --user-name "clusteruser" --new-password "new-p@ssw0rd"
```

3. グループを作成します。

```
$ adcli create-group "clusterteam" --domain "corp.example.com" -U "Admin"
```

4. ユーザーをグループに追加します。

```
$ adcli add-member "clusterteam" "clusteruser" --domain "corp.example.com" -U
 "Admin"
```

5. ユーザーとグループについて説明します。

   すべてのユーザーついて説明します。

```
$ ldapsearch "(&(objectClass=user))" -x -h "192.0.2.254" -b
 "DC=corp,DC=example,DC=com" -D
 "CN=Admin,OU=Users,OU=CORP,DC=corp,DC=example,DC=com" -w "p@ssw0rd"
```

   特定のユーザーについて説明します。

```
$ ldapsearch "(&(objectClass=user)(cn=clusteruser))"
 -x -h "192.0.2.254" -b "DC=corp,DC=example,DC=com" -D
 "CN=Admin,OU=Users,OU=CORP,DC=corp,DC=example,DC=com" -w "p@ssw0rd"
```

   すべてのユーザーを名前パターンで説明します。

```
$ ldapsearch "(&(objectClass=user)(cn=user*))" -x -h "192.0.2.254" -b
 "DC=corp,DC=example,DC=com" -D
 "CN=Admin,OU=Users,OU=CORP,DC=corp,DC=example,DC=com" -w "p@ssw0rd"
```

   特定のグループに属するすべてのユーザーについて説明します。

```
$ ldapsearch "(&(objectClass=user)
(memberOf=CN=clusterteam,OU=Users,OU=CORP,DC=corp,DC=example,DC=com))"
 -x -h "192.0.2.254" -b "DC=corp,DC=example,DC=com" -D
 "CN=Admin,OU=Users,OU=CORP,DC=corp,DC=example,DC=com" -w "p@ssw0rd"
```

   全てのグループについて説明します。

```
$ ldapsearch "objectClass=group" -x -h "192.0.2.254" -b "DC=corp,DC=example,DC=com"
 -D "CN=Admin,OU=Users,OU=CORP,DC=corp,DC=example,DC=com" -w "p@ssw0rd"
```

特定のグループについて説明します。

```
$ ldapsearch "(&(objectClass=group)(cn=clusterteam))"
 -x -h "192.0.2.254" -b "DC=corp,DC=example,DC=com" -D
 "CN=Admin,OU=Users,OU=CORP,DC=corp,DC=example,DC=com" -w "p@ssw0rd"
```

6. グループからユーザーを削除します。

```
$ adcli remove-member "clusterteam" "clusteruser" --domain "corp.example.com" -U
 "Admin"
```

7. ユーザーを削除します。

```
$ adcli delete-user "clusteruser" --domain "corp.example.com" -U "Admin"
```

8. グループを削除します。

```
$ adcli delete-group "clusterteam" --domain "corp.example.com" -U "Admin"
```

クラスターを作成する

Amazon EC2 インスタンスを終了していない場合は、ここで終了します。

Active Directory (AD) に対してユーザーを認証できるクラスターを作成するように環境が設定されます。

簡単なクラスター設定を作成し、AD への接続に関連する設定を行います。詳細については「DirectoryService」セクションを参照してください。

次のクラスター設定のいずれかを選択し、`ldaps_config.yaml`、`ldaps_nocert_config.yaml`、または `ldap_config.yaml` という名前のファイルにコピーします。

証明書検証付きの LDAPS 設定を選択することをお勧めします。この設定を選択した場合は、ブートストラップスクリプトを `active-directory.head.post.sh` という名前のファイルにコピーする

必要もあります。また、設定ファイルに示しているように Amazon S3 バケットに保存する必要があります。

証明書検証設定のある LDAPS (推奨)

> **Note**
>
> 以下のコンポーネントは変更する必要があります。
>
> - `KeyName`: Amazon EC2 キーペアのいずれか。
> - `SubnetId` / `SubnetIds`: CloudFormation スタックのクイック作成 (自動チュートリアル) または Python スクリプト (手動チュートリアル) の出力で提供されるサブネット ID のいずれか。
> - `Region`: AD インフラストラクチャを作成したリージョン。
> - `DomainAddr`: この IP アドレスは AD サービスの DNS アドレスの 1 つ。
> - `PasswordSecretArn`: `DomainReadOnlyUser` のパスワードが含まれているシークレットの Amazon リソースネーム (ARN)。
> - `BucketName`: ブートストラップスクリプトを保持するバケットの名前。
> - `AdditionalPolicies/Policy`: 読み取りドメイン認証ポリシー ReadCertExampl の Amazon リソースネーム (ARN)。
> - `CustomActions/OnNodeConfigured/Args`: ドメイン認定ポリシーを保持するシークレットの Amazon リソースネーム (ARN)。
>
> セキュリティ体制を強化するために、`HeadNode/Ssh/AllowedIps` 設定を使用してヘッドノードへの SSH アクセスを制限することをお勧めします。

```
Region: region-id
Image:
  Os: alinux2
HeadNode:
  InstanceType: t2.micro
  Networking:
    SubnetId: subnet-abcdef01234567890
  Ssh:
    KeyName: keypair
  Iam:
```

```
    AdditionalIamPolicies:
      - Policy: arn:aws:iam::123456789012:policy/ReadCertExample
    S3Access:
      - BucketName: amzn-s3-demo-bucket
        EnableWriteAccess: false
        KeyName: bootstrap/active-directory/active-directory.head.post.sh
  CustomActions:
    OnNodeConfigured:
      Script: s3://amzn-s3-demo-bucket/bootstrap/active-directory/active-
directory.head.post.sh
      Args:
        - arn:aws:secretsmanager:region-id:123456789012:secret:example-cert-123abc
        - /opt/parallelcluster/shared/directory_service/domain-certificate.crt
Scheduling:
  Scheduler: slurm
  SlurmQueues:
    - Name: queue0
      ComputeResources:
        - Name: queue0-t2-micro
          InstanceType: t2.micro
          MinCount: 1
          MaxCount: 10
      Networking:
        SubnetIds:
          - subnet-abcdef01234567890
DirectoryService:
  DomainName: corp.example.com
  DomainAddr: ldaps://corp.example.com
  PasswordSecretArn: arn:aws:secretsmanager:region-
id:123456789012:secret:ADSecretPassword-1234
  DomainReadOnlyUser: cn=ReadOnlyUser,ou=Users,ou=CORP,dc=corp,dc=example,dc=com
  LdapTlsCaCert: /opt/parallelcluster/shared/directory_service/domain-certificate.crt
  LdapTlsReqCert: hard
```

ブートストラップスクリプト

ブートストラップファイルを作成し、S3 バケットにアップロードする前に、 `chmod +x active-directory.head.post.sh`を実行して AWS ParallelCluster 実行アクセス許可を付与します。

```
#!/bin/bash
set -e

CERTIFICATE_SECRET_ARN="$1"
```

```
CERTIFICATE_PATH="$2"

[[ -z $CERTIFICATE_SECRET_ARN ]] && echo "[ERROR] Missing CERTIFICATE_SECRET_ARN" &&
 exit 1
[[ -z $CERTIFICATE_PATH ]] && echo "[ERROR] Missing CERTIFICATE_PATH" && exit 1

source /etc/parallelcluster/cfnconfig
REGION="${cfn_region:?}"

mkdir -p $(dirname $CERTIFICATE_PATH)
aws secretsmanager get-secret-value --region $REGION --secret-id
 $CERTIFICATE_SECRET_ARN --query SecretString --output text > $CERTIFICATE_PATH
```

証明書検証設定なしの LDAPS

> **Note**
>
> 以下のコンポーネントは変更する必要があります。
>
> - `KeyName`: Amazon EC2 キーペアのいずれか。
> - `SubnetId` / `SubnetIds`: CloudFormation スタックのクイック作成 (自動チュートリアル) または Python スクリプト (手動チュートリアル) の出力にあるサブネット ID の 1 つ。
> - `Region`: AD インフラストラクチャを作成したリージョン。
> - `DomainAddr`: この IP アドレスは AD サービスの DNS アドレスの 1 つ。
> - `PasswordSecretArn`: `DomainReadOnlyUser` のパスワードが含まれているシークレットの Amazon リソースネーム (ARN)。
>
> セキュリティ体制を強化するために、HeadNode/Ssh/AllowedIps 設定を使用して、ヘッドノードへの SSH アクセスを制限することをお勧めします。

```
Region: region-id
Image:
  Os: alinux2
HeadNode:
  InstanceType: t2.micro
  Networking:
    SubnetId: subnet-abcdef01234567890
  Ssh:
```

```
    KeyName: keypair
Scheduling:
  Scheduler: slurm
  SlurmQueues:
    - Name: queue0
      ComputeResources:
        - Name: queue0-t2-micro
          InstanceType: t2.micro
          MinCount: 1
          MaxCount: 10
      Networking:
        SubnetIds:
          - subnet-abcdef01234567890
DirectoryService:
  DomainName: corp.example.com
  DomainAddr: ldaps://corp.example.com
  PasswordSecretArn: arn:aws:secretsmanager:region-
id:123456789012:secret:ADSecretPassword-1234
  DomainReadOnlyUser: cn=ReadOnlyUser,ou=Users,ou=CORP,dc=corp,dc=example,dc=com
  LdapTlsReqCert: never
```

LDAP 設定

> **Note**
>
> 以下のコンポーネントは変更する必要があります。
>
> - KeyName: Amazon EC2 キーペアのいずれか。
> - SubnetId / SubnetIds: CloudFormation スタックのクイック作成 (自動チュートリアル) または Python スクリプト (手動チュートリアル) の出力で提供されるサブネット ID のいずれか。
> - Region: AD インフラストラクチャを作成したリージョン。
> - DomainAddr: この IP アドレスは AD サービスの DNS アドレスの 1 つ。
> - PasswordSecretArn: DomainReadOnlyUser のパスワードが含まれているシークレットの Amazon リソースネーム (ARN)。
>
> セキュリティ体制を強化するために、HeadNode/Ssh/AllowedIps 設定を使用して、ヘッドノードへの SSH アクセスを制限することをお勧めします。

```
Region: region-id
Image:
  Os: alinux2
HeadNode:
  InstanceType: t2.micro
  Networking:
    SubnetId: subnet-abcdef01234567890
  Ssh:
    KeyName: keypair
Scheduling:
  Scheduler: slurm
  SlurmQueues:
    - Name: queue0
      ComputeResources:
        - Name: queue0-t2-micro
          InstanceType: t2.micro
          MinCount: 1
          MaxCount: 10
      Networking:
        SubnetIds:
          - subnet-abcdef01234567890
DirectoryService:
  DomainName: dc=corp,dc=example,dc=com
  DomainAddr: ldap://192.0.2.254,ldap://203.0.113.237
  PasswordSecretArn: arn:aws:secretsmanager:region-
id:123456789012:secret:ADSecretPassword-1234
  DomainReadOnlyUser: cn=ReadOnlyUser,ou=Users,ou=CORP,dc=corp,dc=example,dc=com
  AdditionalSssdConfigs:
    ldap_auth_disable_tls_never_use_in_production: True
```

以下のコマンドを使用して、クラスターを作成します。

```
$ pcluster create-cluster --cluster-name "ad-cluster" --cluster-configuration "./
ldaps_config.yaml"
{
  "cluster": {
    "clusterName": "pcluster",
    "cloudformationStackStatus": "CREATE_IN_PROGRESS",
    "cloudformationStackArn": "arn:aws:cloudformation:region-id:123456789012:stack/ad-
cluster/1234567-abcd-0123-def0-abcdef0123456",
    "region": "region-id",
    "version": 3.13.2,
    "clusterStatus": "CREATE_IN_PROGRESS"
```

```
  }
}
```

ユーザーとしてクラスターに接続する

クラスターのステータスは、次のコマンドを使用して確認できます。

```
$ pcluster describe-cluster -n ad-cluster --region "region-id" --query "clusterStatus"
```

出力は次のとおりです。

```
"CREATE_IN_PROGRESS" / "CREATE_COMPLETE"
```

ステータスが "CREATE_COMPLETE" になったら、作成したユーザー名とパスワードでログインします。

```
$ HEAD_NODE_IP=$(pcluster describe-cluster -n "ad-cluster" --region "region-id" --query
 headNode.publicIpAddress | xargs echo)
```

```
$ ssh user000@$HEAD_NODE_IP
```

/home/user000@HEAD_NODE_IP/.ssh/id_rsa で新しいユーザー用に作成した SSH キーを提供することで、パスワードなしでログインできます。

ssh コマンドが成功すると、Active Directory (AD) を使用するように認証されたユーザーとしてクラスターに正常に接続されたことになります。

クリーンアップ

1. ローカルマシンからクラスターを削除します。

```
$ pcluster delete-cluster --cluster-name "ad-cluster" --region "region-id"
{
  "cluster": {
    "clusterName": "ad-cluster",
    "cloudformationStackStatus": "DELETE_IN_PROGRESS",
    "cloudformationStackArn": "arn:aws:cloudformation:region-id:123456789012:stack/
ad-cluster/1234567-abcd-0123-def0-abcdef0123456",
    "region": "region-id",
    "version": "3.13.2",
```

```
    "clusterStatus": "DELETE_IN_PROGRESS"
  }
}
```

2. 削除中のクラスターの進行状況を確認します。

```
$ pcluster describe-cluster --cluster-name "ad-cluster" --region "region-id" --
query "clusterStatus"
"DELETE_IN_PROGRESS"
```

クラスターが正常に削除されたら、次のステップに進みます。

自動化

Active Directory リソースの削除

1. https://console.aws.amazon.com/cloudformation/ から。
2. ナビゲーションペインで、[Stacks] を選択します。
3. スタックのリストから AD スタック (例: `pcluster-ad`) を選択します。
4. [削除] を選択します。

手動

1. Amazon EC2 インスタンスを削除します。
   a. https://console.aws.amazon.com/ec2/ から、ナビゲーションペインで [インスタンス] を選択します。
   b. インスタンスのリストから、ユーザーをディレクトリに追加するために作成したインスタンスを選択します。
   c. [インスタンスの状態]、[インスタンスの終了] の順に選択します。
2. ホストゾーンを削除します。
   a. 以下のコンテンツで `recordset-delete.json` を作成します。この例では、HostedZoneId はロードバランサーの正規のホストゾーン ID です。

```
{
  "Changes": [
    {
```

```
      "Action": "DELETE",
      "ResourceRecordSet": {
        "Name": "corp.example.com",
        "Type": "A",
        "Region": "region-id",
        "SetIdentifier": "pcluster-active-directory",
        "AliasTarget": {
          "HostedZoneId": "Z2IFOLAFXWLO4F",
          "DNSName": "CorpExampleCom-NLB-3afe296bf4ba80d4.elb.region-
id.amazonaws.com",
          "EvaluateTargetHealth": true
        }
      }
    }
  ]
}
```

b. ホストゾーン ID を使用して、レコードセットの変更をホストゾーンに送信します。

```
$ aws route53 change-resource-record-sets --hosted-zone-
id Z09020002B5MZQNXMSJUB \
  --change-batch file://recordset-delete.json
{
 "ChangeInfo": {
     "Id": "/change/C04853642A0TH2TJ5NLNI",
     "Status": "PENDING",
     "SubmittedAt": "2022-05-05T14:25:51.046000+00:00"
 }
}
```

c. ホストゾーンを削除します。

```
$ aws route53 delete-hosted-zone --id Z09020002B5MZQNXMSJUB
{
 "ChangeInfo": {
     "Id": "/change/C0468051QFABTVHMDEG9",
     "Status": "PENDING",
     "SubmittedAt": "2022-05-05T14:26:13.814000+00:00"
 }
}
```

3. LB リスナーを削除します。

```
$ aws elbv2 delete-listener \
  --listener-arn arn:aws:elasticloadbalancing:region-id:123456789012:listener/net/
CorpExampleCom-NLB/3afe296bf4ba80d4/a8f9d97318743d4b --region region-id
```

4. ターゲットグループを削除します。

```
$ aws elbv2 delete-target-group \
  --target-group-arn arn:aws:elasticloadbalancing:region-
id:123456789012:targetgroup/CorpExampleCom-Targets/44577c583b695e81 --
region region-id
```

5. ロードバランサーを削除します。

```
$ aws elbv2 delete-load-balancer \
  --load-balancer-arn arn:aws:elasticloadbalancing:region-
id:123456789012:loadbalancer/net/CorpExampleCom-NLB/3afe296bf4ba80d4 --
region region-id
```

6. クラスターが Secrets Manager から証明書を読み取るために使用するポリシーを削除します。

```
$ aws iam delete-policy --policy-arn arn:aws:iam::123456789012:policy/
ReadCertExample
```

7. ドメイン証明書を含むシークレットを削除します。

```
$ aws secretsmanager delete-secret \
  --secret-id arn:aws:secretsmanager:region-id:123456789012:secret:example-
cert-123abc \
  --region region-id
{
 "ARN": "arn:aws:secretsmanager:region-id:123456789012:secret:example-cert-123abc",
 "Name": "example-cert",
 "DeletionDate": "2022-06-04T16:27:36.183000+02:00"
}
```

8. ACM から証明書を削除します。

```
$ aws acm delete-certificate \
  --certificate-arn arn:aws:acm:region-
id:123456789012:certificate/343db133-490f-4077-b8d4-3da5bfd89e72 --region region-id
```

9. Active Directory (AD) リソースを削除します。

   a. Python スクリプト `ad.py` の出力から次のリソース ID を取得します。

      - AD ID
      - AD サブネット ID
      - AD VPC ID

   b. 次のコマンドを実行して、ディレクトリを削除します。

```
$ aws ds delete-directory --directory-id d-abcdef0123456789 --region region-id
{
   "DirectoryId": "d-abcdef0123456789"
}
```

   c. VPC 内のセキュリティグループを一覧表示します。

```
$ aws ec2 describe-security-groups --filters '[{"Name":"vpc-id","Values":
["vpc-07614ade95ebad1bc"]}]' --region region-id
```

   d. カスタムセキュリティグループを削除します。

```
$ aws ec2 delete-security-group --group-id sg-021345abcdef6789 --region region-
id
```

   e. サブネットを削除します。

```
$ aws ec2 delete-subnet --subnet-id subnet-1234567890abcdef --region region-id
```

```
$ aws ec2 delete-subnet --subnet-id subnet-021345abcdef6789 --region region-id
```

   f. インターネットゲートウェイについて説明します。

```
$ aws ec2 describe-internet-gateways \
  --filters Name=attachment.vpc-id,Values=vpc-021345abcdef6789 \
  --region region-id
{
  "InternetGateways": [
    {
      "Attachments": [
        {
          "State": "available",
```

```
            "VpcId": "vpc-021345abcdef6789"
          }
        ],
        "InternetGatewayId": "igw-1234567890abcdef",
        "OwnerId": "123456789012",
        "Tags": []
      }
    ]
  }
```

g. インターネットゲートウェイをデタッチします。

```
$ aws ec2 detach-internet-gateway \
  --internet-gateway-id igw-1234567890abcdef \
  --vpc-id vpc-021345abcdef6789 \
  --region region-id
```

h. インターネットゲートウェイを削除します。

```
$ aws ec2 delete-internet-gateway \
  --internet-gateway-id igw-1234567890abcdef \
  --region region-id
```

i. VPC を削除します。

```
$ aws ec2 delete-vpc \
  --vpc-id vpc-021345abcdef6789 \
  --region region-id
```

j. `ReadOnlyUser` パスワードを含むシークレットを削除します。

```
$ aws secretsmanager delete-secret \
  --secret-id arn:aws:secretsmanager:region-
id:123456789012:secret:ADSecretPassword-1234" \
  --region region-id
```

AWS KMS キーを使用した共有ストレージ暗号化の設定

用に設定されたクラスターファイルストレージシステム内のデータを暗号化および保護するための力スタマーマネージド AWS KMS キーを設定する方法について説明します AWS ParallelCluster。

AWS ParallelCluster コマンドラインインターフェイス (CLI) または API を使用する場合、 AWS ParallelCluster イメージとクラスターを作成または更新するときに作成された AWS リソースに対してのみ料金が発生します。詳細については、「AWS が使用する サービス AWS ParallelCluster」を参照してください。

AWS ParallelCluster は、以下の共有ストレージ設定オプションをサポートしています。

- `SharedStorage` / `EbsSettings` / `KmsKeyId`
- `SharedStorage` / `EfsSettings` / `KmsKeyId`
- `SharedStorage` / `FsxLustreSettings` / `KmsKeyId`

これらのオプションを使用して、Amazon EBS、Amazon EFS、FSx for Lustre 共有ストレージシステムの暗号化用のカスタマーマネージド AWS KMS キーを提供できます。これらを使用するには、以下の IAM ポリシーを作成して設定する必要があります。

- `HeadNode` / `Iam` / `AdditionalIamPolicies` / `Policy`
- `Scheduler` / `SlurmQueues` / `Iam` / `AdditionalIamPolicies` / `Policy`

前提条件

- AWS ParallelCluster がインストールされます。
- AWS CLI がインストールされ、設定されています。
- Amazon EC2 のキーペアがある。
- `pcluster` CLI の実行に必要なアクセス許可を持つ IAM ロールがある。

トピック

- ポリシーの作成
- クラスターの設定と作成

ポリシーの作成

このチュートリアルでは、 AWS KMS キーで共有ストレージ暗号化を設定するためのポリシーを作成します。

ポリシーを作成します。

1. IAM コンソール (https://console.aws.amazon.com/iam/home) に移動します。
2. [ポリシー] を選択します。
3. [Create policy] (ポリシーの作成) を選択します。
4. [JSON] タブを選択し、以下のポリシーを貼り付けます。のすべての出現を *123456789012* AWS アカウント ID とキー Amazon リソースネーム (ARN) AWS リージョン に置き換え、 を独自のものに置き換えてください。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:DescribeKey",
                "kms:ReEncrypt*",
                "kms:CreateGrant",
                "kms:Decrypt"
            ],
            "Resource": [
                "arn:aws:kms:region-id:123456789012:key/abcd1234-ef56-gh78-ij90-
abcd1234efgh5678"
            ]
        }
    ]
}
```

5. ポリシーの名前 `ParallelClusterKmsPolicy` を入力し、[ポリシーの作成] を選択します。
6. ポリシー ARN をメモします。クラスターの設定に必要です。

クラスターの設定と作成

以下は、暗号化された Amazon Elastic Block Store 共有ファイルシステムを含むクラスター設定の例です。

```
Region: eu-west-1
```

```
Image:
  Os: alinux2
HeadNode:
  InstanceType: t2.micro
  Networking:
    SubnetId: subnet-abcdef01234567890
  Ssh:
    KeyName: my-ssh-key
  Iam:
    AdditionalIamPolicies:
      - Policy: arn:aws:iam::123456789012:policy/ParallelClusterKmsPolicy
Scheduling:
  Scheduler: slurm
  SlurmQueues:
    - Name: q1
      ComputeResources:
        - Name: t2micro
          InstanceType: t2.micro
          MinCount: 0
          MaxCount: 10
      Networking:
        SubnetIds:
          - subnet-abcdef01234567890
      Iam:
        AdditionalIamPolicies:
          - Policy: arn:aws:iam::123456789012:policy/ParallelClusterKmsPolicy
SharedStorage:
  - MountDir: /shared/ebs1
    Name: shared-ebs1
    StorageType: Ebs
    EbsSettings:
      Encrypted: True
      KmsKeyId: abcd1234-ef56-gh78-ij90-abcd1234efgh5678
```

赤い文字の項目を独自の値に置き換えます。次に、 AWS KMS キーを使用して Amazon EBS 内のデータを暗号化するクラスターを作成します。

Amazon EFS および FSx for Lustre ファイルシステムの設定は同様です。

Amazon EFS `SharedStorage` の設定は以下のとおりです。

```
...
SharedStorage:
```

```
  - MountDir: /shared/efs1
    Name: shared-efs1
    StorageType: Efs
    EfsSettings:
      Encrypted: True
      KmsKeyId: abcd1234-ef56-gh78-ij90-abcd1234efgh5678
```

FSx for Lustre `SharedStorage` の設定は以下のとおりです。

```
...
SharedStorage:
  - MountDir: /shared/fsx1
    Name: shared-fsx1
    StorageType: FsxLustre
    FsxLustreSettings:
      StorageCapacity: 1200
      DeploymentType: PERSISTENT_1
      PerUnitStorageThroughput: 200
      KmsKeyId: abcd1234-ef56-gh78-ij90-abcd1234efgh5678
```

マルチキューモードのクラスターでジョブを実行する

このチュートリアルでは、複数のキューモードで最初のHello World「」ジョブを AWS ParallelCluster で実行する方法について説明します。

AWS ParallelCluster コマンドラインインターフェイス (CLI) または API を使用する場合、 AWS ParallelCluster イメージとクラスターを作成または更新したときに作成された AWS リソースに対してのみ料金が発生します。詳細については、「AWS が使用する サービス AWS ParallelCluster」を参照してください。

前提条件

- AWS ParallelCluster がインストールされます。
- AWS CLI がインストールされ、設定されています。
- Amazon EC2 のキーペアがある。
- `pcluster` CLI の実行に必要なアクセス許可を持つ IAM ロールがある。

クラスターを設定する

まず、次のコマンドを実行して、 が正しくインストール AWS ParallelCluster されていることを確認します。

```
$ pcluster version
```

pcluster version の詳細については、「pcluster version」を参照してください。

このコマンドは、実行中の のバージョンを返します AWS ParallelCluster。

次に、pcluster configure を実行して、基本的な設定ファイルを生成します。このコマンドに続くすべてのプロンプトに従います。

```
$ pcluster configure --config multi-queue-mode.yaml
```

pcluster configure コマンドの詳細については、「pcluster configure」を参照してください。

このステップを完了すると、multi-queue-mode.yaml という名前の基本設定ファイルが表示されます。このファイルには、基本的なクラスター設定が含まれています。

次のステップでは、新しい設定ファイルを変更して、複数のキューを持つクラスターを起動します。

> **Note**
>
> このチュートリアルで使用される一部のインスタンスは、無料利用枠の対象外です。

このチュートリアルでは、設定ファイルを以下の設定に合わせて変更します。赤で強調表示されている項目は、設定ファイルの値を表します。お客様の値をご使用ください。

```
Region: region-id
Image:
 Os: alinux2
HeadNode:
 InstanceType: c5.xlarge
 Networking:
   SubnetId: subnet-abcdef01234567890
```

```
  Ssh:
    KeyName: yourkeypair
Scheduling:
  Scheduler: slurm
  SlurmQueues:
  - Name: spot
    ComputeResources:
    - Name: c5xlarge
      InstanceType: c5.xlarge
      MinCount: 1
      MaxCount: 10
    - Name: t2micro
      InstanceType: t2.micro
      MinCount: 1
      MaxCount: 10
    Networking:
      SubnetIds:
      - subnet-abcdef01234567890
  - Name: ondemand
    ComputeResources:
    - Name: c52xlarge
      InstanceType: c5.2xlarge
      MinCount: 0
      MaxCount: 10
    Networking:
      SubnetIds:
      - subnet-021345abcdef6789
```

クラスターを作成する

設定ファイルに基づいて `multi-queue-cluster` という名前が付けられたクラスターを作成します。

```
$ pcluster create-cluster --cluster-name multi-queue-cluster --cluster-configuration
 multi-queue-mode.yaml
{
 "cluster": {
   "clusterName": "multi-queue-cluster",
   "cloudformationStackStatus": "CREATE_IN_PROGRESS",
   "cloudformationStackArn": "arn:aws:cloudformation:eu-west-1:123456789012:stack/
multi-queue-cluster/1234567-abcd-0123-def0-abcdef0123456",
   "region": "eu-west-1",
   "version": "3.13.2",
```

```
    "clusterStatus": "CREATE_IN_PROGRESS"
  }
}
```

pcluster create-cluster コマンドの詳細については、「pcluster create-cluster」を参照してください。

次のコマンドを実行して、クラスターのステータスを確認します。

```
$ pcluster list-clusters
{
  "cluster": {
    "clusterName": "multi-queue-cluster",
    "cloudformationStackStatus": "CREATE_IN_PROGRESS",
    "cloudformationStackArn": "arn:aws:cloudformation:eu-west-1:123456789012:stack/
multi-queue-cluster/1234567-abcd-0123-def0-abcdef0123456",
    "region": "eu-west-1",
    "version": "3.13.2",
    "clusterStatus": "CREATE_IN_PROGRESS"
  }
}
```

クラスターが作成されると、clusterStatus フィールドは、CREATE_COMPLETE を表示します。

ヘッドノードにログインします。

プライベート SSH キーファイルを使用して、ヘッドノードにログインします。

```
$ pcluster ssh --cluster-name multi-queue-cluster -i ~/path/to/yourkeyfile.pem
```

pcluster ssh の詳細については、「pcluster ssh」を参照してください。

ログインしたら、sinfo スケジューラキューがセットアップされ設定されていることを確認します。

sinfo の詳細については、「Slurm ドキュメンテーション」の「sinfo」を参照してください。

```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
spot*        up   infinite     18  idle~ spot-dy-c5xlarge-[1-9],spot-dy-t2micro-[1-9]
```

```
spot*        up   infinite      2  idle  spot-st-c5xlarge-1,spot-st-t2micro-1
ondemand     up   infinite     10  idle~ ondemand-dy-c52xlarge-[1-10]
```

この出力では、1 つの t2.micro コンピューティングノードと 1 つのc5.xlarge コンピューティングノードが idle 状態で存在し、クラスター内で利用可能であることがわかります。

その他のノードはすべて省電力状態にあることが、ノード状態の ~ サフィックスで示されています。これらは Amazon EC2 インスタンスによってバッキングされていません。デフォルトのキューには、キュー名の後に * というサフィックスが付きますので、spot はデフォルトのジョブキューとなります。

マルチキューモードでジョブを実行する

次に、しばらくの間、ジョブをスリープ状態にして実行してみます。このジョブは、後で自分のホスト名を出力します。このスクリプトが現在のユーザーで実行できることを確認します。

```
$ tee <<EOF hellojob.sh
#!/bin/bash
sleep 30
echo "Hello World from \$(hostname)"
EOF

$ chmod +x hellojob.sh
$ ls -l hellojob.sh
-rwxrwxr-x 1 ec2-user ec2-user 57 Sep 23 21:57 hellojob.sh
```

sbatch コマンドを使用してジョブを送信します。このジョブに対して -N 2 オプションで 2 つのノードを要求し、ジョブが正常に送信されることを確認します。sbatch の詳細については、「Slum ドキュメンテーション」の「sbatch」を参照してください。

```
$ sbatch -N 2 --wrap "srun hellojob.sh"
Submitted batch job 1
```

squeue コマンドでは、キューの表示やジョブの状態を確認することができます。特定のキューを指定していないため、デフォルトのキュー (spot) が使用されます。squeue の詳細については、「Slurm ドキュメンテーション」の「squeue」を参照してください。

```
$ squeue
JOBID PARTITION     NAME     USER  ST      TIME  NODES NODELIST(REASON)
```

```
   1      spot     wrap ec2-user  R       0:10      2 spot-st-c5xlarge-1,spot-st-
t2micro-1
```

ジョブが現在実行ステータスであることが出力に示されます。ジョブが終了するまで待機してください。これには約 30 秒かかります。その後、もう一度 squeue を実行してください。

```
$ squeue
JOBID PARTITION     NAME     USER          ST       TIME  NODES NODELIST(REASON)
```

キュー内のジョブがすべて終了したので、カレントディレクトリで slurm-1.out という名前の出力ファイル探します。

```
$ cat slurm-1.out
Hello World from spot-st-t2micro-1
Hello World from spot-st-c5xlarge-1
```

この出力では、spot-st-t2micro-1 と spot-st-c5xlarge-1 の各ノードでジョブが正常に実行されたことが示されています。

次のコマンドで特定のインスタンスに制約条件を指定して、同じジョブを送信します。

```
$ sbatch -N 3 -p spot -C "[c5.xlarge*1&t2.micro*2]" --wrap "srun hellojob.sh"
Submitted batch job 2
```

これらのパラメータを sbatch に使用しました。

- -N 3 — 3 つのノードを要求します
- -p spot — ジョブを spot キューへ送信します また、-p ondemand を指定して ondemand キューにジョブを送信することもできます。
- -C "[c5.xlarge*1&t2.micro*2]" — このジョブの特定のノード制約を指定します。これは、このジョブに使用される 1 つの c5.xlarge ノードと 2 つの t2.micro ノードを要求します。

sinfo コマンドを実行して、ノードとキューを表示します。のキュー AWS ParallelCluster は、 のパーティションと呼ばれますSlurm。

```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
```

```
spot*        up   infinite      1  alloc# spot-dy-t2micro-1
spot*        up   infinite     17  idle~ spot-dy-c5xlarge-[2-10],spot-dy-t2micro-[2-9]
spot*        up   infinite      1  mix   spot-st-c5xlarge-1
spot*        up   infinite      1  alloc spot-st-t2micro-1
ondemand     up   infinite     10  idle~ ondemand-dy-c52xlarge-[1-10]
```

ノードの電源が入っています。これは、ノードの状態に # というサフィックスが付いていることで示されます。squeue コマンドを実行して、クラスター内のジョブに関する情報を表示します。

```
$ squeue
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
   2      spot     wrap ec2-user CF       0:04      3 spot-dy-c5xlarge-1,spot-dy-
t2micro-1,spot-st-t2micro-1
```

ジョブは CF (CONFIGURING) の状態で、インスタンスがスケールアップしてクラスターに参加するのを待っています。

約 3 分後、ノードが利用可能になり、ジョブは R (RUNNING) の状態になります。

```
$ squeue
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
   2      spot     wrap ec2-user  R       0:07      3 spot-dy-t2micro-1,spot-st-
c5xlarge-1,spot-st-t2micro-1
```

ジョブが終了すると、3 つのノードはすべて idle の状態になります。

```
$ squeue
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
spot*        up   infinite     17  idle~ spot-dy-c5xlarge-[1-9],spot-dy-t2micro-[2-9]
spot*        up   infinite      3  idle  spot-dy-t2micro-1,spot-st-c5xlarge-1,spot-st-
t2micro-1
ondemand     up   infinite     10  idle~ ondemand-dy-c52xlarge-[1-10]
```

そして、キューにジョブが残っていない状態になってから、ローカルディレクトリに slurm-2.out があるかどうかを確認します。

```
$ cat slurm-2.out
Hello World from spot-st-t2micro-1
```

```
Hello World from spot-dy-t2micro-1
Hello World from spot-st-c5xlarge-1
```

これがクラスターの最終的な状態です。

```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
spot*        up   infinite     17  idle~ spot-dy-c5xlarge-[1-9],spot-dy-t2micro-[2-9]
spot*        up   infinite      3  idle  spot-dy-t2micro-1,spot-st-c5xlarge-1,spot-st-
t2micro-1
ondemand     up   infinite     10  idle~ ondemand-dy-c52xlarge-[1-10]
```

クラスターからログオフした後、pcluster delete-cluster を実行してクリーンアップすることができます。詳細については、「pcluster list-clusters」および「pcluster delete-cluster」を参照してください。

```
$ pcluster list-clusters
{
 "clusters": [
   {
     "clusterName": "multi-queue-cluster",
     "cloudformationStackStatus": "CREATE_COMPLETE",
     "cloudformationStackArn": "arn:aws:cloudformation:eu-west-1:123456789012:stack/
multi-queue-cluster/1234567-abcd-0123-def0-abcdef0123456",
     "region": "eu-west-1",
     "version": "3.1.4",
     "clusterStatus": "CREATE_COMPLETE"
   }
 ]
}
$ pcluster delete-cluster -n multi-queue-cluster
{
 "cluster": {
   "clusterName": "multi-queue-cluster",
   "cloudformationStackStatus": "DELETE_IN_PROGRESS",
   "cloudformationStackArn": "arn:aws:cloudformation:eu-west-1:123456789012:stack/
multi-queue-cluster/1234567-abcd-0123-def0-abcdef0123456",
   "region": "eu-west-1",
   "version": "3.1.4",
   "clusterStatus": "DELETE_IN_PROGRESS"
 }
}
```

AWS ParallelCluster API の使用

このチュートリアルでは、Amazon API Gateway と CloudFormation テンプレートを使用して API を構築してテストします。 AWS ParallelCluster CloudFormation 次に、GitHub にあるサンプルクライアントを使用して API を使用します。API の使用の詳細については、「AWS ParallelCluster API」を参照してください。

詳細については、EC2 Image Builder ユーザーガイドの「Image Builder を使用してカスタムコンポーネントを作成する」を参照してください。 EC2

AWS ParallelCluster コマンドラインインターフェイス (CLI) または API を使用する場合、 AWS ParallelCluster イメージとクラスターを作成または更新したときに作成された AWS リソースに対してのみ料金が発生します。詳細については、「AWS が使用する サービス AWS ParallelCluster」を参照してください。

前提条件

- AWS CLI はコンピューティング環境にインストールおよび設定されます。
- AWS ParallelCluster は仮想環境にインストールされます。詳細については、「仮想環境に AWS ParallelCluster をインストールする (推奨)」を参照してください。
- Amazon EC2 のキーペアがある。
- `pcluster` CLI を実行するために必要なアクセス許可を持つ IAM ロールがあること。

ステップ 1: Amazon API ゲートウェイを使用して API を構築する

ホームユーザーディレクトリに留まり、仮想環境を有効化します。

1. 便利な JSON コマンドラインプロセッサをインストールしてください。

```
$ sudo yum groupinstall -y "Development Tools"
 sudo yum install -y jq python3-devel
```

2. 次のコマンドを実行してバージョンを取得し AWS ParallelCluster 、環境変数に割り当てます。

```
$ PCLUSTER_VERSION=$(pcluster version | jq -r '.version')
 echo "export PCLUSTER_VERSION=${PCLUSTER_VERSION}" |tee -a ~/.bashrc
```

3. 環境変数を作成し、リージョン ID を割り当てます。

```
$ export AWS_DEFAULT_REGION="us-east-1"
 echo "export AWS_DEFAULT_REGION=${AWS_DEFAULT_REGION}" |tee -a ~/.bashrc
```

4. 以下のコマンドを実行して、API をデプロイします。

```
API_STACK_NAME="pc-api-stack"
 echo "export API_STACK_NAME=${API_STACK_NAME}" |tee -a ~/.bashrc
```

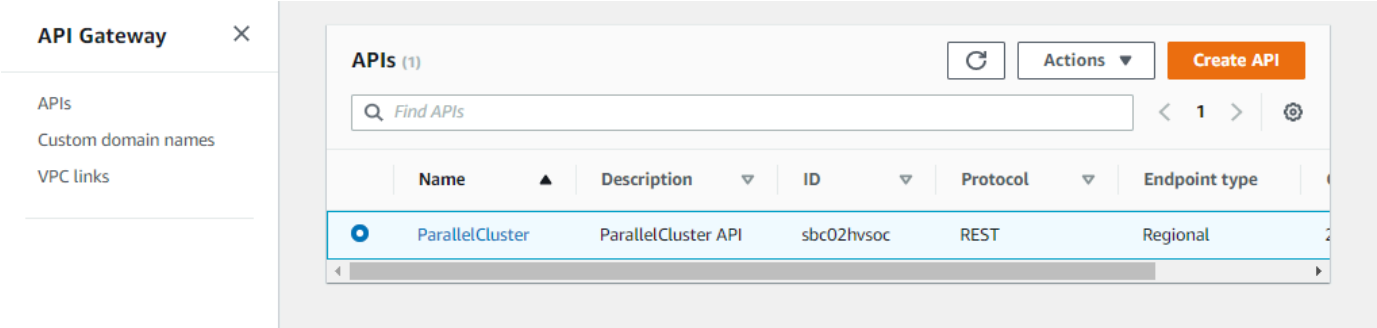
```
aws cloudformation create-stack \
   --region ${AWS_DEFAULT_REGION} \
   --stack-name ${API_STACK_NAME} \
   --template-url https://${AWS_DEFAULT_REGION}-aws-parallelcluster.s3.
${AWS_DEFAULT_REGION}.amazonaws.com/parallelcluster/${PCLUSTER_VERSION}/api/
parallelcluster-api.yaml \
   --capabilities CAPABILITY_NAMED_IAM CAPABILITY_AUTO_EXPAND \
   --parameters ParameterKey=EnableIamAdminAccess,ParameterValue=true

    {
       "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/my-api-
stack/abcd1234-ef56-gh78-ei90-1234abcd5678"
    }
```

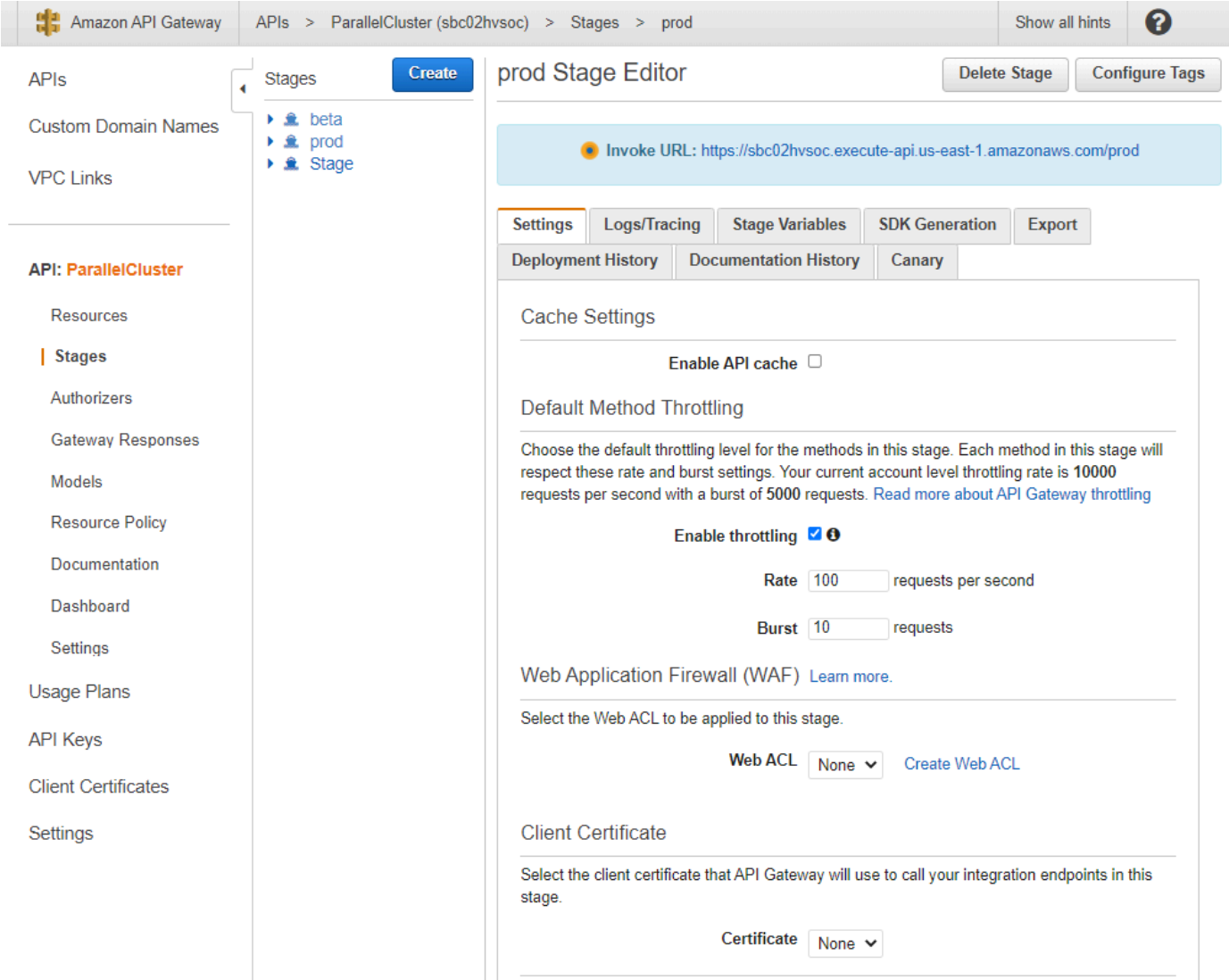
プロセスが完了したら、次のステップに進みます。

ステップ 2: Amazon API Gateway コンソールで API をテストする

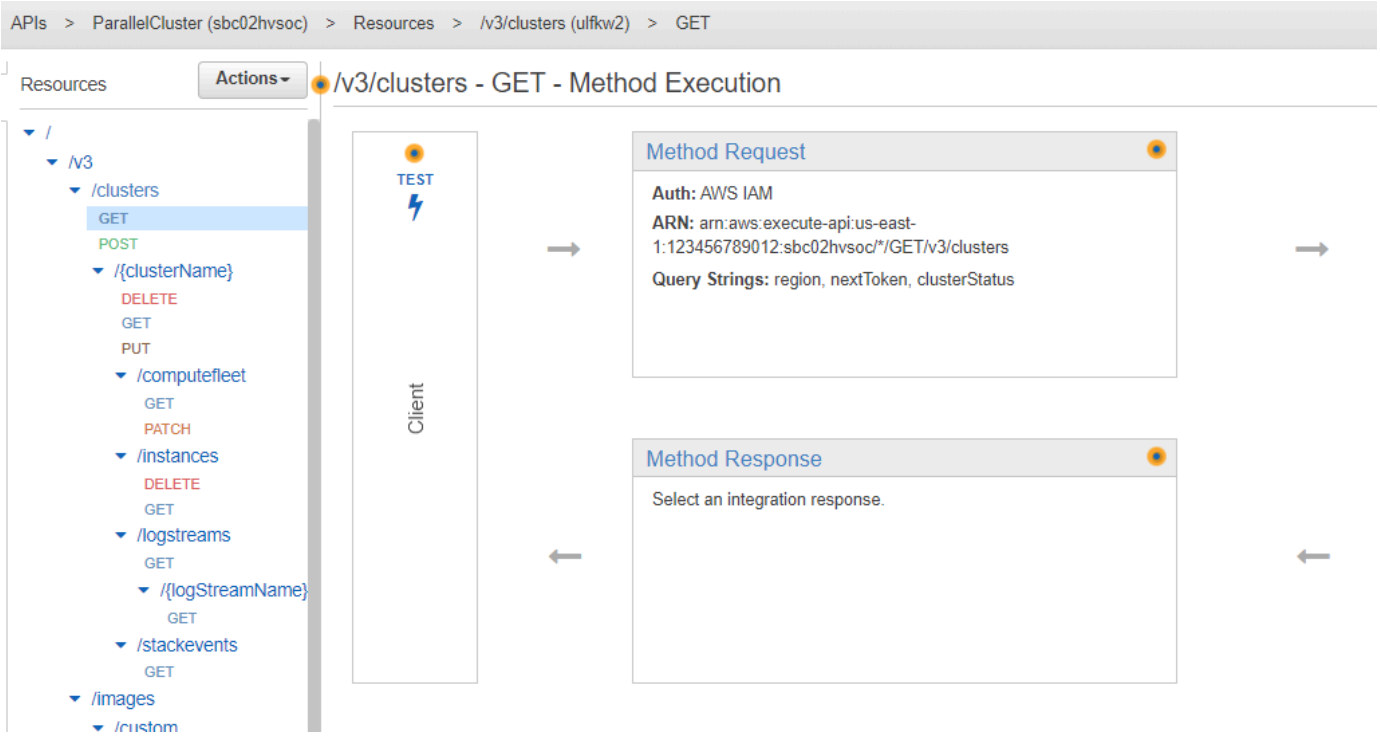
1. AWS Management Consoleにサインインします。
2. Amazon API Gateway コンソールに移動します。
3. API デプロイを選択します。



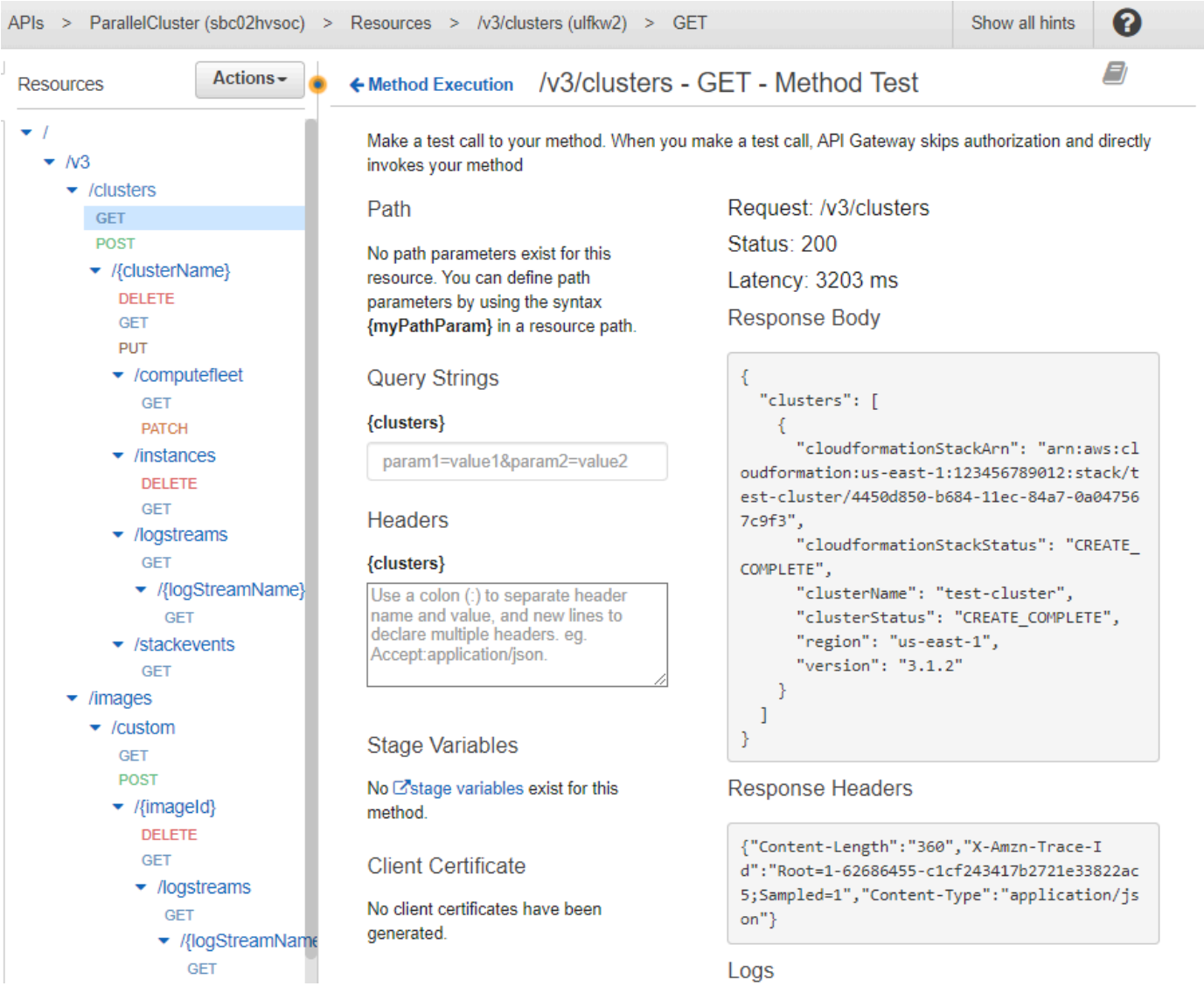
4. [ステージ] を選択し、ステージを選択します。



5. API Gateway が API にアクセスしたり呼び出したりするために提供する URL を書き留めておきます。青色で強調表示されています。
6. [リソース] を選択し、**`/clusters`** の下の **`GET`** を選択します。
7. [テスト] アイコンを選択し、下にスクロールして [テスト] アイコンを選択します。



`/clusters GET` に対するレスポンスが表示されます。



ステップ 3: API を呼び出すサンプルクライアントを準備してテストする

AWS ParallelCluster ソースコードを `api` ディレクトリcdにクローンし、Python クライアントライブラリをインストールします。

1.
```
$ git clone -b v${PCLUSTER_VERSION} https://github.com/aws/aws-parallelcluster aws-parallelcluster-v${PCLUSTER_VERSION}
 cd aws-parallelcluster-v${PCLUSTER_VERSION}/api
```

```
$ pip3 install client/src
```

2. ホームユーザーディレクトリに戻ります。
3. 実行時にクライアントが使用する API Gateway ベース URL をエクスポートします。

```
$ export PCLUSTER_API_URL=$( aws cloudformation describe-stacks
 --stack-name ${API_STACK_NAME} --query 'Stacks[0].Outputs[?
OutputKey==`ParallelClusterApiInvokeUrl`].OutputValue' --output text )
 echo "export PCLUSTER_API_URL=${PCLUSTER_API_URL}" |tee -a ~/.bashrc
```

4. クライアントがクラスターの作成に使用するクラスター名をエクスポートします。

```
$ export CLUSTER_NAME="test-api-cluster"
 echo "export CLUSTER_NAME=${CLUSTER_NAME}" |tee -a ~/.bashrc
```

5. 以下のコマンドを実行して、サンプルクライアントが API へのアクセスに使用する認証情報を保存します。

```
$ export PCLUSTER_API_USER_ROLE=$( aws cloudformation describe-
stacks --stack-name ${API_STACK_NAME} --query 'Stacks[0].Outputs[?
OutputKey==`ParallelClusterApiUserRole`].OutputValue' --output text )
 echo "export PCLUSTER_API_USER_ROLE=${PCLUSTER_API_USER_ROLE}" |tee -a ~/.bashrc
```

ステップ 4: クライアントコードスクリプトをコピーしてクラスターテストを実行する

1. 次のクライアントコード例をホームユーザーディレクトリの `test_pcluster_client.py` にコピーします。クライアントコードは次の処理を実行するように要求します。
   - クラスターを作成する。
   - クラスターを記述する。
   - クラスターを一覧表示する。
   - コンピューティングフリートについて記述する。
   - クラスターインスタンスについて記述する。

```
# Copyright 2021 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
```

```
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
#  Author: Evan F. Bollig (Github: bollig)

import time, datetime
import os
import pcluster_client
from pprint import pprint
from pcluster_client.api import (
    cluster_compute_fleet_api,
    cluster_instances_api,
    cluster_operations_api
)
from pcluster_client.model.create_cluster_request_content import
 CreateClusterRequestContent
from pcluster_client.model.cluster_status import ClusterStatus
region=os.environ.get("AWS_DEFAULT_REGION")

# Defining the host is optional and defaults to http://localhost
# See configuration.py for a list of all supported configuration parameters.
configuration = pcluster_client.Configuration(
    host = os.environ.get("PCLUSTER_API_URL")
)
cluster_name=os.environ.get("CLUSTER_NAME")

# Enter a context with an instance of the API client
with pcluster_client.ApiClient(configuration) as api_client:
    cluster_ops = cluster_operations_api.ClusterOperationsApi(api_client)
    fleet_ops = cluster_compute_fleet_api.ClusterComputeFleetApi(api_client)
```

```
    instance_ops = cluster_instances_api.ClusterInstancesApi(api_client)

    # Create cluster
    build_done = False
    try:
        with open('cluster-config.yaml', encoding="utf-8") as f:
            body = CreateClusterRequestContent(cluster_name=cluster_name,
cluster_configuration=f.read())
            api_response = cluster_ops.create_cluster(body, region=region)
    except pcluster_client.ApiException as e:
        print("Exception when calling create_cluster: %s\n" % e)
        build_done = True
    time.sleep(60)

    # Confirm cluster status with describe_cluster
    while not build_done:
        try:
            api_response = cluster_ops.describe_cluster(cluster_name,
region=region)
            pprint(api_response)
            if api_response.cluster_status == ClusterStatus('CREATE_IN_PROGRESS'):
                print('. . . working . . .', end='', flush=True)
                time.sleep(60)
            elif api_response.cluster_status == ClusterStatus('CREATE_COMPLETE'):
                print('READY!')
                build_done = True
            else:
                print('ERROR!!!!')
                build_done = True
        except pcluster_client.ApiException as e:
            print("Exception when calling describe_cluster: %s\n" % e)

    # List clusters
    try:
        api_response = cluster_ops.list_clusters(region=region)
        pprint(api_response)
    except pcluster_client.ApiException as e:
        print("Exception when calling list_clusters: %s\n" % e)

    # DescribeComputeFleet
    try:
        api_response = fleet_ops.describe_compute_fleet(cluster_name,
region=region)
        pprint(api_response)
```

```
    except pcluster_client.ApiException as e:
        print("Exception when calling compute fleet: %s\n" % e)

    # DescribeClusterInstances
    try:
        api_response = instance_ops.describe_cluster_instances(cluster_name,
 region=region)
        pprint(api_response)
    except pcluster_client.ApiException as e:
        print("Exception when calling describe_cluster_instances: %s\n" % e)
```

2. クラスター設定を作成します。

```
$ pcluster configure --config cluster-config.yaml
```

3. API Client ライブラリは、環境変数 (`AWS_ACCESS_KEY_ID`、`AWS_SECRET_ACCESS_KEY`、`AWS_SESSION_TOKEN` など) または `$HOME/.aws` から設定の詳細を自動的に検出します。次のコマンドは、現在の IAM ロールを指定された ParallelClusterApiUserRole に切り替えます。

```
$  eval $(aws sts assume-role --role-arn ${PCLUSTER_API_USER_ROLE} --role-
session-name ApiTestSession | jq -r '.Credentials | "export AWS_ACCESS_KEY_ID=
\(.AccessKeyId)\nexport AWS_SECRET_ACCESS_KEY=\(.SecretAccessKey)\nexport
 AWS_SESSION_TOKEN=\(.SessionToken)\n"')
```

注意すべきエラー:

次のようなエラーが表示される場合は、すでに ParallelClusterApiUserRole および `AWS_SESSION_TOKEN` の有効期限が切れていると考えられます。

```
An error occurred (AccessDenied) when calling the AssumeRole operation:
User: arn:aws:sts::XXXXXXXXXXXX:assumed-role/ParallelClusterApiUserRole-XXXXXXXX-
XXXX-XXXX-XXXX-XXXXXXXXXXXX/ApiTestSession
is not authorized to perform: sts:AssumeRole on resource:
 arn:aws:iam::XXXXXXXXXXXX:role/ParallelClusterApiUserRole-XXXXXXXX-XXXX-XXXX-XXXX-
XXXXXXXXXXXX
```

ロールを削除し、`aws sts assume-role` コマンドを再実行して ParallelClusterApiUserRole を使用します。

```
$ unset AWS_SESSION_TOKEN
```

```
unset AWS_SECRET_ACCESS_KEY
unset AWS_ACCESS_KEY_ID
```

API アクセス用の現在のユーザーアクセス許可を付与するには、リソースポリシーを拡張する必要があります。

4. サンプルクライアントをテストするには、次のコマンドを実行します。

```
$ python3 test_pcluster_client.py
{'cluster_configuration': 'Region: us-east-1\n'
                          'Image:\n'
                          '  Os: alinux2\n'
                          'HeadNode:\n'
                          '  InstanceType: t2.micro\n'
                          '  Networking . . . :\n'
                          '    SubnetId: subnet-1234567890abcdef0\n'
                          '  Ssh:\n'
                          '    KeyName: adpc\n'
                          'Scheduling:\n'
                          '  Scheduler: slurm\n'
                          '  SlurmQueues:\n'
                          '  - Name: queue1\n'
                          '    ComputeResources:\n'
                          '    - Name: t2micro\n'
                          '      InstanceType: t2.micro\n'
                          '      MinCount: 0\n'
                          '      MaxCount: 10\n'
                          '    Networking . . . :\n'
                          '      SubnetIds:\n'
                          '      - subnet-1234567890abcdef0\n',
 'cluster_name': 'test-api-cluster'}
{'cloud_formation_stack_status': 'CREATE_IN_PROGRESS',
 'cloudformation_stack_arn': 'arn:aws:cloudformation:us-east-1:123456789012:stack/
test-api-cluster/abcd1234-ef56-gh78-ij90-1234abcd5678',
 'cluster_configuration': {'url': 'https://parallelcluster-021345abcdef6789-v1-do-
not-delete...},
 'cluster_name': 'test-api-cluster',
 'cluster_status': 'CREATE_IN_PROGRESS',
 'compute_fleet_status': 'UNKNOWN',
 'creation_time': datetime.datetime(2022, 4, 28, 16, 18, 47, 972000,
 tzinfo=tzlocal()),
 'last_updated_time': datetime.datetime(2022, 4, 28, 16, 18, 47, 972000,
 tzinfo=tzlocal()),
```

```
 'region': 'us-east-1',
 'tags': [{'key': 'parallelcluster:version', 'value': '3.1.3'}],
 'version': '3.1.3'}
        .
        .
        .
. . . working . . . {'cloud_formation_stack_status': 'CREATE_COMPLETE',
 'cloudformation_stack_arn': 'arn:aws:cloudformation:us-east-1:123456789012:stack/
test-api-cluster/abcd1234-ef56-gh78-ij90-1234abcd5678',
 'cluster_configuration': {'url': 'https://parallelcluster-021345abcdef6789-v1-do-
not-delete...},
 'cluster_name': 'test-api-cluster',
 'cluster_status': 'CREATE_COMPLETE',
 'compute_fleet_status': 'RUNNING',
 'creation_time': datetime.datetime(2022, 4, 28, 16, 18, 47, 972000,
 tzinfo=tzlocal()),
 'head_node': {'instance_id': 'i-abcdef01234567890',
               'instance_type': 't2.micro',
               'launch_time': datetime.datetime(2022, 4, 28, 16, 21, 46,
 tzinfo=tzlocal()),
               'private_ip_address': '172.31.27.153',
               'public_ip_address': '52.90.156.51',
               'state': 'running'},
 'last_updated_time': datetime.datetime(2022, 4, 28, 16, 18, 47, 972000,
 tzinfo=tzlocal()),
 'region': 'us-east-1',
 'tags': [{'key': 'parallelcluster:version', 'value': '3.1.3'}],
 'version': '3.1.3'}
READY!
```

ステップ 5: クライアントコードスクリプトをコピーしてクラスターを削除する

1. 次のサンプルクライアントコードを `delete_cluster_client.py` にコピーします。クライアントコードはクラスターの削除をリクエストします。

```
# Copyright 2021 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
```

```
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
#  Author: Evan F. Bollig (Github: bollig)

import time, datetime
import os
import pcluster_client
from pprint import pprint
from pcluster_client.api import (
    cluster_compute_fleet_api,
    cluster_instances_api,
    cluster_operations_api
)
from pcluster_client.model.create_cluster_request_content import
 CreateClusterRequestContent
from pcluster_client.model.cluster_status import ClusterStatus
region=os.environ.get("AWS_DEFAULT_REGION")

# Defining the host is optional and defaults to http://localhost
# See configuration.py for a list of all supported configuration parameters.
configuration = pcluster_client.Configuration(
    host = os.environ.get("PCLUSTER_API_URL")
)
cluster_name=os.environ.get("CLUSTER_NAME")

# Enter a context with an instance of the API client
with pcluster_client.ApiClient(configuration) as api_client:
    cluster_ops = cluster_operations_api.ClusterOperationsApi(api_client)

    # Delete the cluster
    gone = False
```

```
    try:
        api_response = cluster_ops.delete_cluster(cluster_name, region=region)
    except pcluster_client.ApiException as e:
        print("Exception when calling delete_cluster: %s\n" % e)
    time.sleep(60)

    # Confirm cluster status with describe_cluster
    while not gone:
        try:
            api_response = cluster_ops.describe_cluster(cluster_name,
 region=region)
            pprint(api_response)
            if api_response.cluster_status == ClusterStatus('DELETE_IN_PROGRESS'):
                print('. . . working . . .', end='', flush=True)
                time.sleep(60)
        except pcluster_client.ApiException as e:
            gone = True
            print("DELETE COMPLETE or Exception when calling describe_cluster: %s
\n" % e)
```

2. 次のコマンドを実行して、クラスターを削除します。

```
$ python3 delete_cluster_client.py
{'cloud_formation_stack_status': 'DELETE_IN_PROGRESS',
'cloudformation_stack_arn': 'arn:aws:cloudformation:us-east-1:123456789012:stack/
test-api-cluster/abcd1234-ef56-gh78-ij90-1234abcd5678',
'cluster_configuration': {'url': 'https://parallelcluster-021345abcdef6789-v1-do-
not-delete...},
'cluster_name': 'test-api-cluster',
'cluster_status': 'DELETE_IN_PROGRESS',
'compute_fleet_status': 'UNKNOWN',
'creation_time': datetime.datetime(2022, 4, 28, 16, 50, 47, 943000,
 tzinfo=tzlocal()),
'head_node': {'instance_id': 'i-abcdef01234567890',
              'instance_type': 't2.micro',
              'launch_time': datetime.datetime(2022, 4, 28, 16, 53, 48,
 tzinfo=tzlocal()),
              'private_ip_address': '172.31.17.132',
              'public_ip_address': '34.201.100.37',
              'state': 'running'},
'last_updated_time': datetime.datetime(2022, 4, 28, 16, 50, 47, 943000,
 tzinfo=tzlocal()),
'region': 'us-east-1',
'tags': [{'key': 'parallelcluster:version', 'value': '3.1.3'}],
```

```
'version': '3.1.3'}
       .
       .
       .
. . . working . . . {'cloud_formation_stack_status': 'DELETE_IN_PROGRESS',
'cloudformation_stack_arn': 'arn:aws:cloudformation:us-east-1:123456789012:stack/
test-api-cluster/abcd1234-ef56-gh78-ij90-1234abcd5678',
'cluster_configuration': {'url': 'https://parallelcluster-021345abcdef6789-v1-do-
not-delete...},
'cluster_name': 'test-api-cluster',
'cluster_status': 'DELETE_IN_PROGRESS',
'compute_fleet_status': 'UNKNOWN',
'creation_time': datetime.datetime(2022, 4, 28, 16, 50, 47, 943000,
 tzinfo=tzlocal()),
'last_updated_time': datetime.datetime(2022, 4, 28, 16, 50, 47, 943000,
 tzinfo=tzlocal()),
'region': 'us-east-1',
'tags': [{'key': 'parallelcluster:version', 'value': '3.1.3'}],
'version': '3.1.3'}
. . . working . . . DELETE COMPLETE or Exception when calling describe_cluster:
 (404)
Reason: Not Found
        .
        .
        .
HTTP response body: {"message":"Cluster 'test-api-cluster' does not exist or
 belongs to an incompatible ParallelCluster major version."}
```

3. テストが終了したら、環境変数を設定解除します。

```
$ unset AWS_SESSION_TOKEN
unset AWS_SECRET_ACCESS_KEY
unset AWS_ACCESS_KEY_ID
```

ステップ 6: クリーンアップする

AWS Management Console または を使用して API AWS CLI を削除できます。

1. AWS CloudFormation コンソールから API スタックを選択し、削除を選択します。
2. AWS CLIを使用している場合、次のコマンドを実行します。

   の使用 AWS CloudFormation。

```
$ aws cloudformation delete-stack --stack-name ${API_STACK_NAME}
```

Slurm アカウンティングによるクラスターの作成

Slurm アカウンティングを使用してクラスターを構成および作成する方法について説明します。詳細については、「Slurm による アカウンティング AWS ParallelCluster」を参照してください。

AWS ParallelCluster コマンドラインインターフェイス (CLI) または API を使用する場合、 AWS ParallelCluster イメージとクラスターを作成または更新するときに作成された AWS リソースに対してのみ料金が発生します。詳細については、「AWS が使用する サービス AWS ParallelCluster」を参照してください。

このチュートリアルでは、CloudFormation クイック作成テンプレート (us-east-1) を使用して Amazon Aurora for MySQL サーバーレスデータベースを作成します。このテンプレートは、クラスターと同じ VPC に Amazon Aurora サーバーレスデータベースをデプロイするために必要なすべてのコンポーネントを作成するよう CloudFormation に指示します。このテンプレートは、クラスターとデータベース間の接続のための基本的なネットワークとセキュリティの設定も作成します。

> **Note**
> バージョン 3.3.0 以降、 はクラスター設定パラメータ SlurmSettings / Database による Slurmアカウンティング AWS ParallelCluster をサポートしています。

> **Note**
> クイック作成テンプレートはその一例です。このテンプレートは、Slurm アカウンティングデータベースサーバーで考えられるすべてのユースケースを網羅しているわけではありません。本番環境のワークロードに適した構成と容量を備えたデータベースサーバーを作成するのはお客様の責任です。

前提条件:

- AWS ParallelCluster がインストールされます。
- AWS CLI がインストールされ、設定されています。

- Amazon EC2 のキーペアがある。
- `pcluster` CLI を実行するために必要なアクセス許可を持つ IAM ロールがあること。
- クイック作成テンプレートをデプロイするリージョンは、Amazon Aurora MySQL Serverless v2 をサポートしています。詳細については、「Aurora MySQL を使用した Aurora Serverless v2」を参照してください。

ステップ 1: の VPC とサブネットを作成する AWS ParallelCluster

提供されている CloudFormation テンプレートを Slurm アカウンティングデータベースに使用するには、クラスター用の VPC を準備しておく必要があります。これは手動で行うことも、「Configure and create a cluster with the AWS ParallelCluster command line interface」 の手順の一部として行うこともできます。既に AWS ParallelClusterを使用している場合は、クラスターとデータベースサーバーをデプロイするための VPC の準備ができている可能性があります。

ステップ 2: データベーススタックを作成する

CloudFormation クイック作成テンプレート (us-east-1) を使用して、Slurm アカウンティング用のデータベーススタックを作成します。テンプレートには以下の入力が必要です。

- データベースサーバーの認証情報、具体的には管理者ユーザー名とパスワード。
- Amazon Aurora サーバーレスクラスターのサイズ。これは予想されるクラスターの負荷によって異なります。
- ネットワークパラメータ、具体的にはターゲット VPC とサブネット、またはサブネットを作成するための CIDR ブロック。

データベースサーバーに適した認証情報とサイズを選択してください。ネットワークオプションでは、 AWS ParallelCluster クラスターがデプロイされているのと同じ VPC を使用する必要があります。データベースのサブネットを作成し、テンプレートへの入力として渡すことができます。または、2 つのサブネットに独立した 2 つの CIDR ブロックを指定し、CloudFormation テンプレートで CIDR ブロックの 2 つのサブネットを作成します。CIDR ブロックが既存のサブネットと重複しないようにしてください。CIDR ブロックが既存のサブネットと重複している場合、スタックの作成は失敗します。

データベースサーバーの作成には数分かかります。

ステップ 3: Slurm アカウンティングを有効にしたクラスターを作成する

提供されている CloudFormation テンプレートは、いくつかの定義済みの出力を含む CloudFormation スタックを生成します。から AWS Management Console、CloudFormation スタックビューの出力タブで出力を表示できます。Slurm アカウンティングを有効にするには、以下の出力の一部を AWS ParallelCluster クラスターの設定ファイルで使用する必要があります。

- `DatabaseHost`: `SlurmSettings`/`Database`/`Uri` クラスターの設定パラメータに使用される。
- `DatabaseAdminUser`: `SlurmSettings`/`Database`/`UserName` クラスターの設定パラメータ値に使用される。
- `DatabaseSecretArn`: `SlurmSettings`/`Database`/`PasswordSecretArn` クラスターの設定パラメータに使用される。
- `DatabaseClientSecurityGroup`: これは、`HeadNode`/`Networking`/`SecurityGroups` 設定パラメータで定義されているクラスターのヘッドノードにアタッチされているセキュリティグループです。

クラスターの設定ファイルの `Database` パラメータを出力値で更新します。`pcluster` CLI を使用してクラスターを作成します。

```
$ pcluster create-cluster -n cluster-3.x -c path/to/cluster-config.yaml
```

クラスターが作成されたら、`sacctmgr` や `sacct` などの Slurm アカウンティングコマンドを使い始めることができます。

外部 Slurmdbd アカウンティングによるクラスターの作成

外部 Slurmdbd アカウンティングを使用してクラスターを設定および作成する方法について説明します。詳細については、Slurm「 でのアカウンティング AWS ParallelCluster」を参照してください。

AWS ParallelCluster コマンドラインインターフェイス (CLI) または API を使用する場合、 AWS ParallelCluster イメージとクラスターを作成または更新するときに作成されたリソースに対して AWS のみ料金が発生します。詳細については、AWS 「 で使用される のサービス AWS ParallelCluster」を参照してください。

AWS ParallelCluster UI はサーバーレスアーキテクチャ上に構築されており、ほとんどの場合、AWS Free Tierカテゴリ内で使用できます。詳細については、「AWS ParallelCluster UI のコスト」を参照してください。

このチュートリアルでは、 AWS CloudFormation クイック作成テンプレートを使用して、クラスターと同じ VPC に Slurmdbd インスタンスをデプロイするために必要なコンポーネントを作成します。このテンプレートは、クラスターとデータベース間の接続のための基本的なネットワークとセキュリティの設定を作成します。

> **Note**
>
> 以降`version 3.10.0`、 はクラスター設定パラメータ を使用して外部 Slurmdbd AWS ParallelCluster をサポートします`SlurmSettings / ExternelSlurmdbd`。

> **Note**
>
> クイック作成テンプレートはその一例です。このテンプレートは、すべての想定されるユースケースをカバーしているわけではありません。本番環境のワークロードに適した設定と容量を備えた外部 Slurmdbd を作成することはお客様の責任です。

前提条件:

- AWS ParallelCluster がインストールされます。
- AWS CLI がインストールされ、設定されています。
- Amazon Elastic Compute Cloud のキーペアがある。
- CLI の実行に必要なアクセス許可を持つ `pcluster` AWS Identity and Access Management ロールがあります。
- Slurm アカウンティングデータベースがある。Slurm アカウンティングデータベースを作成するチュートリアルを実行するには、「Slurm アカウンティングデータベーススタックを作成する」のステップ 1 および 2 に従います。

ステップ 1: Slurmdbd スタックを作成する

このチュートリアルでは、CloudFormation クイック作成テンプレート (`us-east-1`) を使用して Slurmdbd スタックを作成します。テンプレートには以下の入力が必要です。

ネットワーク

- VPCId: Slurmdbd インスタンスを起動するための VPC ID。

- SubnetId: Slurmdbd インスタンスを起動するためのサブネット ID。
- PrivatePrefix: VPC の CIDR プレフィックス。
- PrivateIp: Slurmdbd インスタンスに割り当てるセカンダリプライベート IP。

データベース接続

- DBMSClientSG: Slurmdbd インスタンスにアタッチするセキュリティグループ。このセキュリティグループは、データベースサーバーと Slurmdbd インスタンス間の接続を許可する必要があります。
- DBMSDatabaseName: データベースの名前。
- DBMSUsername: データベースのユーザー名。
- DBMSPasswordSecretArn: データベースへのパスワードを含むシークレット。
- DBMSUri: データベースサーバーの URI。

インスタンス設定

- InstanceType: slurmdbd インスタンスに使用するインスタンスタイプ。
- KeyName: slurmdbd インスタンスに使用する Amazon EC2 キーペア。

Slurmdbd 設定

- AMIID : Slurmdbd インスタンスの AMI。AMI は ParallelCluster AMI である必要があります。ParallelCluster AMI のバージョンにより、Slurmdbd のバージョンが決まります。
- MungeKeySecretArn: Slurmdbd とクラスター間の通信を認証するために使用する munge キーを含むシークレット。
- SlurmdbdPort: slurmdbd で使用するポート番号。
- EnableSlurmdbdSystemService: slurmdbd をシステムサービスとして有効にし、インスタンスの起動時に実行します。

> **Warning**
>
> データベースが別のバージョンの SlurmDB で作成した場合は、Slurmdbd をシステムサービスとして使用しないでください。

データベースに多数のエントリが含まれている場合、Slurm Database Daemon (SlurmDBD) はデータベースの更新に数十分かかり、この間は応答しなくなることがあります。SlurmDB をアップグレードする前に、データベースのバックアップを作成します。詳細については、Slurm のドキュメントを参照してください。

ステップ 2: 外部 Slurmdbd を有効にしたクラスターを作成する

提供された AWS CloudFormation テンプレートは、いくつかの定義された出力を持つ AWS CloudFormation スタックを生成します。

から AWS Management Console、 AWS CloudFormation スタックの出力タブを表示して、作成されたエンティティを確認します。Slurm アカウンティングを有効にするには、これらの出力の一部 AWS ParallelCluster を設定ファイルで使用する必要があります。

- SlurmdbdPrivateIp: SlurmSettings / ExternalSlurmdbd / Host クラスター設定パラメータに使用します。
- SlurmdbdPort: SlurmSettings / ExternalSlurmdbd / ポートクラスター設定パラメータの値に使用されます。
- AccountingClientSecurityGroup: これは、HeadNode / Networking / AdditionalSecurityGroups 設定パラメータで定義されているクラスターのヘッドノードにアタッチされているセキュリティグループです。

さらに、 AWS CloudFormation スタックビューのパラメータタブから以下を実行します。

- MungeKeySecretArn: SlurmSettings / MungeKeySecretArn クラスター設定パラメータ値に使用します。

クラスター設定ファイルのデータベースパラメータを出力値で更新します。pcluster AWS CLI を使用してクラスターを作成します。

```
$ pcluster create-cluster -n cluster-3.x-c path/to/cluster-config.yaml
```

クラスターを作成すると、`sacctmgr` や `sacct` などの Slurm アカウンティングコマンドを使い始めることができます。

> **Warning**
>
> `ParallelCluster` と外部 SlurmDB との間のトラフィックは暗号化されません。クラスターと外部 SlurmDB は、信頼できるネットワークで実行することをお勧めします。

以前の AWS Systems Manager ドキュメントバージョンに戻す

以前の AWS Systems Manager ドキュメントバージョンに戻す方法について説明します。SSM ドキュメントの詳細については、「AWS Systems Manager ユーザーガイド」の「AWS Systems Manager ドキュメント」を参照してください。

AWS ParallelCluster コマンドラインインターフェイス (CLI) または API を使用する場合、 AWS ParallelCluster イメージとクラスターを作成または更新するときに作成された AWS リソースに対してのみ料金が発生します。詳細については、「AWS が使用する サービス AWS ParallelCluster」を参照してください。

前提条件:

- SSM ドキュメントを管理するアクセス許可 AWS アカウント を持つ 。
- AWS CLI がインストールされ、設定されています。

SSM ドキュメントを以前のバージョンに戻す

1. ターミナルで次のコマンドを実行して、所有している既存の SSM ドキュメントの一覧を取得します。

   ```
   $ aws ssm list-documents --document-filter "key=Owner,value=Self"
   ```

2. SSM ドキュメントを以前のバージョンに戻します。この例では、`SessionManagerRunShell` ドキュメントを以前のバージョンに戻します。SSM `SessionManagerRunShell` ドキュメントを使用して、開始するすべての SSM シェルセッションをカスタマイズできます。

   a. 次のコマンドを実行して、`SessionManagerRunShell` の `DocumentVersion` パラメータを検索します。

   ```
   $ aws ssm describe-document --name "SSM-SessionManagerRunShell"
   ```

```
{
    "Document": {
        "Hash": "...",
        "HashType": "Sha256",
        "Name": "SSM-SessionManagerRunShell",
        "Owner": "123456789012",
        "CreatedDate": "2023-02-20T19:04:32.390000+00:00",
        "Status": "Active",
        "DocumentVersion": "1",
        "Parameters": [
            {
                "Name": "linuxcmd",
                "Type": "String",
                "Description": "The command to run on connection...",
                "DefaultValue": "if [ -d '/opt/parallelcluster' ]; then
 source /opt/parallelcluster/cfnconfig; sudo su - $cfn_cluster_user; fi; /bin/
bash"
            }
        ],
        "PlatformTypes": [
            "Windows",
            "Linux",
            "MacOS"
        ],
        "DocumentType": "Session",
        "SchemaVersion": "1.0",
        "LatestVersion": "2",
        "DefaultVersion": "1",
        "DocumentFormat": "JSON",
        "Tags": []
    }
}
```

最新バージョンは 2 です。

b. 次のコマンドを実行して、以前のバージョンに戻します。

```
$ aws ssm delete-document --name "SSM-SessionManagerRunShell" --document-
version 2
```

3. `describe-document` コマンドをもう一度実行して、ドキュメントのバージョンが戻ったことを確認します。

```
$ aws ssm describe-document --name "SSM-SessionManagerRunShell"
{
    "Document": {
        "Hash": "...",
        "HashType": "Sha256",
        "Name": "SSM-SessionManagerRunShell",
        "Owner": "123456789012",
        "CreatedDate": "2023-02-20T19:04:32.390000+00:00",
        "Status": "Active",
        "DocumentVersion": "1",
        "Parameters": [
            {
                "Name": "linuxcmd",
                "Type": "String",
                "Description": "The command to run on connection...",
                "DefaultValue": "if [ -d '/opt/parallelcluster' ]; then source /
opt/parallelcluster/cfnconfig; sudo su - $cfn_cluster_user; fi; /bin/bash"
            }
        ],
        "PlatformTypes": [
            "Windows",
            "Linux",
            "MacOS"
        ],
        "DocumentType": "Session",
        "SchemaVersion": "1.0",
        "LatestVersion": "1",
        "DefaultVersion": "1",
        "DocumentFormat": "JSON",
        "Tags": []
    }
}
```

最新バージョンは 1 です。

を使用したクラスターの作成 AWS CloudFormation

AWS ParallelCluster CloudFormation カスタムリソースを使用してクラスターを作成する方法について説明します。詳細については、「AWS CloudFormation カスタムリソース」を参照してください。

を使用する場合 AWS ParallelCluster、 AWS ParallelCluster イメージとクラスターを作成または更新したときに作成された AWS リソースに対してのみ料金が発生します。詳細については、「AWS が使用する サービス AWS ParallelCluster」を参照してください。

前提条件:

- AWS CLI がインストールされ、設定されています。
- Amazon EC2 のキーペア。
- `pcluster` CLI を実行するために必要なアクセス許可を持つ IAM ロール。

CloudFormation クイック作成スタックによるクラスター作成

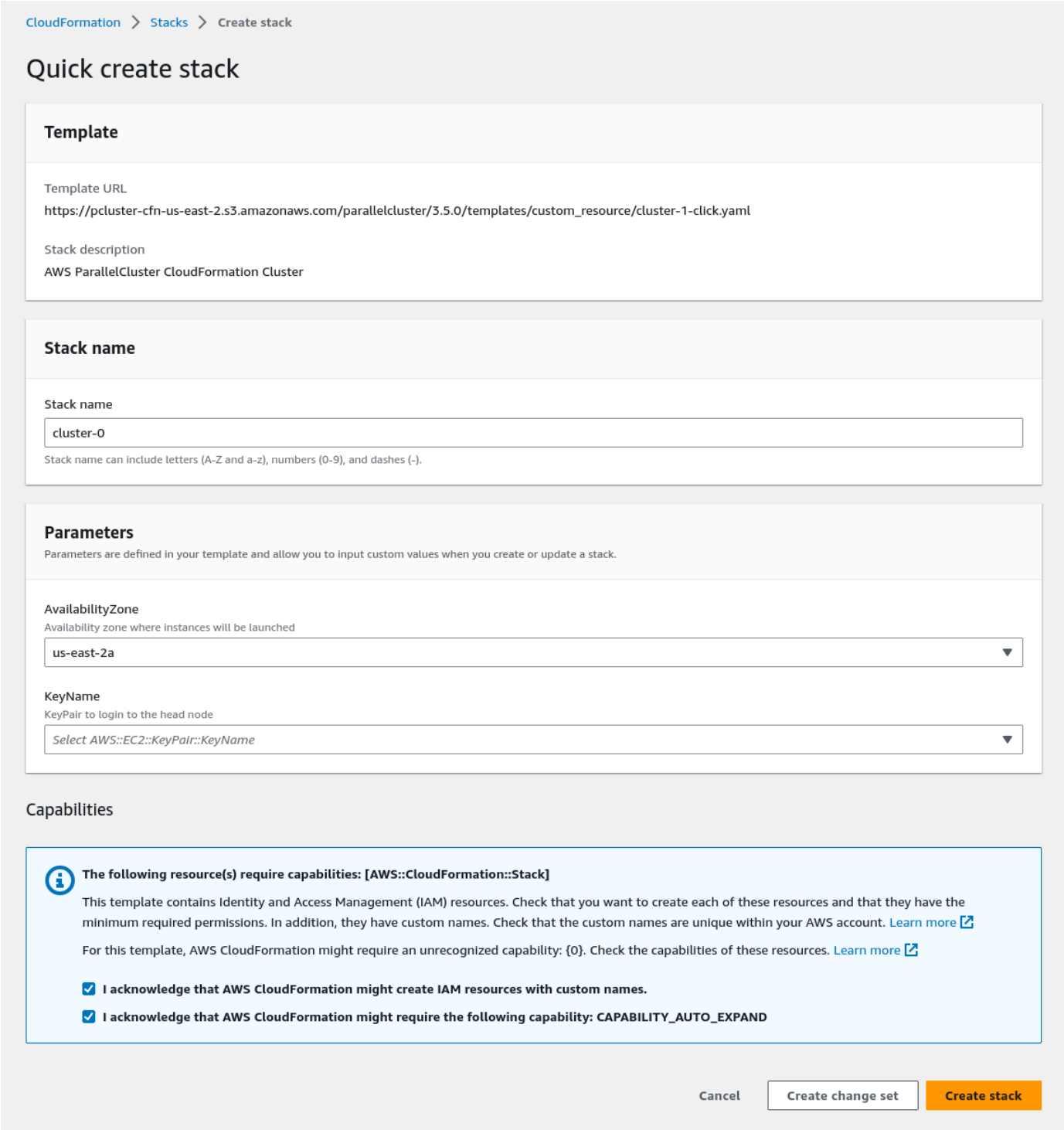
このチュートリアルでは、クイック作成スタックを使用して、クラスターと次の AWS リソースを作成する CloudFormation テンプレートをデプロイします。

- CloudFormation クイック作成スタックを使用して作成されたルート CloudFormation スタック。
- デフォルトポリシー、デフォルト VPC セットアップ、カスタムリソースプロバイダーを含むネストされた CloudFormation スタック。
- AWS ParallelCluster クラスタースタックの例と、 にログインしてジョブを実行できるクラスター。

を使用してクラスターを作成する AWS CloudFormation

1. AWS Management Consoleにサインインします。
2. CloudFormation クイック作成リンクを開いて、CloudFormation コンソールに以下のリソースを作成します。
   - クラスターヘッドノードとコンピューティングノードをそれぞれ実行するための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 を備えた、ネストされた CloudFormation スタック。
   - クラスターを管理するための AWS ParallelCluster カスタムリソースを持つネストされた CloudFormation スタック。
   - クラスターを管理するためのデフォルトポリシーを含むネストされた CloudFormation スタック。
   - ネストされたスタックのルート CloudFormation スタック。

- スSlurmケジューラと定義された数のコンピューティングノードを持つ AWS ParallelCluster クラスター。



3. [クイック作成スタック] の [パラメータ] セクションで、以下のパラメータの値を入力します。

   a. [KeyName] に、Amazon EC2 のキーペア名を入力します。
   b. [AvailabilityZone] では、クラスターノード用のアベイラビリティーゾーンを選択します (`us-east-1a` など)。
4. ページの下部で、アクセス機能を必要とすることを確認するチェックボックスを選択します。
5. [スタックの作成] を選択してください。
6. CloudFormation スタックが `CREATE_COMPLETE` の状態に達するまで待ちます。

コマンドラインインターフェイス (CLI) AWS CloudFormation を使用したクラスターの作成

このチュートリアルでは、CloudFormation の AWS コマンドラインインターフェイス (CLI) を使用して、クラスターを作成する CloudFormation テンプレートをデプロイします。

次の AWS リソースを作成します。

- CloudFormation クイック作成スタックを使用して作成されたルート CloudFormation スタック。
- デフォルトポリシー、デフォルト VPC セットアップ、カスタムリソースプロバイダーを含むネストされた CloudFormation スタック。
- AWS ParallelCluster クラスタースタックの例と、 にログインしてジョブを実行できるクラスター。

`####`など、`############`を独自の値に置き換えます。

を使用してクラスターを作成する AWS CloudFormation

1. 以下の内容で `cluster_template.yaml` という名前の CloudFormation テンプレートを作成します。

```
AWSTemplateFormatVersion: '2010-09-09'
Description: > AWS ParallelCluster CloudFormation Template

Parameters:
  KeyName:
    Description: KeyPair to login to the head node
    Type: AWS::EC2::KeyPair::KeyName

  AvailabilityZone:
```

```
    Description: Availability zone where instances will be launched
    Type: AWS::EC2::AvailabilityZone::Name
    Default: us-east-2a

Mappings:
  ParallelCluster:
    Constants:
      Version: 3.13.2

Resources:
  PclusterClusterProvider:
    Type: AWS::CloudFormation::Stack
    Properties:
      TemplateURL: !Sub
        - https://${AWS::Region}-aws-parallelcluster.s3.${AWS::Region}.
${AWS::URLSuffix}/parallelcluster/${Version}/templates/custom_resource/cluster.yaml
        - { Version: !FindInMap [ParallelCluster, Constants, Version] }

  PclusterVpc:
    Type: AWS::CloudFormation::Stack
    Properties:
      Parameters:
        PublicCIDR: 10.0.0.0/24
        PrivateCIDR: 10.0.16.0/20
        AvailabilityZone: !Ref AvailabilityZone
      TemplateURL: !Sub
        - https://${AWS::Region}-aws-parallelcluster.s3.${AWS::Region}.
${AWS::URLSuffix}/parallelcluster/${Version}/templates/networking/public-private-
${Version}.cfn.json
        - { Version: !FindInMap [ParallelCluster, Constants, Version ] }

  PclusterCluster:
    Type: Custom::PclusterCluster
    Properties:
      ServiceToken: !GetAtt [ PclusterClusterProvider , Outputs.ServiceToken ]
      ClusterName: !Sub 'c-${AWS::StackName}'
      ClusterConfiguration:
        Image:
          Os: alinux2
        HeadNode:
          InstanceType: t2.medium
          Networking:
            SubnetId: !GetAtt [ PclusterVpc , Outputs.PublicSubnetId ]
          Ssh:
```

```
            KeyName: !Ref KeyName
        Scheduling:
          Scheduler: slurm
          SlurmQueues:
          - Name: queue0
            ComputeResources:
            - Name: queue0-cr0
              InstanceType: t2.micro
            Networking:
              SubnetIds:
              -  !GetAtt [ PclusterVpc , Outputs.PrivateSubnetId ]
Outputs:
  HeadNodeIp:
    Description: The Public IP address of the HeadNode
    Value: !GetAtt [ PclusterCluster, headNode.publicIpAddress ]
```

2. 次の AWS CLI コマンドを実行して、クラスターの作成と管理のために CloudFormation スタックをデプロイします。

```
$ aws cloudformation deploy --template-file ./cluster_template.yaml \
  --stack-name mycluster \
  --parameter-overrides KeyName=keypair \
                        AvailabilityZone=us-east-2b \
  --capabilities CAPABILITY_NAMED_IAM CAPABILITY_AUTO_EXPAND
```

CloudFormation クラスターの出力を表示する

CloudFormation クラスターの出力を表示すると、クラスターの詳細情報を得ることができます。追加された `ValidationMessages` プロパティにより、クラスターの作成および更新操作からの検証メッセージにアクセスできます。

1. CloudFormation コンソールに移動し、 AWS ParallelCluster カスタムリソースを含むスタックを選択します。
2. [スタックの詳細] を選択し、[出力] タブを選択します。

Outputs (2)

Search outputs

Key	Value	Description
HeadNodeIp	1.2.3.4	The Public IP address of the HeadNode
ValidationMessages	[{"level": "WARNING", "type": "KeyPairValidator", "message": "If you do not specify a key pair, you can't connect to the instance unless you choose an AMI that is configured to allow users another way to log in"}]	Any warnings from cluster create or update operations.

検証メッセージは切り詰められることがあります。ログの取得の詳細については、「AWS ParallelCluster トラブルシューティング」を参照してください。

クラスターへのアクセス

クラスターへのアクセス

クラスターヘッドノードの ssh

1. CloudFormation スタックのデプロイが完了したら、次のコマンドを使用してヘッドノードの IP アドレスを取得します。

```
$ HEAD_NODE_IP=$(aws cloudformation describe-stacks --stack-name=mycluster --query
 "Stacks|[0].Outputs[?OutputKey=='HeadNodeIp']|[0].OutputValue" --output=text)
```

   CloudFormation コンソールのクラスタースタックの [出力] タブにある [HeadNodeIP] パラメータからヘッドノードの IP アドレスを取得することもできます。

   ヘッドノードの IP アドレスは、クラスター CloudFormation テンプレートの `Outputs` セクションに、特にこのサンプルクラスター用に追加されたため、ここで確認できます。

2. 次のコマンドを実行して、クラスターヘッドノードに接続します。

```
$ ssh -i keyname.pem ec2-user@$HEAD_NODE_IP
```

クリーンアップ

クラスターを削除します。

1. 次の AWS CLI コマンドを実行して、CloudFormation スタックとクラスターを削除します。

```
$ aws cloudformation delete-stack --stack-name=mycluster
```

2. 次のコマンドを実行してスタックの削除ステータスを確認します。

```
$ aws cloudformation describe-stacks --stack-name=mycluster
```

Terraform を使用して ParallelCluster API をデプロイする

このチュートリアルでは、ParallelCluster API をデプロイするためのシンプルな Terraform プロジェクトを定義します。

前提条件

- Terraform v1.5.7+ をインストール済みである。
- ParallelCluster API をデプロイするアクセス許可を持つ IAM ロール。「the section called “必要なアクセス許可”」を参照してください。

Terraform プロジェクトを定義する

このチュートリアルでは、Terraform プロジェクトを定義します。

1. `my-pcluster-api` という名前のディレクトリを作成します。

   作成するすべてのファイルは、このディレクトリに入ります。

2. ファイルを作成して`provider.tf`、 AWS プロバイダーを設定します。

```
provider "aws" {
  region  = var.region
  profile = var.profile
}
```

3. ParallelCluster モジュールを使用してリソースを定義するファイル `main.tf` を作成します。

```
module "parallelcluster_pcluster_api" {
  source = "aws-tf/parallelcluster/aws//modules/pcluster_api"
  version = "1.1.0"
```

```
  region                = var.region
  api_stack_name        = var.api_stack_name
  api_version           = var.api_version

  parameters = {
    EnableIamAdminAccess = "true"
  }
}
```

4. このプロジェクトに挿入できる変数を定義するファイル `variables.tf` を作成します。

```
variable "region" {
  description = "The region the ParallelCluster API is deployed in."
  type        = string
  default     = "us-east-1"
}

variable "profile" {
  type        = string
  description = "The AWS profile used to deploy the clusters."
  default     = null
}

variable "api_stack_name" {
  type        = string
  description = "The name of the CloudFormation stack used to deploy the
 ParallelCluster API."
  default     = "ParallelCluster"
}

variable "api_version" {
  type        = string
  description = "The version of the ParallelCluster API."
}
```

5. 変数の任意の値を設定するファイル `terraform.tfvars` を作成します。

   以下のファイルは、スタック名 `us-east-1`を使用して ParallelCluster API 3.11.1 を にデプロイします`MyParallelClusterAPI-3111`。この ParallelCluster API デプロイは、スタック名を使用して参照できます。

> **Note**
>
> 次のコードapi_versionの割り当ては、サポートされている任意の AWS ParallelCluster バージョンに置き換えることができます。

```
region = "us-east-1"
api_stack_name = "MyParallelClusterAPI-3111"
api_version = "3.11.1"
```

6. このプロジェクトで返す出力を定義するファイル outputs.tf を作成します。

```
output "pcluster_api_stack_outputs" {
  value = module.parallelcluster_pcluster_api.stack_outputs
}
```

プロジェクトディレクトリ:

```
my-pcluster-api
### main.tf - Terraform entrypoint to define the resources using the
 ParallelCluster module.
### outputs.tf - Defines the outputs returned by Terraform.
### providers.tf - Configures the AWS provider.
### terraform.tfvars - Set the arbitrary values for the variables, i.e. region,
 PCAPI version, PCAPI stack name
### variables.tf - Defines the variables, e.g. region, PCAPI version, PCAPI stack
 name.
```

API をデプロイする

API をデプロイするには、標準の Terraform コマンドを順に実行します。

1. プロジェクトをビルドします。

```
terraform init
```

2. デプロイプランを定義します。

```
terraform plan -out tfplan
```

3. プランをデプロイします。

```
terraform apply tfplan
```

必要なアクセス許可

Terraform を使用して ParallelCluster API をデプロイするには、以下のアクセス許可が必要です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "cloudformation:DescribeStacks",
                "cloudformation:GetTemplate"
            ],
            "Resource": "arn:PARTITION:cloudformation:REGION:ACCOUNT:stack/*",
            "Effect": "Allow",
            "Sid": "CloudFormationRead"
        },
        {
            "Action": [
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:CreateChangeSet"
            ],
            "Resource": "arn:PARTITION:cloudformation:REGION:ACCOUNT:stack/
MyParallelClusterAPI*",
            "Effect": "Allow",
            "Sid": "CloudFormationWrite"
        },
        {
            "Action": [
                "cloudformation:CreateChangeSet"
            ],
            "Resource": [
                "arn:PARTITION:cloudformation:REGION:aws:transform/Include",
                "arn:PARTITION:cloudformation:REGION:aws:transform/
Serverless-2016-10-31"
```

```
            ],
            "Effect": "Allow",
            "Sid": "CloudFormationTransformWrite"
        },
        {
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:PARTITION:s3:::*-aws-parallelcluster/parallelcluster/*/api/
ParallelCluster.openapi.yaml",
                "arn:PARTITION:s3:::*-aws-parallelcluster/parallelcluster/*/layers/aws-
parallelcluster/lambda-layer.zip"
            ],
            "Effect": "Allow",
            "Sid": "S3ParallelClusterArtifacts"
        },
        {
            "Action": [
                "iam:CreateRole",
                "iam:DeleteRole",
                "iam:GetRole",
                "iam:CreatePolicy",
                "iam:DeletePolicy",
                "iam:GetPolicy",
                "iam:GetRolePolicy",
                "iam:AttachRolePolicy",
                "iam:DetachRolePolicy",
                "iam:PutRolePolicy",
                "iam:DeleteRolePolicy",
                "iam:ListPolicyVersions"
            ],
            "Resource": [
                "arn:PARTITION:iam::ACCOUNT:role/*",
                "arn:PARTITION:iam::ACCOUNT:policy/*"
            ],
            "Effect": "Allow",
            "Sid": "IAM"
        },
        {
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
```

```
                "arn:PARTITION:iam::ACCOUNT:role/ParallelClusterLambdaRole-*",
                "arn:PARTITION:iam::ACCOUNT:role/APIGatewayExecutionRole-*"
            ],
            "Effect": "Allow",
            "Sid": "IAMPassRole"
        },
        {
            "Action": [
                "lambda:CreateFunction",
                "lambda:DeleteFunction",
                "lambda:GetFunction",
                "lambda:PublishLayerVersion",
                "lambda:DeleteLayerVersion",
                "lambda:GetLayerVersion",
                "lambda:TagResource",
                "lambda:UntagResource"
            ],
            "Resource": [
                "arn:PARTITION:lambda:REGION:ACCOUNT:layer:PCLayer-*",
                "arn:PARTITION:lambda:REGION:ACCOUNT:function:*-
ParallelClusterFunction-*"
            ],
            "Effect": "Allow",
            "Sid": "Lambda"
        },
        {
            "Action": [
                "logs:CreateLogGroup",
                "logs:DeleteLogGroup",
                "logs:DescribeLogGroups",
                "logs:PutRetentionPolicy",
                "logs:TagLogGroup",
                "logs:UntagLogGroup"
            ],
            "Resource": [
                "arn:PARTITION:logs:REGION:ACCOUNT:log-group:/aws/lambda/*-
ParallelClusterFunction-*"
            ],
            "Effect": "Allow",
            "Sid": "Logs"
        },
        {
            "Action": [
                "apigateway:DELETE",
```

```
                "apigateway:GET",
                "apigateway:PATCH",
                "apigateway:POST",
                "apigateway:PUT",
                "apigateway:UpdateRestApiPolicy"
            ],
            "Resource": [
                "arn:PARTITION:apigateway:REGION::/restapis",
                "arn:PARTITION:apigateway:REGION::/restapis/*",
                "arn:PARTITION:apigateway:REGION::/tags/*"
            ],
            "Effect": "Allow",
            "Sid": "APIGateway"
        }
    ]
}
```

Terraform を使用したクラスターの作成

を使用する場合 AWS ParallelCluster、 AWS ParallelCluster イメージとクラスターを作成または更新するときに作成された AWS リソースに対してのみ料金が発生します。詳細については、「the section called “AWS が使用する サービス AWS ParallelCluster”」を参照してください。

前提条件

- Terraform v1.5.7+ をインストール済みである。
- the section called “AWS ParallelCluster API” v3.8.0+ がアカウントにデプロイ済みである。「the section called “Terraform を使用して ParallelCluster API をデプロイする”」を参照してください。
- ParallelCluster API を呼び出すアクセス許可を持つ IAM ロール。「必要な IAM 許可」を参照してください。

Terraform プロジェクトを定義する

このチュートリアルでは、クラスターをデプロイするためのシンプルな Terraform プロジェクトを定義します。

1. `my-clusters` という名前のディレクトリを作成します。

   作成するすべてのファイルは、このディレクトリに入ります。

2. ParallelCluster プロバイダーをインポートするファイル `terraform.tf` を作成します。

```
terraform {
  required_version = ">= 1.5.7"
  required_providers {
    aws-parallelcluster = {
      source  = "aws-tf/aws-parallelcluster"
      version = "~> 1.0"
    }
  }
}
```

3. ParallelCluster と AWS プロバイダーを設定するファイル `providers.tf` を作成します。

```
provider "aws" {
  region  = var.region
  profile = var.profile
}

provider "aws-parallelcluster" {
  region         = var.region
  profile        = var.profile
  api_stack_name = var.api_stack_name
  use_user_role  = true
}
```

4. ParallelCluster モジュールを使用してリソースを定義するファイル `main.tf` を作成します。

```
module "pcluster" {
  source  = "aws-tf/parallelcluster/aws"
  version = "1.1.0"

  region                = var.region
  api_stack_name        = var.api_stack_name
  api_version           = var.api_version
  deploy_pcluster_api   = false

  template_vars         = local.config_vars
  cluster_configs       = local.cluster_configs
  config_path           = "config/clusters.yaml"
}
```

5. 複数のクラスターを Terraform ローカル変数として定義するファイル `clusters.tf` を作成します。

> **Note**
>
> `cluster_config` 要素内で複数のクラスターを定義できます。クラスターごとに、ローカル変数内でクラスタープロパティを明示的に定義する (「`DemoCluster01`」を参照) ことも、外部ファイルを参照する (「`DemoCluster02`」を参照) こともできます。

設定要素内で設定できるクラスタープロパティを確認するには、「the section called “クラスター設定ファイル”」を参照してください。

クラスターを作成するために設定できるオプションを確認するには、「the section called “`pcluster create-cluster`”」を参照してください。

```
locals {
  cluster_configs = {
    DemoCluster01 : {
      region : local.config_vars.region
      rollbackOnFailure : false
      validationFailureLevel : "WARNING"
      suppressValidators : [
        "type:KeyPairValidator"
      ]
      configuration : {
        Region : local.config_vars.region
        Image : {
          Os : "alinux2"
        }
        HeadNode : {
          InstanceType : "t3.small"
          Networking : {
            SubnetId : local.config_vars.subnet
          }
          Iam : {
            AdditionalIamPolicies : [
              { Policy : "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore" }
            ]
          }
        }
```

```
        Scheduling : {
          Scheduler : "slurm"
          SlurmQueues : [{
            Name : "queue1"
            CapacityType : "ONDEMAND"
            Networking : {
              SubnetIds : [local.config_vars.subnet]
            }
            Iam : {
              AdditionalIamPolicies : [
                { Policy : "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore" }
              ]
            }
            ComputeResources : [{
              Name : "compute"
              InstanceType : "t3.small"
              MinCount : "1"
              MaxCount : "4"
            }]
          }]
          SlurmSettings : {
            QueueUpdateStrategy : "TERMINATE"
          }
        }
      }
    }
    DemoCluster02 : {
      configuration : "config/cluster_config.yaml"
    }
  }
}
```

6. 複数のクラスターを YAML 設定として定義するファイル `config/clusters.yaml` を作成します。

```
DemoCluster03:
  region: ${region}
  rollbackOnFailure: true
  validationFailureLevel: WARNING
  suppressValidators:
    - type:KeyPairValidator
  configuration: config/cluster_config.yaml
DemoCluster04:
```

```
  region: ${region}
  rollbackOnFailure: false
  configuration: config/cluster_config.yaml
```

7. Terraform 変数を挿入できる標準の ParallelCluster 設定ファイルであるファイル `config/cluster_config.yaml` を作成します。

   設定要素内で設定できるクラスタープロパティを確認するには、「the section called “クラスター設定ファイル”」を参照してください。

```
Region: ${region}
Image:
 Os: alinux2
HeadNode:
 InstanceType: t3.small
 Networking:
   SubnetId: ${subnet}
 Iam:
   AdditionalIamPolicies:
     - Policy: arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
Scheduling:
 Scheduler: slurm
 SlurmQueues:
   - Name: queue1
     CapacityType: ONDEMAND
     Networking:
       SubnetIds:
         - ${subnet}
     Iam:
       AdditionalIamPolicies:
         - Policy: arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
     ComputeResources:
       - Name: compute
         InstanceType: t3.small
         MinCount: 1
         MaxCount: 5
 SlurmSettings:
   QueueUpdateStrategy: TERMINATE
```

8. クラスター設定に挿入できる変数を定義するファイル `clusters_vars.tf` を作成します。

   このファイルにより、リージョンやサブネットなど、クラスター設定で使用できる動的な値を定義できます。

この例では、プロジェクト変数から値を直接取得しますが、値を決定するにはカスタムロジックの使用が必要になる場合があります。

```
locals {
  config_vars = {
    subnet = var.subnet_id
    region = var.cluster_region
  }
}
```

9. このプロジェクトに挿入できる変数を定義するファイル `variables.tf` を作成します。

```
variable "region" {
  description = "The region the ParallelCluster API is deployed in."
  type        = string
  default     = "us-east-1"
}

variable "cluster_region" {
  description = "The region the clusters will be deployed in."
  type        = string
  default     = "us-east-1"
}

variable "profile" {
  type        = string
  description = "The AWS profile used to deploy the clusters."
  default     = null
}

variable "subnet_id" {
  type        = string
  description = "The id of the subnet to be used for the ParallelCluster
 instances."
}

variable "api_stack_name" {
  type        = string
  description = "The name of the CloudFormation stack used to deploy the
 ParallelCluster API."
  default     = "ParallelCluster"
}
```

```
variable "api_version" {
  type        = string
  description = "The version of the ParallelCluster API."
}
```

10. 変数の任意の値を設定するファイル `terraform.tfvars` を作成します。

    以下のファイルはsubnet-123456789、スタック名 us-east-1で に既にデプロイされている既存の ParallelCluster API 3.11.1 を使用して、サブネット eu-west-1内の にクラスターをデプロイしますMyParallelClusterAPI-3111。

```
region = "us-east-1"
api_stack_name = "MyParallelClusterAPI-3111"
api_version = "3.11.1"

cluster_region = "eu-west-1"
subnet_id = "subnet-123456789"
```

11. このプロジェクトで返す出力を定義するファイル `outputs.tf` を作成します。

```
output "clusters" {
  value = module.pcluster.clusters
}
```

プロジェクトディレクトリ:

```
my-clusters
### config
#   ### cluster_config.yaml - Cluster configuration, where terraform variables can
 be injected..
#   ### clusters.yaml - File listing all the clusters to deploy.
### clusters.tf - Clusters defined as Terraform local variables.
### clusters_vars.tf - Variables that can be injected into cluster configurations.
### main.tf - Terraform entrypoint where the ParallelCluster module is configured.
### outputs.tf - Defines the cluster as a Terraform output.
### providers.tf - Configures the providers: ParallelCluster and AWS.
### terraform.tf - Import the ParallelCluster provider.
### terraform.tfvars - Defines values for variables, e.g. region, PCAPI stack name.
### variables.tf - Defines the variables, e.g. region, PCAPI stack name.
```

クラスターをデプロイする

クラスターをデプロイするには、標準の Terraform コマンドを順に実行します。

> **Note**
> この例では、アカウントに ParallelCluster API を既にデプロイしていることを前提としています。

1. プロジェクトをビルドします。

```
terraform init
```

2. デプロイプランを定義します。

```
terraform plan -out tfplan
```

3. プランをデプロイします。

```
terraform apply tfplan
```

クラスターを使用して ParallelCluster API をデプロイする

ParallelCluster API をまだデプロイしておらず、クラスターを使用してデプロイする場合は、以下のファイルを変更します。

- `main.tf`

```
module "pcluster" {
  source  = "aws-tf/aws/parallelcluster"
  version = "1.0.0"

  region                = var.region
  api_stack_name        = var.api_stack_name
  api_version           = var.api_version
  deploy_pcluster_api   = true
  parameters = {
    EnableIamAdminAccess = "true"
  }
```

```
  template_vars         = local.config_vars
  cluster_configs       = local.cluster_configs
  config_path           = "config/clusters.yaml"
}
```

- `providers.tf`

```
provider "aws-parallelcluster" {
  region   = var.region
  profile  = var.profile
  endpoint = module.pcluster.pcluster_api_stack_outputs.ParallelClusterApiInvokeUrl
  role_arn = module.pcluster.pcluster_api_stack_outputs.ParallelClusterApiUserRole
}
```

必要なアクセス許可

Terraform を使用してクラスターをデプロイするには、以下のアクセス許可が必要です。

- ParallelCluster API とのやり取りを担当する ParallelCluster API ロールを引き受ける
- ParallelCluster API の AWS CloudFormation スタックを記述して、そのスタックが存在することを確認し、そのパラメータと出力を取得します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "sts:AssumeRole",
            "Resource": "arn:PARTITION:iam::ACCOUNT:role/PCAPIUserRole-*",
            "Effect": "Allow",
            "Sid": "AssumePCAPIUserRole"
        },
        {
            "Action": [
                "cloudformation:DescribeStacks"
            ],
            "Resource": "arn:PARTITION:cloudformation:REGION:ACCOUNT:stack/*",
            "Effect": "Allow",
            "Sid": "CloudFormation"
        }
```

```
    ]
}
```

Terraform を使用したカスタム AMI の作成

を使用する場合 AWS ParallelCluster、 AWS ParallelCluster イメージとクラスターを作成または更新するときに作成された AWS リソースに対してのみ料金が発生します。詳細については、「the section called “AWS が使用する サービス AWS ParallelCluster”」を参照してください。

前提条件

- Terraform v1.5.7+ をインストール済みである。
- the section called “AWS ParallelCluster API” v3.8.0+ がアカウントにデプロイ済みである。「the section called “Terraform を使用したクラスターの作成”」を参照してください。
- ParallelCluster API を呼び出すアクセス許可を持つ IAM ロール。「the section called “必要なアクセス許可”」を参照してください。

Terraform プロジェクトを定義する

このチュートリアルでは、ParallelCluster のカスタム AMI をデプロイするためのシンプルな Terraform プロジェクトを定義します。

1. `my-amis` という名前のディレクトリを作成します。

   作成するすべてのファイルは、このディレクトリに入ります。

2. ParallelCluster プロバイダーをインポートするファイル `terraform.tf` を作成します。

```
terraform {
  required_version = ">= 1.5.7"
  required_providers {
    aws-parallelcluster = {
      source  = "aws-tf/aws-parallelcluster"
      version = "~> 1.0"
    }
  }
}
```

3. ParallelCluster と AWS プロバイダーを設定するファイル `providers.tf` を作成します。

```
provider "aws" {
  region  = var.region
  profile = var.profile
}

provider "aws-parallelcluster" {
  region         = var.region
  profile        = var.profile
  api_stack_name = var.api_stack_name
  use_user_role  = true
}
```

4. ParallelCluster モジュールを使用してリソースを定義するファイル `main.tf` を作成します。

   `image_configuration` 要素内で設定できるイメージプロパティを確認するには、「the section called “ビルドイメージの設定ファイル”」を参照してください。

   イメージ作成用に設定できるオプション (`image_id` や `rollback_on_failure` など) については、「the section called “`pcluster build-image`”」を参照してください。

```
data "aws-parallelcluster_list_official_images" "parent_image" {
  region = var.region
  os = var.os
  architecture = var.architecture
}

resource "aws-parallelcluster_image" "demo01" {
  image_id            = "demo01"
  image_configuration = yamlencode({
    "Build":{
      "InstanceType": "c5.2xlarge",
      "ParentImage": data.aws-
parallelcluster_list_official_images.parent_image.official_images[0].amiId,
      "UpdateOsPackages": {"Enabled": false}
    }
  })
  rollback_on_failure = false
}
```

5. このプロジェクトに挿入できる変数を定義するファイル `variables.tf` を作成します。

```
variable "region" {
```

```
  description = "The region the ParallelCluster API is deployed in."
  type        = string
  default     = "us-east-1"
}

variable "profile" {
  type        = string
  description = "The AWS profile used to deploy the clusters."
  default     = null
}

variable "api_stack_name" {
  type        = string
  description = "The name of the CloudFormation stack used to deploy the
 ParallelCluster API."
  default     = "ParallelCluster"
}

variable "api_version" {
  type        = string
  description = "The version of the ParallelCluster API."
}

variable "os" {
  type        = string
  description = "The OS of the ParallelCluster image."
}

variable "architecture" {
  type        = string
  description = "The architecture of the ParallelCluster image."
}
```

6. ファイルを作成して`terraform.tfvars`、変数の任意の値を設定します。

   以下の ファイルでは、スタック名 `us-east-1`で に既にデプロイされている既存の ParallelCluster API 3.11.1 を使用して、x86_64 アーキテクチャ用の Amazon Linux 2 `us-east-1`に基づいて にカスタム AMI をデプロイします`MyParallelClusterAPI-3111`。

```
region = "us-east-1"
api_stack_name = "MyParallelClusterAPI-3111"
api_version = "3.11.1"
```

```
os = "alinux2"
architecture = "x86_64"
```

7. このプロジェクトで返す出力を定義するファイル `outputs.tf` を作成します。

```
output "parent_image" {
  value = data.aws-
parallelcluster_list_official_images.parent_image.official_images[0]
}

output "custom_image" {
  value = aws-parallelcluster_image.demo01
}
```

プロジェクトディレクトリ:

```
my-amis
### main.tf - Terraform entrypoint where the ParallelCluster module is configured.
### outputs.tf - Defines the cluster as a Terraform output.
### providers.tf - Configures the providers: ParallelCluster and AWS.
### terraform.tf - Import the ParallelCluster provider.
### terraform.tfvars - Defines values for variables, e.g. region, PCAPI stack name.
### variables.tf - Defines the variables, e.g. region, PCAPI stack name.
```

AMI をデプロイする

AMI をデプロイするには、標準の Terraform コマンドを順に実行します。

1. プロジェクトをビルドします。

```
terraform init
```

2. デプロイプランを定義します。

```
terraform plan -out tfplan
```

3. プランをデプロイします。

```
terraform apply tfplan
```

必要なアクセス許可

Terraform でカスタム AMI をデプロイするには、以下のアクセス許可が必要です。

- ParallelCluster API とのやり取りを担当する ParallelCluster API ロールを引き受ける
- ParallelCluster API の AWS CloudFormation スタックを記述して、そのスタックが存在することを確認し、そのパラメータと出力を取得します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "sts:AssumeRole",
            "Resource": "arn:PARTITION:iam::ACCOUNT:role/PCAPIUserRole-*",
            "Effect": "Allow",
            "Sid": "AssumePCAPIUserRole"
        },
        {
            "Action": [
                "cloudformation:DescribeStacks"
            ],
            "Resource": "arn:PARTITION:cloudformation:REGION:ACCOUNT:stack/*",
            "Effect": "Allow",
            "Sid": "CloudFormation"
        }
    ]
}
```

AWS ParallelCluster UI と Identity Center の統合

このチュートリアルの目的は、 AWS ParallelCluster クラスターと共有できる Active Directory のユーザーを統合するシングルサインオンソリューションのために AWS ParallelCluster UI を IAM Identity Center と統合する方法を示すことです。

を使用する場合 AWS ParallelCluster、 AWS ParallelCluster イメージとクラスターを作成または更新するときに作成された AWS リソースに対してのみ料金が発生します。詳細については、「AWS が使用する サービス AWS ParallelCluster」を参照してください。

前提条件:

- こちらの手順に従ってインストールできる既存の AWS ParallelCluster UI
- 既存の Managed Active Directory、できれば統合 AWS ParallelClusterにも使用するもの。

IAM Identity Center を有効にする

既に AWS Managed Microsoft AD (Active Directory) に接続されているアイデンティティセンターがある場合は、それを使用できます。「IAM アイデンティティセンターへのアプリケーションの追加」セクションに進んでください。

にまだ ID センターが接続されていない場合は AWS Managed Microsoft AD、以下の手順に従ってセットアップします。

アイデンティティセンターを有効にする

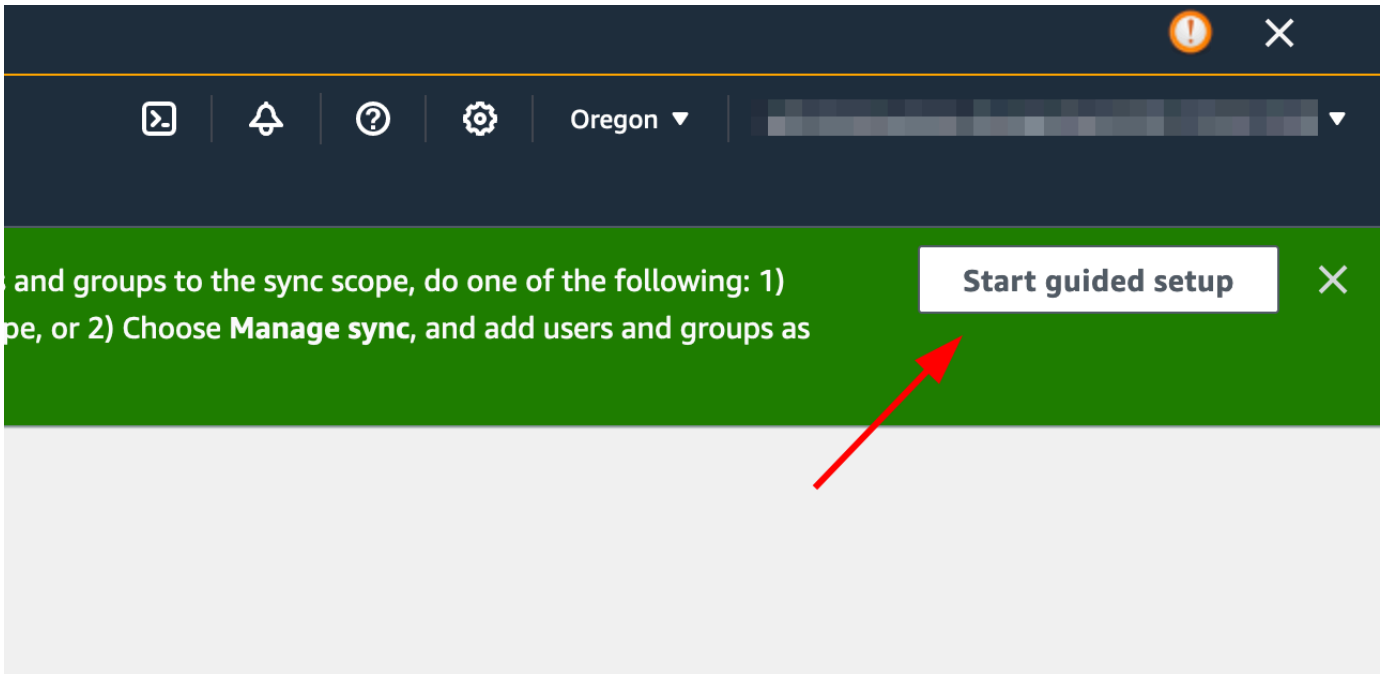
1. コンソールで、IAM アイデンティティセンターに移動します (があるリージョンにいることを確認してください) AWS Managed Microsoft AD。
2. [有効化] ボタンをクリックすると、組織を有効にするかどうかを確認するメッセージが表示される場合があります。これは必須事項であるため、有効にすることを選択します。注: これにより、アカウントの管理者に対して、リンクをたどって確認する必要があるという確認メッセージが E メールで送信されます。

アイデンティティセンターを Managed AD に接続する

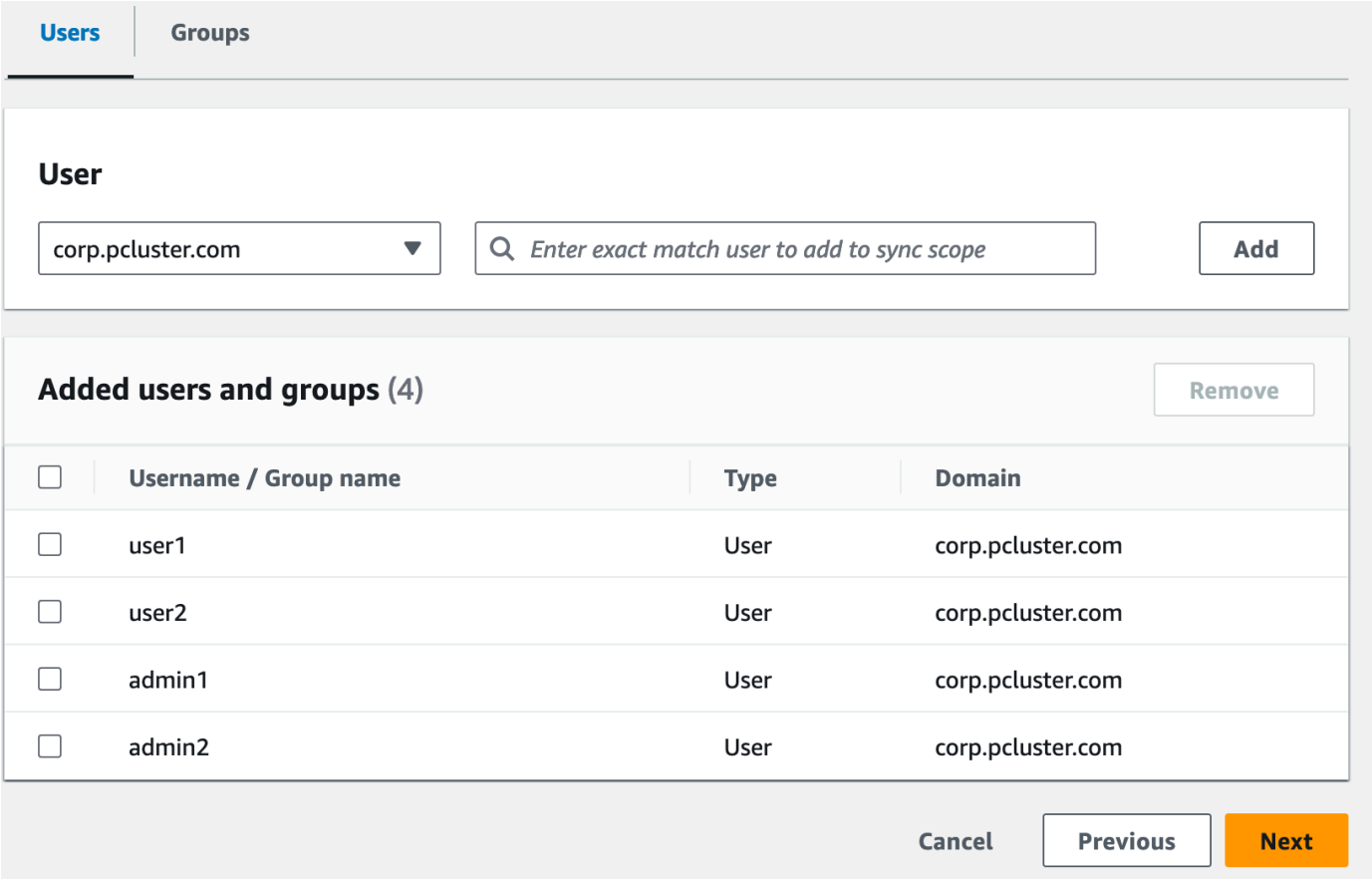
1. アイデンティティセンターを有効にした後の次のページに、推奨されるセットアップ手順が表示されます。ステップ 1 で [アイデンティティソースを選択] を選択します。
2. [アイデンティティソース] セクションで、[アクション] ドロップダウンメニュー (右上) をクリックし、[アイデンティティソースを変更] を選択します。
3. [Active Directory] を選択します。
4. [既存のディレクトリ] で、ディレクトリを選択します。
5. [次へ] をクリックします。
6. 変更を確認して下にスクロールし、テキストボックスに「ACCEPT」と入力して確認します。次に、[アイデンティティソースを変更] をクリックします。
7. 変更が完了するまで待つと、上部に緑色のバナーが表示されます。

ユーザーとグループをアイデンティティセンターに同期する

1. 緑色のバナーで、[ガイド付き設定を開始] (右上のボタン) をクリックします。

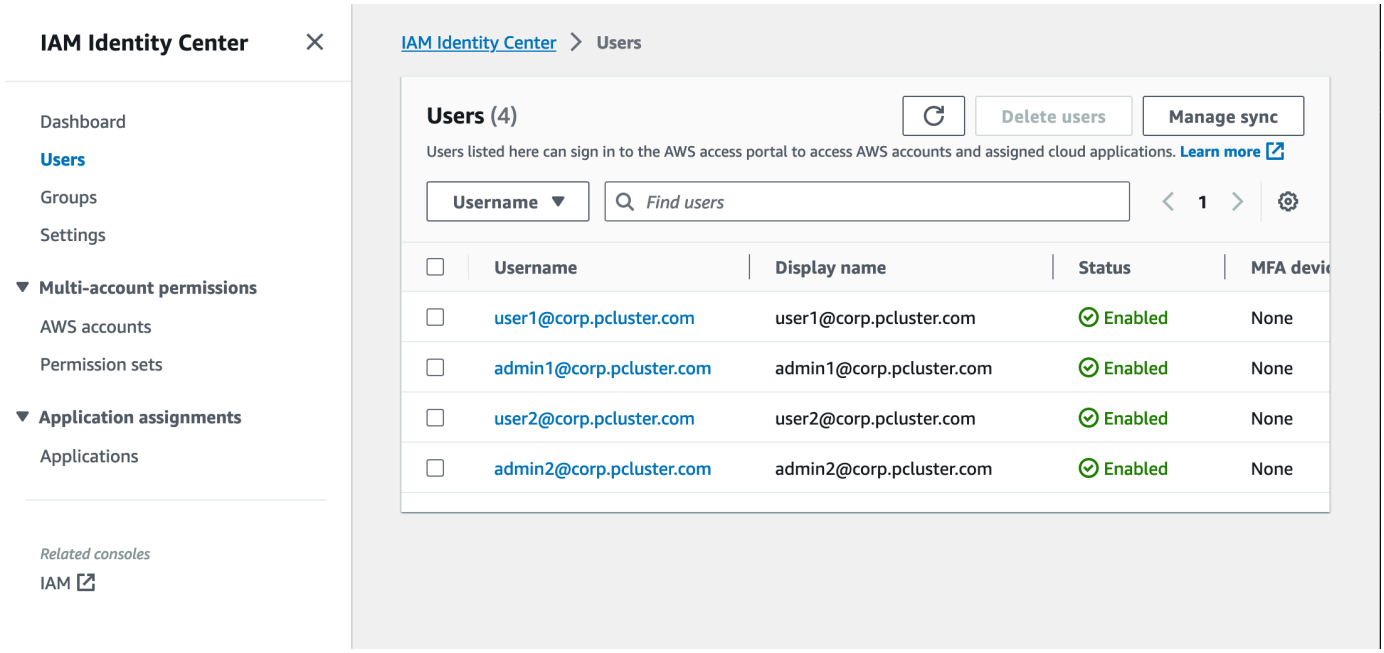


2. [属性マッピングを設定] で、[次へ] をクリックします。
3. [同期の範囲を設定] セクションで、アイデンティティセンターに同期するユーザーの名前を入力し、[追加] をクリックします。
4. ユーザーとグループの追加が完了したら、[次へ] をクリックします。



5. 変更内容を確認してから、[設定を保存] をクリックします。
6. 次の画面にユーザーが同期されていないという警告が表示された場合は、右上の [同期を再開] ボタンを選択します。
7. 次に、ユーザーを有効にするには、左側の [ユーザー] タブでユーザーを選択し、[ユーザーアクセスを有効化] > [ユーザーアクセスを有効化] をクリックします。

   注: 上部に警告バナーが表示されている場合は、[同期を再開] を選択し、ユーザーが同期するまで待つ必要があります (ユーザーが同期されたかどうかを確認するには、[更新] ボタンを試してください)。



アプリケーションを IAM アイデンティティセンターに追加する

ユーザーを IAM アイデンティティセンターと同期したら、新しいアプリケーションを追加する必要があります。これにより、IAM アイデンティティセンターポータルから使用できる SSO 対応アプリケーションを設定します。この場合、アプリケーションとして AWS ParallelCluster UI を追加し、IAM アイデンティティセンターを ID プロバイダーにします。

次のステップでは、IAM Identity Center のアプリケーションとして AWS ParallelCluster UI を追加します。 AWS ParallelCluster UI は、ユーザーがクラスターを管理するのに役立つウェブポータルです。詳細については、「AWS ParallelCluster UI」を参照してください。

アイデンティティセンターでアプリケーションをセットアップする

1. [IAM アイデンティティセンター] > [アプリケーション] (左側のメニューバー) で [アプリケーション] をクリックします。
2. [アプリケーションを追加] をクリックします。
3. [カスタム SAML 2.0 アプリケーションを追加] を選択します。
4. [次へ] をクリックします。
5. 使用する表示名と説明を選択します (PCUI や AWS ParallelCluster UI など)
6. [IAM アイデンティティセンターのメタデータ] で、IAM アイデンティティセンター SAML メタデータファイルへのリンクをコピーして保存します。これは、後ほどウェブアプリケーションで SSO を設定するときに使用します。

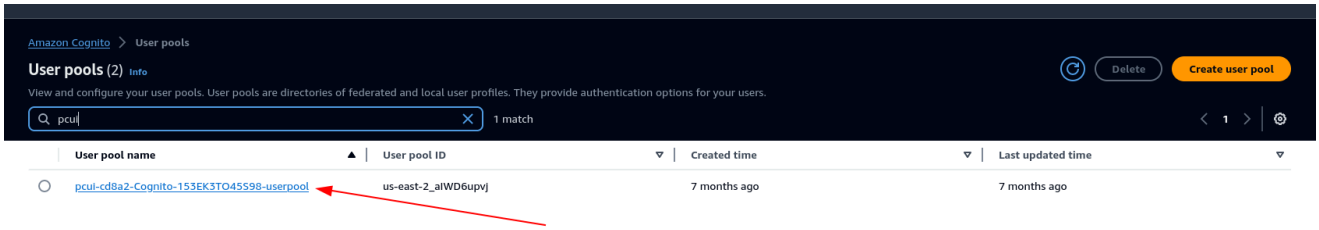
7. [アプリケーションのプロパティ] の [アプリケーション開始 URL] で、PCUI アドレスを入力します。これを確認するには、CloudFormation コンソールに移動して、PCUI に対応するスタック (例: parallelcluster-ui) を選択し、[出力] タブに移動して ParallelClusterUIUrl を見つけます。

   例: https://m2iwazsi1j.execute-api.us-east-1.amazonaws.com

8. [アプリケーションメタデータ] で [メタデータの値を手動で入力] を選択します。次に、以下の値を指定します。

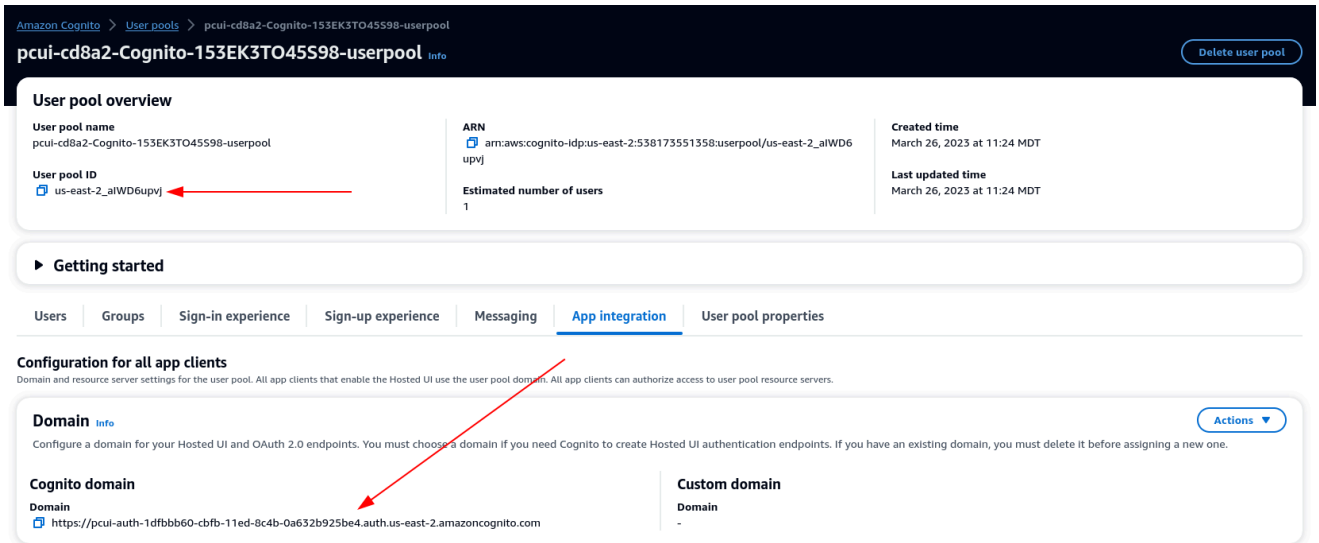
   a. 重要: domain-prefix、region、userpool-id の値は、環境固有の情報に置き換えてください。

   b. domain-prefix、region、userpool-id は、[Amazon Cognito] > [ユーザープールコンソール] を開いて取得できます。



   c. PCUI に対応するユーザープールを選択します (このユーザープールの名前は pcui-cd8a2-Cognito-153EK3TO45S98-userpool のように表示されます)。

   d. [アプリケーションの統合] に移動します。



9. [アプリケーションのアサーションコンシューマーサービス (ACS) の URL]: https://<domain-prefix>.auth.<region>.amazoncognito.com/saml2/idpresponse

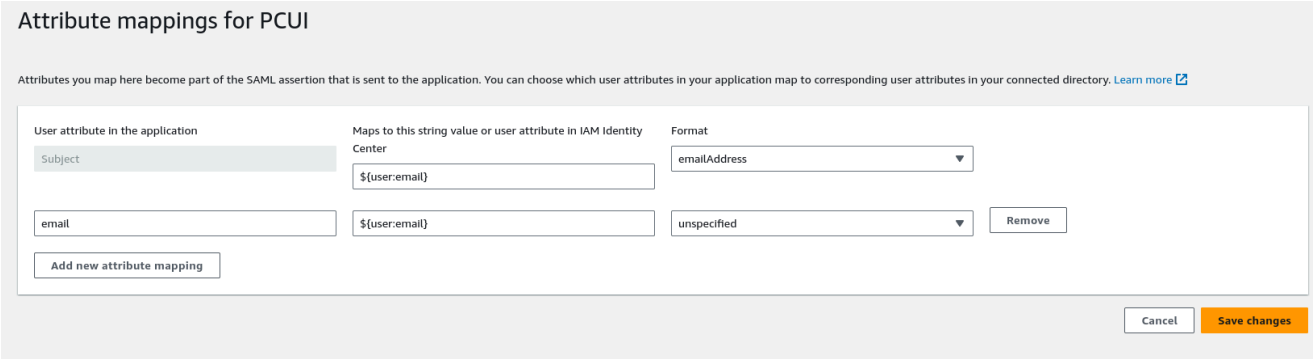
   [アプリケーション SAML 対象者]: urn:amazon:cognito:sp:<userpool-id>

10. [送信] を選択します。次に、追加したアプリケーションの [詳細] ページに移動します。

11. [アクション] ドロップダウンリストを選択し、[属性マッピングを編集] を選択します。次に、以下の属性を指定します。

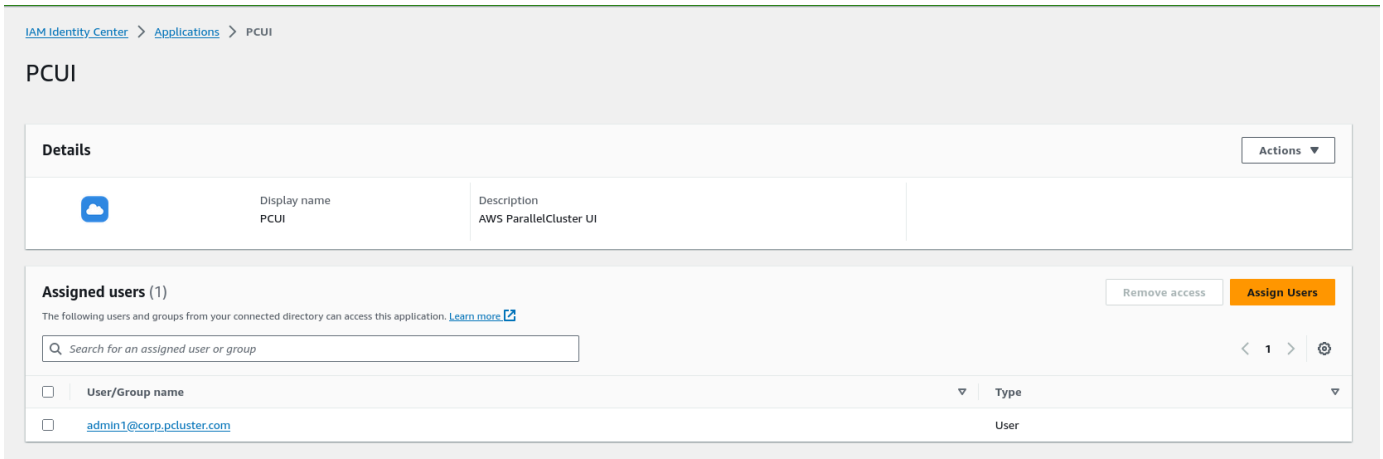
    a. [アプリケーションのユーザー属性]: subject (注: subject は事前入力済み) → [この文字列値または IAM アイデンティティセンターのユーザー属性にマッピング]: ${user:email}、[形式]: emailAddress

    b. [アプリケーションのユーザー属性]: email → [この文字列値または IAM アイデンティティセンターのユーザー属性にマッピング]: ${user:email}、[形式]: 未指定



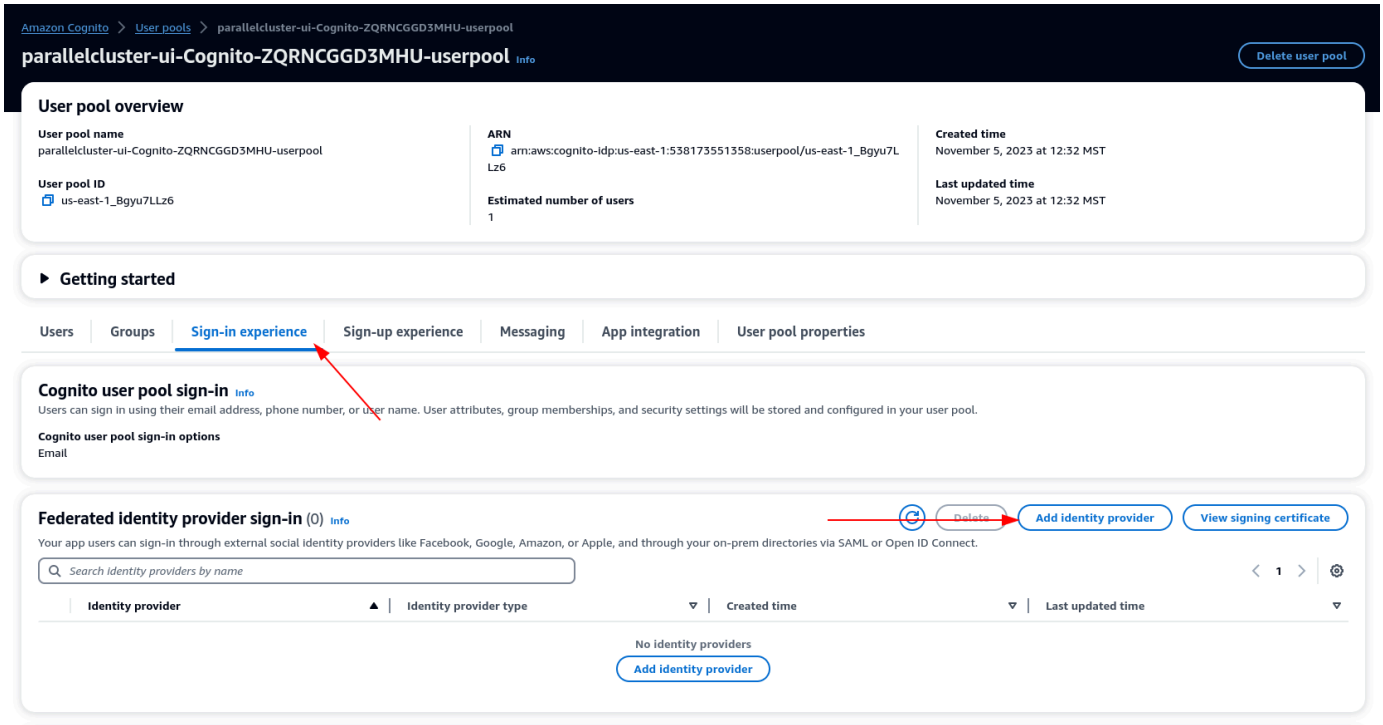
12. 変更内容を保存します。

13. [ユーザーを割り当て] ボタンを選択し、ユーザーをアプリケーションに割り当てます。これらは、PCUI インターフェイスにアクセスできる Active Directory 内のユーザーです。

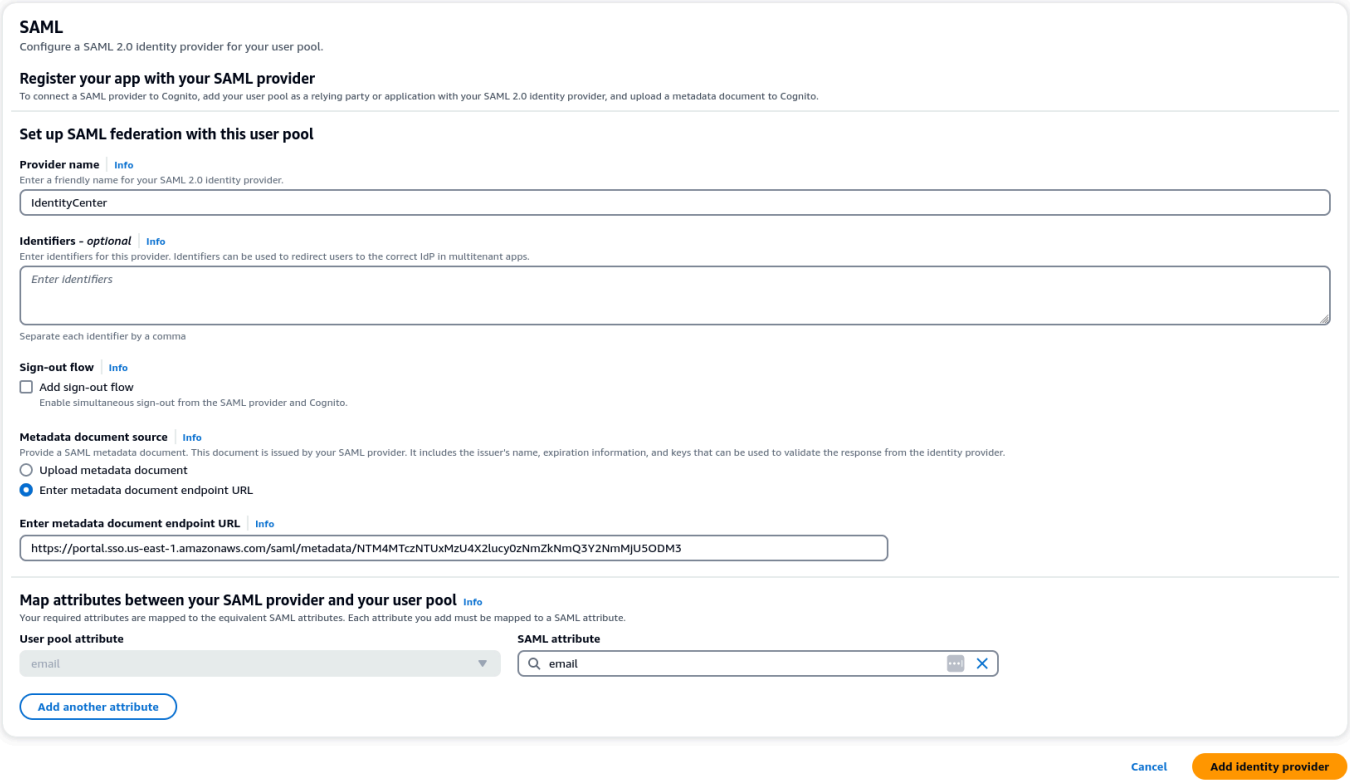


ユーザープールで IAM アイデンティティセンターを SAML IdP として設定する

1. ユーザープール設定で、[サインインエクスペリエンス] > [ID プロバイダーを追加] を選択します。



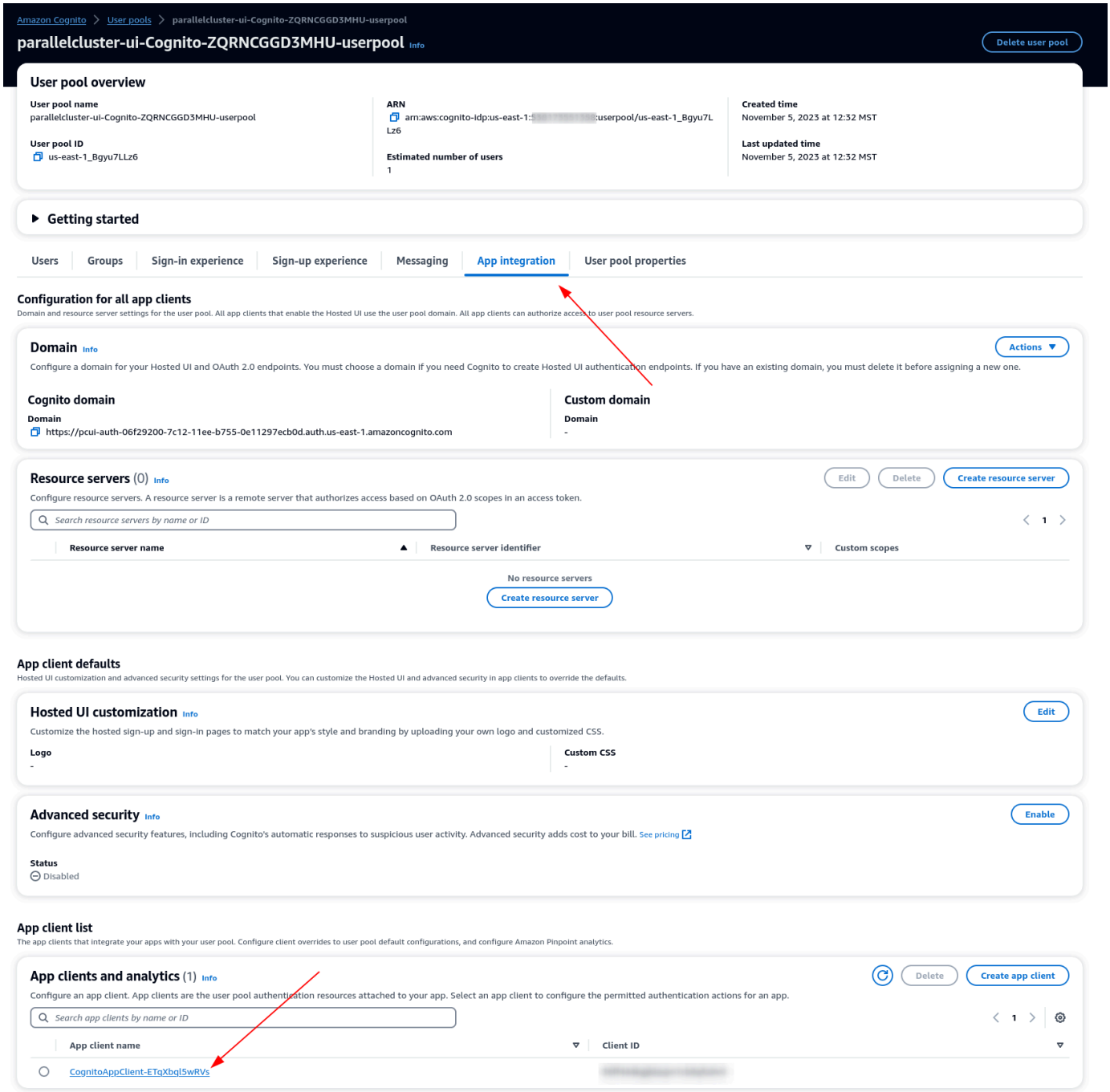
2. SAML IdP を選択します。
3. [プロバイダー名] に [IdentityCenter] を指定します。
4. [メタデータドキュメントソース] で [メタデータドキュメントのエンドポイント URL を入力] を選択し、アイデンティティセンターのアプリケーションをセットアップしたときにコピーした URL を指定します。
5. [属性] の [email] で、E メールを選択します。



6. [ID プロバイダーを追加] を選択します。

IdP をユーザープールアプリケーションクライアントと統合する

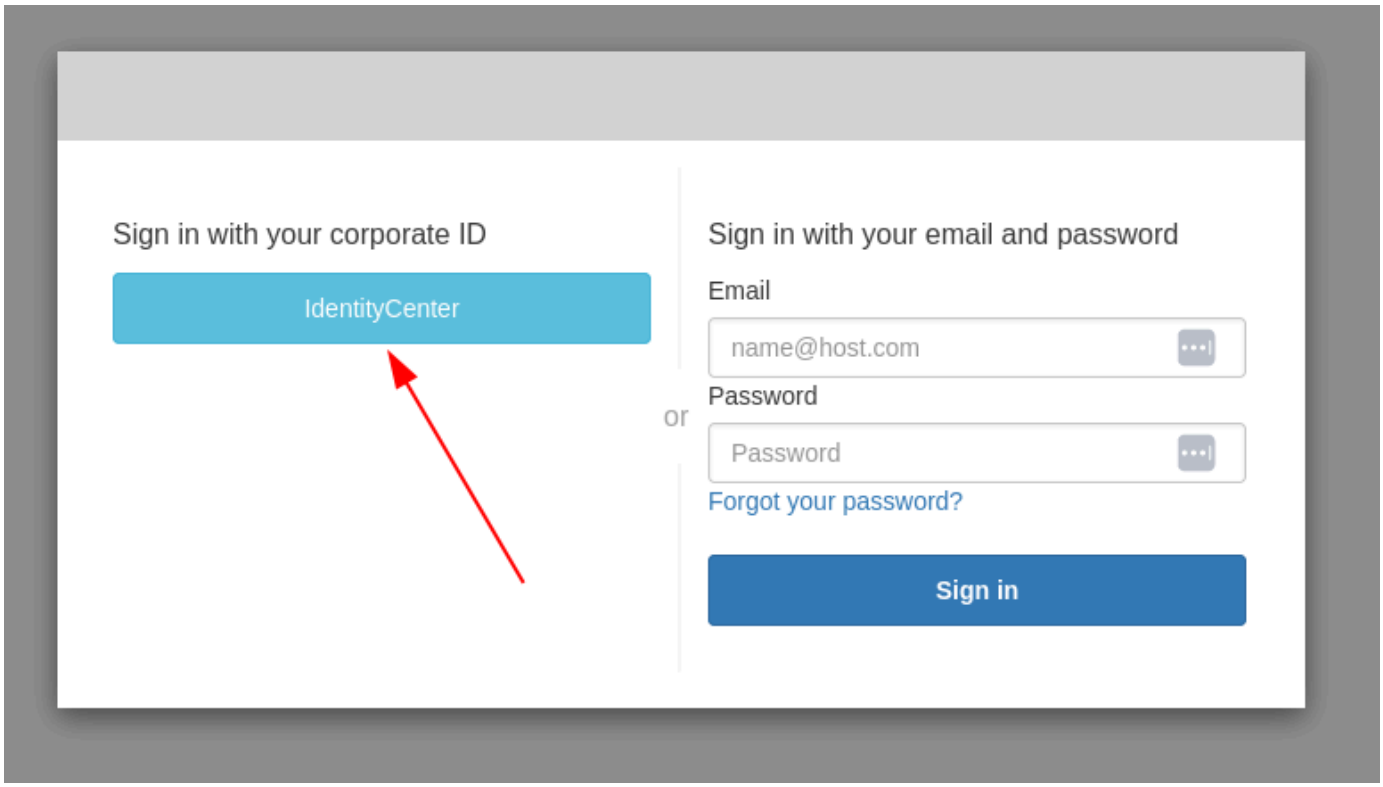
1. 次に、ユーザープールの [アプリケーションの統合] セクションで、[アプリケーションクライアントのリスト] にリストされているクライアントを選択します。



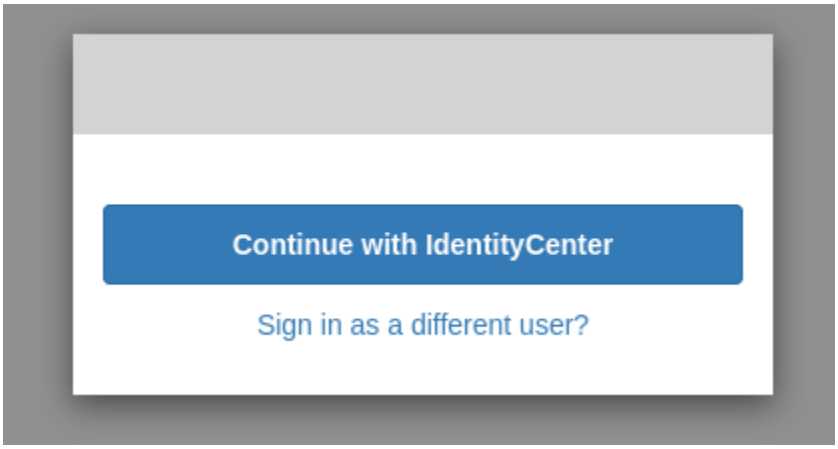
2. [ホストされた UI] で、[編集] を選択します。
3. [ID プロバイダー] で、[IdentityCenter] も選択します。
4. [変更を保存] を選択します。

セットアップを検証する

1. 次に、PCUI にログインして、先ほど作成したセットアップを検証します。PCUI ポータルにサインインすると、企業 ID でサインインするオプションが表示されます。

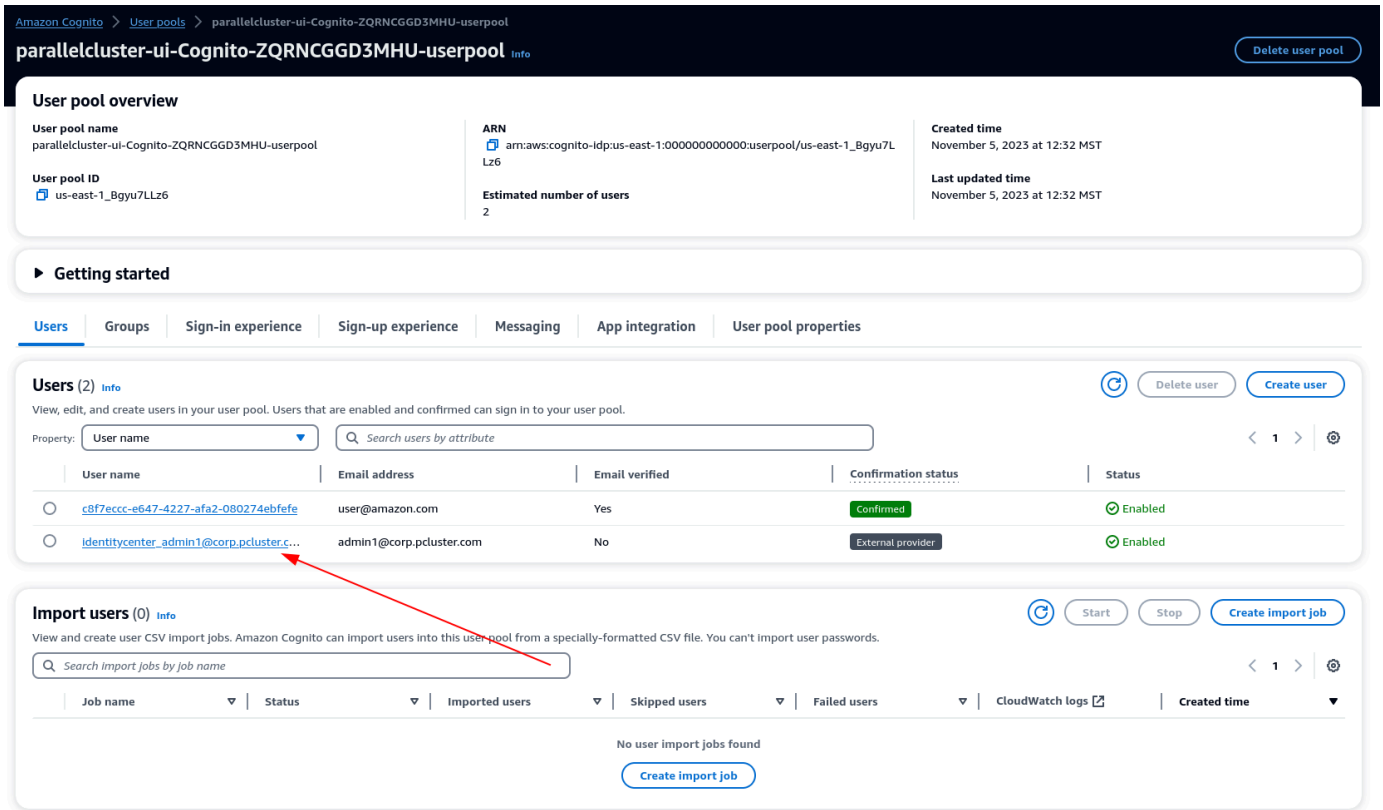


2. [IdentityCenter] ボタンをクリックすると、IAM アイデンティティセンターの IdP ログインに移動し、アプリケーション (PCUI を含む) を示すページが表示されます。PCUI アプリケーションを開きます。
3. 次の画面に移動すると、ユーザーが Cognito ユーザープールに追加されます。



ユーザーを管理者にする

1. ここで、[Amazon Cognito] > ユーザープールコンソールに移動し、identitycenter をプレフィックスとする、新しく作成したユーザーを選択します。



2. [グループメンバーシップ] で、[ユーザーをグループに追加]、[管理者] の順に選択し、[追加] をクリックします。
3. IdentityCenter で続行をクリックすると、UI AWS ParallelCluster ページに移動します。

Pyxis でコンテナ化されたジョブを実行する

Pyxis を使用して、コンテナ化されたジョブを実行できるクラスターを作成する方法について説明します。Pyxis は、SLURM でコンテナ化されたジョブを管理する SPANK プラグインです。Pyxis のコンテナは Enroot で管理します。Enroot は、従来のコンテナ/OS イメージを特権のないサンドボックスに変換するツールです。詳細については、「NVIDIA Pyxis と 「NVIDIA Enroot」を参照してください。

> **Note**
>
> - この機能は v3.11.1 AWS ParallelCluster で使用できます。

- このチュートリアルのスクリプトは、一部のファイルを移動 (mv) し、元の場所から削除します。これらのファイルのコピーを元の場所に保持する場合は、代わりにコピー (cp) コマンドを使用するようにスクリプトを変更します。

を使用する場合 AWS ParallelCluster、 AWS ParallelCluster イメージとクラスターを作成または更新するときに作成された AWS リソースに対してのみ料金が発生します。詳細については、「AWS が使用する サービス AWS ParallelCluster」を参照してください。

前提条件:

- AWS CLI がインストールされ、設定されます。
- Amazon EC2 の キーペア。
- pcluster CLI を実行するために必要なアクセス許可を持つ IAM ロール。

クラスターを作成する

AWS ParallelCluster 3.11.1 以降、すべての公式 AMIs Pyxis と Enroot がプリインストールされています。特に、SLURM は Pyxis サポートで再コンパイルされ、Enroot はバイナリとしてシステムにインストールされています。ただし、特定のニーズに合わせて設定する必要があります。Enroot と Pyxis で使用するフォルダは、クラスターのパフォーマンスに重大な影響を与えます。詳細については、Pyxis のドキュメントと Enroot のドキュメントを参照してください。

便宜上、Pyxis (SPANK プラグイン) と Enroot の両方のサンプル設定を `/opt/parallelcluster/examples/` 内に用意しています。

用意されているサンプル設定を使用してクラスターをデプロイするには、次のチュートリアルを完了します。

サンプル設定を使用してクラスターを作成するには

Pyxis と Enroot をヘッドノードで設定する必要があります。そのためには、まず Enroot の永続ディレクトリと揮発性ディレクトリを作成します。次に Pyxis のランタイムディレクトリを作成します。最後に Pyxis をクラスター全体の SPANK プラグインとして有効にします。

1. 次のスクリプトを OnNodeConfigured カスタムアクションとしてヘッドノードで実行し、ヘッドノードで Pyxis と Enroot を設定します。

```
#!/bin/bash
set -e

echo "Executing $0"

# Configure Enroot
ENROOT_PERSISTENT_DIR="/var/enroot"
ENROOT_VOLATILE_DIR="/run/enroot"

sudo mkdir -p $ENROOT_PERSISTENT_DIR
sudo chmod 1777 $ENROOT_PERSISTENT_DIR
sudo mkdir -p $ENROOT_VOLATILE_DIR
sudo chmod 1777 $ENROOT_VOLATILE_DIR
sudo mv /opt/parallelcluster/examples/enroot/enroot.conf /etc/enroot/enroot.conf
sudo chmod 0644 /etc/enroot/enroot.conf

# Configure Pyxis
PYXIS_RUNTIME_DIR="/run/pyxis"

sudo mkdir -p $PYXIS_RUNTIME_DIR
sudo chmod 1777 $PYXIS_RUNTIME_DIR

sudo mkdir -p /opt/slurm/etc/plugstack.conf.d/
sudo mv /opt/parallelcluster/examples/spank/plugstack.conf /opt/slurm/etc/
sudo mv /opt/parallelcluster/examples/pyxis/pyxis.conf /opt/slurm/etc/
plugstack.conf.d/
sudo -i scontrol reconfigure
```

2. Pyxis と Enroot をコンピューティングフリートで設定するには、Enroot の永続ディレクトリと揮発性ディレクトリを作成し、Pyxis のランタイムディレクトリを作成します。次のスクリプトを OnNodeStart カスタムアクションとしてコンピューティングノードで実行し、コンピューティングフリートで Pyxis と Enroot を設定します。

```
#!/bin/bash
set -e

echo "Executing $0"

# Configure Enroot
ENROOT_PERSISTENT_DIR="/var/enroot"
ENROOT_VOLATILE_DIR="/run/enroot"
```

```
ENROOT_CONF_DIR="/etc/enroot"

sudo mkdir -p $ENROOT_PERSISTENT_DIR
sudo chmod 1777 $ENROOT_PERSISTENT_DIR
sudo mkdir -p $ENROOT_VOLATILE_DIR
sudo chmod 1777 $ENROOT_VOLATILE_DIR
sudo mkdir -p $ENROOT_CONF_DIR
sudo chmod 1777 $ENROOT_CONF_DIR
sudo mv /opt/parallelcluster/examples/enroot/enroot.conf /etc/enroot/enroot.conf
sudo chmod 0644 /etc/enroot/enroot.conf

# Configure Pyxis
PYXIS_RUNTIME_DIR="/run/pyxis"

sudo mkdir -p $PYXIS_RUNTIME_DIR
sudo chmod 1777 $PYXIS_RUNTIME_DIR

# In Ubuntu24.04 Apparmor blocks the creation of unprivileged user namespaces,
# which is required by Enroot. So to run Enroot, it is required to disable this
 restriction.
# See https://ubuntu.com/blog/ubuntu-23-10-restricted-unprivileged-user-namespaces
source /etc/os-release
if [ "${ID}${VERSION_ID}" == "ubuntu24.04" ]; then
    echo "kernel.apparmor_restrict_unprivileged_userns = 0" | sudo tee /etc/
sysctl.d/99-pcluster-disable-apparmor-restrict-unprivileged-userns.conf
    sudo sysctl --system
fi
```

ジョブを送信する

Pyxis をクラスター内に設定したので、sbatch および srun コマンドを使用して、コンテナ化してコンテナ固有のオプションで強化したジョブを送信できます。

```
# Submitting an interactive job
srun -N 2 --container-image docker://ubuntu:22.04 hostname

# Submitting a batch job
sbatch -N 2 --wrap='srun --container-image docker://ubuntu:22.04 hostname'
```

EFA 対応 FSx Lustre を使用したクラスターの作成

このチュートリアルでは、EFA 対応の FSx Lustre ファイルシステムを共有ストレージとして使用するクラスターを作成します。EFA を有効にして FSx Lustre ファイルシステムを使用すると、パフォーマンスが最大 8 倍向上します。EFA 対応ファイルシステムが必要なものかどうかを確認するには、FSx for Lustre ユーザーガイド」の「EFA 対応ファイルシステムの使用」を参照してください。

を使用すると AWS ParallelCluster、 AWS ParallelCluster イメージとクラスターを作成または更新したときに作成された AWS リソースに対してのみ料金が発生します。詳細については、「AWS が使用する サービス AWS ParallelCluster」を参照してください。

要件

- CLI AWS がインストールされ、設定されます。
- ParallelCluster CLI がインストールされ、設定されています。
- クラスターにログインするための Amazon EC2 キーペア。
- ParallelCluster CLI の実行に必要なアクセス許可を持つ IAM ロール。

セキュリティグループを作成する

クラスターとファイルシステムがデプロイされる同じ VPC に 2 つのセキュリティグループを作成します。1 つはクラスターノードで実行されているクライアント用、もう 1 つはファイルシステム用です。

```
# Create security group for the FSx client
aws ec2 create-security-group \
    --group-name Fsx-Client-SecurityGroup \
    --description "Allow traffic for the FSx Lustre client" \
    --vpc-id vpc-cluster \
    --region region

# Create security group for the FSx file system
aws ec2 create-security-group \
    --group-name Fsx-FileSystem-SecurityGroup \
    --description "Allow traffic for the FSx Lustre File System" \
    --vpc-id vpc-cluster \
    --region region
```

チュートリアルの残りの部分では、 sg-client と sg-file-systemがそれぞれクライアントとファイルシステムのセキュリティグループ ID であることを前提としています。

EFA の必要に応じて、ファイルシステムへのすべてのアウトバウンドトラフィックを許可するように、クライアントのセキュリティグループを設定します。

```
# Allow all outbound traffic from the client to the file system
aws ec2 authorize-security-group-egress \
 --group-id sg-client \
 --protocol -1 \
 --port -1 \
 --source-group sg-file-system \
 --region region
```

EFA の必要に応じて、ファイルシステムのセキュリティグループを設定して、そのシステム内のすべてのインバウンド/アウトバウンドトラフィックとクライアントからのすべてのインバウンドトラフィックを許可します。

```
# Allow all inbound traffic within this security group
aws ec2 authorize-security-group-ingress \
    --group-id sg-file-system \
    --protocol -1 \
    --port -1 \
    --source-group sg-file-system \
    --region region

# Allow all outbound traffic within this security group
aws ec2 authorize-security-group-egress \
    --group-id sg-file-system \
    --protocol -1 \
    --port -1 \
    --source-group sg-file-system \
    --region region

# Allow all inbound traffic from the client
aws ec2 authorize-security-group-ingress \
    --group-id sg-file-system \
    --protocol -1 \
    --port -1 \
    --source-group sg-client \
    --region region
```

```
# Allow all outbound traffic to the client
aws ec2 authorize-security-group-egress \
    --group-id sg-file-system \
    --protocol -1 \
    --port -1 \
    --source-group sg-client \
    --region region
```

ファイルシステムを作成する

コンピューティングノードがあるのと同じアベイラビリティーゾーン (AZ) 内にファイルシステムを作成し、次のコードで を ID *subnet-compute-nodes*に置き換えます。これは、EFA がファイルシステムと連携できるようにするために必要です。ファイルシステムの作成の一環として、EfaEnable プロパティを使用して EFA を有効にすることに注意してください。

```
aws fsx create-file-system \
    --file-system-type LUSTRE \
    --storage-capacity 38400 \
    --storage-type SSD \
    --subnet-ids subnet-compute-nodes \
    --security-group-ids sg-file-system \
    --lustre-configuration
 DeploymentType=PERSISTENT_2,PerUnitStorageThroughput=125,EfaEnabled=true,MetadataConfiguration
 \
    --region region
```

前のコマンドで返されたファイルシステム ID を書き留めます。チュートリアルの残りの部分では、をこのファイルシステム ID *fs-id*に置き換えます。

クラスターを作成する

1. YAML 設定ファイルに次の設定が設定された AWS ParallelCluster クラスターを作成します。
   a. Ubuntu 22.04 など、サポートされている OS に基づく AMI。
   b. コンピューティングノードは、g6.16xlarge などの Nitro v4+ を持つ EFA でサポートされているインスタンスタイプを使用する必要があります。
      - コンピューティングノードは、ファイルシステムと同じ AZ に存在する必要があります。
      - コンピューティングノードでは、Efa/Enabled を true に設定する必要があります。
      - コンピューティングノードは、OnNodeStart カスタムアクション`configure-efa-fsx-lustre-client.sh`として設定スクリプトを実行する必要があります。スクリプトは、FSx

の公式ドキュメントで提供されており、お客様の便宜上、パブリックバケットで提供されています。これは、コンピューティングノードで FSx Lustre クライアントを設定して EFA を使用できるようにすることを目的としています。

2. クラスター設定ファイル を作成しますconfig.yaml。

```
Region: region
Image:
  Os: ubuntu2204
HeadNode:
  InstanceType: c5.xlarge
  Networking:
    SubnetId: subnet-xxxxxxxxxx
    AdditionalSecurityGroups:
        - sg-client
  Ssh:
    KeyName: my-ssh-key
Scheduling:
  Scheduler: slurm
  SlurmQueues:
    - Name: q1
      ComputeResources:
        - Name: cr1
          Instances:
            - InstanceType: g6.16xlarge
          MinCount: 1
          MaxCount: 3
          Efa:
            Enabled: true
      Networking:
        SubnetIds:
          - subnet-xxxxxxxxxx # Subnet in the same AZ where the file system is
        AdditionalSecurityGroups:
          - sg-client
        PlacementGroup:
          Enabled: false
      CustomActions:
        OnNodeStart:
          Script: https://us-east-1-aws-parallelcluster.s3.us-east-1.amazonaws.com/
scripts/fsx-lustre-efa/configure-efa-fsx-lustre-client.sh
SharedStorage:
  - MountDir: /fsx
    Name: my-fsxlustre-efa-external
    StorageType: FsxLustre
```

```
    FsxLustreSettings:
      FileSystemId: fs-id
```

次に、その設定を使用してクラスターを作成します。

```
pcluster create-cluster \
    --cluster-name fsx-efa-tutorial \
    --cluster-configuration config.yaml \
    --region region
```

EFA で FSx が動作していることを検証する

Lustre ネットワークトラフィックが EFA を使用していることを確認するには、特定のネットワークインターフェイスのネットワークトラフィックを表示できる Lustre `lnetctl` ツールを使用します。この目的を達成するには、コンピューティングノードで次のコマンドを実行します。

```
# Take note of the number of packets flowing through the interface,
# which are specified in statistics:send_count and statistics:recv_count
sudo lnetctl net show --net efa -v

# Generate traffic to the file system
echo 'Hello World' > /fsx/hello-world.txt

# Take note of the number of packets flowing through the interface,
# which are specified in statistics:send_count and statistics:recv_count
sudo lnetctl net show --net efa -v
```

機能が機能している場合、インターフェイスを通過するパケットの数は増加すると予想されます。

AWS ParallelCluster トラブルシューティング

以下のセクションでは、 AWS ParallelClusterの使用中に発生する可能性がある問題のトラブルシューティングのヒントを示します。 AWS ParallelCluster コミュニティは、AWS ParallelCluster GitHub Wiki で多くのトラブルシューティングのヒントを提供する Wiki ページを保持しています。既知の問題のリストは、「既知の問題」を参照してください。

トピック

- クラスターの作成を試行する
- ジョブの実行を試行する
- クラスターの更新を試行する
- ストレージへのアクセスを試行する
- クラスターの削除を試行する
- AWS ParallelCluster API スタックのアップグレードを試行する
- コンピューティングノードの初期化のエラーが表示されている
- クラスターヘルスメトリクスのトラブルシューティング
- クラスターデプロイの問題のトラブルシューティング
- Terraform を使用したクラスターのデプロイのトラブルシューティング
- スケーリング問題のトラブルシューティング
- プレイスメントグループとインスタンスの起動に関する問題
- ディレクトリの置換
- Amazon DCV の問題のトラブルシューティング
- AWS Batch 統合によるクラスターの問題のトラブルシューティング
- Active Directory を使用したマルチユーザー統合のトラブルシューティング
- カスタム AMI の問題のトラブルシューティング
- cfn-hup が実行していない場合のクラスター更新タイムアウトのトラブルシューティング
- ネットワークのトラブルシューティング
- クラスターの更新が onNodeUpdated カスタムアクションで失敗した
- カスタム Slurm 設定でエラーが表示されている
- クラスターのアラーム

クラスターの作成を試行する

AWS ParallelCluster バージョン 3.5.0 以降を使用してクラスターを作成し、 を `--rollback-on-failure`に設定してクラスターの作成が失敗した場合`false`、 `pcluster describe-cluster` CLI コマンドを使用してステータスと障害情報を取得します。この場合、`pcluster describe-cluster` の `clusterStatus` の正常な出力は `CREATE_FAILED` です。出力の `failures` セクションを確認して、`failureCode` と `failureReason` を見つけます。次のセクションで一致する `failureCode` を探して、その他のトラブルシューティングについてのヘルプを見つけます。詳細については、「`pcluster describe-cluster`」を参照してください。

次のセクションでは、`/var/log/cfn-init.log` や `/var/log/chef-client.log` ファイルなど、ヘッドノードのログを確認することをお勧めします。 AWS ParallelCluster ログとその表示方法の詳細については、デバッグ用のキーログ「」および「」を参照してくださいログの取得と保存。

がない場合は`failureCode`、 AWS CloudFormation コンソールに移動してクラスタースタックを表示します。`HeadNodeWaitCondition` の `Status Reason`、または他のリソースの障害を確認して、失敗に関するその他の詳細を確認します。詳細については、「で AWS CloudFormation イベントを表示する `CREATE_FAILED`」を参照してください。ヘッドノードの `/var/log/cfn-init.log` および `/var/log/chef-client.log` ファイルを確認します。ヘッドノードの作成に失敗したためにクラスターの作成に失敗し、クラスターログがクラスターロググループで利用できない場合は、クラスターを失敗時に保持し、 `--rollback-on-failure =` を指定`True`して、ヘッドノード自体内からログを取得する必要があります。

`failureCode` が `OnNodeConfiguredExecutionFailure`

- 失敗した原因

  クラスターを作成するために、設定のヘッドノードセクションの `OnNodeConfigured` にカスタムスクリプトを指定しました。しかし、カスタムスクリプトの実行に失敗しました。

- 解決方法

  `/var/log/cfn-init.log` ファイルを確認して、障害の詳細とカスタムスクリプトの問題の修正方法を確認します。このログの最後の方で、`Running command runpostinstall` メッセージの後に `OnNodeConfigured` スクリプトに関連する実行情報が表示される場合があります。

`failureCode` が `OnNodeConfiguredDownloadFailure`

- 失敗した原因

クラスターを作成するために、設定のヘッドノードセクションの OnNodeConfigured にカスタムスクリプトを指定しました。しかし、カスタムスクリプトのダウンロードに失敗しました。

- 解決方法

  URL が有効で、アクセスが正しく設定されていることを確認します。カスタムブートストラップスクリプトの設定に関する詳細については、「カスタムブートストラップアクション」を参照してください。

  /var/log/cfn-init.log ファイルを確認してください。このログの最後の方で、Running command runpostinstall メッセージの後に、ダウンロードを含め OnNodeConfigured スクリプトの処理に関連する実行情報が表示される場合があります。

failureCode が OnNodeConfiguredFailure

- 失敗した原因

  クラスターを作成するために、設定のヘッドノードセクションの OnNodeConfigured にカスタムスクリプトを指定しました。ただし、クラスターのデプロイにおいてカスタムスクリプトの使用に失敗しました。即時に原因を判断できないため、追加の調査が必要です。

- 解決方法

  /var/log/cfn-init.log ファイルを確認してください。このログの最後の方で、Running command runpostinstall メッセージの後に OnNodeConfigured スクリプトの処理に関連する実行情報が表示される場合があります。

failureCode が OnNodeStartExecutionFailure

- 失敗した原因

  クラスターを作成するために、設定のヘッドノードセクションの OnNodeStart にカスタムスクリプトを指定しました。しかし、カスタムスクリプトの実行に失敗しました。

- 解決方法

  /var/log/cfn-init.log ファイルを確認して、障害の詳細とカスタムスクリプトの問題の修正方法を確認します。このログの最後の方で、Running command runpreinstall メッセージの後に OnNodeStart スクリプトに関連する実行情報が表示される場合があります。

failureCode が OnNodeStartDownloadFailure

- 失敗した原因

  クラスターを作成するために、設定のヘッドノードセクションの OnNodeStart にカスタムスクリプトを指定しました。しかし、カスタムスクリプトのダウンロードに失敗しました。

- 解決方法

  URL が有効で、アクセスが正しく設定されていることを確認します。カスタムブートストラップスクリプトの設定に関する詳細については、「カスタムブートストラップアクション」を参照してください。

  /var/log/cfn-init.log ファイルを確認してください。このログの最後の方で、Running command runpreinstall メッセージの後に、ダウンロードを含め OnNodeStart スクリプトの処理に関連する実行情報が表示される場合があります。

failureCode が OnNodeStartFailure

- 失敗した原因

  クラスターを作成するために、設定のヘッドノードセクションの OnNodeStart にカスタムスクリプトを指定しました。ただし、クラスターのデプロイにおいてカスタムスクリプトの使用に失敗しました。即時に原因を判断できないため、追加の調査が必要です。

- 解決方法

  /var/log/cfn-init.log ファイルを確認してください。このログの最後の方で、Running command runpreinstall メッセージの後に OnNodeStart スクリプトの処理に関連する実行情報が表示される場合があります。

failureCode が EbsMountFailure

- 失敗した原因

  クラスター設定で定義されている EBS ボリュームのマウントに失敗しました。

- 解決方法

  失敗の詳細について、/var/log/chef-client.log ファイルを確認します。

failureCode が EfsMountFailure

- 失敗した原因

  クラスター設定で定義されている Amazon EFS ボリュームのマウントに失敗しました。

- 解決方法

  既存の Amazon EFS ファイルシステムを定義した場合は、クラスターとファイルシステムの間のトラフィックが許可されていることを確認します。詳細については、「SharedStorage」/「EfsSettings」/「FileSystemId」を参照してください。

  失敗の詳細について、/var/log/chef-client.log ファイルを確認します。

failureCode が FsxMountFailure

- 失敗した原因

  クラスター設定で定義されている Amazon FSx ファイルシステムのマウントに失敗しました。

- 解決方法

  既存の Amazon FSx ファイルシステムを定義した場合は、クラスターとファイルシステムの間のトラフィックが許可されていることを確認します。詳細については、「SharedStorage」/「FsxLustreSettings」/「FileSystemId」を参照してください。

  失敗の詳細について、/var/log/chef-client.log ファイルを確認します。

failureCode が RaidMountFailure

- 失敗した原因

  クラスター設定で定義されている RAID ボリュームのマウントに失敗しました。

- 解決方法

  失敗の詳細について、/var/log/chef-client.log ファイルを確認します。

failureCode が AmiVersionMismatch

- 失敗した原因

  カスタム AMI の作成に使用される AWS ParallelCluster バージョンは、クラスターの設定に使用される AWS ParallelCluster バージョンとは異なります。CloudFormation コンソールで、クラスターの CloudFormation スタックの詳細を表示し、 `Status Reason`で をチェックHeadNodeWaitConditionして、 AWS ParallelCluster バージョンと AMI の詳細を確認します。詳細については、「で AWS CloudFormation イベントを表示する CREATE_FAILED」を参照してください。

- 解決方法

  カスタム AMI の作成に使用した AWS ParallelCluster バージョンが、クラスターの設定に使用した AWS ParallelCluster バージョンと同じであることを確認します。カスタム AMI のバージョン、または `pcluster` CLI のバージョンのいずれかを変更して同じにすることができます。

failureCode が InvalidAmi

- 失敗した原因

  カスタム AMI は、 を使用して構築されていないため、無効です AWS ParallelCluster。

- 解決方法

  `pcluster build-image` コマンドを使用し、独自の AMI を親イメージにして AMI を作成します。詳細については、「`pcluster build-image`」を参照してください。

failureCode が HeadNodeBootstrapFailure と failureReason で、ヘッドノードの設定に失敗した。

- 失敗した原因

  即時に原因を判断できないため、追加の調査が必要です。例えば、クラスターが保護ステータスにある場合や、静的コンピューティングフリートのプロビジョニングの失敗により発生した可能性があります。

- 解決方法

  失敗の詳細について、`/var/log/chef-client.log.` ファイルを確認します。

> **Note**
>
> `RuntimeError` 例外 `Cluster state has been set to PROTECTED mode due to failures detected in static node provisioning` が表示された場合、クラスターは保護ステータスにあります。詳細については、「保護モードをデバッグする方法」を参照してください。

`failureCode` は `HeadNodeBootstrapFailure` で、`failureReason` クラスター作成がタイムアウトした。

- 失敗した原因

  デフォルトでは、クラスターの作成が完了するのに 30 分の時間制限があります。このタイムフレーム内でクラスターの作成が完了しない場合、クラスターの作成はタイムアウトエラーで失敗します。クラスターの作成は、さまざまな理由でタイムアウトになる可能性があります。例えば、タイムアウトによる失敗は、ヘッドノード作成の失敗、ネットワークの問題、ヘッドノードでの実行に時間がかかりすぎるカスタムスクリプト、コンピューティングノードで実行されるカスタムスクリプトのエラー、またはコンピューティングノードのプロビジョニングの待ち時間が長いことにより発生する可能性があります。即時に原因を判断できないため、追加の調査が必要です。

- 解決方法

  失敗の詳細について、`/var/log/cfn-init.log` と `/var/log/chef-client.log` ファイルを確認します。 AWS ParallelCluster ログとその取得方法に関する詳細については、「デバッグ用のキーログ」と「ログの取得と保存」を参照してください。

  これらのログで、次のことが見つかることがあります。

  - **`chef-client.log` の最後の方にある `Waiting for static fleet capacity provisioning` が表示されている**

    これは、静的ノードの電源が入るのを待機しているときにクラスターの作成がタイムアウトしたことを示しています。詳細については、「コンピューティンティングノードの初期化のエラーが表示されている」を参照してください。

  - **`OnNodeConfigured` または `OnNodeStart` ヘッドノードスクリプトが `cfn-init.log` の最後で終了していないことが表示されている**

これは、OnNodeConfigured または OnNodeStart で、カスタムスクリプトの実行に時間がかかり、タイムアウトエラーが発生したことを示しています。カスタムスクリプトに、実行に長い時間がかかる問題がないか確認します。カスタムスクリプトの実行に長い時間が必要な場合は、次の例に示されているようにクラスター設定ファイルに DevSettings セクションを追加してタイムアウト制限を変更することを考慮してください。

```
DevSettings:
  Timeouts:
    HeadNodeBootstrapTimeout: 1800 # default setting: 1800 seconds
```

- ログが見つからない、またはヘッドノードが正常に作成されない

  ヘッドノードが正常に作成されず、ログが見つからない可能性があります。CloudFormation コンソールでクラスタースタックの詳細を表示して、失敗に関するその他の詳細を確認します。

failureCode は HeadNodeBootstrapFailure で、failureReason はヘッドノードのブートストラップに失敗した。

- 失敗した原因

  即時に原因を判断できないため、追加の調査が必要です。

- 解決方法

  /var/log/cfn-init.log と /var/log/chef-client.log のファイルを確認します。

failureCode が ResourceCreationFailure

- 失敗した原因

  クラスター作成プロセス中に、一部のリソースの作成に失敗しました。さまざまな理由で失敗が発生します。例えば、リソース作成の失敗は、容量の問題や IAM ポリシーが誤って設定されていることにより発生することがあります。

- 解決方法

  CloudFormation コンソールでクラスタースタックを表示して、リソース作成の失敗に関するその他の詳細を確認します。

failureCode が ClusterCreationFailure

- 失敗した原因

  即時に原因を判断できないため、追加の調査が必要です。

- 解決方法

  CloudFormation コンソールでクラスタースタックを表示し、HeadNodeWaitCondition の Status Reason を確認して、失敗に関するその他の詳細を見つけます。

  /var/log/cfn-init.log と /var/log/chef-client.log のファイルを確認します。

CloudFormation スタックの WaitCondition timed out... が表示されている

詳細については、「failureCode は HeadNodeBootstrapFailure で、failureReason クラスター作成がタイムアウトした。」を参照してください。

CloudFormation スタックの Resource creation cancelled が表示されている

詳細については、「failureCode が ResourceCreationFailure」を参照してください。

AWS CloudFormation スタック内の表示Failed to run cfn-init...またはその他のエラー

失敗に関するその他の詳細について、/var/log/cfn-init.log と /var/log/chef-client.log を確認します。

INFO: Waiting for static fleet capacity provisioning の最後に chef-client.log が表示されている

これは、静的ノードの電源が入るのを待機しているときにクラスターの作成がタイムアウトになることと関係しています。詳細については、「コンピューティングノードの初期化のエラーが表示されている」を参照してください。

`Failed to run preinstall or postinstall in cfn-init.log` が表示されている

クラスター設定 の `HeadNode` セクションに `OnNodeConfigured` または `OnNodeStart` スクリプトがあります。このスクリプトが正しく動作していません。カスタムスクリプトのエラーの詳細について、`/var/log/cfn-init.log` ファイルを確認します。

CloudFormation スタックの `This AMI was created with xxx, but is trying to be used with xxx...` が表示されている

詳細については、「`failureCode` が `AmiVersionMismatch`」を参照してください。

CloudFormation スタックの `This AMI was not baked by AWS ParallelCluster...` が表示されている

詳細については、「`failureCode` が `InvalidAmi`」を参照してください。

`pcluster create-cluster` コマンドがローカルで実行できないことが表示されている

失敗の詳細について、ローカルファイルシステムの `~/.parallelcluster/pcluster-cli.log` を確認します。

追加のサポート

クラスターデプロイの問題のトラブルシューティング のトラブルシューティングガイダンスに従ってください。

シナリオが GitHub の にある GitHub の既知の問題 AWS ParallelCluster でカバーされているかどうかを確認します。 GitHub

ジョブの実行を試行する

次のセクションでは、ジョブを実行しようとして問題が発生した場合に役立つトラブルシューティングソリューションを示します。

srun のインタラクティブなジョブがエラー srun: error: fwd_tree_thread: can't find address for <host>, check slurm.conf で失敗する

- 失敗した原因

  srun コマンドを実行してジョブを送信し、更新が完了した後、Slurm デーモンを再起動せずに pcluster update-cluster コマンドを使用してキューサイズを増やしました。

  Slurm は、Slurm デーモンをツリー階層に整理して通信を最適化します。この階層は、デーモンが開始するときにのみ更新されます。

  srun を使用してジョブを起動し、pcluster update-cluster コマンドを実行してキューサイズを増やすとします。新しいコンピューティングノードは、更新の一環として起動します。次に、Slurm は、新しいコンピューティングノードの 1 つに、ジョブをキューに入れます。この場合、Slurm デーモンと srun の両方は新しいコンピューティングノードを検出しません。srun は新しいノードを検出しないため、エラーを返します。

- 解決方法

  すべてのコンピューティングノードで Slurm デーモンを再起動し、srun を使用してジョブを送信します。コンピューティングノードを再起動する scontrol reboot コマンドを実行することにより、Slurm デーモンの再起動をスケジュールできます。詳細については、「Slurm ドキュメント」の「scontrol reboot」を参照してください。また、対応する systemd サービスの再起動をリクエストすることにより、コンピューティングノードで Slurm デーモンを手動で再起動することもできます。

ジョブが squeue コマンドで、CF 状態でスタックしている

これは、動的ノードの電源を入れる時の問題である可能性があります。詳細については、「コンピューティンティングノードの初期化のエラーが表示されている」を参照してください。

大規模なジョブを実行し、nfsd: too many open connections, consider increasing the number of threads in /var/log/messages が表示されている

ネットワークに接続されたファイルシステムでは、ネットワークの制限に到達すると、I/O の待機時間も増加します。ネットワークと I/O メトリクスの両方のデータを書き込むのにネットワークが使用されるため、これによりソフトロックアップになることがあります。

第 5 世代のインスタンスでは、ENA ドライバーを使用してパケットカウンターを公開します。これらのカウンターは、ネットワークがインスタンス帯域幅制限に達した AWS ときに によって形成されたパケットをカウントします。これらのカウンターを確認して 0 より大きいかどうかを確認できます。その場合、帯域幅制限を超えていることになります。`ethtool -S eth0 | grep exceeded` を実行するとこれらのカウンターが表示されます。

サポートする NFS 接続が多すぎると、多くの場合、ネットワーク制限を超えます。これは、ネットワークの制限に到達したり、それを超えたりしたときに最初に確認することの 1 つです。

例えば、次の出力にドロップされたパッケージを示します。

```
$ ethtool -S eth0 | grep exceeded
  bw_in_allowance_exceeded: 38750610
  bw_out_allowance_exceeded: 1165693
  pps_allowance_exceeded: 103
  conntrack_allowance_exceeded: 0
  linklocal_allowance_exceeded: 0
```

このメッセージの表示を回避するには、ヘッドノードのインスタンスタイプをよりパフォーマンスの高いインスタンスタイプに変更することを検討します。データストレージを、Amazon EFS や Amazon FSx などの NFS 共有としてエクスポートされていない共有ストレージファイルシステムに移行することを検討します。詳細については、共有ストレージ「」および GitHub の AWS ParallelCluster Wiki の「ベストプラクティス」を参照してください。

MPI ジョブを実行する

デバッグモードを有効にする

OpenMPI デバッグモードを有効にするには、「What controls does Open MPI have that aid in debugging」を参照してください。

IntelMPI デバッグモードを有効にするには、「Other Environment Variables」を参照してください。

ジョブ出力の MPI_ERRORS_ARE_FATAL と OPAL ERROR が表示されている

これらのエラーコードはアプリケーションの MPI レイヤーから発生します。アプリケーションから MPI デバッグログを取得する方法については、「デバッグモードを有効にする」を参照してください。

このエラーの考えられる原因として、アプリケーションが OpenMPI などの特定の MPI 実装用にコンパイルされており、IntelMPI などの別の MPI 実装で実行しようとしていることがあります。アプリケーションのコンパイルと実行の両方で、同じ MPI 実装を使用していることを確認します。

マネージド DNS を無効にして mpirun を使用する

SlurmSettings/Dns/DisableManagedDns および UseEc2Hostnames を `true` に設定して作成されたクラスターの場合、Slurm ノード名は DNS によって解決されません。Slurm は、MPI ジョブが Slurm コンテキストで実行されていて、`nodenames` が有効になっていないとき、MPI プロセスをブートストラップできます。「Slurm MPI User's Guide」のガイダンスに従い、Slurm を使用して MPI ジョブを実行することをお勧めします。

クラスターの更新を試行する

次のセクションでは、クラスターを更新しようとして問題が発生した場合に役立つトラブルシューティングソリューションを示します。

pcluster update-cluster コマンドのローカル実行に失敗する

失敗の詳細について、ローカルファイルシステムの `~/.parallelcluster/pcluster-cli.log` を確認します。

pcluster describe-cluster コマンドで clusterStatus が UPDATE_FAILED であることが表示されている

クラスタースタックの更新がロールバックされた場合、エラーの詳細について `/var/log/chef-client.log` ファイルを確認します。

問題が GitHub の にある GitHub の既知の問題 AWS ParallelCluster に記載されているかどうかを確認します。 GitHub

クラスターの更新がタイムアウトになる

これは cfn-hup が実行されていないことに関連する問題の可能性があります。cfn-hup デーモンが外部の原因により終了させられる場合、自動的に再開されることはありません。cfn-hup が実行されていない場合、クラスターの更新中に CloudFormation スタックは期待どおりに更新プロセスを開始しますが、更新手順はヘッドノードでアクティブ化されず、最終的にスタックのデプロイはタイムアウトになります。詳細については、「cfn-hup が実行していない場合のクラスター更新タイムアウトのトラブルシューティング」を参照してトラブルシューティングと問題からの復旧を行ってください。

ストレージへのアクセスを試行する

ストレージにアクセスする際のトラブルシューティングのヒントについて説明します。

外部の Amazon FSx for Lustre ファイルシステムを使用する

クラスターとファイルシステム間のトラフィックが許可されていることを確認します。ファイルシステムは、ポート 988、1021、1022、および 1023 経由のインバウンドおよびアウトバウンドの TCP トラフィックを許可するセキュリティグループに関連付けられている必要があります。セキュリティグループの設定方法の詳細については、「FileSystemId」を参照してください。

外部の Amazon Elastic File System ファイルシステムを使用する

クラスターとファイルシステム間のトラフィックが許可されていることを確認します。ファイルシステムは、ポート 988、1021、1022、および 1023 経由のインバウンドおよびアウトバウンドの TCP トラフィックを許可するセキュリティグループに関連付けられている必要があります。セキュリティグループの設定方法の詳細については、「FileSystemId」を参照してください。

クラスターの削除を試行する

以下のセクションでは、クラスターを削除しようとしてエラーが発生した場合の一般的なシナリオのトラブルシューティングのヒントを示します。

pcluster delete-cluster コマンドのローカル実行に失敗する

ローカルファイルシステムの ~/.parallelcluster/pcluster-cli.log ファイルを確認します。

クラスタースタックが削除に失敗する

クラスタースタックが削除に失敗する場合、CloudFormation スタックイベントのメッセージを確認します。

問題が GitHub の AWS ParallelCluster の「GitHub Known Issues」に記載されているか、確認します。

AWS ParallelCluster API スタックのアップグレードを試行する

AWS ParallelCluster API スタックをアップグレードしようとして `UPDATE_FAILED` などのエラーが発生した場合は、GitHub の AWS ParallelCluster Wiki の「Known Issues」セクションで解決策を確認することをお勧めします。例えば、「ParallelCluster API Stack Upgrade Fails for ECR resources」を参照してください。これは、考えられる 1 つの問題を特定し、緩和オプションを提供します。

コンピューティングノードの初期化のエラーが表示されている

以下のセクションでは、コンピューティングノードの初期化中にエラーが発生した場合のトラブルシューティングのヒントを提供します。これには、ブートストラップエラー、ログでのエラーの確認、特定の状況に当てはまるシナリオがない場合の問い合わせ先が含まれます。

トピック

- clustermgtd.log に Node bootstrap error が表示されている
- オンデマンドキャパシティ予約 (ODCR) またはゾーンレベルのリザーブドインスタンスを設定しました。
- ジョブの実行に失敗したとき slurm_resume.log で、またはクラスターの作成に失敗したとき clustermgtd.log で An error occurred (VcpuLimitExceeded) が表示されている
- ジョブの実行に失敗したとき slurm_resume.log で、またはクラスターの作成に失敗したとき clustermgtd.log で An error occurred (InsufficientInstanceCapacity) が表示されている
- ノードが Reason (Code:InsufficientInstanceCapacity)... と共に DOWN ステータスで表示されている
- slurm_resume.log に cannot change locale (en_US.utf-8) because it has an invalid name が表示されている
- 前のシナリオはどれも私の状況には当てはまりません。

clustermgtd.log に Node bootstrap error が表示されている

この問題は、コンピューティングノードがブートストラップで失敗していることに関連しています。クラスター保護モードの問題をデバッグする方法の詳細については、「保護モードをデバッグする方法」を参照してください。

オンデマンドキャパシティ予約 (ODCR) またはゾーンレベルのリザーブドインスタンスを設定しました。

P4d, P4deT AWS rainium (Trn) など、複数のネットワークインターフェイスを持つインスタンスを含む ODCRs

クラスター設定ファイルで、HeadNode がパブリックサブネットにあり、コンピューティングノードがプライベートサブネットにあることを確認します。

ODCR が ターゲット ODCRS

オンデマンドキャパシティ予約 (ODCR) を使用してインスタンスを起動する の指示に従って既に **/opt/slurm/etc/pcluster/run_instances_overrides.json** を配置したのに **Unable to read file '/opt/slurm/etc/pcluster/run_instances_overrides.json'.** が表示されている

ターゲット ODCRs で AWS ParallelCluster バージョン 3.1.1 から 3.2.1 を使用していて、実行インスタンスも JSON ファイルを上書きする場合、JSON ファイルが正しくフォーマットされていない可能性があります。`clustermgtd.log` で次のような エラーが表示されることがあります。

```
Unable to read file '/opt/slurm/etc/pcluster/run_instances_overrides.json'.
Using default: {} in  /var/log/parallelcluster/clustermgtd.
```

次を実行して、JSON ファイル形式が正しいことを確認します。

```
$ echo /opt/slurm/etc/pcluster/run_instances_overrides.json | jq
```

クラスターの作成に失敗したときは **clustermgtd.log** で、またはジョブの実行に失敗したときは **slurm_resume.log** で **Found RunInstances parameters override.** が表示されている

JSON ファイルをオーバーライドしてインスタンスを実行するを使用している場合は、`/opt/slurm/etc/pcluster/run_instances_overrides.json` ファイルでキュー名とコンピューティングリソース名を正しく設定していることを確認します。

ジョブの実行に失敗したとき **slurm_resume.log** で、またはクラスターの作成に失敗したとき **clustermgtd.log** で **An error occurred (InsufficientInstanceCapacity)** が表示されている

PG-ODCR (プレイスメントグループ ODCR) を使用する

関連するプレイスメントグループを使用して ODCR を作成する場合、設定ファイルでは同じプレイスメントグループ名を使用する必要があります。クラスター設定で対応するプレイスメントグループ名を設定します。

ゾーンレベルのリザーブドインスタンスを使用する

クラスター設定で `PlacementGroup/Enabled` を `true` としてゾーンレベルのリザーブドインスタンスを使用している場合、次のようなエラーが表示されることがあります。

```
We currently do not have sufficient trn1.32xlarge capacity in the Availability Zone
 you requested (us-east-1d). Our system will be working on provisioning additional
 capacity.
You can currently get trn1.32xlarge capacity by not specifying an Availability Zone in
 your request or choosing us-east-1a, us-east-1b, us-east-1c, us-east-1e, us-east-1f.
```

これは、ゾーンレベルのリザーブドインスタンスが同じ UC (またはスパイン) に配置されていないために発生することがあります。プレイスメントグループを使用しているときに、容量不足エラー (ICE) が発生することがあります。クラスター設定の `PlacementGroup` グループ設定を無効にして、クラスターがインスタンスを割り当てることができるかどうかを判断することにより、このケースについて確認できます。

ジョブの実行に失敗したとき slurm_resume.log で、またはクラスターの作成に失敗したとき clustermgtd.log で An error occurred (VcpuLimitExceeded) が表示されている

使用している特定の EC2 インスタンスタイプについて、アカウントの vCPU 制限を確認します。vCPU の数がゼロまたはリクエストしている数より少ない場合は、制限の引き上げをリクエストします。現在の制限を確認する方法と新しい制限をリクエストする方法については、「Amazon EC2 ユーザーガイド」の「Amazon EC2 のサービスクォータ」を参照してください。

ジョブの実行に失敗したとき slurm_resume.log で、またはクラスターの作成に失敗したとき clustermgtd.log で An error occurred (InsufficientInstanceCapacity) が表示されている

容量不足の問題が発生しています。https://aws.amazon.com/premiumsupport/knowledge-center/ec2-insufficient-capacity-errors/ に従って問題をトラブルシューティングします。

ノードが Reason (Code:InsufficientInstanceCapacity)... と共に DOWN ステータスで表示されている

容量不足の問題が発生しています。https://aws.amazon.com/premiumsupport/knowledge-center/ec2-insufficient-capacity-errors/ に従って問題をトラブルシューティングします。容量不足 AWS ParallelClusterの高速フェイルオーバーモードの詳細については、「」を参照してくださいSlurm クラスタ高速容量不足フェイルオーバー。

slurm_resume.log に cannot change locale (en_US.utf-8) because it has an invalid name が表示されている

これは、yum のインストールプロセスで失敗してロケール設定に一貫性がない状態のままになっている場合に発生することがあります。例えば、これはユーザーがインストールプロセスを終了したときに発生することがあります。

原因を確認するには、次のアクションを実行します。

- `su - pcluster-admin` を実行します。

  シェルに、`cannot change locale...no such file or directory` などのエラーが表示されます。

- `localedef --list` を実行します。

  空のリストを返すか、デフォルトロケールを含んでいません。

- 最後の yum コマンドおよび `yum history` と `yum history info #ID` を確認します。最後の ID に `Return-Code: Success` が含まれていますか。

  最後の ID に `Return-Code: Success` が含まれていない場合、インストール後のスクリプトが正常に実行されていない可能性があります。

問題を解決するには、`yum reinstall glibc-all-langpacks` を使用してロケールを再構築してください。再構築後、問題が修正されているなら `su - pcluster-admin` でエラーや警告は表示されません。

前のシナリオはどれも私の状況には当てはまりません。

コンピューティングノードの初期化の問題のトラブルシューティングについては、「ノードの初期化に関する問題のトラブルシューティング」を参照してください。

シナリオが GitHub の にある GitHub の既知の問題 AWS ParallelCluster でカバーされているかどうかを確認します。 GitHub

クラスターヘルスメトリクスのトラブルシューティング

クラスターのヘルスメトリクスは、 AWS ParallelCluster バージョン 3.6.0 から AWS ParallelCluster Amazon CloudWatch ダッシュボードに追加されます。以降のセクションで、ダッシュボードヘルスメトリクスと、問題のトラブルシューティングと解決のために実行できるアクションについて説明します。

トピック

- インスタンスプロビジョニングエラーグラフが表示されている
- 「異常なインスタンスエラー」グラフが表示されている
- コンピューティングフリートのアイドル時間グラフが表示されている

インスタンスプロビジョニングエラーグラフが表示されている

`Instance Provisioning Errors` グラフに 0 以外の値が表示される場合は、Slurm ノードをバッキングする Amazon EC2 インスタンスが `CreateFleet` または `RunInstance` API で起動できなかったことを示しています。

`IAMPolicyErrors` が表示されている

- 何が起きたのか。

  多数のインスタンスが起動できませんでした。これは、権限が不十分であることが原因であり、エラーコード `UnauthorizedOperation` が出ています。

- 解決方法

カスタム `InstanceRole` または `InstanceProfile` を設定している場合は、IAM ポリシーを調べて、正しい認証情報を使用していることを確認してください。

`clustermgtd` ファイルにスタティックノードのエラーの詳細がないか確認してください。動的ノードエラーの詳細については `slurm_resume.log` ファイルを確認してください。詳細を参照して、追加する必要のある不足している権限について詳しく調べてください。

`VcpuLimitErrors` が表示されている

- 何が起きたのか。

  AWS ParallelCluster クラスターコンピューティングノード用に設定した AWS アカウント 特定の Amazon EC2 インスタンスタイプの の vCPU 制限に達したため、 はインスタンスを起動できませんでした。

- 解決方法

  静的ノードの場合は `clustermgtd` ファイルに `VcpuLimitExceeded` エラーがないか確認し、動的ノードの場合は `slurm_resume.log` ファイルで詳細を確認してください。この問題を解決するため、vCPU 制限の引き上げをリクエストできます。現在の制限を確認する方法と新しい制限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 Amazon Elastic Compute Cloud ユーザーガイド」の「Amazon Elastic Compute Cloud のサービスクォータ」を参照してください。

`VolumeLimitErrors` が表示されている

- 何が起きたのか。

  で Amazon EBS ボリュームの制限に達しており AWS アカウント、エラーコード AWS ParallelCluster `InsufficientVolumeCapacity`または でインスタンスを起動できませんVolumeLimitExceeded。

- 解決方法

  静的ノードの場合は `clustermgtd` ファイルを確認し、動的ノードの場合は `slurm_resume.log` ファイルでボリューム制限の詳細を確認してください。この問題を解決するには、別のボリュームを使用するか AWS リージョン、既存のボリュームを AWS クリーンアップするか、サポートセンターに連絡して Amazon EBS ボリューム制限の引き上げリクエストを送信してください。

`InsufficientCapacityErrors` が表示されている

- 何が起きたのか。

  AWS ParallelCluster には、Amazon EC2 インスタンスを起動してノードをバックアップするのに十分な容量がありません。

- 解決方法

  静的ノードについては `clustermgtd` ファイルを確認し、動的ノードについては `slurm_resume.log` ファイルで容量不足エラーの詳細を確認してください。この問題のトラブルシューティングを行うには、https://aws.amazon.com/premiumsupport/knowledge-center/ec2-insufficient-capacity-errors/ のガイダンスに従ってください。

`OtherInstanceLaunchFailures`

- 何が起きたのか。

  コンピューティングノードをバッキングするための Amazon EC2 インスタンスを `CreateFleet` または `RunInstance` API で起動できませんでした。

- 解決方法

  静的ノードについては `clustermgtd` ファイルを確認し、動的ノードについては `slurm_resume.log` ファイルでエラーの詳細を確認してください。

「異常なインスタンスエラー」グラフが表示されている

- 何が起きたのか。

  多数のコンピューティングインスタンスが起動されたものの、後に異常として終了しました。

- 解決方法

  異常なノードのトラブルシューティングの詳細については、「予期しないノードの置換や終了のトラブルシューティング」を参照してください。

`InstanceBootstrapTimeoutError` が表示されている

- 何が起きたのか。

インスタンスは resume_timeout (動的ノードの場合) または node_replacement_timeout (静的ノードの場合) 内のクラスターに参加できません。これは、コンピューティングノード用にネットワークが正しく設定されていない場合や、コンピューティングノードで実行されているカスタムスクリプトが終了するまでに時間がかかりすぎる場合に発生する可能性があります。

- 解決方法

  動的ノードの場合は、clustermgtd ログ (/var/log/parallelcluster/clustermgtd) でコンピューティングノードの IP アドレスと次のようなエラーを確認してください。

  ```
  Node bootstrap error: Resume timeout expires for node
  ```

  静的ノードの場合は、clustermgtd ログ (/var/log/parallelcluster/clustermgtd) でコンピューティングノードの IP アドレスと次のようなエラーを確認してください。

  ```
  Node bootstrap error: Replacement timeout expires for node ... in replacement.
  ```

  詳細については、/var/log/cloud-init-output.log ファイルでエラーを確認してください。問題のあるコンピューティングノードの IP アドレスは、clustermgtd および slurm_resume ログファイルから取得できます。

EC2HealthCheckErrors が表示されている

- 何が起きたのか。

  インスタンスが Amazon EC2 ヘルスチェックに失敗しました。

- 解決方法

  この問題のトラブルシューティングについては、「ステータスチェックに失敗したインスタンスのトラブルシューティング」を参照してください。

ScheduledEventHealthCheckErrors が表示されている

- 何が起きたのか。

  インスタンスが Amazon EC2 のスケジュールされたイベントのヘルスチェックに失敗し、正常ではありません。

- 解決方法

  この問題のトラブルシューティングについては、「インスタンスの予定されたイベント」を参照してください。

NoCorrespondingInstanceErrors が表示されている

- 何が起きたのか。

  AWS ParallelCluster はインスタンスバッキングノードを見つけることができません。ブートストラップ操作中にノードが自動的に終了した可能性があります。SlurmQueues/CustomActions/OnNodeStart | OnNodeConfigured スクリプト、またはネットワークエラーが、NoCorrespondingInstanceErrors を発生させている可能性があります。

- 解決方法

  詳細については、コンピューティングノードの /var/log/cloud-init-output.log を確認してください。

コンピューティングフリートのアイドル時間グラフが表示されている

アイドル時間のスケールダウンのしきい値よりも大幅に長い MaxDynamicNodeIdleTime が表示されている

- 何が起きたのか。

  インスタンスが正しく終了していません。MaxDynamicNodeIdleTime は、Amazon EC2 インスタンスにバッキングされた動的ノードがアイドル状態になる最大時間 (秒) を示します。アイドル時間スケールダウンのしきい値は、クラスター設定の ScaledownIdletime パラメータから算出されます。コンピューティングノードがアイドル時間スケールダウンの秒数を超えてアイドル状態になると、 はノードSlurmの電源を切り、バッキングインスタンスを AWS ParallelCluster 終了します。この場合、何かがインスタンスの終了を妨げています。

- 解決方法

  この問題の詳細については、「スケーリング問題のトラブルシューティング」の「問題のあるインスタンスやノードの置換、終了、電源オフ」を参照してください。

クラスターデプロイの問題のトラブルシューティング

クラスターの作成に失敗し、スタックの作成がロールバックされる場合は、ログファイルを参照して問題を解決します。失敗した場合のメッセージは、次の出力のようになります。

```
$ pcluster create-cluster --cluster-name mycluster --region eu-west-1 \
 --cluster-configuration cluster-config.yaml
{
  "cluster": {
    "clusterName": "mycluster",
    "cloudformationStackStatus": "CREATE_IN_PROGRESS",
    "cloudformationStackArn": "arn:aws:cloudformation:eu-west-1:xxx:stack/
mycluster/1bf6e7c0-0f01-11ec-a3b9-024fcc6f3387",
    "region": "eu-west-1",
    "version": "3.13.2",
    "clusterStatus": "CREATE_IN_PROGRESS"
  }
}

$ pcluster describe-cluster --cluster-name mycluster --region eu-west-1
{
  "creationTime": "2021-09-06T11:03:47.696Z",
  ...
  "cloudFormationStackStatus": "ROLLBACK_IN_PROGRESS",
  "clusterName": "mycluster",
  "computeFleetStatus": "UNKNOWN",
  "cloudformationStackArn": "arn:aws:cloudformation:eu-west-1:xxx:stack/
mycluster/1bf6e7c0-0f01-11ec-a3b9-024fcc6f3387",
  "lastUpdatedTime": "2021-09-06T11:03:47.696Z",
  "region": "eu-west-1",
  "clusterStatus": "CREATE_FAILED"
}
```

トピック

- で AWS CloudFormation イベントを表示する CREATE_FAILED
- CLI を使用してログストリームを表示する
- rollback-on-failure を使用して失敗したクラスターを再作成する

で AWS CloudFormation イベントを表示する CREATE_FAILED

コンソールまたは CLI AWS ParallelCluster を使用してCREATE_FAILED、エラーに関する CloudFormation イベントを表示し、根本原因を見つけることができます。

トピック

- CloudWatch コンソールでイベントを表示する
- CLI を使用して、CREATE_FAILED で CloudFormation イベントを表示し、フィルタリングします。

CloudWatch コンソールでイベントを表示する

"CREATE_FAILED" ステータスの原因に関する情報を確認するのに、CloudFormation コンソールを使用できます。

コンソールから CloudFormation のエラーメッセージを表示する。

1. にログイン AWS Management Console し、https://console.aws.amazon.com/cloudformation に移動します。
2. *cluster_name* という名前のスタックを選択します。
3. [イベント] タブを選択します。
4. [論理 ID] でリソースイベントのリストをスクロールして、作成に失敗したリソースの [ステータス] を確認します。サブタスクの作成に失敗した場合は、失敗したリソースイベントを見つけるために逆算します。
5. 例えば、次のステータスメッセージが表示される場合は、現在の vCPU 制限を超えないインスタンスタイプを使用するか、vCPU 容量の追加をリクエストする必要があります。

```
2022-02-04 16:09:44 UTC-0800 HeadNode CREATE_FAILED You have requested more vCPU
 capacity than your current vCPU limit of 0 allows
     for the instance bucket that the specified instance type belongs to. Please
 visit http://aws.amazon.com/contact-us/ec2-request to request an adjustment to
 this limit.
     (Service: AmazonEC2; Status Code: 400; Error Code: VcpuLimitExceeded; Request
 ID: a9876543-b321-c765-d432-dcba98766789; Proxy: null).
```

CLI を使用して、**CREATE_FAILED** で CloudFormation イベントを表示し、フィルタリングします。

クラスター作成の問題を診断するには、`pcluster get-cluster-stack-events` コマンドを使用して CREATE_FAILED ステータスのフィルタリングを行います。詳細については、「 AWS Command Line Interface ユーザーガイド」の AWS CLI 「出力のフィルタリング」を参照してください。

```
$ pcluster get-cluster-stack-events --cluster-name mycluster --region eu-west-1 \
    --query 'events[?resourceStatus==`CREATE_FAILED`]'
  [
    {
      "eventId": "3ccdedd0-0f03-11ec-8c06-02c352fe2ef9",
      "physicalResourceId": "arn:aws:cloudformation:eu-west-1:xxx:stack/
mycluster/1bf6e7c0-0f02-11ec-a3b9-024fcc6f3387",
      "resourceStatus": "CREATE_FAILED",
      "resourceStatusReason": "The following resource(s) failed to create: [HeadNode].
 ",
      "stackId": "arn:aws:cloudformation:eu-west-1:xxx:stack/
mycluster/1bf6e7c0-0f02-11ec-a3b9-024fcc6f3387",
      "stackName": "mycluster",
      "logicalResourceId": "mycluster",
      "resourceType": "AWS::CloudFormation::Stack",
      "timestamp": "2021-09-06T11:11:51.780Z"
    },
    {
      "eventId": "HeadNode-CREATE_FAILED-2021-09-06T11:11:50.127Z",
      "physicalResourceId": "i-04e91cc1f4ea796fe",
      "resourceStatus": "CREATE_FAILED",
      "resourceStatusReason": "Received FAILURE signal with UniqueId
 i-04e91cc1f4ea796fe",
      "resourceProperties": "{\"LaunchTemplate\":{\"Version\":\"1\",\"LaunchTemplateId
\":\"lt-057d2b1e687f05a62\"}}",
      "stackId": "arn:aws:cloudformation:eu-west-1:xxx:stack/
mycluster/1bf6e7c0-0f02-11ec-a3b9-024fcc6f3387",
      "stackName": "mycluster",
      "logicalResourceId": "HeadNode",
      "resourceType": "AWS::EC2::Instance",
      "timestamp": "2021-09-06T11:11:50.127Z"
    }
  ]
```

前の例では、失敗はヘッドノードの設定にありました。

CLI を使用してログストリームを表示する

この種の問題をデバッグするには、node-type でフィルタリングして pcluster list-cluster-log-streams のヘッドノードから利用可能なログストリームをリストアップし、ログストリームのコンテンツを分析します。

```
$ pcluster list-cluster-log-streams --cluster-name mycluster --region eu-west-1 \
--filters 'Name=node-type,Values=HeadNode'
{
  "logStreams": [
    {
      "logStreamArn": "arn:aws:logs:eu-west-1:xxx:log-group:/aws/parallelcluster/
mycluster-202109061103:log-stream:ip-10-0-0-13.i-04e91cc1f4ea796fe.cfn-init",
      "logStreamName": "ip-10-0-0-13.i-04e91cc1f4ea796fe.cfn-init",
      ...
    },
    {
      "logStreamArn": "arn:aws:logs:eu-west-1:xxx:log-group:/aws/parallelcluster/
mycluster-202109061103:log-stream:ip-10-0-0-13.i-04e91cc1f4ea796fe.chef-client",
      "logStreamName": "ip-10-0-0-13.i-04e91cc1f4ea796fe.chef-client",
      ...
    },
    {
      "logStreamArn": "arn:aws:logs:eu-west-1:xxx:log-group:/aws/parallelcluster/
mycluster-202109061103:log-stream:ip-10-0-0-13.i-04e91cc1f4ea796fe.cloud-init",
      "logStreamName": "ip-10-0-0-13.i-04e91cc1f4ea796fe.cloud-init",
      ...
    },
    ...
  ]
}
```

初期化エラーを見つけるために使用できる 2 つの主要なログストリームは次のとおりです。

- cfn-init は cfn-init のスクリプトのログです。まず、このログストリームを確認します。このログには Command chef failed エラーが表示される可能性があります。エラーメッセージの詳細については、この行の直前の行を参照してください。詳細については、「cfn-init」を参照してください。

- cloud-init は cloud-init のログです。cfn-init に何も表示されない場合は、次にこのログを確認してください。

ログストリームの内容を pcluster get-cluster-log-events で取得することができます (取得するイベント数を制限する --limit 5 オプションに注意してください)。

```
$ pcluster get-cluster-log-events --cluster-name mycluster \
  --region eu-west-1 --log-stream-name ip-10-0-0-13.i-04e91cc1f4ea796fe.cfn-init \
  --limit 5
{
  "nextToken": "f/36370880979637159565202782352491087067973952362220945409/s",
  "prevToken": "b/36370880752972385367337528725601470541902663176996585497/s",
  "events": [
    {
      "message": "2021-09-06 11:11:39,049 [ERROR] Unhandled exception during build:
 Command runpostinstall failed",
      "timestamp": "2021-09-06T11:11:39.049Z"
    },
    {
      "message": "Traceback (most recent call last):\n  File \"/opt/aws/bin/
cfn-init\", line 176, in <module>\n    worklog.build(metadata, configSets)\n
 File \"/usr/lib/python3.7/site-packages/cfnbootstrap/construction.py\", line
 135, in build\n    Contractor(metadata).build(configSets, self)\n  File \"/
usr/lib/python3.7/site-packages/cfnbootstrap/construction.py\", line 561, in
 build\n    self.run_config(config, worklog)\n  File \"/usr/lib/python3.7/
site-packages/cfnbootstrap/construction.py\", line 573, in run_config\n
 CloudFormationCarpenter(config, self._auth_config).build(worklog)\n  File \"/usr/
lib/python3.7/site-packages/cfnbootstrap/construction.py\", line 273, in build\n
   self._config.commands)\n  File \"/usr/lib/python3.7/site-packages/cfnbootstrap/
command_tool.py\", line 127, in apply\n    raise ToolError(u\"Command %s failed\" %
 name)",
      "timestamp": "2021-09-06T11:11:39.049Z"
    },
    {
      "message": "cfnbootstrap.construction_errors.ToolError: Command runpostinstall
 failed",
      "timestamp": "2021-09-06T11:11:39.049Z"
    },
    {
      "message": "2021-09-06 11:11:49,212 [DEBUG] CloudFormation client initialized
 with endpoint https://cloudformation.eu-west-1.amazonaws.com",
      "timestamp": "2021-09-06T11:11:49.212Z"
```

```
    },
    {
      "message": "2021-09-06 11:11:49,213 [DEBUG] Signaling resource HeadNode in stack
 mycluster with unique ID i-04e91cc1f4ea796fe and status FAILURE",
      "timestamp": "2021-09-06T11:11:49.213Z"
    }
  ]
}
```

前述の例では、失敗の原因は runpostinstall の失敗なので、CustomActions の OnNodeConfigured 設定パラメータで使われているカスタムブートストラップスクリプトの内容と厳密に関係しています。

rollback-on-failure を使用して失敗したクラスターを再作成する

AWS ParallelCluster は、ロググループにクラスター CloudWatch ログストリームを作成します。これらのログは、CloudWatch コンソールのカスタムダッシュボードまたはロググループで表示できます。詳細については、「Amazon CloudWatch Logs との統合」および「Amazon CloudWatch ダッシュボード」を参照してください。利用可能なログストリームがない場合は、CustomActions カスタムブートストラップスクリプトや AMI 関連の問題が失敗の原因である可能性があります。この場合の作成問題を診断するには、--rollback-on-failure パラメータを false に設定した上で、pcluster create-cluster を使用してクラスターを再度作成します。次に、SSH を使用して、次に示すようにクラスターを表示します。

```
$ pcluster create-cluster --cluster-name mycluster --region eu-west-1 \
   --cluster-configuration cluster-config.yaml --rollback-on-failure false
 {
   "cluster": {
     "clusterName": "mycluster",
     "cloudformationStackStatus": "CREATE_IN_PROGRESS",
     "cloudformationStackArn": "arn:aws:cloudformation:eu-west-1:xxx:stack/
mycluster/1bf6e7c0-0f01-11ec-a3b9-024fcc6f3387",
     "region": "eu-west-1",
     "version": "3.13.2",
     "clusterStatus": "CREATE_IN_PROGRESS"
   }
 }
 $ pcluster ssh --cluster-name mycluster
```

ヘッドノードにログインすると、エラーの原因を見つけるための 3 つの主要なログファイルが見つかります。

- `/var/log/cfn-init.log` は `cfn-init` のスクリプトのログです。最初に、このログを確認してください。このログには `Command chef failed` などのエラーが表示される可能性があります。エラーメッセージの詳細については、この行の直前の行を参照してください。詳細については、「cfn-init」を参照してください。
- `/var/log/cloud-init.log` は cloud-init のログです。`cfn-init.log` に何も表示されない場合は、次にこのログを確認してください。
- `/var/log/cloud-init-output.log` は cloud-init で実行されたコマンドの出力です。これには `cfn-init` からの出力も含まれます。通常、この種の問題のトラブルシューティングには、このログを見る必要はありません。

Terraform を使用したクラスターのデプロイのトラブルシューティング

このセクションは、Terraform を使用してデプロイしたクラスターに関連しています。

ParallelCluster API が見つからない

ParallelCluster API が見つからないため、計画は失敗する可能性があります。この場合、次のようなエラーが返されます。

```
Planning failed. Terraform encountered an error while generating this plan.

#
# Error: Unable to retrieve ParallelCluster API cloudformation stack.
#
#   with provider["registry.terraform.io/aws-tf/aws-parallelcluster"],
#   on providers.tf line 6, in provider "aws-parallelcluster":
#    6: provider "aws-parallelcluster" {
#
# operation error CloudFormation: DescribeStacks, https response error StatusCode: 400,
 RequestID: REQUEST_ID, api error ValidationError: Stack with id PCAPI_STACK_NAME does
 not exist
```

このエラーを解決するには、クラスターを作成する先のアカウントに ParallelCluster API をデプロイします。「the section called “Terraform を使用したクラスターの作成”」を参照してください。

ParallelCluster API を呼び出す権限がない

Terraform プロジェクトをデプロイするために引き受けた IAM ロール/ユーザーには ParallelCluster API を操作するアクセス許可がないため、計画が失敗する可能性があります。この場合、次のようなエラーが返されます。

```
Planning failed. Terraform encountered an error while generating this plan.

# Error: 403 Forbidden
#
#   with
 module.parallelcluster_clusters.module.clusters[0].pcluster_cluster.managed_configs["DemoClust
#   on .terraform/modules/parallelcluster_clusters/modules/clusters/main.tf line 35, in
 resource "pcluster_cluster" "managed_configs":
#   35: resource "pcluster_cluster" "managed_configs" {
#
# {{"Message":"User: USER_ARN is not authorized to perform: execute-api:Invoke on
 resource: PC_API_REST_RESOURCE with an explicit deny"}
# }
```

このエラーを解決するには、ParallelCluster API ロールを使用して API を操作するように ParallelCluster プロバイダーを設定します。

```
provider "aws-parallelcluster" {
  region         = var.region
  profile        = var.profile
  api_stack_name = var.api_stack_name
  **use_user_role** **= true**
}
```

スケーリング問題のトラブルシューティング

このセクションは、Slurmジョブスケジューラで AWS ParallelCluster バージョン 3.0.0 以降を使用してインストールされたクラスターに関連しています。複数のキューの設定の詳細については、「複数のキューの設定」を参照してください。

稼働中のクラスターの 1 つに問題が発生した場合、トラブルシューティングを開始する前に次のコマンドを実行してクラスターを STOPPED 状態にします。これにより、予想外のコストが発生することを防ぐことができます。

```
$ pcluster update-compute-fleet --cluster-name mycluster \
   --status STOP_REQUESTED
```

pcluster list-cluster-log-streams コマンドを使用し、障害ノードまたはヘッドノードのいずれかの private-dns-name を使用してフィルタリングすることで、クラスターノードから利用可能なログストリームをリストアップすることができます。

```
$ pcluster list-cluster-log-streams --cluster-name mycluster --region eu-west-1 \
 --filters 'Name=private-dns-name,Values=ip-10-0-0-101'
```

そして、pcluster get-cluster-log-events コマンドを使用して、次のセクションで述べたキーログの 1 つに対応する --log-stream-name を渡すことで、ログストリームの内容を取り出して分析することができます。

```
$ pcluster get-cluster-log-events --cluster-name mycluster \
--region eu-west-1 --log-stream-name ip-10-0-0-13.i-04e91cc1f4ea796fe.cfn-init
```

AWS ParallelCluster は、ロググループにクラスター CloudWatch ログストリームを作成します。これらのログは、CloudWatch コンソールのカスタムダッシュボードまたはロググループで表示できます。詳細については、「Amazon CloudWatch Logs との統合」および「Amazon CloudWatch ダッシュボード」を参照してください。

トピック

- デバッグ用のキーログ
- ジョブの実行に失敗したとき slurm_resume.log で、またはクラスターの作成に失敗したとき clustermgtd.log で InsufficientInstanceCapacity エラーが表示されている
- ノードの初期化に関する問題のトラブルシューティング
- 予期しないノードの置換や終了のトラブルシューティング
- 問題のあるインスタンスやノードの置換、終了、電源オフ
- キュー (パーティション) の Inactive ステータス
- その他の既知のノードやジョブの問題のトラブルシューティング

デバッグ用のキーログ

次の表は、ヘッドノードのキーログの概要を示したものです。

- `/var/log/cfn-init.log` - これは init AWS CloudFormation ログです。インスタンスがセットアップされた時に実行されたすべてのコマンドが含まれています。初期化時の問題のトラブルシューティングに使用します。
- `/var/log/chef-client.log` - これは Chef クライアントログです。Chef/CINC で実行されたすべてのコマンドが含まれています。初期化時の問題のトラブルシューティングに使用します。
- `/var/log/parallelcluster/slurm_resume.log` - これは `ResumeProgram` ログです。動的ノードのインスタンスを起動します。ダイナミックノードの起動に関する問題のトラブルシューティングに使用します。
- `/var/log/parallelcluster/slurm_suspend.log` - これが `SuspendProgram` のログです。動的ノードのインスタンスが終了する際に呼び出されます。ダイナミックノードの終了に関する問題のトラブルシューティングに使用します。このログを確認する際には、`clustermgtd` のログも確認する必要があります。
- `/var/log/parallelcluster/clustermgtd` - これが `clustermgtd` のログです。ほとんどのクラスターオペレーションのアクションを管理する集中型デーモンとして動作します。起動、終了、またはクラスターの動作の問題のトラブルシューティングに使用します。
- `/var/log/slurmctld.log` - これはSlurmコントロールデーモン log. AWS ParallelCluster does がスケーリングを決定しません。むしろ、Slurm の要求を満たすためにリソースの起動を試みます。スケーリングや割り当ての問題、ジョブ関連の問題、スケジューラ関連の起動や終了の問題などに有効です。
- `/var/log/parallelcluster/compute_console_output` - このログには、予期せず終了した静的コンピューティングノードのサンプルサブセットからのコンソール出力が記録されます。静的コンピューティングノードが終了し、そのコンピューティングノードログが CloudWatch で利用できない場合は、このログを使用します。Amazon EC2 コンソールまたは を使用してインスタンスコンソールの出力 AWS CLI を取得する場合、受け取る`compute_console_output log`コンテンツは同じです。

以下は、コンピューティングノードの主要なログです。

- `/var/log/cloud-init-output.log` - これは cloud-init のログです。インスタンスがセットアップされた時に実行されたすべてのコマンドが含まれています。初期化時の問題のトラブルシューティングに使用します。
- `/var/log/parallelcluster/computemgtd` - これが `computemgtd` のログです。各コンピューティングノードで実行し、ヘッドノードの `clustermgtd` デーモンがオフラインであるという一般的でなイベントのノードをモニタリングします。予期せぬ終了の問題のトラブルシューティングに使用します。

- /var/log/slurmd.log - Slurm コンピューティングデーモンのログです。初期化およびコンピュートの不具合の問題に関するトラブルシューティングに使用します。

ジョブの実行に失敗したとき slurm_resume.log で、またはクラスターの作成に失敗したとき clustermgtd.log で InsufficientInstanceCapacity エラーが表示されている

クラスターが Slurm スケジューラを使用している場合、容量不足の問題が発生しています。インスタンス起動のリクエスト時に十分な数のインスタンスが用意されていない場合は、InsufficientInstanceCapacity エラーが返されます。

静的インスタンス容量の場合、/var/log/parallelcluster/clustermgtd の clustermgtd ログでエラーを見つけることができます。

動的インスタンス容量の場合、/var/log/parallelcluster/slurm_resume.log の ResumeProgram ログでエラーを見つけることができます。

メッセージは、次の例のようになります。

```
An error occurred (InsufficientInstanceCapacity) when calling the RunInstances/
CreateFleet operation...
```

ユースケースによっては、これらの種類のエラーメッセージが表示されないように、次のいずれかの方法を使用することを考慮します。

- プレイスメントグループが有効になっている場合は、無効にします。詳細については、「プレイスメントグループとインスタンスの起動に関する問題」を参照してください。
- インスタンスのキャパシティを予約し、ODCR (オンデマンドキャパシティ予約) を使用して起動します。詳細については、「オンデマンドキャパシティ予約 (ODCR) を使用してインスタンスを起動する」を参照してください。
- 異なるインスタンスタイプを持つ複数のコンピューティングリソースを設定します。ワークロードに特定のインスタンスタイプが必要でない場合、複数のコンピューティングリソースによる容量不足の高速フェイルオーバーを活用できます。詳細については、「Slurm クラスタ高速容量不足フェイルオーバー」を参照してください。
- 同じコンピューティングリソースに複数のインスタンスタイプを設定し、複数のインスタンスタイプの割り当てを活用します。複数のインスタンスの設定に関す

る詳細については、Slurm による複数のインスタンスタイプの割り当て および Scheduling/SlurmQueues/ComputeResources/Instances を参照してください。

- クラスター設定 Scheduling/SlurmQueues/Networking/SubnetIds のサブネット ID を変更して、キューを異なるアベイラビリティーゾーンに移動します。
- ワークロードが密接に結合されていない場合は、キューをさまざまなアベイラビリティーゾーンにまたがるようにしてください。複数のサブネットの設定に関する詳細については、「Scheduling」/「SlurmQueues」/「Networking」/「SubnetIds」を参照してください。

ノードの初期化に関する問題のトラブルシューティング

このセクションでは、ノードの初期化に関するトラブルを解決する方法について説明します。これには、ノードが起動しない、電源が入らない、またはクラスターが参加できない、などの問題が含まれます。

トピック

- ヘッドノード
- コンピューティングノード

ヘッドノード

該当するログ:

- /var/log/cfn-init.log
- /var/log/chef-client.log
- /var/log/parallelcluster/clustermgtd
- /var/log/parallelcluster/slurm_resume.log
- /var/log/slurmctld.log

/var/log/cfn-init.log および /var/log/chef-client.log のログまたは対応するログストリームを確認してください。これらのログには、ヘッドノードのセットアップ時に実行されたすべてのアクションが含まれています。セットアップ中に発生するほとんどのエラーは、/var/log/chef-client.log ログにエラーメッセージが表示されます。クラスターの構成で OnNodeStart または OnNodeConfigured のスクリプトが指定されている場合、スクリプトが正常に実行されていることをログメッセージで再確認します。

クラスターが作成されると、ヘッドノードはコンピューティングノードがクラスターに参加するのを待ってからクラスターに参加する必要があります。このため、コンピューティングノードがクラスターに参加できないと、ヘッドノードも失敗してしまいます。このような問題を解決するためには、使用しているコンピュートノートの種類に応じて、次の手順のいずれかを実行します。

コンピューティングノード

- 該当するログ:
  - /var/log/cloud-init-output.log
  - /var/log/slurmd.log
- コンピューティングノードが起動した場合、まず /var/log/cloud-init-output.log を確認します。ヘッドノードの /var/log/chef-client.log ログと同様のセットアップログが含まれているはずです。セットアップ中に発生するほとんどのエラーは、/var/log/cloud-init-output.log ログにエラーメッセージが表示されます。クラスター構成でプレインストールまたはポストインストールスクリプトが指定されている場合、それらが正常に実行されたかどうかを確認します。
- Slurm の設定を変更したカスタム AMI を使用している場合は、Slurm 関連のエラーが発生し、コンピューティングノードがクラスターに参加できない可能性があります。スケジューラ関連のエラーについては、/var/log/slurmd.log のログを確認してください。

動的コンピューティングノード:

- ResumeProgram ログ (/var/log/parallelcluster/slurm_resume.log) でコンピューティングノード名を検索し、そのノードで ResumeProgram が呼び出されたことがあるかどうかを調べます (ResumeProgramが呼び出されたことがない場合は、 slurmctld ログ (/var/log/slurmctld.log) をチェックして、ノードResumeProgramで を呼び出そうとしたSlurmかどうかを判断できます) 。
- なお、ResumeProgram のパーミッションが正しくないと、ResumeProgram がバックグラウンドで失敗する可能性があります。ResumeProgram の設定を変更したカスタム AMI を使用している場合は、ResumeProgram の所有者が slurm ユーザーであり、744 (rwxr--r--) の許可を持っていることを確認してください。
- ResumeProgram が呼ばれた場合、そのノードのインスタンスが起動しているかどうかを確認します。インスタンスが起動しなかった場合は、起動の失敗を示すエラーメッセージが表示されます。
- インスタンスが起動する場合は、セットアップ時に問題が発生した可能性があります。ResumeProgram のログに対応するプライベート IP アドレスとインスタンス ID が表示され

ます。さらに、特定のインスタンスに対応するセットアップログを確認することができます。コンピューティングノードのセットアップエラーのトラブルシューティングについては、次のセクションを参照してください。

静的コンピューティングノード:

- `clustermgtd` (`/var/log/parallelcluster/clustermgtd`) ログをチェックして、ノードに対してインスタンスが起動されたかどうかを確認します。起動されていない場合は、起動失敗の詳細を示す明確なエラーメッセージが表示されるはずです。
- インスタンスが起動した場合、セットアップ中に何らかの問題が発生します。`ResumeProgram` のログに対応するプライベート IP アドレスとインスタンス ID が表示されます。さらに、特定のインスタンスに対応するセットアップログを確認することができます。

スポットインスタンスにバックアップされたコンピューティングノード:

- スポットインスタンスを初めて使用し、ジョブが PD (保留中) のままである場合は、`/var/log/parallelcluster/slurm_resume.log` ファイルを再確認します。次のようなエラーが見つかる可能性があります。

```
2022-05-20 13:06:24,796 - [slurm_plugin.common:add_instances_for_nodes] - ERROR -
 Encountered exception when launching instances for nodes (x1) ['spot-dy-t2micro-2']:
 An error occurred (AuthFailure.ServiceLinkedRoleCreationNotPermitted) when calling
 the RunInstances operation: The provided credentials do not have permission to
 create the service-linked role for Amazon EC2 Spot Instances.
```

スポットインスタンスを使用する場合、お客様のアカウントに `AWSServiceRoleForEC2Spot` サービスにリンクしたロールが存在する必要があります。を使用してアカウントにこのロールを作成するには AWS CLI、次のコマンドを実行します。

```
$ aws iam create-service-linked-role --aws-service-name spot.amazonaws.com
```

詳細については、 AWS ParallelCluster 「 ユーザーガイドスポットインスタンスの操作」の「」および「Amazon EC2 ユーザーガイド」の「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

予期しないノードの置換や終了のトラブルシューティング

このセクションでは、ノードに関連する問題、特にノードが予期せず置換されたり終了したりした場合のトラブルシューティング方法について引き続き説明します。

- 該当するログ:
  - `/var/log/parallelcluster/clustermgtd` (ヘッドノード)
  - `/var/log/slurmctld.log` (ヘッドノード)
  - `/var/log/parallelcluster/computemgtd` (コンピューティングノード)

予期せず置換または終了したノード

- `clustermgtd` がノードを置換または終了させたかどうかを確認するには、`clustermgtd` のログ (`/var/log/parallelcluster/clustermgtd`) を確認します。なお、`clustermgtd` は通常のノードメンテナンスのアクションをすべて行います。
- `clustermgtd` がノードを置換または終了させた場合、なぜそのアクションがノードに対して行われたのかを詳細に説明するメッセージが必要です。スケジューラ関連の理由 (例えば、ノードが `DOWN` にあるため) の場合は、`slurmctld` ログで確認してください。理由が Amazon EC2 に関連するものであれば、置換を必要とする Amazon EC2 に関連する問題の詳細を示す情報メッセージが表示されるはずです。
- `clustermgtd` がノードを終了しなかった場合は、まず、これが Amazon EC2 によって予期された終了、具体的にはスポット終了であるかどうかを確認します。コンピューティングノードでcomputemgtd実行されている は、 clustermgtdが異常であると判断された場合にノードを終了することもできます。`computemgtd` のログ (`/var/log/parallelcluster/computemgtd`) を確認し、`computemgtd` がノードを終了したかどうかを確認します。

ノードが失敗しました

- `slurmctld` ログ (`/var/log/slurmctld.log`) で、ジョブやノードが失敗した理由を確認します。なお、ノードに障害が発生した場合、ジョブは自動的に再キューされます。
- `slurm_resume` がノードの起動を報告し、`clustermgtd` が数分後にそのノードに対応するインスタンスが Amazon EC2 に存在しないと報告した場合、ノードはセットアップ中に失敗する可能性があります。コンピュート (`/var/log/cloud-init-output.log`) からログを取得するには、次の手順で行います。
  - Slurm が新しいノードを立ち上げるためのジョブを送信します。
  - コンピューティングノードが起動するまで待ちます。

- インスタンスにより開始したシャットダウン動作を変更して、障害が発生したコンピューティングノードを終了するのではなく停止するようにします。

```
$ aws ec2 modify-instance-attribute \
    --instance-id i-1234567890abcdef0 \
    --instance-initiated-shutdown-behavior "{\"Value\": \"stop\"}"
```

- 削除保護を有効にします。

```
$ aws ec2 modify-instance-attribute \
    --instance-id i-1234567890abcdef0 \
    --disable-api-termination
```

- 簡単に識別できるようにノードにタグを付けます。

```
$ aws ec2 create-tags \
    --resources i-1234567890abcdef0 \
    --tags Key=Name,Value=QUARANTINED-Compute
```

- `parallelcluster:cluster-name` タグを変更して、クラスターからノードをデタッチします。

```
$ aws ec2 create-tags \
    --resources i-1234567890abcdef0 \
    --tags Key=parallelcluster:clustername,Value=QUARANTINED-ClusterName
```

- このコマンドで、ノードのコンソール出力を取得します。

```
$ aws ec2 get-console-output --instance-id i-1234567890abcdef0 --output text
```

問題のあるインスタンスやノードの置換、終了、電源オフ

- 該当するログ:
  - `/var/log/parallelcluster/clustermgtd` (ヘッドノード)
  - `/var/log/parallelcluster/slurm_suspend.log` (ヘッドノード)
- 通常、`clustermgtd` は期待されるすべてのインスタンス終了アクションを処理します。`clustermgtd` ログを見て、ノードの置換または終了に失敗した理由を確認します。

- SlurmSettings プロパティ に失敗した動的ノードの場合、SuspendProgram ログで SuspendProgram が slurmctld から特定のノードを引数として呼び出されたかどうかを確認します。なお、SuspendProgram は実際には何のアクションもしません。呼び出されたときだけログに記録されます。インスタンスの終了や NodeAddr のリセットは全て clustermgtd が行います。Slurm は SuspendTimeout の後、自動的にノードを POWER_SAVING 状態に戻します。
- ブートストラップの失敗のためコンピューティングノードに継続的に障害が発生する場合は、Slurm クラスター保護モード が有効な状態で起動されているかどうかを確認してください。保護モードが有効になっていない場合は、保護モードの設定を変更して保護モードを有効にします。ブートストラップスクリプトのトラブルシューティングをして修正してください。

キュー (パーティション) の Inactive ステータス

sinfo を実行して出力に inact の AVAIL ステータスのキューが表示される場合は、クラスターで Slurm クラスター保護モード が有効になっており、キューが事前に定義した期間に INACTIVE ステータスに設定されていた可能性があります。

その他の既知のノードやジョブの問題のトラブルシューティング

もう 1 つの既知の問題は、ジョブの割り当てやスケーリングの決定に失敗 AWS ParallelCluster する可能性があることです。このタイプの発行では、Slurm の指示に従って、リソースの起動、 AWS ParallelCluster の終了、または維持のみが行われます。これらの問題については、slurmctld ログを確認してトラブルシューティングを行ってください。

プレイスメントグループとインスタンスの起動に関する問題

ノード間のレイテンシーを最小にするには、プレースメントグループを使用します。プレイスメントグループにより、インスタンスが同じネットワーキングバックボーンに配置されます。リクエスト時に十分な数のインスタンスが用意されていない場合は、InsufficientInstanceCapacity エラーが返ります。クラスタープレイスメントグループを使用する際にこのエラーが発生する可能性を減らすために、SlurmQueues/Networking/PlacementGroup/Enabled パラメータを false に設定します。

キャパシティアクセスをさらに制御するには、「Launching instances with ODCR (On-Demand Capacity Reservations)」を考慮します。

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「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」および「プレイスメントグループの役割と制限」を参照してください。

ディレクトリの置換

一部のディレクトリは置換できません。ディレクトリの置き換えに問題がある場合、置換不能が原因となっていることがあります。以下のディレクトリはノード間で共有され、置換できません。

- `/opt/intel` - これには、インテル MPI、インテル Parallel Studio、および関連ファイルが含まれます。
- `/opt/slurm` - これには、Slurm ワークロードマネージャーおよび関連ファイルが含まれます。(条件付き、`Scheduler: slurm` の場合のみ)

Amazon DCV の問題のトラブルシューティング

トピック

- Amazon DCV のログ
- Ubuntu Amazon DCV の問題

Amazon DCV のログ

Amazon DCV のログは `/var/log/dcv/` ディレクトリのファイルに書き込まれます。これらのログを確認することは、問題の解決に役立ちます。

インスタンスタイプは、Amazon DCV を実行するために、少なくとも1.7 ギビバイト (GiB) の RAM を必要とします。nano や micro のインスタンスタイプには、Amazon DCV を実行するのに十分なメモリがありません。

AWS ParallelCluster は、ロググループに Amazon DCV ログストリームを作成します。これらのログは、CloudWatch コンソールのカスタムダッシュボードまたはロググループで表示できます。詳細については、「Amazon CloudWatch Logs との統合」および「Amazon CloudWatch ダッシュボード」を参照してください。

Ubuntu Amazon DCV の問題

Ubuntu の Amazon DCV セッションで Gnome ターミナルを実行すると、 AWS ParallelCluster でログインシェルから利用可能となるユーザー環境に自動的にアクセスできない場合があります。ユーザー環境には、openmpi や intelmpi などの環境モジュールやその他のユーザー設定が用意されています。

Gnome ターミナルのデフォルト設定では、シェルをログインシェルとして起動することはできません。つまり、シェルプロファイルは自動的にソース化されず、 AWS ParallelCluster ユーザー環境はロードされません。

シェルプロファイルを適切にソース化し、 AWS ParallelCluster ユーザー環境にアクセスするには、次のいずれかを実行します。

- デフォルトのターミナル設定を変更します。
  1. Gnome ターミナルで [編集] メニューを選択します。
  2. [設定]、[プロファイル] の順に選択します。
  3. [コマンド] を選択し、[ログインシェルとしてコマンドを実行] を選択します。
  4. [新しいターミナル] を開きます。
- コマンドラインを使用して、使用可能なプロファイルを取得します。

```
$ source /etc/profile && source $HOME/.bashrc
```

AWS Batch 統合によるクラスターの問題のトラブルシューティング

このセクションでは、特にヘッドノードの問題、コンピューティングの問題、ジョブの失敗、タイムアウトエラーなど、ス AWS Batch ケジューラ統合を使用するクラスターで考えられるトラブルシューティングのヒントを提供します。

トピック

- ヘッドノードの問題
- コンピュートの問題
- ジョブの失敗
- エンドポイント URL の接続タイムアウトエラー

ヘッドノードの問題

ヘッドノードのセットアップに関連する問題は、Slurm クラスターと同様にトラブルシューティングを行うことができます (Slurm 固有のログは除く)。これらの問題を解決する方法の詳細については、「ヘッドノード」を参照してください。

コンピュートの問題

AWS Batch は、サービスのスケーリングとコンピューティングの側面を管理します。コンピューティング関連の問題が発生した場合は、 AWS Batch トラブルシューティングドキュメントを参照してください。

ジョブの失敗

ジョブが失敗した場合は、`awsbout` コマンドを実行してジョブの出力を取得することができます。また、`awsbstat` コマンドを実行して、Amazon CloudWatch が保存しているジョブログへのリンクを取得することもできます。

エンドポイント URL の接続タイムアウトエラー

マルチノード並列ジョブが、エラー `Connect timeout on endpoint URL` で失敗する場合:

- `awsbout` 出力ログで、出力からジョブがマルチノード並列であることを確認します。`Detected 3/3 compute nodes. Waiting for all compute nodes to start.`
- コンピューティングノードのサブネットがパブリックかどうかを確認します。

マルチノード並列ジョブは、 AWS Batch で を使用する場合のパブリックサブネットの使用をサポートしていません AWS ParallelCluster。コンピューティングノードとジョブにはプライベートサブネットを使用します。詳細については、「AWS Batch ユーザーガイド」の「Compute environment considerations」を参照してください。コンピューティングノードにプライベートサブネットを設定するには、「AWS ParallelCluster スケジューラを使用する AWS Batch」を参照してください。

Active Directory を使用したマルチユーザー統合のトラブルシューティング

このセクションは、Active Directory と統合されたクラスターに関連するものです。

Active Directory 統合機能が期待どおりに動作していない場合、SSSD ログは役立つ診断情報を提供します。これらのログは、クラスターノードの `/var/log/sssd` に配置されています。デフォルトでは、クラスターの Amazon CloudWatch ロググループにも保存されます。

トピック

- Active Directory 固有のトラブルシューティング

- デバッグモードを有効にする
- LDAPS から LDAP に移行する方法
- LDAPS サーバー証明書の検証を無効にする方法
- パスワードではなく SSH キーを使用してログインする方法
- ユーザーパスワードと失効しているパスワードをリセットする方法
- 参加しているドメインを確認する方法
- 証明書に関する問題をトラブルシューティングする方法
- Active Directory との統合が動作していることを確認する方法
- コンピューティングノードへのログインのトラブルシューティング方法
- マルチユーザー環境での SimCenter StarCCM+ ジョブに関する既知の問題
- ユーザー名解決に関する既知の問題
- ホームディレクトリ作成の問題の解決方法

Active Directory 固有のトラブルシューティング

このセクションは、Active Directory タイプ固有のトラブルシューティングに関連するものです。

Simple AD

- `DomainReadOnlyUser` 値は、ユーザーの Simple AD ディレクトリベース検索と一致している必要があります。

  `cn=ReadOnlyUser,cn=Users,dc=corp,dc=example,dc=com`

  `Users` の `cn` に注意してください。
- デフォルトの管理者ユーザーは `Administrator` です。
- `Ldapsearch` ユーザー名の前に NetBIOS 名が必要です。

  `Ldapsearch` の構文は次のようである必要があります。

```
$ ldapsearch -x -D "corp\\Administrator" -w "Password" -H ldap://192.0.2.103 \
   -b "cn=Users,dc=corp,dc=example,dc=com"
```

AWS Managed Microsoft AD

- `DomainReadOnlyUser` 値は、ユーザーの AWS Managed Microsoft AD ディレクトリベース検索と一致する必要があります。

  `cn=ReadOnlyUser,ou=Users,ou=CORP,dc=corp,dc=example,dc=com`
- デフォルトの管理者ユーザーは `Admin` です。
- `Ldapsearch` の構文は次のようである必要があります。

```
$ ldapsearch -x -D "Admin" -w "Password" -H ldap://192.0.2.103 \
   -b "ou=Users,ou=CORP,dc=corp,dc=example,dc=com"
```

デバッグモードを有効にする

SSSD からのデバッグログは問題をトラブルシューティングするのに役立ちます。デバッグモードを有効にするには、クラスター設定に次の変更を加えてクラスターを更新する必要があります。

```
DirectoryService:
  AdditionalSssdConfigs:
    debug_level: "0x1ff"
```

LDAPS から LDAP に移行する方法

LDAP だけでは暗号化を提供できないため、LDAPS (TLS/SSL を使用する LDAP) から LDAP への移行は推奨されていません。それでも、テスト目的やトラブルシューティングに役立ちます。

クラスターを以前の設定定義で更新することにより、クラスターを以前の設定に復元できます。

LDAPS から LDAP に移行するには、クラスター設定に次の変更を加えてクラスターを更新する必要があります。

```
DirectoryService:
  LdapTlsReqCert: never
  AdditionalSssdConfigs:
    ldap_auth_disable_tls_never_use_in_production: True
```

LDAPS サーバー証明書の検証を無効にする方法

テストまたはトラブルシューティングの目的でヘッドノードでの LDAPS サーバー証明書の検証を一時的に無効にすることは役立ちます。

クラスターを以前の設定定義で更新することにより、クラスターを以前の設定に復元できます。

LDAPS サーバー証明書の検証を無効にするには、クラスター設定に次の変更を加えてクラスターを更新する必要があります。

```
DirectoryService:
  LdapTlsReqCert: never
```

パスワードではなく SSH キーを使用してログインする方法

SSH キーは、パスワードを使用して初めてログインした後、`/home/$user/.ssh/id_rsa` に作成されます。SSH キーを使用してログインするには、パスワードを使用してログインし、SSH キーをローカルにコピーしてから、通常どおりパスワードなしで SSH キーを使用する必要があります。

```
$ ssh -i $LOCAL_PATH_TO_SSH_KEY $username@$head_node_ip
```

ユーザーパスワードと失効しているパスワードをリセットする方法

ユーザーがクラスターにアクセスできなくなった場合、AWS Managed Microsoft AD パスワードが失効している可能性があります。

パスワードをリセットするには、ディレクトリの書き込みアクセス許可を持つユーザーとロールを使用し、次のコマンドを実行します。

```
$ aws ds reset-user-password \
  --directory-id "d-abcdef01234567890" \
  --user-name "USER_NAME" \
  --new-password "NEW_PASSWORD" \
  --region "region-id"
```

`DirectoryService`/`DomainReadOnlyUser` のパスワードをリセットする場合。

1. 必ず新しいパスワードで `DirectoryService`/`PasswordSecretArn` シークレットを更新します。

2. 新しいシークレット値にクラスターを更新します。
   a. `pcluster update-compute-fleet` コマンドを使用してコンピューティングフリートを停止します。
   b. クラスターヘッドノード内から、次のコマンドを実行します。

```
$ sudo /opt/parallelcluster/scripts/directory_service/
update_directory_service_password.sh
```

パスワードのリセットとクラスターの更新の後、ユーザーのクラスターアクセスは復元されています。

詳細については、「AWS Directory Service 管理ガイド」の「ユーザーパスワードをリセットする」を参照してください。

参加しているドメインを確認する方法

次のコマンドは、ヘッドノードではなく、ドメインに参加しているインスタンスから実行する必要があります。

```
$ realm list corp.example.com \
  type: kerberos \
  realm-name: CORP.EXAMPLE.COM \
  domain-name: corp.example.com \
  configured: kerberos-member \
  server-software: active-directory \
  client-software: sssd \
  required-package: oddjob \
  required-package: oddjob-mkhomedir \
  required-package: sssd \
  required-package: adcli \
  required-package: samba-common-tools \
  login-formats: %U \
  login-policy: allow-realm-logins
```

証明書に関する問題をトラブルシューティングする方法

LDAPS 通信が動作していない場合、TLS 通信のエラーが原因である可能性があり、それは証明書の問題である可能性があります。

証明書に関する注意事項。

- クラスターの設定 `LdapTlsCaCert` で指定する証明書は、ドメインコントローラーの証明書を発行した認証局 (CA) チェーンの証明書全体を含む PEM 証明書のバンドルである必要があります。
- PEM 証明書のバンドルとは PEM 証明書を連結したファイルのことです。
- PEM 形式の証明書 (通常 Linux で使用) は base64 DER 形式の証明書 (通常 Windows でエクスポート) と同等です。
- ドメインコントローラーの証明書が下位 CA によって発行された場合、証明書バンドルには下位およびルート CA の両方の証明書が含まれている必要があります。

トラブルシューティングの検証手順:

次の検証手順は、コマンドがクラスターヘッドノード内から実行されていて、ドメインコントローラーに *SERVER*:*PORT* でアクセスできることを前提としています。

証明書に関連する問題をトラブルシューティングするには、次の検証手順に従ってください。

検証手順:

1. Active Directory ドメインコントローラーへの接続を確認します。

   ドメインコントローラーに接続できることを確認します。この手順が成功したら、ドメインコントローラーへの SSL 接続は成功し、証明書が検証されます。問題は証明書とは関係ありません。

   この手順に失敗した場合は、次の検証に進みます。

   ```
   $ openssl s_client -connect SERVER:PORT -CAfile PATH_TO_CA_BUNDLE_CERTIFICATE
   ```

2. 証明書の検証を確認します。

   ローカル CA 証明書バンドルがドメインコントローラから提供された証明書を検証できることを確認します。この手順が成功した場合、問題は証明書に関するものではなく、他のネットワークの問題に関係しています。

   この手順に失敗した場合は、次の検証に進みます。

   ```
   $ openssl verify -verbose -
   CAfile PATH_TO_CA_BUNDLE_CERTIFICATE PATH_TO_A_SERVER_CERTIFICATE
   ```

3. Active Directory ドメインコントローラーから提供された証明書を確認します。

   ドメインコントローラーから提供された証明書の内容が期待どおりのものであることを確認します。この手順が成功した場合、コントローラーの検証に使用した CA 証明書に問題がある可能性があります。次のトラブルシューティングの手順に進んでください。

   この手順に失敗した場合は、ドメインコントローラー用に発行された証明書を修正し、トラブルシューティングの手順を再実行する必要があります。

```
$ openssl s_client -connect SERVER:PORT -showcerts
```

4. 証明書の内容を確認します。

   ドメインコントローラーから提供された証明書の内容が期待どおりのものであることを確認します。この手順が成功した場合、コントローラーの検証に使用した CA 証明書に問題がある可能性があります。次のトラブルシューティングの手順に進んでください。

   この手順に失敗した場合は、ドメインコントローラー用に発行された証明書を修正し、トラブルシューティングの手順を再実行する必要があります。

```
$ openssl s_client -connect SERVER:PORT -showcerts
```

5. ローカル CA 証明書バンドルの内容を確認します。

   ドメインコントローラーの証明書の検証に使用されるローカル CA 証明書バンドルの内容が期待どおりのものであることを確認します。この手順が成功した場合、ドメインコントローラーから提供される証明書に問題がある可能性があります。

   この手順に失敗した場合は、ドメインコントローラー用に発行された CA 証明書バンドルを修正し、トラブルシューティングの手順を再実行する必要があります。

```
$ openssl x509 -in PATH_TO_A_CERTIFICATE -text
```

Active Directory との統合が動作していることを確認する方法

次の 2 つのチェックが成功すれば、Active Directory との統合は動作しています。

チェック:

1. ディレクトリに定義されているユーザーを検出できる。

クラスターヘッドノード内から、ec2-user として次のようにします。

```
$ getent passwd $ANY_AD_USER
```

2. ユーザーパスワードを指定してヘッドノードに SSH 接続できる。

```
$ ssh $ANY_AD_USER@$HEAD_NODE_IP
```

チェック 1 が失敗すると、チェック 2 も失敗することが予想されます。

その他のトラブルシューティングの確認。

- ユーザーがディレクトリに存在することを確認します。
- デバッグログを有効にします。
- LDAPS の問題を除外するために、LDAPS から LDAP に移行して暗号化を一時的に無効にすることを検討します。

コンピューティングノードへのログインのトラブルシューティング方法

このセクションは、Active Directory と統合されたクラスターのコンピューティングノードへのログインに関連するものです。

では AWS ParallelCluster、クラスターコンピューティングノードへのパスワードログインは設計上無効になっています。

すべてのユーザーは、独自の SSH キーを使用してコンピューティングノードにログインする必要があります。

クラスター設定で `GenerateSshKeysForUsers` が有効になっている場合、ユーザーは初回認証 (ログインなど) の後にヘッドノードで SSH キーを取得できます。

ユーザーはヘッドノードで初めて認証されるとき、ディレクトリユーザーとして自動的に生成された SSH キーを取得できます。ユーザーのホームディレクトリも作成されます。これは sudo ユーザーが初めてヘッドノードのユーザーに切り替えたときにも生じます。

ユーザーがヘッドノードにログインしたことがない場合、SSH キーは生成されず、ユーザーはコンピューティングノードにログインすることはできません。

マルチユーザー環境での SimCenter StarCCM+ ジョブに関する既知の問題

このセクションは、Siemens の計算流体力学ソフトウェア Simcenter StarCCM+ がマルチユーザー環境で起動したジョブに関連するものです。

埋め込みの IntelMPI を使用するように設定された StarCCM+ v16 ジョブを実行すると、デフォルトで SSH を使用して MPI プロセスがブートストラップされます。

ユーザー名の解決エラーを起こす既知の Slurm のバグにより、ジョブは `error setting up the bootstrap proxies` などのエラーで失敗することがあります。このバグは、 AWS ParallelCluster バージョン 3.1.1 および 3.1.2 にのみ影響します。

これを防ぐには、MPI ブートストラップ方式として Slurm を使用するよう IntelMPI を強制します。「IntelMPI 公式ドキュメント」で説明されているように、StarCCM+ を起動するジョブスクリプトに環境変数 `I_MPI_HYDRA_BOOTSTRAP=slurm` をエクスポートします。

ユーザー名解決に関する既知の問題

このセクションは、ジョブ内でのユーザー名の取得に関連するものです。

既知の Slurm のバグにより、`srun` なしでジョブを実行する場合、ジョブプロセス内で取得されるユーザー名は、nobody になることがあります。このバグは、 AWS ParallelCluster バージョン 3.1.1 および 3.1.2 にのみ影響します。

例えば、ディレクトリユーザーとして `sbatch --wrap 'srun id'` コマンドを実行すると、正しいユーザー名が返されます。ただし、`sbatch --wrap 'id'` をディレクトリユーザーとして実行すると、nobody がユーザー名として返されることがあります。

次の回避策を使用できます。

1. 可能であれば、`'sbatch'` の代わりに `'srun'` を使用してジョブを起動します。
2. クラスター設定で AdditionalSssdConfigs を次のように設定して、SSSD 列挙を有効にします。

```
AdditionalSssdConfigs:
  enumerate: true
```

ホームディレクトリ作成の問題の解決方法

このセクションは、ホームディレクトリ作成の問題に関連するものです。

次の例に示されるようなエラーが表示される場合は、ヘッドノードに初めてログインしたときにホームディレクトリが作成されていません。または、ヘッドノードで初めて sudoer から Active Directory ユーザーに切り替えたときに、ホームディレクトリが作成されませんでした。

```
$ ssh AD_USER@$HEAD_NODE_IP
/opt/parallelcluster/scripts/generate_ssh_key.sh failed: exit code 1

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
Could not chdir to home directory /home/PclusterUser85: No such file or directory
```

ホームディレクトリの作成の失敗は、クラスターヘッドノードにインストールされている oddjob および oddjob-mkhomedir パッケージにより発生することがあります。

ホームディレクトリと SSH キーがない場合、ユーザーはジョブや SSH をクラスターノードに送信できません。

システムに oddjob パッケージが必要な場合、oddjobd サービスが実行中であることを確認し、PAM 設定ファイルを更新してホームディレクトリが作成されていることを確認します。これを行うには、次の例に示されるようにヘッドノードでコマンドを実行します。

```
sudo systemctl start oddjobd
sudo authconfig --enablemkhomedir --updateall
```

システムに oddjob パッケージが必要でない場合、アンインストールし、PAM 設定ファイルを更新してホームディレクトリが作成されていることを確認します。これを行うには、次の例に示されるようにヘッドノードでコマンドを実行します。

```
sudo yum remove -y oddjob oddjob-mkhomedir
sudo authconfig --enablemkhomedir --updateall
```

カスタム AMI の問題のトラブルシューティング

このセクションでは、カスタム AMI の問題のトラブルシューティングに役立つヒントを提供します。

カスタム AMI を使用すると、次の警告が表示されます。

```
"validationMessages": [
  {
    "level": "WARNING",
    "type": "CustomAmiTagValidator",
    "message": "The custom AMI may not have been created by pcluster. You can ignore
 this warning if the AMI is shared or copied from another pcluster AMI. If the
 AMI is indeed not created by pcluster, cluster creation will fail. If the cluster
 creation fails, please go to https://docs.aws.amazon.com/parallelcluster/latest/ug/
troubleshooting.html#troubleshooting-stack-creation-failures for troubleshooting."
  },
  {
   "level": "WARNING",
   "type": "AmiOsCompatibleValidator",
   "message": "Could not check node AMI ami-0000012345 OS and cluster OS alinux2
 compatibility, please make sure they are compatible before cluster creation and update
 operations."
  }
]
```

正しい AMI が使用されていることを確認できる場合、これらの警告は無視できます。

今後これらの警告を表示しない場合は、カスタム AMI に次のタグを付けます。 *my-os*は alinux2、、、alinux2023ubuntu2404、ubuntu2204ubuntu2004、、または rhel9 のいずれかでrhel8、*#3.13.2*」は使用中pclusterのバージョンです。

```
$ aws ec2 create-tags \
  --resources ami-yourcustomAmi \
  --tags Key="parallelcluster:version",Value="3.13.2"
 Key="parallelcluster:os",Value="my-os"
```

cfn-hup が実行していない場合のクラスター更新タイムアウトのトラブルシューティング

cfn-hup ヘルパーは、リソースメタデータの変更を検出し、変更が検出された場合に、ユーザーが指定した操作を実行するデーモンです。これは、UpdateStack API アクションを介して、実行中の Amazon EC2 インスタンスで構成を更新する方法です。

現在、cfn-hup デーモンは supervisord によって起動されます。しかし、起動の後、cfn-hup プロセスは supervisord のコントロールからデタッチされます。cfn-hup デーモンが外部攻撃者

により強制終了される場合、自動的に再開されることはありません。cfn-hup が実行されていない場合、クラスターの更新中に CloudFormation スタックは期待どおりに更新プロセスを開始しますが、更新手順はヘッドノードでアクティブ化されず、最終的にスタックはタイムアウトになります。クラスターログ /var/log/chef-client から、更新レシピが呼び出されていないことを確認できます。

失敗した場合、**cfn-hup** を確認して再起動します

1. ヘッドノードで、cfn-hup が実行されているかどうかを確認します。

```
$ ps aux | grep cfn-hup
```

2. ヘッドノードで cfn-hup ログ /var/log/cfn-hup.log と /var/log/supervisord.log を確認してください。
3. cfn-hup が実行されていない場合、次を実行して再起動してみます。

```
$ sudo /opt/parallelcluster/pyenv/versions/cookbook_virtualenv/bin/supervisorctl
 start cfn-hup
```

ネットワークのトラブルシューティング

このセクションでは、ネットワークの問題、特に単一のパブリックサブネットの問題でクラスターを処理する場合のトラブルシューティングのヒントを提供します。

単一のパブリックサブネットのクラスターに関する問題

1 つのコンピューティングノードから cloud-init-output.log を確認します。ノードが Slurm の初期化においてスタックしていることを示す次のようなものが見つかった場合は、DynamoDB VPC エンドポイントが見つからないことが原因である可能性が高いです。DynamoDB エンドポイントを追加してください。詳細については、「AWS ParallelCluster インターネットアクセスのない単一のサブネット内」を参照してください。

```
ruby_block[retrieve compute node info] action run[2022-03-11T17:47:11+00:00] INFO:
 Processing ruby_block[retrieve compute node info] action run (aws-parallelcluster-
slurm::init line 31)
```

クラスターの更新が onNodeUpdated カスタムアクションで失敗した

HeadNode/CustomActions/OnNodeUpdated スクリプトが失敗するとき、更新は失敗し、ロールバック時間にスクリプトは実行されません。ロールバックが完了した後に必要なクリーンアップを手動で実行するのはユーザーの責任です。例えば、OnNodeUpdated スクリプトが設定ファイルのフィールドのステータスを変更 (true から false など) して失敗した場合、そのフィールド値を手動で更新前の状態 (false から true など) に復元する必要があります。詳細については、「カスタムブートストラップアクション」を参照してください。

カスタム Slurm 設定でエラーが表示されている

AWS ParallelCluster バージョン 3.6.0 以降、カスタムSlurm設定に含めることで、単一の prologまたはepilogスクリプトをターゲットにすることはできなくなりました。 AWS ParallelCluster バージョン 3.6.0 以降のバージョンでは、カスタムスクリプトprologとepilogスクリプトをそれぞれのPrologフォルダと Epilogフォルダに配置する必要があります。これらのフォルダは、デフォルトで次を指すように設定されています。

- Prolog は、/opt/slurm/etc/scripts/prolog.d/ を指します。
- Epilog は、/opt/slurm/etc/scripts/epilog.d/ を指します。

90_plcuster_health_check_manager Prolog スクリプトと 90_pcluster_noop Epilog スクリプトは保持しておくことをお勧めします。

Slurm は、スクリプトをアルファベットの降順で実行します。Prolog および Epilog フォルダの両方に、少なくとも 1 つのファイルが含まれている必要があります。詳細については、「Slurmprolog と epilog」および「Slurm 設定のカスタマイズ」を参照してください。

クラスターのアラーム

クラスターのヘルスモニタリングは、最適なパフォーマンスを確保するために不可欠です。 AWS ParallelCluster では、クラスターのヘッドノードで複数の CloudWatch ベースのアラームをモニタリングできます。

このセクションでは、命名規則、アラームをトリガーする特定の条件、推奨されるトラブルシューティング手順など、ヘッドノードにおけるクラスターのアラームについてタイプ別に詳しく説明します。

クラスターのアラームの命名規則は `CLUSTER_NAME-COMPONENT-METRIC` です (例: `mycluster-HeadNode-Cpu`)。

- `CLUSTER_NAME-HeadNode`: ヘッドノードの全体的なステータスを示します。以下のアラームのうち少なくとも 1 つが該当する場合は、赤色になります。
- `CLUSTER_NAME-HeadNode-Health`: Amazon EC2 ヘルスチェックエラーが少なくとも 1 つあると、赤色になります。アラームが発生した場合は、「ステータスチェックに失敗したインスタンスをトラブルシューティングする」を参照することをお勧めします。
- `CLUSTER_NAME-HeadNode-Cpu`: CPU 使用率が 90% を超えると、赤色になります。アラームが発生した場合は、`ps -aux --sort=-%cpu | head -n 10` を使用して CPU を最も多く消費しているプロセスを確認します。
- `CLUSTER_NAME-HeadNode-Mem`: メモリ使用率が 90% を超えると、赤色になります。アラームが発生した場合は、`ps -aux --sort=-%mem | head -n 10` を使用してメモリを最も多く消費しているプロセスを確認します。
- `CLUSTER_NAME-HeadNode-Disk`: 占有ディスク容量がパス / で 90% を超えると、赤色になります。アラームが発生した場合は、`du -h --max-depth=2 / 2> /dev/null | sort -hr` を使用してスペースの大部分を消費しているフォルダを確認します。

AWS ParallelCluster サポートポリシー

AWS ParallelCluster は複数のリリースを同時にサポートします。すべての AWS ParallelCluster リリースには、サポート終了日 (EOSL) がスケジュールされています。EOSL の日付を過ぎると、そのリリースに対するサポートやメンテナンスは提供されません。

AWS ParallelCluster は`major.minor.`*`patch`*バージョンスキームを使用します。最新のメジャーバージョンリリースの新しいマイナーバージョンリリースには、新機能、パフォーマンスの向上、セキュリティアップデート、バグ修正が含まれます。マイナーバージョンには、メジャーバージョン内での下位互換性があります。重大な問題については、 AWS はパッチリリースを通じて修正を提供しますが、EOSL に達していないリリースの最新のマイナーバージョンに限られます。新しいバージョンリリースのアップデートを使用する場合は、新しいマイナーバージョンまたはパッチバージョンにアップグレードする必要があります。

AWS ParallelCluster バージョン	サポート終了日(EOSL)
3.0。*x*	3/31/2023
3.1。*x*	8/31/2023
3.2。*x*	1/31/2024
3.3。*x*	5/31/2024
3.4。*x*	6/28/2024
3.5。*x*	8/31/2024
3.6。*x*	11/30/2024
3.7。*x*	2/28/2025
3.8。*x*	6/30/2025
3.9。*x*	09/05/2025
3.10。*x*	12/27/2025
3.11。*x*	03/25/2026

AWS ParallelCluster バージョン	サポート終了日(EOSL)
3.12。*x*	06/30/2026
3.13。*x*	09/30/2026

のセキュリティ AWS ParallelCluster

でのクラウドセキュリティが最優先事項 AWS です。お客様は AWS 、最もセキュリティの影響を受けやすい組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャを活用できます。

セキュリティは、 AWS お客様とお客様の間の責任共有です。責任共有モデル、、は、これをクラウドのセキュリティ、およびクラウド内のセキュリティとして説明しています。

- クラウドのセキュリティ – AWS クラウドで AWS サービスを実行するインフラストラクチャを保護する AWS 責任があります。 AWS また、 では、安全に使用できるサービスも提供しています。サードパーティーの監査者は、AWS コンプライアンスプログラムコンプライアンスプログラムの一環として、当社のセキュリティの有効性を定期的にテストおよび検証。が適用されるコンプライアンスプログラムの詳細については AWS ParallelCluster、「コンプライアンスプログラムAWS による対象範囲内のサービスコンプライアンスプログラム」を参照してください。
- クラウドのセキュリティ – お客様の責任は、使用する特定の AWS サービスによって決まります。また、お客様は、データの機密性、企業の要件、および適用される法律や規制など、その他のいくつかの関連要素についても責任を負います。

このドキュメントでは、 を使用する際の責任共有モデルの適用方法について説明します AWS ParallelCluster。以下のトピックでは、セキュリティおよびコンプライアンスの目的を達成する AWS ParallelCluster ように を設定する方法について説明します。また、リソースのモニタリングと保護に役立つ AWS ParallelCluster 方法で を使用する方法についても説明します AWS 。

トピック

- が使用する サービスのセキュリティ情報 AWS ParallelCluster
- でのデータ保護 AWS ParallelCluster
- の Identity and Access Management AWS ParallelCluster
- のコンプライアンス検証 AWS ParallelCluster
- TLS の最小バージョン 1.2 の指定

が使用する サービスのセキュリティ情報 AWS ParallelCluster

- Amazon EC2 でのセキュリティ
- Amazon API Gateway でのセキュリティ

- のセキュリティ AWS Batch
- AWS CloudFormationでのセキュリティ
- Amazon CloudWatch でのセキュリティ
- AWS CodeBuildでのセキュリティ
- Amazon DynamoDB でのセキュリティ
- Amazon ECR でのセキュリティ
- Amazon ECS でのセキュリティ
- Amazon EFS でのセキュリティ
- FSx for Lustre でのセキュリティー
- AWS Identity and Access Management (IAM) のセキュリティ
- EC2 Image Builder でのセキュリティ
- のセキュリティ AWS Lambda
- Amazon Route 53 でのセキュリティ
- Amazon SNS でのセキュリティ
- Amazon SQS のセキュリティ（AWS ParallelCluster バージョン 2.x の場合）
- Amazon S3 でのセキュリティ
- Amazon VPC でのセキュリティ

でのデータ保護 AWS ParallelCluster

責任 AWS 共有モデル、 でのデータ保護に適用されます AWS ParallelCluster。このモデルで説明されているように、 AWS はすべての 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 AWS クラウド。ユーザーは、このインフラストラクチャでホストされるコンテンツに対する管理を維持する責任があります。また、使用する「 AWS のサービス 」のセキュリティ設定と管理タスクもユーザーの責任となります。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に関する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に投稿された AWS 責任共有モデルおよび GDPR のブログ記事を参照してください。

データ保護の目的で、認証情報を保護し AWS アカウント 、 AWS IAM Identity Center または AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定することをお勧めします。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権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- 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。CloudTrail 証跡を使用して AWS アクティビティをキャプチャする方法については、「 AWS CloudTrail ユーザーガイド」のCloudTrail 証跡の使用」を参照してください。
- AWS 暗号化ソリューションと、 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 AWS のサービス。
- Amazon Macie などの高度な管理されたセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密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イスまたは API AWS を介して にアクセスするときに FIPS 140-3 検証済み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) 140-3」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報を、タグ、または [名前] フィールドなどの自由形式のテキストフィールドに含めないことを強くお勧めします。これは、コンソール AWS ParallelCluster 、API、または SDK を使用して 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 タグ、または名前に使用される自由記述のテキストフィールドに入力したデータは、請求または診断ログに使用される場合があります。外部サーバーに URL を提供する場合、そのサーバーへのリクエストを検証できるように、認証情報を URL に含めないことを強くお勧めします。

データの暗号化

セキュリティで保護されたサービスの重要な特徴として、情報はアクティブに使用されていないときに暗号化されます。

保管中の暗号化

AWS ParallelCluster 自体は、ユーザーに代わって AWS サービスとやり取りするために必要な認証情報以外の顧客データを保存しません。

クラスター内のノード上のデータについては、保管時に暗号化できます。

Amazon EBS ボリュームの場合、`EbsSettings` セクションの `EbsSettings`/`Encrypted` および `EbsSettings`/`KmsKeyId` の設定を使用して暗号化を設定します。詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EBS の暗号化」を参照してください。

Amazon EFS ボリュームの場合、`EfsSettings` セクションの `EfsSettings`/`Encrypted` および `EfsSettings`/`KmsKeyId` の設定を使用して暗号化を設定します。詳細については、「Amazon Elastic File System ユーザーガイド」の「How encryption at rest works」を参照してください。

FSx for Lustre ファイルシステムでは、Amazon FSx ファイルシステムの作成時に、保管中のデータの暗号化が自動的に有効になります。詳細については、「FSx for Lustre ユーザーガイド」の「保管中のデータの暗号化」を参照してください。

NVMe ボリュームを持つインスタンスタイプでは、NVMe インスタンスストアボリューム内のデータは、インスタンスのハードウェアモジュールに実装されている XTS-AES-256 暗号を使用して暗号化されます。暗号化キーはハードウェアモジュールで作成され、NVMe インスタンスストレージデバイスごとに固有です。すべての暗号化キーはインスタンスが停止または終了して復元できないときに破棄されます。この暗号化を無効にしたり、独自の暗号キーを指定したりすることはできません。詳細については、「Amazon EC2 ユーザーガイド」の「保管中の暗号化」を参照してください。

AWS ParallelCluster を使用して、カスタマーデータをローカルコンピュータに送信して保存するための AWS サービスを呼び出す場合は、そのサービスのユーザーガイドの「セキュリティとコンプライアンス」の章を参照して、データの保存、保護、暗号化の方法を確認してください。

転送中の暗号化

デフォルトでは、 AWS ParallelCluster および AWS サービスエンドポイントを実行しているクライアントコンピュータから送信されるすべてのデータは、HTTPS/TLS 接続を介してすべてを送信することで暗号化されます。クラスター内のノード間のトラフィックは、選択したインスタンスタイプに応じて自動的に暗号化することができます。詳細については、「Amazon EC2 ユーザーガイド」の「転送中の暗号化」を参照してください。

関連情報

- Amazon EC2 でのデータ保護
- EC2 Image Builder でのデータ保護
- でのデータ保護 AWS CloudFormation
- Amazon EFS でのデータ保護
- Amazon S3 でのデータ保護
- Amazon FSx for Lustre のデータ保護

の Identity and Access Management AWS ParallelCluster

AWS ParallelCluster は、 ロールを使用して AWS リソースとそのサービスにアクセスします。 AWS ParallelCluster がアクセス許可を付与するために使用するインスタンスポリシーとユーザーポリシーについては、「」を参照してください[AWS Identity and Access Management での アクセス許可 AWS ParallelCluster](#)。

唯一の大きな違いは、標準の ユーザーと長期の認証情報を使用する場合の認証方法です。ユーザーは AWS サービスのコンソールにアクセスするためにパスワードが必要ですが、同じユーザーは を使用して同じオペレーションを実行するためにアクセスキーペアが必要です AWS ParallelCluster。他のすべての短期の認証情報については、使用方法がコンソールと同じです。

で使用される認証情報 AWS ParallelCluster はプレーンテキストファイルに保存され、暗号化されません。

- `$HOME/.aws/credentials` ファイルは、 AWS リソースへのアクセスに必要な長期的な認証情報を保存します。これには、アクセスキー ID とシークレットアクセスキーが含まれます。
- 引き受けるロールや AWS IAM Identity Center サービスで使用するような短期の認証情報は、それぞれ `$HOME/.aws/cli/cache` フォルダと `$HOME/.aws/sso/cache` フォルダに保存されます。

リスクの軽減

- `$HOME/.aws` フォルダとその子フォルダおよびファイルに対して、許可されたユーザーにのみアクセスを制限するようにファイルシステムのアクセス許可を設定することを強くお勧めします。
- 一時的な認証情報を持つロールをできるだけ使用し、認証情報が漏洩した場合の損害の可能性を減らします。長期の認証情報は、短期のロールの認証情報を要求および更新する場合にのみ使用します。

のコンプライアンス検証 AWS ParallelCluster

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 AWS サービスのセキュリティと AWS コンプライアンスを評価します。 AWS ParallelCluster を使用してサービスにアクセスしても、そのサービスのコンプライアンスは変更されません。

特定のコンプライアンスプログラムの対象となる AWS サービスのリストについては、「コンプライアンスAWS プログラムの対象となるサービスコンプライアンス」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、「AWS Artifactでレポートをダウンロードする」を参照してください。

を使用する際のお客様のコンプライアンス責任 AWS ParallelCluster は、お客様のデータの機密性、貴社のコンプライアンス目的、適用される法律および規制によって決まります。 は、コンプライアンスに役立つ以下のリソース 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デプロイする手順について説明します AWS。
- 「Amazon Web Services での HIPAA セキュリティとコンプライアンスの設計 AWS 」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 デベロッパーガイドのルールによるリソースの評価 – この AWS Config サービスは、リソース設定が内部プラクティス、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。

TLS の最小バージョン 1.2 の指定

AWS サービスと通信する際のセキュリティを強化するには、TLS 1.2 以降を使用するように を設定 AWS ParallelCluster する必要があります。を使用すると AWS ParallelCluster、Python を使用して TLS バージョンを設定します。

が TLS 1.2 より前の TLS バージョン AWS ParallelCluster を使用しないようにするには、OpenSSL を再コンパイルしてこの最小値を適用し、新しく構築された OpenSSL を使用するように Python OpenSSL を再コンパイルする必要があります。

現在サポートされているプロトコルの確認

まず、OpenSSL を使用して、テストサーバーと Python SDK に使用する自己署名証明書を作成します。

```
$ openssl req -subj '/CN=localhost' -x509 -newkey rsa:4096 -nodes -keyout key.pem -out
 cert.pem -days 365
```

次に、OpenSSL を使用してテストサーバーをスピンアップします。

```
$ openssl s_server -key key.pem -cert cert.pem -www
```

新しいターミナルウィンドウで仮想環境を作成し、Python SDK をインストールします。

```
$ python3 -m venv test-env
source test-env/bin/activate
pip install botocore
```

SDK の基になる HTTP ライブラリを使用する、`check.py` という名前の新しい Python スクリプトを作成します。

```
$ import urllib3
URL = 'https://localhost:4433/'

http = urllib3.PoolManager(
ca_certs='cert.pem',
cert_reqs='CERT_REQUIRED',
)
r = http.request('GET', URL)
print(r.data.decode('utf-8'))
```

新しいスクリプトを実行します。

```
$ python check.py
```

確立された接続に関する詳細が表示されます。出力で「Protocol:」を検索します。出力が「TLSv1.2」以降の場合、SDK のデフォルト設定は TLS v1.2 以降です。それ以前のバージョンの場合は、OpenSSL を再コンパイルして Python を再コンパイルする必要があります。

ただし、Python のインストールがデフォルトで TLS v1.2 以降に設定されている場合でも、サーバーが TLS v1.2 以降をサポートしていないと、Python は TLS v1.2 より前のバージョンに再ネゴシエートする可能性があります。Python が以前のバージョンに自動的に再ネゴシエートしないことを確認するには、次のようにしてテストサーバーを再起動します。

```
$ openssl s_server -key key.pem -cert cert.pem -no_tls1_3 -no_tls1_2 -www
```

以前のバージョンの OpenSSL を使用している場合は、`-no_tls_3` フラグが使用できない可能性があります。この場合は、使用している OpenSSL のバージョンが TLS v1.3 をサポートしていないため、フラグを削除します。次に、Python スクリプトを実行します。

```
$ python check.py
```

Python のインストールが TLS 1.2 より前のバージョンに対して正しく再ネゴシエートしない場合は、SSL エラーが表示されます。

```
$ urllib3.exceptions.MaxRetryError: HTTPSConnectionPool(host='localhost',
 port=4433): Max retries exceeded with url: / (Caused by SSLError(SSLError(1, '[SSL:
 UNSUPPORTED_PROTOCOL] unsupported protocol (_ssl.c:1108)')))
```

接続を確立できる場合は、OpenSSL と Python を再コンパイルして、TLS v1.2 より前のプロトコルのネゴシエーションを無効にする必要があります。

OpenSSL と Python のコンパイル

AWS ParallelCluster が TLS 1.2 より前のものに対してネゴシエートされないようにするには、OpenSSL と Python を再コンパイルする必要があります。これを行うには、次のコンテンツをコピーしてスクリプトを作成し、実行します。

```
#!/usr/bin/env bash
set -e

OPENSSL_VERSION="1.1.1d"
OPENSSL_PREFIX="/opt/openssl-with-min-tls1_2"
PYTHON_VERSION="3.8.1"
PYTHON_PREFIX="/opt/python-with-min-tls1_2"


curl -O "https://www.openssl.org/source/openssl-$OPENSSL_VERSION.tar.gz"
```

```
tar -xzf "openssl-$OPENSSL_VERSION.tar.gz"
cd openssl-$OPENSSL_VERSION
./config --prefix=$OPENSSL_PREFIX no-ssl3 no-tls1 no-tls1_1 no-shared
make > /dev/null
sudo make install_sw > /dev/null


cd /tmp
curl -O "https://www.python.org/ftp/python/$PYTHON_VERSION/Python-$PYTHON_VERSION.tgz"
tar -xzf "Python-$PYTHON_VERSION.tgz"
cd Python-$PYTHON_VERSION
./configure --prefix=$PYTHON_PREFIX --with-openssl=$OPENSSL_PREFIX --disable-shared > /
dev/null
make > /dev/null
sudo make install > /dev/null
```

これにより、静的にリンクされた OpenSSL を持つ Python のバージョンがコンパイルされます。このバージョンは、TLS 1.2 より前のバージョンは自動的にネゴシエートしません。また、`/opt/openssl-with-min-tls1_2` ディレクトリに OpenSSL がインストールされ、`/opt/python-with-min-tls1_2` ディレクトリに Python がインストールされます。このスクリプトを実行した後、新しいバージョンの Python のインストールを確認します。

```
$ /opt/python-with-min-tls1_2/bin/python3 --version
```

これにより、以下が出力されます。

```
Python 3.8.1
```

この新しいバージョンの Python が TLS 1.2 より前のバージョンをネゴシエートしないことを確認するには、新しくインストールされた Python バージョン (つまり現在サポートされているプロトコルの確認) を使用する手順 `/opt/python-with-min-tls1_2/bin/python3` を再実行します。

AWS リージョン でサポートされる AWS ParallelCluster

AWS ParallelCluster バージョン 3 は、以下から入手できます AWS リージョン。

リージョン名	リージョン	リージョンをサポートする最初の AWS ParallelCluster バージョン
米国東部 (オハイオ)	us-east-2	3.0.0
米国東部 (バージニア北部)	us-east-1	3.0.0
米国西部 (北カリフォルニア)	us-west-1	3.0.0
米国西部 (オレゴン)	us-west-2	3.0.0
アフリカ (ケープタウン)	af-south-1	3.0.0
アジアパシフィック (香港)	ap-east-1	3.0.0
アジアパシフィック (ムンバイ)	ap-south-1	3.0.0
アジアパシフィック (ソウル)	ap-northeast-2	3.0.0
アジアパシフィック (シンガポール)	ap-southeast-1	3.0.0
アジアパシフィック (シドニー)	ap-southeast-2	3.0.0
アジアパシフィック (ジャカルタ)	ap-southeast-3	3.10.0
アジアパシフィック (マレーシア)	ap-southeast-5	3.13.0
アジアパシフィック (タイ)	ap-southeast-7	3.13.0
アジアパシフィック (東京)	ap-northeast-1	3.0.0
カナダ (中部)	ca-central-1	3.0.0
中国 (北京)	cn-north-1	3.0.0
中国 (寧夏)	cn-northwest-1	3.0.0

リージョン名	リージョン	リージョンをサポートする最初の AWS ParallelCluster バージョン
欧州 (フランクフルト)	eu-central-1	3.0.0
欧州 (アイルランド)	eu-west-1	3.0.0
欧州 (ロンドン)	eu-west-2	3.0.0
欧州 (ミラノ)	eu-south-1	3.0.0
欧州 (パリ)	eu-west-3	3.0.0
欧州 (ストックホルム)	eu-north-1	3.0.0
中東 (バーレーン)	me-south-1	3.0.0
南米 (サンパウロ)	sa-east-1	3.0.0
AWS GovCloud (米国東部)	us-gov-east-1	3.0.0
AWS GovCloud (米国西部)	us-gov-west-1	3.0.0
イスラエル (テルアビブ)	il-central-1	3.8.0

リリースノートとドキュメント履歴

以下の表では、 AWS ParallelCluster ユーザーガイドの主な更新と新機能について説明します。また、お客様からいただいたフィードバックに対応するために、ドキュメントを頻繁に更新しています。

AWS ParallelCluster

変更	説明	日付
AWS ParallelCluster バージョン 3.13.2 をリリース	アップグレードするには、と入力しますsudo pip install --upgrade aws-parallelcluster 。 バグ修正: • EC2 API 経由でアクセスできなくなった期限切れのキャパシティ予約update-compute-fleet をコンピューティングリソースが参照すると、 update-cluster および が失敗することがあるバグを修正しました。 APIs • 親イメージに最新のカーネルバージョンが付属していない場合に発生する Rocky 9 のbuild-image 障害を修正しました。 https://github.com/aws/aws-parallelcluster/issues/6874 を参照してください。	2025 年 6 月 24 日
AWS ParallelCluster バージョン 3.13.1 をリリース	アップグレードするには、と入力しますsudo pip	2025 年 6 月 4 日

変更	説明	日付

	`install --upgrade aws-parallelcluster` 。
	
	変更:
	
	• Slurm をバージョン 24.05.8 にアップグレードします。
	• EFA インストーラを 1.41.0 (1.38.1 から) にアップグレードします。
	  • Efa-driver: efa-2.15.0-1
	  • Efa-config: efa-config-1.18-1
	  • Efa-profile: efa-profile-1.7-1
	  • Libfabric-aws: libfabric-aws-2.1.0-1
	  • Rdma-core: rdma-core-57.0-1
	  • Open MPI: openmpi40-aws-4.1.7-2 および openmpi50-aws-5.0.6
	• Amazon Linux 以外の AMI では、amazon-efs-utils をバージョン 2.3.1 (v2.1.0 から) にアップグレードします。
	• us-isob-east-1 および us-iso-east-1 で DCV をサポートします。
	• us-isob-east-1 および us-iso-east-1 で FSX for Lustre	

変更	説明	日付
	と Ontap をサポートします。 • ParallelCluster イメージビルド全体でカーネルの一貫性を確保するには、開始時にピン留めし、終了時にピン留めを解除します。 バグ修正: • 分離された環境でビルドイメージが失敗する原因となっていた ARM Performance Library のインストールのバグを修正しました。 • スクリプト「update_directory_service_password.sh」が AD パスワードを更新できないバグを修正しました。	

変更	説明	日付
AWS ParallelCluster バージョン 3.13.0 をリリース	アップグレードするには、と入力しますsudo pip install --upgrade aws-parallelcluster 。 非推奨： • Ubuntu 20.04 は 2025 年 5 月に End-Of-Standard-Support になるため、これは Ubuntu 20.04 をサポートする最後の ParallelCluster リリースです。 機能強化: • Ubuntu 24.04 のサポートを追加します。 • ap-southeast-7 リージョンのサポートを追加します。 • セキュリティを向上させるために、公式 ParallelCluster AMIs から未使用のサービスカップと wpa_supplicant を無効にします。 変更: • Slurm をバージョン 24.05.7 にアップグレードします。 • AL2 を除くすべての OSs で NVIDIA ドライバーをバージョン 570.86.15 (550.127.08	2025 年 4 月 1 日

変更	説明	日付

	から) にアップグレードします。
	• AL2 を除くすべての OSs で CUDA Toolkit をバージョン 12.8.0 (12.4.1 から) にアップグレードします。
	• AL2 (3.9.20 から) を除くすべての OSs で Python を 3.12.8 にアップグレードします。 AL2
	• Ubuntu 22.04 では、カーネルのコンパイルに使用したのと同じコンパイラバージョンで Nvidia ドライバーをインストールします。
	• バージョン 2.0-33 `aws-cfn-bootstrap` にアップグレードします。
	• EFA インストーラを 1.38.0 (1.36.0 から) にアップグレードします。
	  • Efa-driver: efa-2.13.0-1
	  • Efa-config: efa-confi g-1.17-1
	  • Efa-profile: efa-profi le-1.7-1
	  • `Libfabric-aws` : `libfabric-aws-1.22.0-1`
	  • Rdma-core: rdma-core -54.0-1
	  • Open MPI: `openmpi40-aws-4.1.7-1` および	

変更	説明	日付

	`openmpi50-aws-5.0.5`
	• amazon-efs-utils をバージョン 2.1.0 にアップグレードします。
	• サードパーティーのクックブックを削除する: apt-7.5.22 および pyenv-4.2.3。
	• サードパーティのクックブックの依存関係をアップグレード:
	  • line-4.5.21 (line-4.5.13 から)
	  • nfs-5.1.5 (nfs-5.1.2 から)
	  • openssh-2.11.14 (openssh-2.11.12 から)
	  • yum-7.4.20 (yum-7.4.13 から)
	  • yum-epel-5.0.8 (yum-epel-5.0.2 から)
	• Pmix を 5.0.6 (5.0.3 から) にアップグレードします。
	• ARM PL をバージョン 24.10 (23.10 から) にアップグレードします。
	• Lambda レイヤーとインストーラで Python をバージョン 3.12.8 (3.9.17 から) にアップグレードします。
	• Lambda レイヤーとインストーラで NodeJS をバージョン 20.18.3 (18.20.3 か	

変更	説明	日付
	ら) にアップグレードします。 • ログインノードの DSA キーの生成を DSA として削除します。DSA は OpenSSH 9.7 以降ではサポートされなくなりました。 • コンピューティングノードの起動時に Slurm でインスタンス ID とインスタンスタイプ情報を設定します。 • NVIDIA インストーラで廃止されたno-cc-version-check」オプションを使用せずに NVIDIA ドライバーをインストールします。 • 検証機能を追加して、最大 10 個のログインノードプールを適用します。 • デフォルトのルートボリュームサイズを 45 GB に更新します。 バグ修正: • コンピューティングノードのブートストラップに cfn-init を使用すると、ノードのスケールアップ時間を短縮できます。 • プロキシの使用時にコンピューティングノードのブートストラップが失敗する問題を修正しました。	

変更	説明	日付
	• Ubuntu 22.04 では、インストールの失敗を防ぐためにカーネルのコンパイルに使用したのと同じコンパイラバージョンで Nvidia ドライバーをインストールします。- 更新中にヘッドノードでのみ aws-parallelcluster-node パッケージを上書きする実行を修正しました。 • マルチユーザー環境 (Active Directory と統合) で Pyxis/Enroot を介して実行されるコンテナ化されたジョブが失敗する問題を修正しました。 • ディレクトリサービスが使用されているときに、Rocky 9.5+ でノードブートストラップ障害を引き起こす認証選択の使用を修正しました。	

変更	説明	日付
AWS ParallelCluster バージョン 3.12.0 をリリース	アップグレードするには、と入力しますsudo pip install --upgrade aws-parallelcluster 。 機能強化: • 新しいビルドイメージ設定セクションを追加してBuild/Installation 、NVIDIA ソフトウェアと Lustre クライアントのインストールをオン/オフにします。デフォルトでは、NVIDIA ソフトウェアは公式の ParallelCluster AMIs に含まれていますが、 によってインストールされませんbuild-image 。デフォルトでは、Lustre クライアントがインストールされます。 • CLI コマンド export-cluster-logs および export-image-logsは、デフォルトでデフォルトの ParallelCluster バケットまたは 設定で指定されている場合は CustomS3Bucket にログをエクスポートできるようになりました。 • ARM インスタンスで Amazon DCV サポートを	2024 年 12 月 19 日

変更	説明	日付

	Ubuntu2204 に拡張します。
	
	変更:
	
	• NVIDIA ドライバーをバージョン 550.127.08 (550.90.07 から) にアップグレードします。これは NVIDIA の既知の問題に対処します。詳細については、NVIDIA データセンタードキュメントの「既知の問題」を参照してください。
	• Amazon DCV をバージョン にアップグレードします`2024.0-18131` 。
	  • サーバー: `2024.0-18131-1`
	  • xdcv: `2024.0.631-1`
	  • gl: `2024.0.1078-1`
	  • web_viewer: `2024.0-18131-1`
	• EFA インストーラーを `1.36.0` にアップグレードします。
	  • Efa-driver: `efa-2.13.0-1`
	  • Efa-config: `efa-config-1.17-1`
	  • Efa-profile: `efa-profile-1.7-1`	

変更	説明	日付
	• Libfabric-aws: `libfabric-aws-1.22.0-1` • Rdma-core: `rdma-core-54.0-1` • Open MPI: `openmpi40-aws-4.1.7-1` および `openmpi50-aws-5.0.5` • 障害発生時に slurmctld を自動的に再起動します。 • mysql-community-client をバージョン 8.0.39 にアップグレードします。 • 有効期限が切れた Python 3.7 および 3.8 のサポートを削除します。 バグ修正: • クラスターの更新中に一連のカスタムアクションスクリプトの変更が検出されなかった問題を修正しました。 • ログインノードをデプロイするために必要な Elastic Load Balancing と Auto Scaling のサービスにリンクされたロールを作成するための AWS ParallelCluster API のアクセス許可の欠落を追加します。	

変更	説明	日付
	• ボリュームの管理時にリージョンを取得する方法の問題を修正し、ローカルゾーンを正しく処理できるようにします。 • 更新`AccessPointIds` 中に で EFS ファイルシステムを追加すると失敗する問題を修正しました。 • PCAPI を使用する場合、型ではないパラメータ `String` (など) を更新するとクラスターの更新が失敗する問題を修正しました`MaxCount`。 • 外部 OpenZFS をマウントする場合、ポート 111、2049、20001、20002、20003 のアウトバウンドルールを設定する必要がなくなりました。	

変更	説明	日付
AWS ParallelCluster バージョン 3.11.1 をリリース	機能: • Pyxis はデフォルトで無効になったため、製品ドキュメントに記載しているように手動で有効にする必要があります。 • ParallelCluster Lambda Layer で Python ランタイムをバージョン 3.12 にアップグレードしました。 • setuptools のバージョンピン留めを 70.0.0 より前のバージョンから削除しました。 • libjwt をバージョン 1.17.0 にアップグレードしました。 • 完全な変更ログ バグ修正 • ParallelCluster での Pyxis Slurm プラグインの設定方法において、ジョブの送信エラーを起こす可能性がある問題を修正しました。 • ポリシーのパブリックテンプレートでログインノードに必要なアクセス許可がないと、ログインノードの設定でデプロイが失敗する問題を修正しました。https://	2024 年 10 月 21 日

変更	説明	日付
	github.com/aws/aws-parallelcluster/issues/6483	

変更	説明	日付
AWS ParallelCluster バージョン 3.11.0 をリリース	機能強化 • ログインノードでカスタムアクションのサポートを追加します。 • ログインノードへの DCV 接続を許可します。 • ap-southeast-3 リージョンのサポートを追加します。 • ログインノードの Network Load Balancer にセキュリティグループを追加します。 • ログインノード`AllowedIps` の設定を追加します。 • マウント用のオプションの EFS アクセスポイント`SharedStorage/EfsSettings/AccessPointId` を指定する新しい設定を追加する • 最大 10 個のログインノードプールを許可します。 • 公式の pcluster AMIs に enroot と pyxis をインストールする 変更 • 〔BREAKING] API `DescribeCluster` と CLI コマンドによって返される `loginNode`	2024 年 9 月 26 日

変更	説明	日付

	s フィールドdescribe-cluster が、複数のログインノードのプールをサポートするようにディクショナリから配列に変更されました。この変更により、下位互換性が損なわれ、これらのオペレーションは古いバージョンでデプロイされたクラスターと互換性がありません。
	• Slurm を 23.11.10 (23.11.7 から) にアップグレードします。
	• Pmix を 5.0.3 (5.0.2 から) にアップグレードします。
	• EFA インストーラーを 1.34.0 にアップグレードします。
	  • Efa-driver: efa-2.10.0-1
	  • Efa-config: efa-config-1.17-1
	  • Efa-profile: efa-profile-1.7-1
	  • Libfabric-aws: libfabric-aws-1.22.0-1
	  • Rdma-core: rdma-core-52.0-1
	  • Open MPI: openmpi40-aws-4.1.6-3 および	

変更	説明	日付
	`openmpi50-aws-5.0.3-11` • NVIDIA ドライバーをバージョン 550.90.07 (535.183.01 から) にアップグレードします。 • CUDA Toolkit をバージョン 12.4.1 (12.2.2 から) にアップグレードします。 • Python を 3.9.20 (3.9.19 から) にアップグレードします。 • Intel MPI Library を 2021.13.1.769 (2021.12.1.8 から) にアップグレードします。 バグ修正 • キャパシティブロックの使用時にプレイスメントグループを設定することを提案`EfaPlacementGroupValidator` しないようにバリデータを修正しました。 • セキュリティグループルールの後に FSx for Lustre ファイルシステムが作成されるようにすることで、クラスター作成の失敗が時折修正されます。 • プレイスメントグループが有効になっている場合のク	

変更	説明	日付
	ラスター削除の失敗を修正しました。 • SSH アクセスを制限するときにログインノードが異常とマークされる問題を修正しました。 • 正しい S3 URL を取得`retrieve_supported_regions` できるように修正しました。 • ページ分割を使用する`describe_images` ように修正しました。 • LoginNodes/Networking/SubnetIds `No route tables found` にデフォルトの VPC サブネットを指定する際のバグを修正しました。	
AWS ParallelCluster バージョン 3.10.1 をリリース	バグ修正 • 中国リージョンでのイメージビルドの失敗を修正しました。	2024 年 7 月 8 日

変更	説明	日付
AWS ParallelCluster バージョン 3.10.0 をリリース	機能強化: • 新しい設定セクションを追加して`Scheduling/SlurmSettings/ExternalSlurmdbd` 、クラスターを外部 Slurmdbd に接続します。 • 分離されたネットワークで build-image を実行できるようにします。 • Amazon Linux 2023 のサポートを追加します。 • `price-capacity-optimized` として のサポートを追加します`AllocationStrategy` 。 • キャパシティブロックでプレイスメントグループを使用しないように検証機能を追加します。 変更: • CentOS 7 はサポートされなくなりました。 • Cinc Client を 18.4.12 からバージョン 18.2.7 にアップグレードします。 • munge をバージョン 0.5.16 (0.5.15 から) にアップグレードします。	2024 年 6 月 27 日

変更	説明	日付

	• Pmix を 5.0.2 (4.2.9 から) にアップグレードします。
	• サードパーティのクックブックの依存関係をアップグレード:
	  • apt-7.5.22 (apt-7.5.14 から)
	  • openssh-2.11.12 (openssh-2.11.3 から)
	• サードパーティーのクックブック selinux-6.1.12 を削除します。
	• EFA インストーラーを `1.32.0` にアップグレードします。
	  • Efa-driver: `efa-2.8.0-1`
	  • Efa-config: `efa-config-1.16-1`
	  • Efa-profile: `efa-profile-1.7-1`
	  • Libfabric-aws: `libfabric-aws-1.21.0-1`
	  • Rdma-core: `rdma-core-50.0-1`
	  • Open MPI: `openmpi40-aws-4.1.6-3` および `openmpi50-aws-5.0.2-12`
	• NVIDIA ドライバーをバージョン 535.183.01 (535.154.	

変更	説明	日付
	05 から) にアップグレードします。 • Python を 3.9.19 (3.9.17 から) にアップグレードします。 • Intel MPI Library を 2021.12.1.8 (2021 年 9 月 0 日 ~ 43482) にアップグレードします。 バグ修正: • データリポジトリの関連付け設定を修正して、`AutoExportPolicy` と`AutoImportPolicy` オプションにしました。 • クラスターの削除中に、インスタンスがシャットダウン状態または終了状態になったときにコンピューティングフリートのクリーンアップを完了する問題を修正しました。これは、終了サイクルが長いインスタンスタイプのクラスター削除の失敗を回避するためです。 • クラスター設定の `Monitoring` セクションで、cloudwatch ダッシュボードの有効化とアラームの無効化を許可します。	

変更	説明	日付
	• ParallelCluster カスタムリソースが を使用して検証を抑制できるようにしますPclusterCluster/SuppressValidators 。 • 削除/etc/profile.d/pcluster.sh して、すべてのユーザーログインで実行されずcfn_bootstrap_virtualenv 、PATH 環境変数に追加されないようにします。 • DescribeCluster レスポンスで フィールドfailureReason を に置き換えて ParallelCluster API 仕様を修正failuresしました。 • 欠落していた CloudFormation スタックステータスREVIEW_IN_PROGRESS 、、 を追加してIMPORT_*ParallelCluster API 仕様を修正しましたUPDATE_FAILED 。 • クラスターの更新に、転送中の暗号化を使用した EFS ファイルシステムが含まれない問題を修正しました。 • EFS を共有内部データに使用すると、slurmctld および slurmdbd サービスがヘッドノードの再起動時に再起動	

変更	説明	日付
	できない問題を修正しました。 • Ubuntu システムでは、ParallelCluster の設定と競合する cloud-init ログファイルのデフォルトの logrotate 設定を削除します。 • RHEL 8.10 以降でイメージビルドの失敗を修正しました。	

変更	説明	日付
AWS ParallelCluster バージョン 3.9.3 をリリース	アップグレードするには、「`sudo pip install --upgrade aws-parallelcluster`」と入力します。 機能: • 共有ストレージタイプとしての FSx Lustre のサポートを `us-iso-east-1` で追加しました。 バグ修正: • Slurm のファンアウト問題を避けるため、`cloud_dns` を Slurm 設定の `SlurmctldParameters` から削除しました。 インスタンスの起動時に IP アドレスを設定するため、これは必須ではありません。	2024 年 6 月 19 日
AWS ParallelCluster バージョン 3.9.2 をリリース	機能: • Slurm を 23.11.7 (23.11.4 から) にアップグレードします。 • 詳細については、GitHub の「CHANGELOG 3.9.2」を参照してください。	2024 年 5 月 28 日

変更	説明	日付
AWS ParallelCluster バージョン 3.9.1 をリリース	アップグレードするには、次を入力します: `sudo pip install --upgrade aws-parallelcluster` バグ修正 • update-cluster オペレーションの一部としてファイルシステムをアンマウントする場合、共有ストレージ mountdir の再帰的な削除を削除しました。	2024 年 4 月 11 日

変更	説明	日付
AWS ParallelCluster バージョン 3.9.0 をリリース	アップグレードするには、次を入力します: `sudo pip install --upgrade aws-parallelcluster` 機能強化: • 設定パラメータ `DeploymentSettings/DefaultUserHome` を追加して、デフォルトユーザーのホームディレクトリを `/home` (デフォルト) の代わりに `/local/home` に移動できるようにしました。 • コンピューティングフリートを停止することなく、`MinCount`、`MaxCount`、`source` の各設定パラメータを更新できるようにしました。`Scheduling/SlurmSettings/QueueUpdateStrategy` を TERMINATE 設定することで、これらを更新できるようになりました。 AWS ParallelCluster は、クラスターの更新を通じて実行されたクラスター容量のサイズ変更中に削除されたノードのみを終了します。 • コンピューティングフリートとログインフリートを置き換えることなく、Efs、FsxLustre、FsxOnta	2024 年 3 月 5 日

変更	説明	日付
	p、FsxOpenZfs、FileCache の各タイプの外部共有ストレージを更新できるようにしました。 • RHEL9 のサポートを追加しました。 • `build-image` プロセスで作成した `CustomAmi` として Rocky Linux 9 のサポートを追加しました。現時点では、公式 AWS ParallelCluster の Rocky9 Linux AMI は利用できません。 • `CommunicationParameters` カスタム Slurm 設定拒否リストから を削除します。 • `DeploymentSettings/DisableSudoAccessForDefaultUser` パラメータを追加して、サポートされている OS におけるデフォルトユーザーの sudo アクセスを無効にしました。 • ParallelCluster で作成した FSx for Lustre ファイルシステムへの変更: Lustre サーバーバージョンを `2.15.` に変更 • `['cluster']['nvidia']['kernel_open']` クックブックノード属性	

変更	説明	日付
	を使用して、AMI の構築時にオープンソースとクローズドソースの Nvidia ドライバーのどちらかを選択できるようにしました。 • * clustermgtd 設定オプション `ec2_instance_missing_max_count` を追加して、Amazon EC2 の記述インスタンスと実行インスタンスの結果整合性を実現するまでの再試行回数を設定できるようにしました。 変更 • Slurm を 23.11.4 (23.02.7 から) にアップグレードします。 • NVIDIA ドライバーをバージョン 535.154.05 にアップグレードしました。 • pcluster CLI および aws-parallelcluster-batch-cli で Python 3.11、3.12 のサポートを追加しました。 • `MaximumNetworkCards` 範囲をループする代わりに、Amazon EC2 DescribeInstances レスポンスの `NetworkCardIndex` リストから、ネットワークカードインデックスを使用して	

変更	説明	日付

	ネットワークインターフェイスを構築します。
	• P3、G3、P2、G2 の各インスタンスタイプを使用する場合、GPU アーキテクチャが 3.8.0 リリースの一部として導入されたオープンソース Nvidia ドライバー (OpenRM) と互換性がないため、クラスターの作成が失敗します。
	• サードパーティのクックブック依存関係をアップグレード: nfs-5.1.2 (nfs-5.0.0 から)
	• EFA インストーラーを `1.30.0.` にアップグレード
	  • Efa-driver: `efa-2.6.0-1`
	  • Efa-config: `efa-config-1.15-1`
	  • Efa-profile: `efa-profile-1.6-1`
	  • Libfabric-aws: `libfabric-aws-1.19.0`
	  • Rdma-core: `rdma-core-46.0-1`
	  • Open MPI: `openmpi40-aws-4.1.6-2` および `openmpi50-aws-5.0.0-11`	

変更	説明	日付

	• NICE DCV をバージョン 2023.1-16388. にアップグレード
	  • サーバー: 2023.1.16388-1
	  • xdcv: 2023.1.565-1
	  • gl: 2023.1.1047-1
	  • web_viewer: 2023.1.16388-1
	
	バグ修正
	
	• ログインノードから Active Directory ユーザーとして送信した場合にジョブが失敗する問題を修正しました。この問題は、ヘッドノードでの外部 Active Directory との統合の設定が不完全であるために発生しました。
	• CloudFormation テンプレートの parallelclutser-policies.yaml で定義されている IAM ポリシーをリファクタリングして、IAM 制限を超えるポリシーによる ParallelCluster API デプロイの失敗を防ぎました。
	• ヘッドノードがキーの書き込みに予想以上に時間がかかる場合、ログインノードがブートストラップに失敗する問題を修正しました。	

変更	説明	日付
	変更点の詳細については、GitHub に掲載されている aws-parallelcluster-ui パッケージの `CHANGELOG` ファイルを参照してください。	

変更	説明	日付
AWS ParallelCluster バージョン 3.8.0 をリリース	AWS ParallelCluster バージョン 3.8.0 がリリースされました。 機能強化: • Amazon EC2 Capacity Blocks for ML のサポートを追加しました。 • `build-image` プロセスで作成した `CustomAmi` として Rocky Linux 8 のサポートを追加しました。現時点では、公式 AWS ParallelCluster の Rocky8 Linux AMI は利用できません。 • `Scheduling/SlurmSettings/ScalingStrategy` パラメータを追加して、Slurm コンピューティングノードの Amazon EC2 インスタンスを起動するときに使用するクラスタースケーリング戦略を制御します。指定できる値は `all-or-nothing` 、`greedy-all-or-nothing` 、`best-effort` で、デフォルトは `all-or-nothing` です。 • クラスター内共有ファイルシステムリソース (ParallelCluster 、Intel、Slurm、/home のデータ) に対して、ヘッドノードルートボ	2023 年 12 月 19 日

変更	説明	日付
	リュームからの NFS エクスポートの代わりに EFS ストレージを使用するための `HeadNode/SharedStorageType` パラメータを追加しました。この機能強化により、ヘッドノードネットワークへの負荷が軽減されます。 • 設定ファイルの `SharedStorage` セクションを使用して EFS または FSx 外部共有ストレージとして `/home` をマウントすることを許可しました。 • Secrets Manager から外部ユーザー定義 MUNGE AWS キーの使用`SlurmSettings/MungeKeySecretArn` を許可する新しいパラメータを追加します。 • クラスターの Amazon CloudWatch アラームを切り替えるための `Monitoring/Alarms/Enabled` パラメータを追加しました。 • Amazon EC2 ヘルスチェック、CPU 使用率、ヘッドノードの全体的なステータスをモニタリングするためのヘッドノードアラームを追加し、クラスターで作成した CloudWatch	

変更	説明	日付
	Dashboard に結果を追加しました。 • マネージド FSx for Lustre で `PERSISTENT_2` を `DeploymentType` として使用するときに、データリポジトリの関連付けのサポートを追加しました。 • `Scheduling/SlurmSettings/Database/DatabaseName` パラメータを追加して、ユーザーが Slurm アカウンティングに使用するデータベースサーバーのデータベースのカスタム名を指定できるようにします。 • コンピューティングリソースで `CapacityReservationTarget/CapacityReservationId` を設定するときに、`InstanceType` をオプションの設定パラメータにしました。 • AWS ParallelCluster API で作成した IAM ロールとポリシーのプレフィックスを指定できるようにしました。 • AWS ParallelCluster API によって作成された IAM ロールとポリシーに適用するアクセス許可の境界を指定する可能性を追加します。	

変更	説明	日付
	変更 • Slurm を 23.02.7 (23.02.6 から) にアップグレードします。 • NVIDIA ドライバーをバージョン 535.129.03 にアップグレードしました。 • CUDA ツールキットをバージョン 12.2.2 にアップグレードしました。 • Linux 用の NVIDIA カーネルモジュールとして、クローズドソースの NVIDIA モジュールの代わりに、オープンソースの NVIDIA GPU ドライバー (OpenRM) を使用しました。 • 新しい`Scheduling/ScalingStrategy` クラスター`all_or_nothing_batch` 設定を優先して、Slurm 再開プログラムの設定パラメータのサポートを削除します。 • クラスターアラームの命名規則を「[cluster-name]-[component-name]-[metric]」に変更しました。 • ルートボリュームと追加ボリュームの両方で、ADC リージョンのデフォルトの EBS ボリュームタイプを	

変更	説明	日付

	gp2 から gp3 に変更しました。
	• AWS ParallelCluster API のオプションのアクセス許可の境界が、API インフラストラクチャによって作成されたすべての IAM ロールに適用されるようになりました。
	  • EFA インストーラーを `1.29.1` にアップグレードします。
	  • Efa-driver: `efa-2.6.0-1`
	  • Efa-config: `efa-config-1.15-1`
	  • Efa-profile: `efa-profile-1.5-1`
	  • Libfabric-aws: `libfabric-aws-1.19.0-1`
	  • Rdma-core: `rdma-core-46.0-1`
	  • Open MPI: `openmpi40-aws-4.1.6-1`
	• サポートされているすべての OS で GDRCopy をバージョン 2.4 にアップグレードしました。ただし、例外として Centos 7 ではバージョン 2.3.1 を使用します。	

変更	説明	日付
	• `aws-cfn-bootstrap` をバージョン 2.0-28 にアップグレードしました。 • aws-parallelcluster-batch-cli で Python 3.10 のサポートを追加しました。 バグ修正 • [コンピューティングリソース] で宣言したインスタンスタイプのリストを変更すると、クラスター更新のロールバック後にスケーリング設定が一貫しない問題を修正しました。 • クラスター設定ファイルを介して外部 LDAP サーバーと統合されたクラスターで、ルート権限のないユーザーを切り替えると、ユーザー SSH キーが生成される問題を修正しました。 • を設定するときの Slurm 省電力モードの無効化を修正しました`Scaledown Idletime = -1` 。 • Slurm アカウンティングの `update_slurm_database_password.sh` スクリプトの Slurm インストールディレクトリへのハードコードされたパスを修正しました。	

変更	説明	日付
AWS ParallelCluster バージョン 3.7.2 のリリース	AWS ParallelCluster バージョン 3.7.2 がリリースされました。 変更: • Slurm を 23.02.6 にアップグレードします。	2023 年 10 月 25 日

変更	説明	日付

AWS ParallelCluster バージョン 3.7.1 をリリース	AWS ParallelCluster バージョン 3.7.1 がリリースされました。
	
	変更:
	
	• Slurm を 23.02.5 (23.02.4 から) にアップグレードします。
	  • Pmix を 4.2.6 に (3.2.3 から) アップグレードしました。
	  • libjwt を 1.15.3 に (1.12.0 から) アップグレードしました。
	• EFA インストーラーを `1.26.1` にアップグレードし、P5 の RDMA データ書き込みの問題を修正しました。
	  • Efa-driver: `efa-2.5.0-1` 。
	  • Efa-config: `efa-config-1.15-1` 。
	  • Efa-profile: `efa-profile-1.5-1` 。
	  • Libfabric-aws: `libfabric-aws-1.18.2-1` 。
	  • ERdma-core: `rdma-core-46.0-1` 。
	  • Open MPI: `openmpi40-aws-4.1.5-4` 。
		2023 年 9 月 22 日

変更	説明	日付
AWS ParallelCluster バージョン 3.7.0 をリリース	AWS ParallelCluster バージョン 3.7.0 がリリースされました。 機能強化: • 設定 YAML ファイルを使用して、コンピューティングリソースの静的ノードと動的ノードの優先順位 AWS ParallelCluster の設定をサポートします。 • Ubuntu 22 のサポートを追加します。RSA キーは、デフォルトではサポートされていません。 • キュー設定 `JobExclusiveAllocation` を追加して、パーティション内のノードを常に 1 つのジョブのみに割り当てるようにします。 • クラスターの作成時と更新時に `aws-parallelcluster-node` パッケージをオーバーライドすることを許可しました。ヘッドノードの場合、これはクラスターの更新に適用されます。開発目的にのみ役立ちます。 • コンピューティングノードで NFS サーバーを起動することを回避します。	2023 年 8 月 30 日

変更	説明	日付
	• ログインノードのサポートを追加します。 • Slurm コンピューティングリソースに複数のインスタンスタイプが指定されている場合に、メモリベースのスケジューリングが可能になりました。 • 既存の Amazon File Cache を共有ストレージとしてマウントするサポートを追加します。 変更: • デフォルトでは、Slurm 動的ノードに優先度 (重み) を 1000 に設定します。これにより、Slurm はアイドル状態の動的ノードよりもアイドル状態の静的ノードを優先できます。 • デーモン`aws-parallelcluster-node` がマネージド Slurm パーティションのみを処理する AWS ParallelCluster ようにします。 • `EFS-utils` ウォッチドッグのポーリング間隔を 10 秒に増やしました。この変更は、ウォッチドッグを実行する唯一の条件として `EncryptionInTransit`	

変更	説明	日付

	が `true` に設定されている場合に適用されます。
	• EFA インストーラーを `1.25.1` にアップグレードします。
	  • Efa-driver: `efa-2.5.0-1` (`efa-2.1.1g` から)
	  • Efa-config: `efa-config-1.15-1` (`efa-config-1.13-1` から)
	  • Efa-profile: `efa-profile-1.5-1` (変更なし)
	  • Libfabric-aws: `libfabric-aws-1.18.1-0` (`libfabric-aws-1.17.1-1` から)
	  • Rdma-core: `rdma-core-46.0-1` (`rdma-core-43.0-1` から)
	  • Open MPI: `openmpi40-aws-4.1.5-4` (`openmpi40-aws-4.1.5-1` から)
	• Slurm をバージョン 23.02.4 にアップグレードします。
	• Imds/ImdsSupport のデフォルト値を v1.0 から v2.0 に変更します。
	• Ubuntu 18 は非推奨になります。
	• Centos 7 の制限を考慮して、デフォルトのルートボ	

変更	説明	日付
	リュームサイズを 40 GB に更新します。 • ヘッドノード内の /tmp/wait_condition_handle.txt のアクセス許可を制限して、root だけが読み取れるようになります。 • ノードパッケージデーモンが PC 管理の Slurm パーティションとノードリストを認識するために使用する Slurm パーティションとノードリストのマッピング JSON ファイルを作成します。 • NVIDIA ドライバーをバージョン 535.54.03 にアップグレードします。 • CUDA ライブラリをバージョン 12.2.0 にアップグレードします。 • NVIDIA ファブリックマネージャーを nvidia-fabricmanager-535 にアップグレードしました。 • Ubuntu 22.04 の場合のみ、ARM PL をバージョン 23.04.1 にアップグレードします。 • NICE DCV をバージョン `2023.0-15487` にアップグレードします。	

変更	説明	日付
	• サーバー: `2023.0.15487-1` • xdcv: `2023.0.551-1` • gl: `2023.0.1039-1` • web_viewer: `2023.0.15487-1` バグ修正: • -1 より小さい値を設定しないように `Scaledown Idletime` 値に検証を追加します。 • DCV が有効になっている GPU インスタンス上の Ubuntu Deep Learning AMI クラスター作成に失敗する問題を修正します。 • CustomLambdaRole を使用して ParallelCluster CloudFormation カスタムリソースプロバイダーを作成する場合に、ダングリング IAM ポリシーが作成される問題を修正します。 • `SlurmSettings/Dns/UseEc2Hostnames` を `True` に設定すると複数のネットワークインターフェイスを持つインスタンスでコンピューティングノードの DNS 名がずれてしまう問題を修正します。	

変更	説明	日付
	変更点の詳細については、GitHub に掲載されている aws-parallelcluster、aws-parallelcluster-cookbook、aws-parallelcluster-node の各パッケージの `CHANGELOG` ファイルを参照してください。	
ドキュメントのみリリース	AWS ParallelCluster バージョン 3 固有のユーザーガイドが公開されました。 ドキュメントのみのリリース: • AWS ParallelCluster バージョン 3 には独自のユーザーガイドがあります。	2023 年 7 月 17 日

変更	説明	日付
AWS ParallelCluster バージョン 3.6.1 をリリース	AWS ParallelCluster バージョン 3.6.1 がリリースされました。 変更: • コンピューティングノードが複数の Slurm パーティションに追加`clustermgtd` されている場合、 で見られるノードの重複を回避します。 バグ修正: • ルートボリュームデバイス名 (`/dev/sda1` および `/dev/xvda`) のハードコーディングを削除し、で使用されている AMIs から取得します`create-cluster` 。 • `ElasticIp` を `True` に設定して CloudFormation カスタムリソースを使用する場合のクラスター作成エラーを修正します。 • 大規模な設定ファイルで AWS CloudFormation カスタムリソースを使用する場合のクラスターの作成と更新の失敗を修正しました。 • Ubuntu で `ptrace` 保護が無効にならず、libfabric でク	2023 年 7 月 5 日

変更	説明	日付
	ロスメモリアタッチ (CMA) が許可されない問題を修正します。 • 複数のインスタンスタイプを使用していてインスタンスが返されない場合の、高速容量不足フェイルオーバーロジックを修正します。 変更点の詳細については、GitHub に掲載されている aws-parallelcluster、aws-parallelcluster-cookbook、aws-parallelcluster-node の各パッケージの CHANGELOG ファイルを参照してください。	

変更	説明	日付
AWS ParallelCluster バージョン 3.6.0 をリリース	AWS ParallelCluster バージョン 3.6.0 がリリースされました。 ドキュメント: • AWS ParallelCluster Python ライブラリ API のドキュメントを追加します。 機能強化: • RHEL8 のサポートを追加します。 • AWS CloudFormation でクラスターを作成および管理するための CloudFormation カスタムリソースを追加します。 CloudFormation • 設定 YAML ファイルでクラスター Slurm 設定をカスタマイズするためのサポートを追加します。 AWS ParallelCluster • LUA をサポートする Slurm を構築します。 • クラスターごとのキューの最大数を 10 から 50 に増やします。各キューには、最大 50 のコンピューティングリソースを設定できます。各クラスターには、最大 50 のコンピューティングリソースを設定できます。	2023 年 5 月 22 日

変更	説明	日付
	• OnNodeStart 、OnNodeConfigured 、および OnNodeUpdated パラメータで設定されたイベントに対して、複数のカスタムアクションスクリプトのシーケンスを指定するサポートを追加します。 • ジョブの実行前にコンピューティングノードに GPU ヘルスチェックを適用するための新しい設定セクションHealthChecks /Gpu を追加しました。 • SlurmQueues および SlurmQueues /ComputeResources 設定に Tags のサポートを追加します。 • Monitoring 設定に DetailedMonitoring のサポートを追加します。 • AWS ParallelCluster CloudWatch ダッシュボードにヘッドノードのメモリとルートボリュームのディスク使用率追跡用の mem_used_percent メトリクスと disk_used_percent メトリクスを追加し、これらのメトリクス	

変更	説明	日付
	を監視するためのアラームを設定します。 • AWS ParallelCluster 管理ログのログローテーションサポートを追加します。 • CloudWatch ダッシュボードで一般的なコンピューティングノードエラーと動的ノードの最長アイドル時間を追跡します。 • SSL ソケットを作成するときに、DCV 認証サーバーが少なくとも `TLS-1.2` プロトコルを使用するように強制します。 • `aarch64` `centos7` と `alinux2` を除くサポートされているすべてのオペレーティングシステムに、NVIDIA Data Center GPU Manager (DCGM) パッケージをインストールします。 • デフォルトでカーネルモジュール nvidia-uvm をロードして、統合仮想メモリ (UVM) 機能を CUDA ドライバーに提供します。 • NVIDIA 永続デーモンをシステムサービスとしてインストールします。 変更:	

変更	説明	日付

	• Slurm をバージョン にアップグレードします 23.02.2 (バージョン から22.05.8) 。
	• munge をバージョン 0.5.14 からバージョン 0.5.15 にアップグレードします。
	• Slurm を 30 TreeWidth に設定します。
	• Slurm prologと epilogの設定をそれぞれターゲットディレクトリ/opt/slurm/etc/scripts/prolog.d/ と に設定します/opt/slurm/etc/scripts/epilog.d/ 。
	• コンピューティングノードの登録中にPrologスクリプトを実行する場合は、Slurm BatchStartTimeout を最大 3 分に設定します。
	• CloudWatch ログのデフォルトの Retention InDays を 14 日から 180 日に増やします。
	• EFA インストーラーを 1.22.1 にアップグレードします。
	  • Dkms: 2.8.3-2
	  • Efa-driver: efa-2.1.1g (変更なし)	

変更	説明	日付
	• Efa-config: `efa-config-1.13-1` (変更なし) • Efa-profile: `efa-profile-1.5-1` (変更なし) • Libfabric-aws: `libfabric-aws-1.17.1-1` (`libfabric-aws-1.17.0-1` から) • Rdma-core: `rdma-core-43.0-1` (変更なし) • Open MPI: `openmpi40-aws-4.1.5-1` (変更なし) • Amazon Linux 2 で Lustre クライアントのバージョンを `2.12` にアップグレードします。Lustre クライアント `2.12` は、Ubuntu 20.04、18.04、CentOS 7.7 以降にインストールされています。 • CentOS 7.6 で Lustre クライアントのバージョンを `2.10.8` にアップグレードします。 • NVIDIA ドライバーをバージョン `470.141.03` からバージョン `470.182.03` にアップグレードします。 • NVIDIA ファブリックマネージャーをバージョン `470.141.03` からバー	

変更	説明	日付

	ジョン 470.182.03 にアップグレードします。
	• NVIDIA CUDA ツールキットをバージョン 11.7.1 からバージョン 11.8.0 にアップグレードします。
	• NVIDIA CUDA サンプルをバージョン 11.8.0 にアップグレードします。
	• Intel MPI ライブラリをバージョン 2021 アップデート 6 からバージョン 2021 アップデート 9 にアップグレードします。詳細については、「Intel® MPI Library 2021 Update 9」を参照してください。
	• NICE DCV をバージョン 2022.2-14521 からバージョン 2023.0-15022 にアップグレードします。
	  • server: 2023.0.15022-1 (バージョン 2022.2-14521-1 から)。
	  • xdcv: 2023.0.547-1 (バージョン 2022.2.519-1 から)。
	  • gl: 2023.0.1027-1 (バージョン 2022.2.1012-1 から)。
	  • web_viewer: 2023.0.15022-1 (バージョン	

変更	説明	日付

	2022.2.14521-1 から)。
	• aws-cfn-bootstrap をバージョン 2.0-24 にアップグレードします。
	• AWS バッチクラスターのコンテナイメージを構築するときに CodeBuild 環境で使用されるアップグレードイメージ：
	  • aws/codebuild/amazonlinux2-x86_64-standard:4.0 (aws/codebuild/amazonlinux2-x86_64-standard:3.0 から)。
	  • aws/codebuild/amazonlinux2-aarch64-standard:2.0 (aws/codebuild/amazonlinux2-aarch64-standard:1.0 から)。
	
	バグ修正:
	
	• Amazon EFS と Amazon FSx のネットワークセキュリティグループバリデーターを修正して、誤ったエラーが報告されないようにします。	

変更	説明	日付
	• `build-image` オペレーション中に Image Builder によって作成されたリソースのタグ付けが欠落していた問題を修正しました。 • `MaxCount` プロパティの数値比較を常に実行するように `MaxCount` 更新ポリシーを修正しました。 • 複数のネットワークカードを搭載したコンピューティングノードインスタンスの IP アラインメントを修正しました。 • キューパラメータの更新が実行され、Slurm アカウンティング設定が更新されない場合の `slurm_parallelcluster_slurmdbd.conf` `StoragePass` の置き換えを修正しました。 • 既存の EFS ファイルシステムでクラスターを作成すると、ダングリングセキュリティグループが作成される問題を修正しました。 • 再起動時に `cfn-hup` デーモンが失敗する問題を修正しました。 • Slurm 保護モードのブートストラップ失敗として `INVALID_REG` フラグ	

変更	説明	日付
	を持つ動的ノードを検討してください。Slurm 登録に失敗した静的ノードは、 の後に既にブートストラップ失敗として扱われますnode_replacement_timeout 。 変更点の詳細については、GitHub に掲載されている aws-parallelcluster、aws-parallelcluster-cookbook、aws-parallelcluster-node の各パッケージの CHANGELOG ファイルを参照してください。	

変更	説明	日付

AWS ParallelCluster バージョン 3.5.1 をリリース	AWS ParallelCluster バージョン 3.5.1 がリリースされました。
	
	機能強化:
	
	• スタンドアロンの `pcluster` CLI インストーラー実行ファイルを追加します。
	
	変更:
	
	• EFA インストーラーを `1.22.0` にアップグレードします。
	  • Efa-driver: `efa-2.1.1g` (`efa-2.1.1-1` から)
	  • Efa-config: `efa-config-1.13-1` (efa-config-1.12-1 から)
	  • Efa-profile: `efa-profile-1.5-1` (変更なし)
	  • Libfabric-aws: `libfabric-aws-1.17.0-1` (`libfabric-aws-1.16.1amzn3.0-1` から)
	  • Rdma-core: `rdma-core-43.0-1` (変更なし)
	  • Open MPI: `openmpi40-aws-4.1.5-1` (`openmpi40-aws-4.1.4-3` から)
		2023 年 3 月 29 日

変更	説明	日付
	NICE DCV をバージョン `2022.2-14521` にアップグレードします。 • サーバー: `2022.2.14521-1` • xdcv: `2022.2.519-1` • gl: `2022.2.1012-1` • web_viewer: `2022.2.14521-1` バグ修正: • クラスター更新の一環として共有 Amazon EBS ボリュームを削除するときに、`MountDir` と `/etc/exports` の間のパターン一致によって引き起こされる潜在的なノード起動エラーを修正しました。 • `clustermgtd` イテレーションのたびに `compute_console_output` ログファイルが切り捨てられないように修正しました。 変更点の詳細については、GitHub に掲載されている aws-parallelcluster、aws-parallelcluster-cookbook、aws-parallelcluster-node の各パッ	

変更	説明	日付
	ケージの CHANGELOG ファイルを参照してください。	

変更	説明	日付
AWS ParallelCluster バージョン 3.5.0 をリリース	AWS ParallelCluster バージョン 3.5.0 がリリースされました。 機能強化: • AWS ParallelCluster UI を使用してクラスターにアクセスして管理します。 • ワークロードで参照できる CloudFormation テンプレートにバージョニング AWS ParallelCluster ポリシーを追加します。 • 独自のコードで使用できる AWS ParallelCluster Python ライブラリを追加します。 • コンピューティングノードのブートストラップ障害時のコンピューティングノードコンソール出力のロギングを Amazon CloudWatch に追加します。 • クラスター作成失敗時の `describe-cluster` 出力にエラーコードと理由を含む [エラー] フィールドを追加しました。 • サブプロセスモジュールの呼び出し中に悪意のある文字列が挿入されるのを防ぐためのバリデーターを追加します。	2023 年 2 月 20 日

変更	説明	日付

	• 静的ノードのプロビジョニング中にクラスターの状態が `PROTECTED` に変わると、クラスターの作成に失敗します。
	
	変更:
	
	• Slurm バージョンへのアップグレード `22.05.8` (バージョン から`22.05.7`)
	• EFA インストーラーを `1.21.0` にアップグレードします。
	  • Efa-driver: `efa-2.1.1-1` (`efa-2.1` から)
	  • Efa-config: `efa-config-1.12-1` (efa-config-1.11-1 から)
	  • Efa-profile: `efa-profile-1.5-1` (変更なし)
	  • Libfabric-aws: `libfabric-aws-1.16.1amzn3.0-1` (`libfabric-aws-1.16.1` から)
	  • Rdma-core: `rdma-core-43.0-1` (`rdma-core-43.0-2` から)
	  • Open MPI: `openmpi40-aws-4.1.4-3` (変更なし)	

変更	説明	日付
	• Slurm コントローラーログをより詳細にし、Slurm 省電力プラグインの追加ログ記録を有効にします。 バグ修正: • Slurm アカウンティングが有効になっている場合、クラスター名が 40 文字以下であることを確認して、クラスターデータベースの作成を修正します。 • Amazon EC2 インスタンスのステータスチェック`clustermgtd` が失敗した場合、Slurm を介して再起動されたコンピューティングノードが置き換えられる の問題を修正しました。 • ヘッドノードの IAM ポリシーが正しくないために、他のアカウントと共有されたキャパシティ予約を持つコンピューティングノードが起動できない問題を修正しました。 変更点の詳細については、GitHub に掲載されている aws-parallelcluster、aws-parallelcluster-cookbook、aws-parallelcluster-node、aws-parallelcluster-ui の各パッケー	

変更	説明	日付
	ジの CHANGELOG ファイルを参照してください。	
AWS ParallelCluster バージョン 3.4.1 をリリース	AWS ParallelCluster バージョン 3.4.1 がリリースされました。 バグ修正: • コンピューティングノードの内部レジストリに更新が正しく適用されない可能性がある Slurm スケジューラの問題を修正しました。その結果、この問題が発生すると、EC2 インスタンスが使用できなくなったり、誤ったインスタンスタイプでバックアップされたりする可能性があります。 変更点の詳細については、GitHub に掲載されている aws-parallelcluster、aws-parallelcluster-cookbook、aws-parallelcluster-node の各パッケージの CHANGELOG ファイルを参照してください。	2023 年 1 月 13 日

変更	説明	日付
AWS ParallelCluster バージョン 3.4.0 をリリース	AWS ParallelCluster バージョン 3.4.0 がリリースされました。 機能強化: • キャパシティの可用性を高めるため、複数のアベイラビリティーゾーンにまたがるノード起動のサポートを追加します。 • キャパシティの可用性を高めるため、各キューに複数のサブネットを指定するサポートを追加します。 • Iam / ResourcePrefix に新しい設定パラメータを追加して、 によって作成された IAM リソースのパスと名前のプレフィックスを指定します AWS ParallelCluster。 • Lambda 関数で使用される Vpc 設定を指定するための新しい設定セクション DeploymentSettings / LambdaFunctionsVpcConfig AWS ParallelCluster を追加します。 LambdaFunctionsVpcConfig • クラスターの更新中にヘッドノードで実行するカスタムスクリプトを指定する機能が追加します。スクリプトは、Slurm をスケジュー	2022 年 12 月 22 日

変更	説明	日付

	ラとして使用するときに HeadNode / CustomActions / OnNodeUpdated で指定できます。
	
	変更:
	
	• 既存のファイルシステム用の Amazon EFS マウントターゲットの作成を削除します。
	• `amazon-efs-utils` を使用して EFS ファイルシステムをマウントします。EFS ファイルシステムは、転送中の暗号化と IAM 認定ユーザーを使用してマウントできます。
	• EFS の転送中暗号化をサポートするには、CentOS7 と Ubuntu に stunnel 5.67 をインストールします。
	• EFA インストーラーを `1.18.0` から `1.20.0` にアップグレードします。
	  • Efa-driver: `efa-2.1` (`efa-1.16.0-1` から)
	  • Efa-config: `efa-config-1.11-1` (変更なし)
	  • Efa-profile: `efa-profile-1.5-1` (変更なし)
	  • Libfabric-aws: `libfabric-aws-1.16`	

変更	説明	日付
	.1 (libfabric-aws-1.16.0~amzn4.0-1 から) • Rdma-core: rdma-core-43.0-2 (rdma-core-41.0-2 から) • Open MPI: openmpi40-aws-4.1.4-3 (openmpi40-aws-4.1.4-2 から) • Slurm をバージョン 22.05.5 から 22.05.7 にアップグレードします。 • Python 3.9.16 および 3.7.16 にアップグレードします。(3.9.15 および 3.7.13 から)。 • Slurm では22.05.7、IDLE+CLOUD+COMPLETING+POWER_DOWN+NOT_RESPONDING ステータスの動的ノードは異常とは見なされません。 変更点の詳細については、GitHub に掲載されている aws-parallelcluster、aws-parallelcluster-cookbook、aws-parallelcluster-node の各パッケージの CHANGELOG ファイルを参照してください。	

変更	説明	日付
AWS ParallelCluster バージョン 3.3.1 をリリース	AWS ParallelCluster バージョン 3.3.1 がリリースされました。 変更: • 公式 AWS ParallelCluster 製品 AMIsは、Amazon EC2 が非推奨になってから 2 年後に利用可能になりました。 • API AWS ParallelCluster Lambda のメモリサイズを 2048 に増やして、コールドスタートペナルティを減らし、タイムアウトを回避します。 バグ修正: • マネージド FSx for Lustre ファイルシステムの置き換えと、コンピューティングフリートのサブネット ID への変更を含むクラスター更新でのデータの損失を防止します。 • SharedStorage `DeletionPolicy` は、クラスターの更新アクションに適用されます。 変更点の詳細については、GitHub に掲載されている	2022 年 12 月 2 日

変更	説明	日付
	aws-parallelcluster パッケージの CHANGELOG ファイルを参照してください。	
AWS ParallelCluster ドキュメントのみ hpc6id メモ	AWS ParallelCluster ドキュメントのみの更新 • AWS ParallelCluster は HeadNode / InstanceType 設定の hpc6id インスタンスタイプをサポートしていません。	2022 年 12 月 2 日

変更	説明	日付

AWS ParallelCluster バージョン 3.1.5 をリリース	AWS ParallelCluster バージョン 3.1.5 がリリースされました。
	
	機能強化:
	
	• アイドル状態のノードを終了させない Slurm の問題を修正しました。
	• EFA インストーラーを 1.18.0 にアップグレード
	  • Efa-driver: `efa-1.16.0-1`
	  • Efa-config: `efa-config-1.11-1` (`efa-config-1.9-1` から)
	  • Efa-profile: `efa-profile-1.5-1` (変更なし)
	  • Libfabric-aws: `libfabric-aws-1.16.0~amzn4.0-1` (`libfabric-1.13.2` から)。
	  • Rdma-core: `rdma-core-41.0-2` (`rdma-core-37.0` から)
	  • Open MPI: `openmpi40-aws-4.1.4-2` (`openmpi40-aws-4.1.1-2` から)
	
	変更:
		2022 年 11 月 16 日

変更	説明	日付
	• クラスターの更新のために AWS ParallelCluster API スタック`ParallelClusterUserRole` で使用される `lambda:UntagResource` に `lambda:ListTags` と を追加します。 • Intel MPI ライブラリを、バージョン 2021 アップデート 4 からバージョン 2021 アップデート 6 にアップグレードします。詳細については、「Intel® MPI Library 2021 Update 6」を参照してください。 • NVIDIA ドライバーをバージョン 470.103.01 から 470.141.03 にアップグレードします。 • NVIDIA ファブリックマネージャーをバージョン 470.103.01 から 470.141.03 にアップグレードします。 変更点の詳細については、GitHub に掲載されている aws-parallelcluster、aws-parallelcluster-cookbook、aws-parallelcluster-node の各パッケージの `CHANGELOG` ファイルを参照してください。	

変更	説明	日付
AWS ParallelCluster バージョン 3.3.0 をリリース	AWS ParallelCluster バージョン 3.3.0 がリリースされました。 機能強化: • Slurm をスケジューラとして使用する場合は、コンピューティングリソースの複数のインスタンス割り当て設定のサポートを追加します。詳細については、「Slurm を使用した複数のインスタンスタイプの割り当て」を参照してください。 • 更新された設定を使用して、クラスターの更新で SharedStorage を追加および削除するサポートを追加します。詳細については、「共有ストレージ」を参照してください。 • Efs および FsxLustre 共有ストレージ設定`DeletionPolicy` の新しい設定パラメータを追加して、ストレージの保持をサポートします。 • 新しい設定パラメータ Scheduling / SlurmSettings / Database で Slurm アカウンティングのサポートを追加します。 ./Schedu	2022 年 11 月 2 日

変更	説明	日付
	ling-v3.html#Scheduling-v3-SlurmSettings-Database詳細については、「 を使用した Slurm アカウンティング AWS ParallelCluster」を参照してください。 • オンデマンドキャパシティ予約 (ODCR) とキャパシティ予約リソースグループのサポートを追加しました。詳細については、「オンデマンドキャパシティ予約 (ODCR) を使用してインスタンスを起動する」を参照してください。 • 新しい設定パラメータを追加して、クラスターでサポートする IMDS バージョンを指定するか、クラスター内のイメージインフラストラクチャ、Imds / ImdsSupport、およびビルド、Imds / ImdsSupport、設定をビルドします。 • SlurmQueues / ComputeResources セクションに Networking / Placement Group のサポートを追加します。 • デバイスごとに 1 つの ENI のみに制限されている、複数のネットワークインターフェイスを持つインスタン	

変更	説明	日付
	スのサポートを追加します。 • アタッチされたセキュリティグループの CIDR ブロックをチェックすることで、外部 Amazon EFS ファイルシステムのネットワークの検証を改善します。 • 設定したインスタンスタイプがプレイスメントグループをサポートしているかどうかを確認するバリデーターを追加します。 • 安定性とパフォーマンスを向上させるため、NFS スレッドを min (256、max (8, num_cores * 4)) に設定します。 • 設定時間を短縮するため、ビルド時に NFS のインストールを移動します。 • AWS ParallelCluster API をデプロイするときに作成され、docker イメージビルドイベントで に通知するために使用される EcrImageBuilder SNS トピックのサーバー側の暗号化を有効にします。 変更: • SlurmQueues / Networking / PlacementGroup / の動作	

変更	説明	日付
	を変更しますEnabled。すべてのコンピューティングリソースに対して単一のマネージドプレースメントグループを作成するのではなく、コンピューティングリソースごとに固有のマネージドプレースメントグループを作成するようになりました。 • 任意の命名方法として SlurmQueues / Networking / PlacementGroup / Name のサポートを追加します。 • タグ更新時にヘッドノードが置換されないように、ヘッドノードタグを起動テンプレートからインスタンス定義に移動します。 • 起動テンプレートで設定された `CpuOptions` ではなく、`cloud-init` によって実行されるスクリプトを通じてマルチスレッドを無効にします。 • API インフラストラクチャ、API Docker コンテナ、クラスター Lambda リソースの Python をバージョン 3.9 に、NodeJS をバージョン 16 にアップグレードします。 • `aws-parallelcluster-batch-cli` での	

変更	説明	日付

	Python 3.6 のサポートを終了します。
	• Slurm をバージョン `21.08.8-2` から `22.05.5` にアップグレードします。
	• NVIDIA ドライバーをバージョン `470.129.06` から `470.141.03` にアップグレードします。
	• NVIDIA ファブリックマネージャーをバージョン `470.129.06` から `470.141.03` にアップグレードします。
	• NVIDIA CUDA ツールキットをバージョン 11.7.1 にアップグレードします (`from 11.4.4`)。
	• AWS ParallelCluster virtualenvs で使用される Python を から `3.7.13`にアップグレードします`3.9.15`。
	• EFA インストーラーを 1.18.0 にアップグレードします。
	  • Efa-driver: `efa-1.16.0-1` (変更なし)
	  • Efa-config: `efa-config-1.11-1` (`from efa-config-1.10-1`)	

変更	説明	日付
	• Efa-profile: `efa-profile-1.5-1` (変更なし) • Libfabric-aws: `libfabric-aws-1.16.0~amzn4.0-1` (`libfabric-aws-1.16.0~amzn2.0-1` から)。 • Rdma-core: `rdma-core-41.0-2` (`rdma-core-37.0` から) • Open MPI: `openmpi40-aws-4.1.4-2` (`openmpi40-aws-4.1.1-2` から) • NICE DCV をバージョン `2022.0-12760` からバージョン `2022.1-13300` にアップグレードします。 • Queues の `SingleSubnetValidator` の抑制を有効にします。 • Epilog がまだ実行中である可能性があるため、ノードが `COMPLETING` の状態では `DRAIN` ノードを交換しません。 バグ修正: • 誤ったフィルターが渡されたときに失敗するように、 AWS ParallelCluster	

変更	説明	日付
	`ListClusterLogStreams` コマンドの filters パラメータの検証を修正しました。 • が他の SharedStorage / EfsSettings パラメータとともに`FileSystemId` 指定されている場合に検証に失敗するように、パラメータ SharedStorage / EfsSettings の検証を修正しました。以前は、`FileSystemId` は含まれていませんでした。 • SharedStorage の順序を変更するときのクラスターの更新と、設定の他の変更を修正しました。 • CloudWatch `UpdateParallelClusterLambdaRole` にログをアップロードするための AWS ParallelCluster API の修正。 • クックブックが実行される前にパッケージをインストールするときに Cinc がローカル CA 証明書バンドルを使用しない問題を修正しました。 • `Build:UpdateOsPackages:Enabled:true` が設定されている場合に、`pcluster build-image` を使用して ubuntu	

変更	説明	日付
	をアップグレードする際のハングを修正しました。 • キーが重複している場合に失敗することによる YAML クラスター設定の解析を修正しました。 変更点の詳細については、GitHub に掲載されている aws-parallelcluster、aws-parallelcluster-cookbook、aws-parallelcluster-node の各パッケージの `CHANGELOG` ファイルを参照してください。	
AWS ParallelCluster ドキュメントのみの API リファレンスを追加しました。	AWS ParallelCluster ドキュメントのみの更新 • バージョン 3 の AWS ParallelCluster API リファレンスをドキュメントに追加しました。	2022 年 10 月 27 日

変更	説明	日付
AWS ParallelCluster バージョン 3.2.1 をリリース	AWS ParallelCluster バージョン 3.2.1 がリリースされました。 機能強化: • 複数の NIC を持つ Amazon EC2 インスタンスをより適切にサポートするために、ホストルーティングテーブルをさまざまなネットワークカードに関連付けるロジックを改善しました。 変更: • NVIDIA ドライバーをバージョン 470.141.03 にアップグレードします。 • NVIDIA ファブリックマネージャーをバージョン 470.141.03 にアップグレードします。 • `cron` ジョブタスク `man-db` および `mlocate` を無効にします。これは、ノードのパフォーマンスに悪影響が及ぶ可能性があります。 • Intel MPI ライブラリを 2021.6.0.602 にアップグレードします。 • このセキュリティリスクに対応して Python を 3.7.10	2022 年 10 月 3 日

変更	説明	日付
	から 3.7.13 にアップグレードします。 バグ修正: • クラスター設定が利用できないときには `DescribeCluster` が失敗することを回避します。 変更点の詳細については、GitHub に掲載されている aws-parallelcluster、aws-parallelcluster-cookbook、aws-parallelcluster-node の各パッケージの `CHANGELOG` ファイルを参照してください。	

変更	説明	日付

AWS ParallelCluster バージョン 3.2.0 をリリース	AWS ParallelCluster バージョン 3.2.0 がリリースされました。
	
	機能強化:
	
	• Slurm にメモリベースのスケジューリングのサポートを追加します。
	  • Slurm クラスター設定でコンピューティングノードの実際のメモリを設定します。
	  • 新しい設定パラメータ Scheduling/SlurmSettings/EnableMemoryBasedScheduling を追加して、Slurm でメモリベースのスケジューリングを有効にします。
	  • 新しい設定パラメータ Scheduling / SlurmQueues / ComputeResources / SchedulableMemory を追加して、コンピューティングノードのスケジューラに表示されるメモリのデフォルト値を上書きします。
	• 可能な限りクラスター全体の停止と開始を回避するために、クラスター設定の更新の柔軟性を向上させます。新しい設定パラメー
		2022 年 7 月 27 日

変更	説明	日付
	タ Scheduling/SlurmSettings/QueueUpdateStrategy を追加して、コンピューティングノードで設定の更新と置換が必要な場合に使用する優先戦略を設定します。 • Amazon EC2 インスタンスで容量不足の問題が発生した場合に、利用可能なコンピューティングリソースに対するフェイルオーバーメカニズムを改善しました。容量不足によりノードの起動が失敗した場合、設定した時間だけコンピューティングノードを無効にします。 • 既存の FSx for ONTAP および FSx for OpenZFS ファイルシステムをマウントするためのサポートを追加します。 • 既存の Amazon Elastic File Systems、FSx for Lustre、FSx for ONTAP、FSx for OpenZFS ファイルシステムの複数のインスタンスをマウントするサポートを追加します。 • 新しいファイルシステムを作成するときに、FSx for Lustre Persistent_2 デプロ	

変更	説明	日付

	イタイプのサポートを追加します。
	• pcluster 設定ウィザードを使用するときに、サポートされているインスタンスタイプの EFA を有効にするようにユーザーに求めます。
	• Slurm を使用してコンピューティングノードを再起動するサポートを追加します。
	• Slurm の電源状態の処理を改善して、ノードの手動による電源切断も考慮に入れます。
	• NVIDIA GdrCopy 2.3 を製品 AMI にインストールして、低レイテンシーの GPU メモリコピーを可能にします。
	
	変更:
	
	• EFA インストーラーをバージョン 1.17.2 にアップグレードしました。
	  • EFA driver: `efa-1.16.0-1`
	  • EFA 設定: `efa-config-1.10-1`
	  • EFA プロファイル: `efa-profile-1.5-1`	

変更	説明	日付

	• Libfabric: `libfabric-aws-1.16.0~amzn2.0-1`
	• RDMA コア: `rdma-core-41.0-2`
	• Open MPI: `openmpi40-aws-4.1.4-2`
	• NICE DCV をバージョン 2022.0-12760 にアップグレードします。
	• NVIDIA ドライバーをバージョン 470.129.06 にアップグレードします。
	• NVIDIA ファブリックマネージャーをバージョン 470.129.06 にアップグレードします。
	• ルートボリュームと追加ボリュームの両方で、デフォルトの EBS ボリュームタイプを gp2 から gp3 に変更します。
	• 以下によって作成された FSx for Lustre ファイルシステムの変更 AWS ParallelCluster：
	  • デフォルトのデプロイタイプを `Scratch_2` に変更します。
	  • Lustre サーバーのバージョンを `2.12` に変更します。	

変更	説明	日付
	• 既存の / を渡すtrueときに PlacementGroup PlacementGroup / Enabled を に設定する必要はありませんId。 • Placement Group /Enabled が明示的に false に設定されている場合、Placement Group /Id を設定することはできません。 • AWS ParallelClusterによって作成されたすべてのリソースに parallelcluster:cluster-name タグを追加します。 • API AWS ParallelClusterスタックがクラスターの更新ParallelClusterUserRole に使用する lambda:ListTags と lambda:UntagResource を追加します。 • 設定パラメータ HeadNode/Imds/Secured が有効な場合、IMDS への IPv6 アクセスを root およびクラスター管理者ユーザーのみに制限します。 • カスタム AMI では、ParallelCluster のデフォルトである 35 GiB の代わりに AMI のルートボリュームサイズ	

変更	説明	日付
	を使用します。この値はクラスター設定ファイルで変更できます。 • 設定パラメータ `Scheduling /SlurmQueues /ComputeResources /SpotPrice` が最低限必要なスポットリクエストフルフィルメント価格を下回ると、コンピューティングフリートが自動的に無効になります。 • 更新中にセクションを追加または削除する場合に、変更セット内の `requested_value` および `current_value` の値を表示します。 • 複数のネットワークカードでインスタンスを設定する場合に、`configure_nw_interface.sh` との競合を避けるために、Deep Learning AMI で利用可能な `aws-ubuntu-eni-helper` サービスを無効にします。 • Python 3.6 のサポートを終了します。 • 複数のネットワークカードでインスタンスを設定する場合、すべてのネットワークインターフェイスの MTU を 9001 に設定します。	

変更	説明	日付
	• コンピューティングノードの FQDN を設定する場合は、末尾のドットを削除します。 • POWERING_DOWN で静的ノードを管理します。 • ジョブがまだ実行中である可能性があるため、POWER_DOWN の動的ノードは置き換えられません。 • クラスター設定で Scheduling パラメータが更新された場合にのみ、クラスター更新時に clustermgtd デーモンと slurmctld デーモンを再起動します。 • slurmctld および slurmd systemd サービスファイルを更新します。 • 設定パラメータ HeadNode/Imds/Secured が有効な場合、IMDS への IPv6 アクセスを root およびクラスター管理者ユーザーのみに制限します。 • Slurm 設定 AuthInfo=cred_expire=70 を設定して、ノードが使用できないときにキューに再キューされたジョブが再開	

変更	説明	日付

	するまで待たなければならない時間を短縮します。
	• サードパーティー製クックブックの依存関係をアップグレードします。
	  • apt-7.4.2 (apt-7.4.0 から)
	  • line-4.5.2 (line-4.0.1 から)
	  • openssh-2.10.3 (openssh-2.9.1 から)
	  • pyenv-3.5.1 (pyenv-3.4.2 から)
	  • selinux-6.0.4 (selinux-3.1.1 から)
	  • yum-7.4.0 (yum-6.1.1 から)
	  • yum-epel-4.5.0 (yum-epel-4.1.2 から)
	
	バグ修正:
	• カスタム AMI を構築するときに AWS ParallelCluster 検証ステップとテストステップをスキップするデフォルトの動作を修正しました。
	• `computemgtd` でのファイルハンドルのリークを修正しました。
	• 起動したインスタンスが EC2 DescribeInstances レスポンスでまだ使用できないために散発的に即座に終	

変更	説明	日付
	了する原因となっていた競合状態を修正しました。 • Arm プロセッサを使用したインスタンスタイプでの `DisableSimultaneousMultithreading` パラメータのサポートを修正しました。 • 以前のバージョンからアップグレードする際の AWS ParallelCluster API スタックの更新エラーを修正しました。`ListImagePipelineImages` アクションに使用されるリソースパターンを `EcrImageDeletionLambdaRole` に追加します。 • FSx for Lustre ファイルシステムを作成するときに Amazon S3 からインポートまたはエクスポートするために必要なアクセス許可が欠落している AWS ParallelCluster API を修正しました。 変更点の詳細については、GitHub に掲載されている aws-parallelcluster、aws-parallelcluster-cookbook、aws-parallelcluster-node の各パッ	

変更	説明	日付
	ケージの CHANGELOG ファイルを参照してください。	

変更	説明	日付
AWS ParallelCluster ドキュメントのみの更新	AWS ParallelCluster ドキュメントのみの更新。 新規セクション • ベストプラクティス: 予算アラート V3 • ベストプラクティス: クラスターを新しいマイナーバージョンまたはパッチバージョン V3 AWS ParallelCluster に移動する V3 • Amazon S3 V3 の使用 V3 • スポットインスタンス V3 の使用 V3 • Slurm クラスター保護モード V3 • AWS ParallelCluster リソースとタグ付け V3 • Amazon CloudWatch ダッシュボード V3 • Amazon CloudWatch Logs V3 との統合 V3 • Elastic Fabric Adapter V3 • AWS ParallelCluster AMI カスタマイズ V3 • オンデマンドキャパシティ予約 (ODCR) V3 を使用してインスタンスを起動する V3 • AMI パッチ適用と Amazon EC2 インスタンス置換 V3 • AWS ParallelCluster の仕組み V3	2022 年 7 月 6 日

変更	説明	日付
	• KMS キー V3 AWS を使用した共有ストレージ暗号化の設定 V3 • マルチキューモードクラスター V3 でのジョブの実行 V3 • AWS ParallelCluster API V3 の使用 V3 セクション更新: • ベストプラクティス: ネットワークパフォーマンス V3: Elastic Fabric Adaptor を使用するためのベストプラクティスを追加しました。 • V3 AWS での Identity and Access Management のアクセス許可 AWS ParallelCluster: Amazon FSx for Lustre AWS ParallelCluster を使用する際のさまざまな更新と追加の pcluster ユーザーポリシーの追加。 • V3 のAWS ParallelCluster トラブルシューティング: さまざまな更新。	

変更	説明	日付
AWS ParallelCluster バージョン 3.1.4 をリリース	AWS ParallelCluster バージョン 3.1.4 がリリースされました。 機能強化: • DirectoryService/PasswordSecretArn の検証を追加して、シークレットが存在しない場合に失敗するようにします。 JWT 認証 Slurm を有効にするサポートを追加します。 変更: • Slurm をバージョン 21.08.8-2 へアップグレードします。 • JWT サポート付きで Slurm をビルドします。 • 既存の / を渡す`true`ときに PlacementGroup `PlacementGroup` / Enabled を に設定する必要はありません`Id`。 • クラスター作成とイメージ作成のために ParallelCluster API スタックに使用される `ParallelClusterUserRole` に `lambda:TagResource` を追加します。	2022 年 5 月 16 日

変更	説明	日付
	バグ修正: • `export-cluster-logs` コマンドを `--filters` オプションとともに使用したときに、クラスターのログをエクスポートする機能を修正しました。 • /home 共有ディレクトリを使用してMulti-node-Parallel ジョブの実行を調整するように AWS バッチ Docker エントリポイントを修正しました。 • Slurm の異常な静的ノードをダウンに設定する際にノードアドレスをリセットし、容量不足で障害が発生した静的ノードをブートストラップ障害ノードとして扱うことを回避します。 変更点の詳細については、GitHub に掲載されている aws-parallelcluster、aws-parallelcluster-cookbook、aws-parallelcluster-node の各パッケージの `CHANGELOG` ファイルを参照してください。	

変更	説明	日付
AWS ParallelCluster バージョン 3.1.3 をリリース	AWS ParallelCluster バージョン 3.1.3 がリリースされました。 機能強化: • SSH キーの作成は、SSH ログイン時、別のユーザーへの切り替え時、別のユーザーとしてコマンドを実行する場合など、ホームディレクトリの作成と同時に実行します。 • DirectoryService / DomainName 設定パラメータに FQDN と LDAP 識別名の両方のサポートを追加します。新しいバリデーターは両方の構文をチェックするようになりました。 • ヘッドノードにデプロイされた新しい `update_directory_service_password.sh` スクリプトは、SSSD 設定の Active Directory パスワードの手動更新をサポートします。パスワードは、クラスター設定から AWS Secrets Manager によって取得されます。 • デフォルト VPC のない環境に API インフラストラクチ	2022 年 4 月 20 日

変更	説明	日付
	ャをデプロイするためのサポートを追加します。 変更: • 高パフォーマンスと低レイテンシーを保証するために、x86_64 公式 AMI および `build-image` コマンドで作成された AMI のより深い C ステートを無効にします。 • OS パッケージの更新とセキュリティ修正。 • カーネル 5.10 で AMI を使用するように Amazon Linux 2 のベースイメージを変更します。 バグ修正: • 新しい EC2 Image Builder ポリシーにより、イメージが正常にビルドされた後の `DELETE_FAILED` のビルドイメージスタックを修正しました。 • 複数のドメインアドレスが含まれている場合、設定パラメータ Directory Service/DomainAddr から ldap_uri SSSD プロパティへの変換を修正しました。	

変更	説明	日付
	変更点の詳細については、GitHub の aws-parallelcluster および aws-parallelcluster-cookbook パッケージの CHANGELOG ファイルを参照してください。	

変更	説明	日付
AWS ParallelCluster バージョン 3.1.2 をリリース	AWS ParallelCluster バージョン 3.1.2 がリリースされました。 変更: • Slurm をバージョン `21.08.5` から `21.08.6` にアップグレードします。 バグ修正: • インターネットにアクセスできないサブネットにクラスターがデプロイされた場合の、コンピューティングノード上の `/etc/hosts` ファイルの更新を修正しました。 • コンピューティングノードのブートストラップを修正して、エフェメラルドライブの初期化を待ってからクラスターに参加するようにしました。 変更点の詳細については、GitHub の aws-parallelcluster パッケージの `CHANGELOG` ファイルを参照してください。	2022 年 3 月 2 日

変更	説明	日付
AWS ParallelCluster バージョン 3.1.1 をリリース	AWS ParallelCluster バージョン 3.1.1 がリリースされました。 • AWS Directory Service で管理される Active Directory (AD) ドメインと統合することで、複数のユーザークラスター環境のサポートを追加します。 • クラスター設定ファイルに UseEc2Hostnames のサポートを追加します。true に設定すると、コンピューティングノードに Amazon EC2 のデフォルトホスト名 (例:ip-1-2-3-4) を使用します。 • インターネットにアクセスできないサブネットでのクラスター作成のサポートを追加します。 • キューごとに複数のコンピューティングインスタンスタイプのサポートを追加します。 • NVIDIA カードを搭載した ARM インスタンス上の Slurm による GPU スケジューリングのサポートを追加します。 • `(-n) cluster-name`、()`region`、`-r image-id` (), / `image-con`	2022 年 2 月 10 日

変更	説明	日付
	figuration (-i) cluster-configuration の省略フラグ-cを AWS ParallelCluster CLI に追加します。 • FSx for Lustre AutoImportPolicy パラメータNEW_CHANGED_DELETED のサポートを追加します。 • コンピュートノードが使用する EC2 LaunchTemplates リソースに parallelcluster:compute-resource-name タグを追加します。 • 一部のヘッドノードやキューに SecurityGroups パラメータが指定されている場合に、カスタムセキュリティグループからのインバウンド接続を許可するように、クラスター内に作成されるセキュリティグループを改善します。 • ARM 用の NVIDIA ドライバーと CUDA ライブラリをインストールします。 変更:	

変更	説明	日付
	• Slurm をバージョン 20.11.8 から 21.08.5 にアップグレードします。 • Slurm プラグインをバージョン 20.11 から 21.08 にアップグレードします。 • NICE DCV をバージョン 2021.1-10851 からバージョン 2021.3-11591 にアップグレードします。 • NVIDIA ドライバーをバージョン 470.57.02 から 470.103.01 にアップグレードします。 • NVIDIA ファブリックマネージャーをバージョン 470.57.02 から 470.103.01 にアップグレードします。 • CUDA をバージョン 11.4.0 から 11.4.4 にアップグレードします。 • Intel MPI を、バージョン 2019 アップデート 8 からバージョン 2021 アップデート 4 に更新しました。詳細については、「Intel® MPI Library 2021 Update 4」を参照してください。 • PMIx をバージョン 3.1.5 から 3.2.3 にアップグレードします。	

変更	説明	日付
	• 障害が発生したコンピューティングノードの `/home/logs/compute` へのダンプを削除します。コンピューティングノードのログファイルは、CloudWatch と Amazon EC2 コンソールのログにあります。 • `SlurmQueues` および `ComputeResources` の長さバリデーターを抑制する可能性を有効にします。 • Amazon Linux 2 では、インスタンス起動時のパッケージ更新を無効にします。 • AWS ParallelCluster カスタムイメージを構築するときに、Amazon EC2 ImageBuilder の拡張イメージメタデータを無効にしました。 • `cloud-init` データソースを明示的に EC2 に設定します。これにより、Ubuntu および CentOS プラットフォームの起動時間を節約できます。 • コンピューティングフリート起動テンプレート名には、インスタンスタイプではなくコンピューティングリソース名を使用します。 • pcluster CLI 出力に不要なテキストが含まれないように、stderr と stdout を CLI	

変更	説明	日付
	ログファイルにリダイレクトします。 • configure/install レシピを、メインのクックブックから呼び出される別々のクックブックに移動します。既存のエントリポイントは維持され、下位互換性があります。 • クラスター作成時にインターネットに接続しないように、AMI ビルド時に Intel HPC プラットフォームの依存関係をダウンロードします。 • Slurm ノードを設定するときに、コンピューティングリソース名から - を削除しないでください。 • NVIDIA ドライバーがインストールされていない場合は Slurm で GPU を設定しないでください。 • `BatchUserRole` の `ecs:ListContainerInstances` 権限を修正しました。 • 以前は None プレフィックスにエクスポートされていた、プレフィックスが指定されていない場合のクラスターログのエクスポートを修正しました。	

変更	説明	日付
	• クラスターの更新に失敗した場合にロールバックが実行されない問題を修正しました。 • `BatchUserRole` の `ecs:ListContainerInstances` 権限を修正しました。 • サポート対象外の `KmsKeyId` が指定された場合にエラーを発生させることで、`HeadNode` の `RootVolume` スキーマを修正しました。 • CloudWatch ダッシュボードに表示される Amazon FSx の欠落しているメトリクスを修正しました。 • `EfaSecurityGroupValidator` を修正しました。以前は、カスタムセキュリティグループが提供され、EFA が有効になっている場合、誤った障害が発生する可能性がありました。 変更点の詳細については、GitHub に掲載されている aws-parallelcluster、aws-parallelcluster-cookbook、aws-parallelcluster-node の各パッケージの `CHANGELOG` ファイルを参照してください。	

変更	説明	日付
AWS ParallelCluster バージョン 3.0.3 のリリース	AWS ParallelCluster バージョン 3.0.3 がリリースされました。 • パフォーマンスが低下する可能性を避けるため、Amazon Linux 2 で `log4j-cve-2021-44228-hotpatch` エージェント (`Log4jHotPatch`) を無効にします。詳細については、「Amazon Linux Hotpatch Announcement for Apache Log4j」を参照してください。 変更点の詳細については、GitHub の aws-parallelcluster および aws-parallelcluster-cookbook パッケージの `CHANGELOG` ファイルを参照してください。	2022 年 1 月 17 日

変更	説明	日付
AWS ParallelCluster バージョン 3.0.2 をリリース	AWS ParallelCluster バージョン 3.0.2 がリリースされました。 Elastic Fabric Adapter インストーラを 1.14.1 にアップグレードする • EFA 設定: `efa-config-1.9-1` (`efa-config-1.9` から) • EFA プロファイル:`efa-profile-1.5-1` (`efa-profile-1.5` から) • EFA カーネルモジュール: `efa-1.14.2` (`efa-1.13.0` から) • RDMA コア: `rdma-core-37.0` (`rdma-core-35` から) • Libfabric: `libfabric-1.13.2` (`libfabric-1.13.0` から) • Open MPI: `openmpi40-aws-4.1.1-2` (変更なし) GPUDirect RDMA は、インスタンスタイプでサポートされていれば常に有効です。GdrSupport 構成オプションは効果がありません。 変更点の詳細については、GitHub に掲載されている	2021 年 11 月 5 日

変更	説明	日付
	aws-parallelcluster、aws-parallelcluster-cookbook、aws-parallelcluster-node の各パッケージの CHANGELOG ファイルを参照してください。	

変更	説明	日付
AWS ParallelCluster バージョン 3.0.1 をリリース	AWS ParallelCluster バージョン 3.0.1 がリリースされました。 クラスター構成移行ツール • お客様は、クラスター設定を AWS ParallelCluster バージョン 2 形式から YAML ベースの AWS ParallelCluster バージョン 3 形式に移行できるようになりました。詳細については、「pcluster3-config-converter」を参照してください。 ヘッドノードを停止できます • コンピューティングフリートを停止した後、Amazon EC2 コンソールまたは stop-instances AWS CLI コマンドを使用して、ヘッドノードを停止し、後で再起動できます。 `~/.aws/config` ファイルから読み取られたデフォルトの AWS リージョン • pcluster コマンドでは、設定ファイル、環境、またはコマンドラインで AWS リージョンが指定されて	2021 年 10 月 27 日

変更	説明	日付
	いない場合、~/.aws/config ファイルの [default] セクションの region設定で指定されたデフォルトの AWS リージョンが使用されます。 変更点の詳細については、GitHub に掲載されている aws-parallelcluster、aws-parallelcluster-cookbook、aws-parallelcluster-node の各パッケージの CHANGELOG ファイルを参照してください。	

変更	説明	日付
AWS ParallelCluster バージョン 3.0.0 をリリース	AWS ParallelCluster バージョン 3.0.0 がリリースされました。 Amazon API Gateway によるクラスター管理のサポート • お客様は、Amazon API Gateway を使用して、HTTP エンドポイントを通じてクラスターを管理およびデプロイできるようになりました。これにより、スクリプトやイベントドリブンのワークフローに新たな可能性が生まれます。 AWS ParallelCluster コマンドラインインターフェイス (CLI) も、この API との互換性のために再設計され、新しい JSON 出力オプションが含まれています。この新機能により、お客様は同様のビルディングブロック機能を CLI でも実装できるようになります。 カスタム AMI 作成の改善 • お客様は、EC2 Image Builder を使用してカスタム AMI を作成および管理するためのより堅牢なプロセスを利用できるようになりま	2021 年 9 月 10 日

変更	説明	日付
	した。カスタム AMIs を別の AWS ParallelCluster 設定ファイルで管理でき、コマンドラインインターフェイスの pcluster build-image AWS ParallelCluster コマンドを使用して作成できるようになりました。 変更点の詳細については、GitHub に掲載されている aws-parallelcluster、aws-parallelcluster-cookbook、aws-parallelcluster-node の各パッケージの `CHANGELOG` ファイルを参照してください。	

PCUI

変更	説明	日付
PCUI バージョン 2025.04.0 をリリース	PCUI バージョン 2025.04.0 をリリース 重大な変更： • PC バージョンのデフォルト値を削除します。これで、ユーザーは使用する PC バージョンを指定する必要があります。 機能:	2025 年 4 月 16 日

変更	説明	日付
	• 新しいスタックパラメータ 'AdditionalPolicies PCAPI ' を追加して、デフォルトのロールに加えて ParallelCluster API Lambda ロールのカスタムアクセス許可を追加します。 バグ修正: • PCUI テンプレートを使用して正しい URLs。 • ジョブステータスタブで 200 以上のジョブをロードできない問題を修正しました。（https://github.com/aws/aws-parallelcluster-ui/issues/376 を参照）。 のセキュリティ • Python を 3.9 から 3.12 にアップグレードします。 • クロススポーンを 7.0.3 から 7.0.6 にアップグレードして、脆弱性 CVE-2024-21538 に対処します。 • CVE-2024-35195 に対応するため、リクエストを 2.31.0 から 2.32.0 にアップグレードします。 • urllib3 を 1.26.18 から 1.26.19 にアップグレードし	

変更	説明	日付
	て CVE-2024-37891 に対処します。 • CWE-1395 に対応するために暗号化を 42.0.4 から 44.0.1 にアップグレードします。 • CVE-2024-39689 に対応するために certifi を 2023.7.22 から 2024.7.4 にアップグレードします。 • jinja2 を 3.1.3 から 3.1.6 にアップグレードして、CVE-2024-56201 および CVE-2024-56326 に対処します。 • serverless_wsgi.py をバージョン 3.0.5 にアップグレードします。 • Werkzeug を 2.3.8 からバージョン 3.0.6 にアップグレードして、CVE-2024-34069、CVE-2024-49766、CVE-2024-49767 に対処します。 • Axios を 1.6.7 からバージョン 1.8.2 にアップグレードして、CVE-2024-39338 に対処します。 • Next.js を 14.1.1 からバージョン 14.2.25 にアップグレードして、CVE-2024-51479、CVE-2024-	

変更	説明	日付
	46982、CVE-2025-29927 に対処します。 • idna を 3.4 からバージョン 3.7 にアップグレードして CVE-2024-3651 に対処します。 • CVE-2024-55565 に対応するために nanoid を 3.3.7 からバージョン 3.3.8 にアップグレードします。 • python-jose を 3.3.0 からバージョン 3.4.0 にアップグレードして、CVE-2022-29217 に対処します。	
PCUI バージョン 2024.11.0 をリリース	PCUI バージョン 2024.11.0 をリリース バグ修正: • ECR プライベートリポジトリのポリシーを明示的に設定して、Lambda 関数に影響するスタック更新でのポリシーの削除を防止します。このポリシーには、Lambda 関数がコードを取得するために必要なアクセス許可が含まれています。	2024 年 11 月 22 日

変更	説明	日付
PCUI バージョン 2024.10.0 をリリース	PCUI バージョン 2024.10.0 をリリース 変更: • AWS ParallelCluster 3.11.1 のサポートを追加します。 • ウィザードでオンデマンドキャパシティ予約とキャパシティブロックのサポートを追加します。 • ウィザードでサポートされているインスタンスタイプのリストに g6、m7、p5 ファミリーを追加します。 • 新しいスタックオプションパラメータを追加して、PCUI と Cognito の両方のカスタムドメインを設定します。 バグ修正: • カスタムドメインのセットアップを中断していたバグを修正しました。 のセキュリティ • Flask-CORS を 3.0.10 から 4.0.2 にアップグレードして、脆弱性 CVE-2024-6221 に対処します。	2024 年 10 月 22 日

変更	説明	日付
	• lint-staged を 13.0.3 から 15.2.5 にアップグレードして、脆弱性 CVE-2024-4068 に対処します。 • 完全な変更ログ	
PCUI バージョン 2024.05.0 をリリース	PCUI バージョン 2024.05.0 をリリースしました。 バグ修正 • ユーザーが [ジョブステータス] パネルを開くときに UI がブロックされるという、フロントエンドのバグを修正しました。 • 完全な変更ログ	2024 年 5 月 14 日
PCUI バージョン 2024.04.0 をリリース	PCUI バージョン 2024.04.0 をリリースしました。 機能: • AWS ParallelCluster バージョン 3.9.1 のサポートを追加 • 完全な変更ログ	2024 年 4 月 17 日

変更	説明	日付
PCUI バージョン 2024.03.0 をリリース	PCUI バージョン 2024.03.0 をリリースしました。 機能: • AWS ParallelCluster バージョン 3.9.0 のサポートを追加 • Ubuntu 22.04 と Red Hat Enterprise Linux 9 のサポートを追加 • Ubuntu 18.04 を廃止 バグ修正 • 多くのクラスターを使用する場合に一部のクラスターが表示されない問題を修正 変更点の詳細については、GitHub で aws-parallelcluster-ui パッケージの CHANGELOG ファイルを参照してください。	2024 年 3 月 12 日

変更	説明	日付
PCUI バージョン 2024.02.0 をリリース	PCUI バージョン 2024.02.0 をリリース 変更: • Lambda ランタイム環境を Python v3.9 に更新 変更点の詳細については、GitHub で aws-parallelcluster-ui パッケージの CHANGELOG ファイルを参照してください。	2024 年 2 月 8 日

変更	説明	日付
PCUI バージョン 2023.12.0 をリリース	PCUI バージョン 2023.12.0 をリリースしました。 機能: • プライベートネットワークを使用した PCUI デプロイのサポートを追加しました。 • PCUI および PCAPI インフラストラクチャで作成したすべての IAM ロールに、必要に応じてアクセス許可の境界を適用できるようにしました • PCUI および PCAPI インフラストラクチャで作成したすべての IAM ロールに、必要に応じてプレフィックスを適用できるようにしました。 • ウィザードで機能パリティなしで ParallelCluster バージョン 3.8.0 のサポートを追加しました。 変更点の詳細については、GitHub に掲載されている aws-parallelcluster-ui パッケージの `CHANGELOG` ファイルを参照してください。	2023 年 12 月 21 日

変更	説明	日付
PCUI バージョン 2023.10.0 をリリース	PCUI バージョン 2023.10.0 をリリースしました。 機能: • ウィザードでの機能パリティを FSx File Cache および複数のインスタンスタイプとのメモリベースのスケジューリング互換性に限定した、ParallelCluster 3.7.2 のサポートを追加しました。 バグ修正: • PCUI に Cost Explorer を操作するアクセス許可がない場合に UI エラーが発生する問題を修正しました。 改良点 • アクセストークンの TTL を 10 分から 5 分に短縮して、セキュリティを向上しました。 変更点の詳細については、GitHub に掲載されている aws-parallelcluster-ui パッケージの `CHANGELOG` ファイルを参照してください。	2023 年 10 月 20 日

変更	説明	日付
PCUI バージョン 2023.06.0 をリリース	PCUI バージョン 2023.06.0 をリリースしました。 変更: • デフォルトの AWS ParallelCluster API バージョンを 3.6.0 にアップグレードしました。 バグ修正: • AWS GovCloud (米国西部) リージョンのデプロイの破損を修正しました。 • 分割パネルで、作成開始後にクラスターの詳細が正しく読み込まれるようになりました。 注記: • コストモニタリング機能は、 AWS GovCloud (米国) リージョンでは使用できません。 変更点の詳細については、GitHub に掲載されている aws-parallelcluster-ui パッケージの `CHANGELOG` ファイルを参照してください。	2023 年 6 月 7 日

変更	説明	日付
PCUI バージョン 2023.05.0 をリリース	PCUI バージョン 2023.05.0 をリリースしました。 機能強化: • AWS ParallelCluster バージョン 3.6.0 以降では、RHEL 8 のサポートを追加します。 • クラスターコストモニタリングを追加します。 • AWS ParallelCluster バージョン 3.6.0 以降では、キューとコンピューティングリソースのクォータを増やします。 変更: • クラスター作成ウィザードのユーザーインターフェイスが改善されました。 • PCUI のデプロイ速度が改善されました。 • 新しいユーザーを追加するためのインターフェイスが改善されました。 • キューはデフォルトでヘッドノードサブネットにあります。 バグ修正:	2023 年 5 月 16 日

変更	説明	日付
	• クラスターの作成が完了したら、正しいリージョンに切り替えます。 • 「クラスターの編集」機能のローディングインジケーター表示を修正しました。 • EBS SnapshotId プロパティが削除された場合のクラスター作成を修正しました。 変更点の詳細については、GitHub に掲載されている aws-parallelcluster-ui パッケージの `CHANGELOG` ファイルを参照してください。	

変更	説明	日付
PCUI バージョン 2023.04.0 をリリース	PCUI バージョン 2023.04.0 をリリースしました。 機能強化: • クラスター作成ウィザードの再設計。 • クラスターログページの再設計。 • 共有ストレージにカスタム名設定を追加します。 • クラスターにストレージを追加するときに、複数のストレージを選択できます。 • Amazon EFS と FSx for Lustre で `DeletionPolicy` のサポートを追加します。 • クラスター設定に `ImdsSupport` 設定を追加します。 • C7 インスタンスタイプのサポートを追加します。 • チュートリアル「以前の AWS Systems Manager ドキュメントバージョンに戻す」を追加しました。 変更: • クラスター設定 YAML のサイズは最大 1 MB。	2023 年 4 月 17 日

変更	説明	日付
	• Boto3 IAM の一時認証情報による認可でも、ユーザーはログアウトされません。 • HPC インスタンスが選択されているときのマルチスレッドオプションが無効になりました。 • クラスター作成ページのロールバックの無効化が削除されました。 • 必要な情報を提供するまで、ユーザーは PCUI を使用できません。 • 最大 10 のキューを追加できます。 • PCUI のインストール中に `SSM-SessionManager RunShell` ドキュメントは上書きされません。 バグ修正: • 壊れたパスワードリセットリンクを修正しました。 • `EcrPrivateRepository` が空でないことが原因で `delete stack` が壊れてしまう問題を修正 • [複数ユーザー管理プロパティ] セクションの [SSH キーを生成] チェックボックスの初期化の問題を修正しました。	

変更	説明	日付
	• 未定義のプロパティを持つジョブが原因でクラッシュする問題を修正しました。 • SCRATCH FSx の設定を修正しました。 • [インスタンスの開始] ボタンと [インスタンスの停止] ボタンが 1 回クリックされた後も有効になる問題を修正しました。 変更点の詳細については、GitHub に掲載されている aws-parallelcluster-ui パッケージの `CHANGELOG` ファイルを参照してください。	

Terraform

変更	説明	日付
Terraform Provider for AWS ParallelCluster 1.1.0 をリリース	バグ修正: • ParallelCluster API 3.11.x を使用してログインノードを持つクラスターをデプロイするときに terraform-apply が失敗する問題を修正しました。	2024 年 12 月 6 日
Terraform Module for AWS ParallelCluster 1.1.0 をリリース	変更: • すべてのモジュール例で AWS ParallelCluster	2024 年 12 月 6 日

変更	説明	日付
	Terraform Provider 1.x を使用します。 • スタック名 ParallelCluster APIのすべての例で ParallelClusterAPI 3.11.1 を使用します。 • すべてのモジュール例でログインノードをデプロイします。	
Terraform Provider for AWS ParallelCluster 1.0.0 をリリース	機能: • 完全な変更ログ	2024 年 6 月 26 日
Terraform Module for AWS ParallelCluster 1.0.0 をリリース	機能: • 完全な変更ログ	2024 年 6 月 26 日

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