



ユーザーガイド

AWS PCS



AWS PCS: ユーザーガイド

Copyright © 2026 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

PCS AWS とは	1
概念	1
PCS AWS の使用を開始する	3
前提条件	4
にサインアップ AWS して管理ユーザーを作成する	5
PCS AWS AWS CLI 用の をインストールする	7
必要な IAM 許可	7
の使用 CloudFormation	8
VPC とサブネットを作成する	8
クラスター VPC のデフォルトのセキュリティグループを検索する	9
セキュリティグループを作成する	10
セキュリティグループを作成する	10
クラスターを作成する	11
Amazon EFS で共有ストレージを作成する	12
FSx for Lustre で共有ストレージを作成する	13
コンピューティングノードグループを作成する	14
インスタンスプロファイルを作成する	15
Create launch templates	16
ログインノードのコンピューティングノードグループを作成する	18
ジョブのコンピューティングノードグループを作成する	19
キューを作成する	20
クラスターに接続する	21
クラスター環境を調べる	22
ユーザーの変更	22
共有ファイルシステムの使用	23
Slurm を操作する	23
単一ノードジョブを実行する	24
Slurm でマルチノード MPI ジョブを実行する	26
AWS リソースを削除する	29
CloudFormation と PCS AWS の使用を開始する	32
CloudFormation を使用してクラスターを作成する	32
クラスターに接続する	34
クラスターのクリーンアップ	35
PCS AWS 用の CloudFormation テンプレートの一部	35

ヘッダー	36
メタデータ	36
パラメータ	37
Mappings	39
リソース	39
アウトプット	43
サンプルクラスターを作成するためのテンプレート	44
クラスター	46
クラスターの作成	46
前提条件	47
PCS AWS クラスターを作成する	47
クラスターの更新	51
クラスター更新の利点	51
サポートされている設定の変更	52
制限	52
クラスター更新の前提条件	52
プロセスとジョブの影響を更新する	53
更新中の請求	53
クラスターの更新	53
よくある質問	55
トラブルシューティング	56
クラスターの削除	57
PCS AWS クラスターを削除する際の考慮事項	58
クラスターを削除する	58
クラスターサイズ	59
クラスターシークレット	60
AWS Secrets Manager を使用してクラスターシークレットを検索する	60
PCS AWS を使用してクラスターシークレットを検索する	61
Slurm クラスターシークレットを取得する	62
シークレットローテーション	63
コンピューティングノードグループ	68
コンピューティングノードグループの作成	68
前提条件	69
PCS AWS でコンピューティングノードグループを作成する	69
コンピューティングノードグループの更新	75
AWS PCS コンピューティングノードグループを更新するためのオプション	75

PCS AWS コンピューティングノードグループを更新する際の考慮事項	75
AWS PCS コンピューティングノードグループを更新するには	77
コンピューティングノードグループの削除	79
コンピューティングノードグループを削除する際の考慮事項	79
コンピューティングノードグループを削除する	79
コンピューティングノードグループの詳細を取得する	80
コンピューティングノードグループインスタンスの検索	84
起動テンプレートの使用	86
概要:	86
基本的な起動テンプレートを作成する	88
Amazon EC2 ユーザーデータの使用	90
例: パッケージリポジトリからソフトウェアをインストールする	92
例: S3 バケットからスクリプトを実行する	92
例: グローバル環境変数の設定	94
例: EFS ファイルシステムを共有ホームディレクトリとして使用する	94
キャパシティ予約	96
PCS での ODCRs AWS の使用	96
キャパシティブロック	99
便利な起動テンプレートパラメータ	105
詳細な CloudWatch モニタリングを有効にする	105
インスタンスメタデータサービスバージョン 2 (IMDS v2)	105
キュー	107
キューの作成	107
前提条件	107
PCS AWS でキューを作成するには	108
キューの更新	110
PCS AWS キューを更新する際の考慮事項	110
PCS AWS キューを更新するには	110
キューの削除	112
キューを削除する際の考慮事項	112
キューの削除	112
ログインノード	114
ログインにコンピューティングノードグループを使用する	114
ログインノードの AWS PCS コンピューティングノードグループの作成	114
ログインノードの PCS AWS コンピューティングノードグループの更新	115
ログインノードの AWS PCS コンピューティングノードグループの削除	115

スタンドアロンインスタンスをログインノードとして使用する	116
ステップ 1 – ターゲット PCS AWS クラスターのアドレスとシークレットを取得する	116
ステップ 2 – EC2 インスタンスを起動する	118
ステップ 3 – インスタンスに Slurm をインストールする	119
ステップ 4 – クラスターシークレットを取得して保存する	119
ステップ 5 – PCS AWS クラスターへの接続を設定する	120
ステップ 6 – (オプション) 接続をテストする	122
スタンドアロンログインノードを複数のクラスターに接続する	122
前提条件	123
スクリプトコード	125
スクリプトの使用	133
ネットワーク	136
VPC およびサブネットの要件	136
VPC の要件と考慮事項	136
サブネットの要件と考慮事項	137
VPC を作成する	139
前提条件	139
Amazon VPC を作成する	140
セキュリティグループ	142
セキュリティグループの要件	142
複数のネットワークインターフェイス	144
プレイスメントグループ	145
Elastic Fabric Adapter (EFA) の使用	146
EFA 対応 EC2 インスタンスを特定する	147
EFA 通信をサポートするセキュリティグループを作成する	148
(オプション) プレイスメントグループを作成する	149
EC2 起動テンプレートを作成または更新する	149
EFA のコンピューティングノードグループを作成または更新する	150
(オプション) EFA をテストする	151
(オプション) CloudFormation テンプレートを使用して EFA 対応の起動テンプレートを作成する	153
ネットワークファイルシステム	155
ネットワークファイルシステムの使用に関する考慮事項	155
ネットワークマウントの例	156
Amazon マシンイメージ (AMI)	161
サンプル AMIs	161

現在の AWS PCS サンプル AMIs	162
AWS PCS サンプル AMIs の詳細	163
PCS と互換性のある独自の AMIs AWS を構築する	163
カスタム AMI	163
ステップ 1 – 一時インスタンスを起動する	164
ステップ 2 – AWS PCS エージェントをインストールする	165
ステップ 3 – Slurm をインストールする	168
ステップ 4 – (オプション) 追加のドライバー、ライブラリ、アプリケーションソフトウェア をインストールする	171
ステップ 5 – PCS と互換性のある AMI AWS を作成する	172
ステップ 6 – PCS コンピューティングノードグループでカスタム AMI AWS を使用する ...	172
ステップ 7 – 一時インスタンスを終了する	174
AMIs を構築するためのインストーラ	175
AWS PCS エージェントソフトウェアインストーラ	175
Slurm インストーラ	175
サポートされるオペレーティングシステム	177
サポートされるインスタンスタイプ	177
サポートされている Slurm バージョン	177
チェックサムを使用してインストーラを検証する	177
AMIs リリースノート	184
x86_64 (AL2) AMIs	184
Arm64 (AL2) AMIs	187
サポートされるオペレーティングシステム	191
AWS PCS エージェントバージョン	193
Slurm	197
Slurm バージョン	197
PCS でサポートされている Slurm AWS バージョン	198
PCS でサポートされていない Slurm AWS バージョン	199
リリースノート	199
よくある質問	201
Slurm アカウンティング	204
アカウンティング設定の変更	205
主要なコンセプト	205
既存の PCS AWS クラスターのアカウンティング設定を取得する	207
Slurm REST API	207
一般的なユースケース	207

要件と制限	208
REST API を有効にする	209
REST API 認証	211
REST API を使用する	215
REST API に関するよくある質問	217
Slurm の再起動	220
Slurm の再起動の利点	220
Slurm の再起動を使用するタイミング	220
制限事項	220
コンピューティングノードを再起動する	221
再起動をキャンセルする	222
よくある質問	223
トラブルシューティング	225
カスタム Slurm 設定	226
カスタム Slurm 設定の利点	226
カスタム設定の構成	226
検証とエラー処理	227
制限事項	228
[クラスター設定]	228
コンピューティングノードグループ設定	230
キュー設定	231
トラブルシューティング	231
SPANK プラグイン	233
SPANK プラグインをインストールする	233
SPANK プラグインを設定する	234
SPANK プラグインに関するよくある質問	235
Slurm CLI フィルタープラグイン	236
要件	236
制限とセキュリティ上の考慮事項	236
CLI フィルタープラグインを設定する	237
Amazon S3 を使用して CLI フィルタープラグインスクリプトをデプロイする	240
ジョブ送信プラグインスクリプトを翻訳する	242
よくある質問	243
トラブルシューティング	245
セキュリティ	247
データ保護	248

保管中の暗号化	249
転送中の暗号化	249
キー管理	250
ネットワーク間トラフィックのプライバシー	250
API トラフィックの暗号化	251
データトラフィックの暗号化	251
暗号化された EBS ボリュームの KMS キーポリシー	251
VPC インターフェイスエンドポイント (AWS PrivateLink)	257
考慮事項	258
インターフェイスエンドポイントの作成	258
エンドポイントポリシーを作成する	259
Identity and Access Management	260
オーディエンス	260
アイデンティティを使用した認証	261
ポリシーを使用したアクセスの管理	262
AWS Parallel Computing Service と IAM の連携方法	264
アイデンティティベースのポリシーの例	269
AWS マネージドポリシー	273
サービスリンクロール	275
EC2 スポットロール	277
最小アクセス許可	278
インスタンスプロファイル	285
トラブルシューティング	289
コンプライアンス検証	291
耐障害性	292
インフラストラクチャセキュリティ	292
脆弱性分析と管理	293
サービス間の混乱した代理の防止	294
コンピューティングノードグループの一部としてプロビジョニングされた Amazon EC2 イ ンスタンスの IAM ロール	295
セキュリティのベストプラクティス	296
AMI 関連のセキュリティ	296
Slurm Workload Manager のセキュリティ	296
モニタリングとログ記録	297
ネットワークセキュリティ	297
ログ記録とモニタリング	298

ジョブ完了ログ	298
前提条件	299
ジョブ完了ログの設定	300
ジョブ完了ログの検索方法	302
ジョブ完了ログフィールド	302
ジョブ完了ログの例	306
スケジューラログ	309
前提条件	309
スケジューラログの設定	310
スケジューラログストリームのパスと名前	312
スケジューラログレコードの例	313
CloudWatch によるモニタリング	313
メトリクスのモニタリング	314
インスタンスのモニタリング	315
CloudTrail ログ	323
AWS CloudTrail の PCS 情報	324
PCS からの CloudTrail AWS ログファイルエントリについて	325
エンドポイントと Service Quotas	328
サービスエンドポイント	328
Service Quotas	331
内部クォータ	332
他の AWS サービスの関連するクォータ	332
トラブルシューティング	334
再起動後に EC2 インスタンスが終了して置き換えられる	334
PCS AWS のコンピューティングノードのブートストラップと登録に関する問題のトラブル シューティング	335
Slurm が PCS AWS で機能する仕組み	336
インスタンスログを取得する	337
インスタンス ID から VPC/Subnet/Securityグループを取得する	338
ノード登録の問題	339
Slurm クラスター結合の問題	341
ドキュメント履歴	345
AWS 用語集	371
.....	ccclxxii

AWS Parallel Computing Service とは

AWS Parallel Computing Service (AWS PCS) は、ハイパフォーマンスコンピューティング (HPC) ワークロードの実行とスケーリングを容易にし、Slurm AWS を使用して科学モデルとエンジニアリングモデルを構築できるマネージドサービスです。AWS PCS を使用して、クラス最高のコンピューティング、ストレージ、ネットワーク、視覚化を統合する AWS コンピューティングクラスターを構築します。シミュレーションを実行したり、科学モデルやエンジニアリングモデルを構築したりします。組み込みの管理機能とオブザーバビリティ機能を使用して、クラスターオペレーションを合理化および簡素化します。使い慣れた環境でアプリケーションやジョブを実行できるようにすることで、ユーザーが研究とイノベーションに集中できるようにします。

トピック

- [PCS AWS の概念](#)

PCS AWS の概念

PCS のクラスターには、少なくとも 1 AWS つのコンピューティングノードグループに関連付けられた 1 つ以上のキューがあります。ジョブはキューに送信され、コンピューティングノードグループで定義された EC2 インスタンスで実行されます。これらの基盤を使用して、高度な HPC アーキテクチャを実装できます。

クラスター

クラスターは、リソースを管理し、ワークロードを実行するためのリソースです。クラスターは、コンピューティング、ネットワーク、ストレージ、アイデンティティ、ジョブスケジューラ設定のアセンブリを定義する AWS PCS リソースです。クラスターを作成するには、使用するジョブスケジューラ (Slurm の現在)、使用するスケジューラ設定、クラスターを管理するサービスコントローラー、クラスターリソースを起動する VPC を指定します。スケジューラはジョブを受け入れてスケジューリングし、それらのジョブを処理するコンピューティングノード (EC2 インスタンス) も起動します。

コンピューティングノードグループ

コンピューティングノードグループは、PCS AWS がジョブを実行したり、クラスターへのインタラクティブなアクセスを提供したりするために使用されるコンピューティングノードのコレクションです。コンピューティングノードグループを定義するときは、Amazon EC2 インスタンスタイプ、最小インスタンス数と最大インスタンス数、ターゲット VPC サブネット、Amazon マシンイメージ

(AMI)、購入オプション、カスタム起動設定などの一般的な特性を指定します。AWS PCS は、これらの設定を使用して、コンピューティングノードグループ内のコンピューティングノードを効率的に起動、管理、および終了します。

キュー

特定のクラスターでジョブを実行する場合は、特定のキュー (パーティションとも呼ばれます) に送信します。ジョブは、AWS PCS がコンピューティングノードグループで実行するようにスケジュールするまでキューに残ります。1 つ以上のコンピューティングノードグループを各キューに関連付けます。キューは、ジョブスケジューラが提供するさまざまなスケジューリングポリシーを使用して、基盤となるコンピューティングノードグループリソースでジョブをスケジュールおよび実行するために必要です。ユーザーは、コンピューティングノードまたはコンピューティングノードグループに直接ジョブを送信しません。

システム管理者

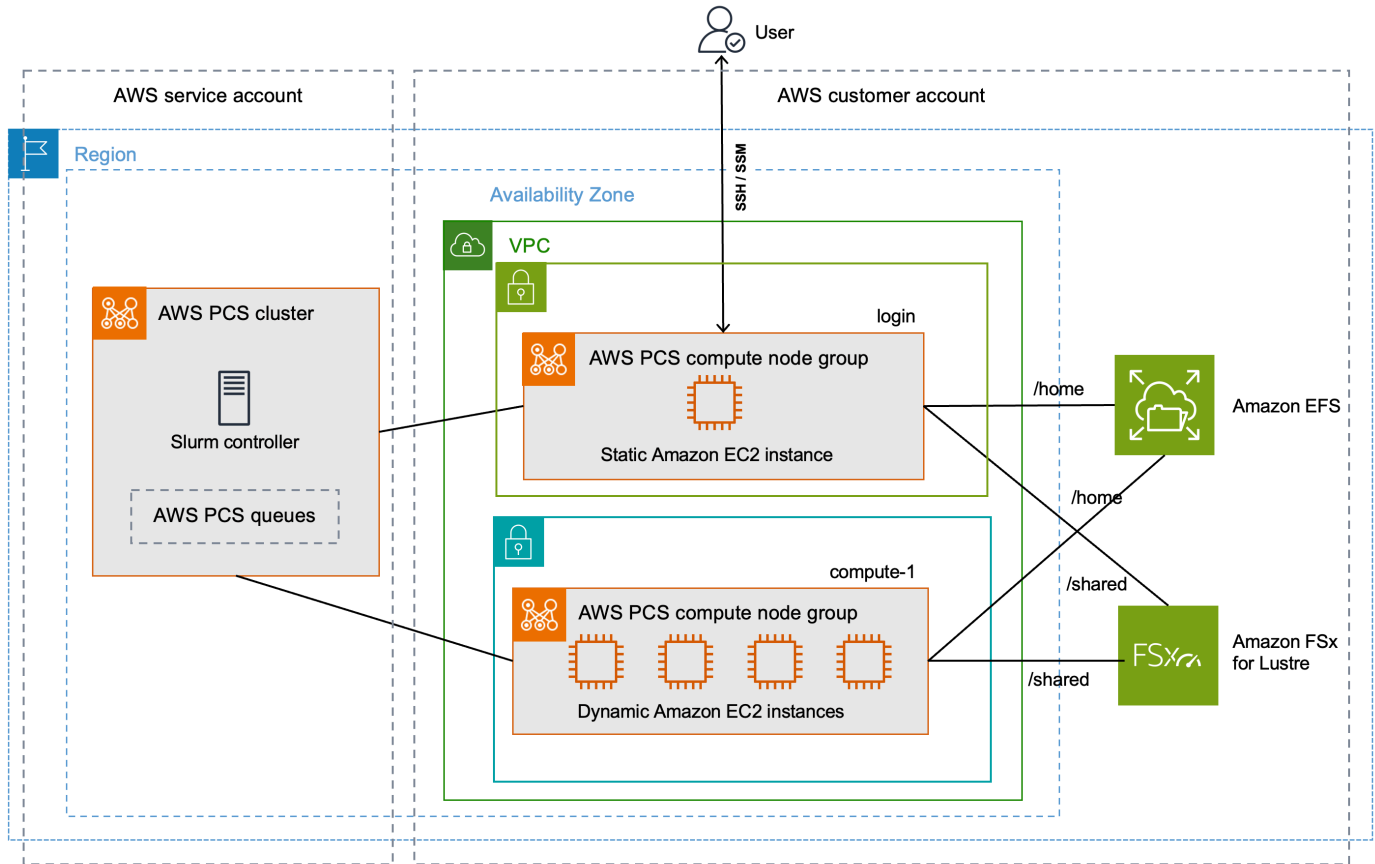
システム管理者は、クラスターをデプロイ、維持、運用します。PCS には、AWS PCS AWS マネジメントコンソール AWS API、および AWS SDK を介してアクセスできます。SSH 経由で、または管理タスクの実行 AWS Systems Manager、ジョブの実行、データの管理、その他のシェルベースのアクティビティの実行が可能な特定のクラスターにアクセスできます。詳細については、[AWS Systems Manager ドキュメント](#) を参照してください。

エンドユーザー

エンドユーザーには、クラスターをデプロイまたは運用する day-to-day 責任はありません。ターミナルインターフェイス (SSH など) を使用して、クラスターリソースへのアクセス、ジョブの実行、データの管理、その他のシェルベースのアクティビティを実行します。

AWS Parallel Computing Service の使用を開始する

これは、PCS AWS を試すために使用できるシンプルなクラスターを作成するためのチュートリアルです。次の図は、クラスターの設計を示しています。



チュートリアルクラスター設計には、次の主要なコンポーネントがあります。

- [AWS PCS ネットワーク要件を満たす](#) VPC とサブネット。
- Amazon EFS ファイルシステム。共有ホームディレクトリとして使用されます。
- Amazon FSx for Lustre ファイルシステム。共有ハイパフォーマンスディレクトリを提供します。
- Slurm AWS コントローラーを提供する PCS クラスター。
- 2 AWS PCS コンピューティングノードグループ。
 - システムへのシェルベースのインタラクティブアクセスを提供するloginノードグループ。
 - compute-1 ノードグループは、ジョブを実行するために伸縮自在にスケーリングされたインスタンスを提供します。
- compute-1 ノードグループの EC2 インスタンスにジョブを送信する 1 つのキュー。

クラスターには、セキュリティグループ、IAM ロール、EC2 起動テンプレートなどの追加の AWS リソースが必要ですが、図には示されていません。

Note

このトピックのコマンドラインステップは、Bash シェルで完了することをお勧めします。Bash シェルを使用していない場合、行継続文字や、変数の設定と使用に関する方法など、一部のスクリプトコマンドのためにシェルの調整が必要となります。さらに、シェルの引用規則とエスケープ規則は異なる場合があります。詳細については、[「バージョン 2 ユーザーガイド」の「の文字列を含む引用符とリテラル AWS CLI」](#)を参照してください。AWS Command Line Interface

トピック

- [PCS AWS の使用を開始するための前提条件](#)
- [PCS AWS チュートリアル AWS CloudFormation での使用](#)
- [PCS の VPC AWS とサブネットを作成する](#)
- [PCS AWS のセキュリティグループを作成する](#)
- [PCS AWS でクラスターを作成する](#)
- [Amazon Elastic File System で AWS PCS の共有ストレージを作成する Amazon Elastic File System](#)
- [Amazon FSx for Lustre で AWS PCS の共有ストレージを作成する](#)
- [PCS AWS でコンピューティングノードグループを作成する](#)
- [AWS PCS でジョブを管理するキューを作成する](#)
- [PCS AWS クラスターに接続する](#)
- [PCS AWS でクラスター環境を調べる](#)
- [PCS AWS で単一ノードジョブを実行する](#)
- [PCS で Slurm を使用してマルチノード MPI AWS ジョブを実行する](#)
- [PCS AWS の AWS リソースを削除する](#)

PCS AWS の使用を開始するための前提条件

AWS PCS 用の AWS アカウント およびローカル開発環境を準備するには、以下のトピックを参照してください。

トピック

- [にサインアップ AWS して管理ユーザーを作成する](#)
- [PCS AWS AWS CLI 用の をインストールする](#)
- [PCS AWS に必要な IAM アクセス許可](#)

にサインアップ AWS して管理ユーザーを作成する

Parallel Computing Service (AWS PCS) AWS を設定するには、次のタスクを実行します。

トピック

- [にサインアップする AWS アカウント](#)
- [管理アクセスを持つユーザーを作成する](#)

にサインアップする AWS アカウント

がない場合は AWS アカウント、次の手順を実行して作成します。

にサインアップするには AWS アカウント

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

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

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

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

管理アクセスを持つユーザーを作成する

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

を保護する AWS アカウントのルートユーザー

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

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

2. ルートユーザーの多要素認証 (MFA) を有効にします。

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

管理アクセスを持つユーザーを作成する

1. IAM アイデンティティセンターを有効にします。

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

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

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

管理アクセス権を持つユーザーとしてサインインする

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

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

追加のユーザーにアクセス権を割り当てる

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

手順については、「AWS IAM アイデンティティセンター ユーザーガイド」の「[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 アイデンティティセンター ユーザーガイド」の「[Add groups](#)」を参照してください。

PCS AWS AWS CLI 用の をインストールする

の最新バージョンを使用する必要があります AWS CLI。詳細については、「[バージョン 2 ユーザーガイド](#)」の「[の最新バージョンのインストールまたは更新 AWS CLI](#)」を参照してください。

を設定する必要があります AWS CLI。詳細については、「バージョン 2 ユーザーガイド」の「[の設定 AWS CLI](#)」を参照してください。 AWS Command Line Interface

コマンドプロンプトで次のコマンドを入力してを確認します AWS CLI。ヘルプ情報が表示されます。

```
aws pcs help
```

PCS AWS に必要な IAM アクセス許可

使用している IAM セキュリティプリンシパルには、PCS IAM AWS ロール、サービスにリンクされたロール、VPC AWS CloudFormation、および関連リソースを操作するためのアクセス許可が必要です。詳細については、「AWS Identity and Access Management ユーザーガイド」の[AWS 並列コンピューティングサービスの Identity and Access Management](#)「」および「[サービスにリンクされたロールの作成](#)」を参照してください。このガイドのすべてのステップは1つのユーザーとして実行する必要があります。現在のユーザーを確認するには次のコマンドを実行してください：

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

PCS AWS チュートリアル AWS CloudFormation での の使用

AWS PCS チュートリアルには多くのステップがあり、PCS AWS クラスターの部分とその作成に必要な手順を理解するのに役立ちます。チュートリアルのステップを少なくとも 1 回実行することをお勧めします。関係する内容を十分に理解できたら、AWS CloudFormation を使用して自動化によりサンプルクラスターをすばやく作成できます。

CloudFormation は、AWS インフラストラクチャのデプロイを予測どおりに繰り返し作成およびプロビジョニングできる AWS サービスです。CloudFormation テンプレートを使用して、サンプルクラスターの AWS リソースをスタックと呼ばれる単一のユニットとして自動的にプロビジョニングできます。スタックは、完了したら削除できます。

詳細については、「[CloudFormation と PCS AWS の使用を開始する](#)」を参照してください。

PCS の VPC AWS とサブネットを作成する

CloudFormation テンプレートを使用して VPC とサブネットを作成できます。次の URL を使用して CloudFormation テンプレートをダウンロードし、[CloudFormation コンソール](#)でテンプレートをアップロードして新しい CloudFormation スタックを作成します。詳細については、「AWS CloudFormation ユーザーガイド」の「[CloudFormation コンソールの使用](#)」を参照してください。

```
https://aws-hpc-recipes.s3.amazonaws.com/main/recipes/net/hpc_large_scale/assets/main.yaml
```

CloudFormation コンソールでテンプレートを開いた状態で、次のオプションを入力します。テンプレートで指定されたデフォルト値を使用できます。

- スタック名を指定します。
 - スタック名に次のように入力します。

```
hpc-networking
```

- パラメータの下:
 - VPC の下:
 - CidrBlock で、次のように入力します。

```
10.3.0.0/16
```

- サブネット A の下:

- CidrPublicSubnetA で、次のように入力します。

10.3.0.0/20

- CidrPrivateSubnetA で、次のように入力します。

10.3.128.0/20

- サブネット B の下:

- CidrPublicSubnetB で、次のように入力します。

10.3.16.0/20

- CidrPrivateSubnetB で、次のように入力します。

10.3.144.0/20

- サブネット C の下:

- ProvisionSubnetsC で、True を選択します。
- CidrPublicSubnetC で、次のように入力します。

10.3.32.0/20

- CidrPrivateSubnetC で、次のように入力します。

10.3.160.0/20

- 機能:

- が IAM リソースを作成する AWS CloudFormation 可能性があることを確認する のチェックボックスをオンにします。

CloudFormation スタックのステータスをモニタリングします。に達したらCREATE_COMPLETE、新しい VPC でデフォルトのセキュリティグループの ID を見つけます。ID はチュートリアルの後半で使用します。

クラスター VPC のデフォルトのセキュリティグループを検索する

新しい VPC でデフォルトのセキュリティグループの ID を検索するには、次の手順に従います。

クラスター VPC のデフォルトのセキュリティグループを検索する

- [\[Amazon VPC コンソール\]](#) に移動します。
- VPC ダッシュボードで、VPC でフィルタリングを選択します。
 - 名前が で始まる VPC を選択します hpc-networking。
 - Security で、Security groups を選択します。
- という名前のグループのセキュリティグループ ID を見つけます default。という説明があります default VPC security group。後で ID を使用して EC2 起動テンプレートを設定します。

PCS AWS のセキュリティグループを作成する

AWS PCS は、セキュリティグループに依存して、クラスターとそのコンピューティングノードグループとの間のネットワークトラフィックを管理します。このトピックの詳細については、「」を参照してください [セキュリティグループの要件と考慮事項](#)。

このステップでは、CloudFormation テンプレートを使用して 2 つのセキュリティグループを作成します。

- AWS PCS コントローラー、コンピューティングノード、ログインノード間の通信を可能にするクラスターセキュリティグループ。
- インバウンド SSH セキュリティグループ。オプションでログインノードに追加して SSH アクセスをサポートできます。

PCS AWS のセキュリティグループを作成する

CloudFormation テンプレートを使用して、セキュリティグループを作成できます。次の URL を使用して CloudFormation テンプレートをダウンロードし、[CloudFormation コンソール](#) でテンプレートをアップロードして新しい CloudFormation スタックを作成します。詳細については、「[ユーザーガイド](#)」の「[CloudFormation コンソールの使用](#)」を参照してください。AWS CloudFormation

```
https://aws-hpc-recipes.s3.amazonaws.com/main/recipes/pcs/getting_started/assets/pcs-cluster-sg.yaml
```

AWS CloudFormation コンソールでテンプレートを開いた状態で、次のオプションを入力します。一部のオプションはテンプレートに事前入力されるため、デフォルト値のままにしておくことができます。

- スタック名を指定する

- スタック名に次のように入力します。

```
getstarted-sg
```

- パラメータの下
 - VpcId で、名前が で始まる VPC を選択します hpc-networking。
 - (オプション) ClientIpCidr で、インバウンド SSH セキュリティグループのより制限の厳しい IP 範囲を入力します。これは、独自の IP/サブネット (独自の IP の場合は x.x.x.x/32、範囲の場合は x.x.x.x/24 に制限することをお勧めします。 x.x.x.x を独自の PUBLIC IP に置き換えます。
<https://ifconfig.co/> などのツールを使用してパブリック IP を取得できます)

CloudFormation スタックのステータスをモニタリングします。セキュリティグループリソースに到達するCREATE_COMPLETEと、リソースの準備が整います。

2 つのセキュリティグループが作成され、その名前が付けられます。

- cluster-getstarted-sg – これはクラスターセキュリティグループです
- inbound-ssh-getstarted-sg – これは、インバウンド SSH アクセスを許可するセキュリティグループです。

PCS AWS でクラスターを作成する

AWS PCS では、クラスターはリソースを管理し、ワークロードを実行するための永続的なリソースです。新規または既存の VPC のサブネットで、特定のスケジューラ (AWS PCS は現在 Slurm をサポートしています) のクラスターを作成します。クラスターはジョブを受け入れてスケジュールし、それらのジョブを処理するコンピューティングノード (EC2 インスタンス) を起動します。

クラスターを作成するには

1. [AWS PCS コンソール](#)を開き、クラスターの作成を選択します。
2. クラスターの詳細セクションで、次のフィールドに入力します。
 - クラスター名 – Enter get-started
 - スケジューラ – Slurm バージョン 25.05 を選択する
 - コントローラーサイズ – スモールを選択
3. ネットワークセクションで、次のフィールドの値を選択します。

- VPC – という名前の VPC を選択します。hpc-networking:Large-Scale-HPC
 - サブネット – 名前が始まるサブネットを選択します。hpc-networking:PrivateSubnetA
 - セキュリティグループ – という名前のクラスターセキュリティグループを選択します。
cluster-getstarted-sg
4. [クラスターを作成] を選択します。

 Note

ステータスフィールドには、クラスターのプロビジョニング中に作成中と表示されます。クラスターの作成には数分かかる場合があります。

Amazon Elastic File System で AWS PCS の共有ストレージを作成する Amazon Elastic File System

Amazon Elastic File System (Amazon EFS) は、サーバーレスで完全に伸縮自在なファイルストレージを提供する AWS サービスです。これにより、ストレージの容量とパフォーマンスをプロビジョニングまたは管理することなく、ファイルデータを共有できます。詳細については、Amazon Elastic File System ユーザーガイドの「[Amazon Elastic File System とは](#)」を参照してください。

AWS PCS デモンストレーションクラスターは、EFS ファイルシステムを使用して、クラスターノード間で共有ホームディレクトリを提供します。クラスターと同じ VPC に EFS ファイルシステムを作成します。

Amazon EFS ; ファイルシステムを作成するには

1. [Amazon EFS コンソール](#)に移動します。
2. AWS PCS を試 AWS リージョン 場所と同じに設定されていることを確認します。
3. ファイルシステムを作成する を選択します。
4. ファイルシステムの作成ページで、次のパラメータを設定します。
 - [名前] に「getstarted-efs」と入力します。
 - Virtual Private Cloud (VPC) で、 という名前の VPC を選択します。hpc-networking:Large-Scale-HPC
 - [Create] (作成) を選択します。これにより、ファイルシステムページに戻ります。

5. `getstarted-efs` ファイルシステムのファイルシステム ID を書き留めます。この情報は後で使用します。

Amazon FSx for Lustre で AWS PCS の共有ストレージを作成する

Amazon FSx for Lustre を使用すると、人気のある高性能 Lustre ファイルシステムを簡単に、費用対効果の高い方法で起動して実行できます。機械学習、ハイパフォーマンスコンピューティング (HPC)、ビデオ処理、財務モデリングなど、速度が重要なワークロードには Lustre を使用します。詳細については、[「Amazon FSx for Lustre ユーザーガイド」の「Amazon FSx for Lustre とは」](#)を参照してください。 FSx

AWS PCS デモンストレーションクラスターは、FSx for Lustre ファイルシステムを使用して、クラスターノード間に高性能な共有ディレクトリを提供できます。クラスターと同じ VPC に FSx for Lustre ファイルシステムを作成します。

FSx for Lustre ファイルシステムを作成するには

1. [Amazon FSx コンソール](#)に移動します。
2. コンソールがクラスター AWS リージョン と同じ を使用するように設定されていることを確認します。
3. ファイルシステムを作成する を選択します。
 - ファイルシステムタイプの選択 で、Amazon FSx for Lustre を選択し、次へ を選択します。
4. ファイルシステムの詳細を指定ページで、次のパラメータを設定します。
 - ファイルシステムの詳細の下
 - [名前] に「`getstarted-fsx`」と入力します。
 - デプロイとストレージタイプで、永続、SSD を選択します。
 - ストレージ単位あたりのスループットについては、125 MB/s/TiB を選択します。
 - ストレージ容量には、1.2 TiB と入力します。
 - メタデータ設定 で、自動 を選択します。
 - データ圧縮タイプで、LZ4 を選択します。
 - ネットワークとセキュリティの下
 - Virtual Private Cloud (VPC) で、 という名前の VPC を選択します。 `hpc-networking:Large-Scale-HPC`

- VPC セキュリティグループの場合、セキュリティグループは のままにします。 default
 - Subnet で、名前が で始まるサブネットを選択します。 hpc-networking:PrivateSubnetA
 - 他のオプションはデフォルト値のままにしておきます。
 - [Next (次へ)] を選択します。
5. 確認と作成ページで、ファイルシステムの作成を選択します。これにより、ファイルシステムページに戻ります。
 6. 作成した FSx for Lustre ファイルシステムの詳細ページに移動します。
 7. ファイルシステム ID とマウント名を書き留めます。この情報は後で使用します。

Note

ステータス フィールドには、ファイルシステムのプロビジョニング中に作成中と表示されます。ファイルシステムの作成には数分かかる場合があります。完了するまで待ってから、チュートリアルに残りの部分に進みます。

PCS AWS でコンピューティングノードグループを作成する

コンピューティングノードグループは、PCS が起動して管理するコンピューティングノード (EC2 インスタンス) AWS の仮想コレクションです。コンピューティングノードグループを定義するときは、EC2 インスタンスタイプ、最小インスタンス数と最大インスタンス数、ターゲット VPC サブネット、優先購入オプション、カスタム起動設定などの一般的な特性を指定します。AWS PCS は、これらの設定に従って、コンピューティングノードグループ内のコンピューティングノードを効率的に起動、管理、終了します。デモンストレーションクラスターは、コンピューティングノードグループを使用してユーザーアクセス用のログインノードを提供し、別のコンピューティングノードグループを使用してジョブを処理します。以下のトピックでは、クラスターでこれらのコンピューティングノードグループを設定する手順について説明します。

トピック

- [PCS AWS のインスタンスプロファイルを作成する](#)
- [PCS AWS の起動テンプレートを作成する](#)
- [AWS PCS でログインノードのコンピューティングノードグループを作成する](#)

- [AWS PCS でコンピューティングジョブを実行するためのコンピューティングノードグループを作成する](#)

PCS AWS のインスタンスプロファイルを作成する

コンピューティングノードグループの作成時にインスタンスプロファイルが必要です。AWS マネジメントコンソール を使用して Amazon EC2 のロールを作成する場合、コンソールはインスタンスプロファイルを自動的に作成し、そのインスタンスプロファイルにロールと同じ名前を付けます。詳細については、「AWS Identity and Access Management ユーザーガイド」の「[インスタンスプロファイルの使用](#)」を参照してください。

次の手順では、AWS マネジメントコンソール を使用して Amazon EC2 のロールを作成します。これにより、コンピューティングノードグループのインスタンスプロファイルも作成されます。

ロールとインスタンスプロファイルを作成するには

- [IAM console](#) (IAM コンソール) に入ります。
- [アクセス管理] で、[ポリシー] を選択します。
 - [Create policy] (ポリシーの作成) を選択します。
 - アクセス許可の指定で、ポリシーエディタで JSON を選択します。
 - テキストエディタの内容を以下に置き換えます。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "pcs:RegisterComputeNodeGroupInstance"
      ],
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}
```

- [次へ] を選択します。
- 「確認と作成」で、ポリシー名に「」と入力しますAWSPCS-getstarted-policy。

- [Create policy] (ポリシーの作成) を選択します。
- [Access management] (アクセス管理) で、[Roles] (ロール) を選択します。
- [ロールの作成] を選択してください。
- 信頼されたエンティティの選択:
 - 信頼されたエンティティタイプで、AWS サービスを選択します。
 - ユースケースで、EC2 を選択します。
 - 次に、指定されたサービスのユースケースを選択するで、EC2 を選択します。
- [次へ] を選択します。
- アクセス許可の追加:
 - アクセス許可ポリシーで、AWSPCS-getstarted-policy を検索します。
 - AWSPCS-getstarted-policy の横にあるチェックボックスをオンにして、ロールに追加します。
 - アクセス許可ポリシーで、AmazonSSMManagedInstanceCore を検索します。
 - AmazonSSMManagedInstanceCore の横にあるチェックボックスをオンにして、ロールに追加します。
 - [次へ] を選択します。
- 「名前」、「確認」、「作成」:
 - ロールの詳細の下:
 - [Role name] (ロール名) にAWSPCS-getstarted-roleと入力します。
 - [ロールの作成] を選択してください。

PCS AWS の起動テンプレートを作成する

コンピューティングノードグループを作成するときは、PCS が起動する EC2 AWS インスタンスの設定に使用する EC2 起動テンプレートを指定します。これには、インスタンスの起動時に実行されるセキュリティグループやスクリプトなどの設定が含まれます。

このステップでは、1 つの CloudFormation テンプレートを使用して 2 つの EC2 起動テンプレートを作成します。1 つのテンプレートはログインノードの作成に使用され、もう 1 つのテンプレートはコンピューティングノードの作成に使用されます。主な違いは、インバウンド SSH アクセスを許可するようにログインノードを設定できることです。

CloudFormation テンプレートにアクセスする

次の URL を使用して CloudFormation テンプレートをダウンロードし、[CloudFormation コンソール](#)でテンプレートをアップロードして新しい CloudFormation スタックを作成します。詳細については、「[AWS CloudFormation ユーザーガイド](#)」の「[CloudFormation コンソールの使用](#)」を参照してください。

```
https://aws-hpc-recipes.s3.amazonaws.com/main/recipes/pcs/getting_started/assets/pcs-lt-efs-fsx1.yaml
```

CloudFormation テンプレートを使用して EC2 起動テンプレートを作成する

CloudFormation コンソールで CloudFormation テンプレートを完了するには、次の手順に従います。

- スタック名を指定します。
 - スタック名に と入力しますgetstarted-lt。
- パラメータの下:
 - セキュリティの下
 - VpcSecurityGroupId で、クラスター VPC default内の という名前のセキュリティグループを選択します。
 - ClusterSecurityGroupId で、 という名前のグループを選択します。cluster-getstarted-sg
 - SshSecurityGroupId の場合は、 という名前のグループを選択します。inbound-ssh-getstarted-sg
 - SshKeyName で、任意の SSH キーペアを選択します。
 - ファイルシステムの下
 - EfsFileSystemId には、チュートリアルの前半で作成した EFS ファイルシステムのファイルシステム ID を入力します。
 - FSxLustreFileSystemId には、チュートリアルの前半で作成した FSx for Lustre ファイルシステムのファイルシステム ID を入力します。
 - FSxLustreFileSystemMountName には、同じ FSx for Lustre ファイルシステムのマウント名を入力します。
- Next を選択し、もう一度 Next を選択します。
- [Submit] を選択してください。

CloudFormation スタックのステータスをモニタリングします。起動テンプレートに到達するCREATE_COMPLETEと、使用する準備が整います。

Note

CloudFormation テンプレートが作成したすべてのリソースを表示するには、[CloudFormation コンソール](#)を開きます。getstarted-1t スタックを選択し、[リソース] タブを選択します。

AWS PCS でログインノードのコンピューティングノードグループを作成する

コンピューティングノードグループは、PCS が起動して管理するコンピューティングノード (EC2 インスタンス) AWS の仮想コレクションです。コンピューティングノードグループを定義するときは、EC2 インスタンスタイプ、最小インスタンス数と最大インスタンス数、ターゲット VPC サブネット、優先購入オプション、カスタム起動設定などの一般的な特性を指定します。AWS PCS は、これらの設定に従って、コンピューティングノードグループ内のコンピューティングノードを効率的に起動、管理、終了します。

このステップでは、クラスターへのインタラクティブなアクセスを提供する静的コンピューティングノードグループを起動します。SSH または Amazon EC2 Systems Manager (SSM) を使用してログインし、シェルコマンドを実行して Slurm ジョブを管理できます。

コンピューティングノードグループを作成するには

- [AWS PCS コンソール](#)を開き、クラスターに移動します。
- という名前のクラスターを選択する get-started
- Compute ノードグループに移動し、作成を選択します。
- Compute ノードグループのセットアップセクションで、以下を指定します。
 - コンピューティングノードグループ名 – と入力しますlogin。
- コンピューティング設定で、次の値を入力または選択します。
 - EC2 起動テンプレート – 名前が である起動テンプレートを選択します。login-getstarted-1t
 - IAM インスタンスプロファイル – という名前のインスタンスプロファイルを選択します。AWSPCS-getstarted-role

- サブネット – 名前が で始まるサブネットを選択しますhpc-networking:PublicSubnetA。
- インスタンス – を選択しますc6i.xlarge。
- スケーリング設定 – 最小インスタンス数には、 と入力します1。最大インスタンス数には、 と入力します1。
- 追加設定で、以下を指定します。
 - AMI ID – 次の形式の名前を持つ、使用する AMI を選択します。

```
aws-pcs-sample_ami-amzn2-platform-slurm-version
```

サンプル AMIs 「」を参照してください[PCS でサンプル Amazon マシンイメージ \(AMIs\) AWS を使用する](#)。

- コンピューティングノードグループの作成 を選択します。

Status フィールドには、コンピューティングノードグループのプロビジョニング中に Create と表示されます。チュートリアルのは進行中は、次のステップに進むことができます。

AWS PCS でコンピューティングジョブを実行するためのコンピューティングノードグループを作成する

このステップでは、クラスターに送信されたジョブを実行するために伸縮自在にスケールするコンピューティングノードグループを起動します。

コンピューティングノードグループを作成するには

- [AWS PCS コンソール](#)を開き、クラスターに移動します。
- という名前のクラスターを選択する get-started
- Compute ノードグループに移動し、作成を選択します。
- Compute ノードグループのセットアップセクションで、以下を指定します。
 - コンピューティングノードグループ名 – と入力しますcompute-1。
- コンピューティング設定で、次の値を入力または選択します。
 - EC2 起動テンプレート – 名前が である起動テンプレートを選択します。 compute-getstarted-1t
 - IAM インスタンスプロファイル – という名前のインスタンスプロファイルを選択します。 AWSPCS-getstarted-role

- サブネット – 名前が で始まるサブネットを選択しますhpc-networking:PrivateSubnetA。
- インスタンス – を選択しますc6i.xlarge。
- スケーリング設定 – 最小インスタンス数には、 と入力します0。最大インスタンス数には、 と入力します4。
- 追加設定で、以下を指定します。
 - AMI ID – 次の形式の名前を持つ、使用する AMI を選択します。

```
aws-pcs-sample_ami-amzn2-platform-slurm-version
```

サンプル AMIs 「」を参照してください[PCS でサンプル Amazon マシンイメージ \(AMIs\) AWS を使用する](#)。

- コンピューティングノードグループの作成 を選択します。

Status フィールドには、コンピューティングノードグループのプロビジョニング中に作成中と表示されます。

Important

Status フィールドがアクティブになるまで待ってから、このチュートリアル次のステップに進みます。

AWS PCS でジョブを管理するキューを作成する

ジョブをキューに送信して実行します。ジョブは、PCS AWS がコンピューティングノードグループで実行するようにスケジュールするまでキューに残ります。各キューは、1 つ以上のコンピューティングノードグループに関連付けられ、処理を実行するために必要な EC2 インスタンスを提供します。

このステップでは、コンピューティングノードグループを使用してジョブを処理するキューを作成します。

キューを作成するには

- [AWS PCS コンソール](#)を開きます。
- get-started という名前のクラスターを選択します。

- Compute ノードグループに移動し、compute-1グループのステータスがアクティブであることを確認します。

 Important

次のステップに進む前に、compute-1グループのステータスがアクティブになっている必要があります。

- キューに移動し、キューの作成を選択します。
 - キュー設定セクションで、次の値を指定します。
 - キュー名 – 次のように入力します。 demo
 - コンピューティングノードグループ – という名前のコンピューティングノードグループを選択しますcompute-1。
- [キューの作成]を選択します。

ステータスフィールドには、キューの作成中に の作成 と表示されます。

 Important

Status フィールドがアクティブになるまで待ってから、このチュートリアル次のステップに進みます。

PCS AWS クラスターに接続する

login コンピューティングノードグループのステータスがアクティブになったら、作成した EC2 インスタンスに接続できます。

ログインノードに接続するには

- [AWS PCS コンソール](#)を開き、クラスターに移動します。
- get-started という名前のクラスターを選択します。
- Compute ノードグループを選択します。
- という名前のコンピューティングノードグループに移動しますlogin。
- Compute ノードグループ ID を見つけます。
- 別のブラウザウィンドウまたはタブで、[Amazon EC2 コンソール](#)を開きます。

- [Instances] を選択します。
- 次のタグが付いた EC2 インスタンスを検索します。*node-group-id* を前のステップの Compute ノードグループ ID の値に置き換えます。インスタンスが 1 つあるはずです。

```
aws:pcs:compute-node-group-id=node-group-id
```

- EC2 インスタンスに接続します。Session Manager または SSH を使用できます。

Session Manager

- インスタンスを選択してください。
- [接続]を選択してください。
- インスタンスに接続するで、セッションマネージャーを選択します。
- [接続]を選択してください。
- [接続]を選択してください。インタラクティブターミナルがブラウザで起動します。

SSH

- インスタンスを選択してください。
- [接続]を選択してください。
- インスタンスに接続する で、SSH クライアントを選択します。
- コンソールの指示に従います。

Note

インスタンスのユーザー名は **ec2-user**ではありません **root**。

PCS AWS でクラスター環境を調べる

クラスターにログインしたら、シェルコマンドを実行できます。例えば、ユーザーの変更、共有ファイルシステムのデータの操作、Slurm の操作を行うことができます。

ユーザーの変更

Session Manager を使用してクラスターにログインしている場合は、として接続されている可能性があります **ssm-user**。これは、Session Manager 用に作成された特別なユーザーです。次のコマンドを使用して、Amazon Linux 2 のデフォルトユーザーに切り替えます。SSH を使用して接続している場合は、これを行う必要はありません。


```
sudo su - ec2-user
```

共有ファイルシステムの使用

コマンドを使用して、EFS ファイルシステムと FSx for Lustre ファイルシステムが使用可能であることを確認できます `df -h`。クラスターの出力は次のようになります。

```
[ec2-user@ip-10-3-6-103 ~]$ df -h
Filesystem                Size      Used Avail Use% Mounted on
devtmpfs                   3.8G         0   3.8G   0% /dev
tmpfs                      3.9G         0   3.9G   0% /dev/shm
tmpfs                      3.9G   556K   3.9G   1% /run
tmpfs                      3.9G         0   3.9G   0% /sys/fs/cgroup
/dev/nvme0n1p1             24G       18G   6.6G  73% /
127.0.0.1:/                8.0E         0   8.0E   0% /home
10.3.132.79@tcp:/z1shxbev  1.2T    7.5M   1.2T   1% /shared
tmpfs                      780M         0   780M   0% /run/user/0
tmpfs                      780M         0   780M   0% /run/user/1000
```

/home ファイルシステムは 127.0.0.1 をマウントし、非常に大きな容量を持ちます。これは、チュートリアル前半で作成した EFS ファイルシステムです。ここで書き込まれたファイルは、クラスター内のすべてのノード/homeの で使用できます。

/shared ファイルシステムはプライベート IP をマウントし、容量は 1.2 TB です。これは、チュートリアル前半で作成した FSx for Lustre ファイルシステムです。ここで書き込まれたファイルは、クラスター内のすべてのノード/sharedの で使用できます。

Slurm を操作する

トピック

- [キューとノードを一覧表示する](#)
- [ジョブの表示](#)

キューとノードを一覧表示する

を使用して、キューとそれらが関連付けられているノードを一覧表示できます `sinfo`。クラスターからの出力は次のようになります。

```
[ec2-user@ip-10-3-6-103 ~]$ sinfo
```

```

PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
demo          up    infinite      4  idle~ compute-1-[1-4]
[ec2-user@ip-10-3-6-103 ~]$

```

という名前のパーティションを書き留めますdemo。ステータスは upで、最大4つのノードがあります。これは、ノードグループのcompute-1ノードに関連付けられます。コンピューティングノードグループを編集し、インスタンスの最大数を8に増やす8と、ノードの数は を読み取り、ノードリストは を読み取りますcompute-1-[1-8]。4つのノードtestを持つ という名前の2番目のコンピューティングノードグループを作成し、demoキューに追加した場合、それらのノードもノードリストに表示されます。

ジョブの表示

を使用して、システム上の任意の状態のすべてのジョブを一覧表示できますsqueue。クラスターからの出力は次のようになります。

```

[ec2-user@ip-10-3-6-103 ~]$ squeue
JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)

```

Slurm ジョブが保留中または実行中になったら、後でsqueueもう一度実行してみてください。

PCS AWS で単一ノードジョブを実行する

Slurm を使用してジョブを実行するには、ジョブ要件を指定する送信スクリプトを準備し、 sbatch コマンドを使用してキューに送信します。通常、これは共有ディレクトリから行われるため、ログインノードとコンピューティングノードにはファイルにアクセスするための共通スペースがあります。

クラスターのログインノードに接続し、シェルプロンプトで次のコマンドを実行します。

- デフォルトユーザーになります。共有ディレクトリに変更します。

```

sudo su - ec2-user
cd /shared

```

- 次のコマンドを使用して、サンプルジョブスクリプトを作成します。

```

cat << EOF > job.sh
#!/bin/bash
#SBATCH -J single
#SBATCH -o single.%j.out

```

```
#SBATCH -e single.%j.err

echo "This is job \${SLURM_JOB_NAME} [\${SLURM_JOB_ID}] running on \
\${SLURMD_NODENAME}, submitted from \${SLURM_SUBMIT_HOST}" && sleep 60 && echo "Job
complete"
EOF
```

- ジョブスクリプトを Slurm スケジューラに送信します。

```
sbatch -p demo job.sh
```

- ジョブが送信されると、ジョブ ID が数値として返されます。その ID を使用してジョブのステータスを確認します。次のコマンドの *job-id* を、 から返された数値に置き換えます sbatch。

```
squeue --job job-id
```

Example

```
squeue --job 1
```

squeue コマンドは、次のような出力を返します。

```
JOBID PARTITION NAME USER      ST TIME NODES NODELIST(REASON)
1      demo      test ec2-user CF 0:47 1      compute-1
```

- (R実行中) ステータスに達するまで、ジョブのステータスを引き続き確認します。ジョブは、squeueが何も返さないときに行われます。
- /shared ディレクトリの内容を確認します。

```
ls -alth /shared
```

コマンドの出力は次のようになります。

```
-rw-rw-r- 1 ec2-user ec2-user 107 Mar 19 18:33 single.1.out
-rw-rw-r- 1 ec2-user ec2-user 0 Mar 19 18:32 single.1.err
-rw-rw-r- 1 ec2-user ec2-user 381 Mar 19 18:29 job.sh
```

single.1.out および という名前のファイルは、クラスターのコンピューティングノードの 1 つによってsingle.1.err書き込まれました。ジョブは共有ディレクトリ (/shared) で実行された

ため、ログインノードでも使用できます。そのため、このクラスターに FSx for Lustre ファイルシステムを設定しました。

- `single.1.out` ファイルの内容を確認します。

```
cat /shared/single.1.out
```

出力は次の例のようになります。

```
This is job test [1] running on compute-1, submitted from ip-10-3-13-181
Job complete
```

PCS で Slurm を使用してマルチノード MPI AWS ジョブを実行する

これらの手順は、Slurm を使用して AWS PCS でメッセージパスインターフェイス (MPI) ジョブを実行する方法を示しています。

ログインノードのシェルスプロンプトで次のコマンドを実行します。

- デフォルトユーザーになります。をホームディレクトリに変更します。

```
sudo su - ec2-user
cd ~/
```

- C プログラミング言語でソースコードを作成します。

```
cat > hello.c << EOF
// * mpi-hello-world - https://www.mpitutorial.com
// Released under MIT License
//
// Copyright (c) 2014 MPI Tutorial.
//
// 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, subject to the following conditions:
// The above copyright notice and this permission notice shall be included in
```

```
// all copies or substantial portions of the Software.
//
// 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.

#include <mpi.h>
#include <stdio.h>
#include <stddef.h>

int main(int argc, char** argv) {
    // Initialize the MPI environment. The two arguments to MPI Init are not
    // currently used by MPI implementations, but are there in case future
    // implementations might need the arguments.
    MPI_Init(NULL, NULL);

    // Get the number of processes
    int world_size;
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);

    // Get the rank of the process
    int world_rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);

    // Get the name of the processor
    char processor_name[MPI_MAX_PROCESSOR_NAME];
    int name_len;
    MPI_Get_processor_name(processor_name, &name_len);

    // Print off a hello world message
    printf("Hello world from processor %s, rank %d out of %d processors\n",
           processor_name, world_rank, world_size);

    // Finalize the MPI environment. No more MPI calls can be made after this
    MPI_Finalize();
}
EOF
```

- OpenMPI モジュールをロードします。

```
module load openmpi
```

- C プログラムをコンパイルします。

```
mpicc -o hello hello.c
```

- Slurm ジョブ送信スクリプトを記述します。

```
cat > hello.sh << EOF
#!/bin/bash
#SBATCH -J multi
#SBATCH -o multi.out
#SBATCH -e multi.err
#SBATCH --exclusive
#SBATCH --nodes=4
#SBATCH --ntasks-per-node=1

srun $HOME/hello
EOF
```

- 共有ディレクトリに変更します。

```
cd /shared
```

- ジョブスクリプトを送信します。

```
sbatch -p demo ~/hello.sh
```

- ジョブが完了するまでジョブをモニタリング `squeue` するには、 `squeue` を使用します。
- の内容を確認します `multi.out`。

```
cat multi.out
```

出力は以下のようになります。異なるノードで実行されたため、各ランクには独自の IP アドレスがあることに注意してください。

```
Hello world from processor ip-10-3-133-204, rank 0 out of 4 processors
Hello world from processor ip-10-3-128-219, rank 2 out of 4 processors
Hello world from processor ip-10-3-141-26, rank 3 out of 4 processors
Hello world from processor ip-10-3-143-52, rank 1 out of 4 processor
```

PCS AWS の AWS リソースを削除する

このチュートリアル用に作成したクラスターとノードグループが完了したら、作成したリソースを削除する必要があります。

Important

で実行されているすべてのリソースに対して料金が発生します。AWS アカウント

このチュートリアル用に作成した AWS PCS リソースを削除するには

- [AWS PCS コンソール](#)を開きます。
- get-started という名前のクラスターに移動します。
- キューセクションに移動します。
- demo という名前のキューを選択します。
- [削除] を選択します。

Important

キューが削除されるまで待ってから続行します。

- Compute ノードグループセクションに移動します。
- compute-1 という名前のコンピューティングノードグループを選択します。
- [削除] を選択します。
- login という名前のコンピューティングノードグループを選択します。
- [削除] を選択します。

Important

続行する前に、両方のコンピューティングノードグループが削除されるまで待ちます。

- 開始するクラスターの詳細ページで、削除を選択します。

 Important

クラスターが削除されるまで待ってから、以降のステップに進みます。

このチュートリアル用に作成した他の AWS リソースを削除するには

- [IAM コンソール](#)を開きます。
 - [ロール] を選択します。
 - AWSPCS-getstarted-role という名前のロールを選択し、削除を選択します。
 - ロールを削除したら、ポリシーを選択します。
 - AWSPCS-getstarted-policy という名前のポリシーを選択し、削除を選択します。
- [CloudFormation コンソール](#)を開きます。
 - getstarted-It という名前のスタックを選択します。
 - [削除] を選択します。

 Important

スタックが削除されるまで待ってから次に進みます。

- [Amazon EFS コンソール](#)を開きます。
 - [File Systems (ファイルシステム)] を選択します。
 - getstarted-efs という名前のファイルシステムを選択します。
 - [削除] を選択します。

 Important

ファイルシステムが削除されるまで待ってから次に進みます。

- [Amazon FSx コンソール](#)を開きます。
 - [File Systems (ファイルシステム)] を選択します。
 - getstarted-fsx という名前のファイルシステムを選択します。
 - [削除] を選択します。

 Important

ファイルシステムが削除されるまで待ってから次に進みます。

- [CloudFormation コンソール](#)を開きます。
 - getstarted-sg という名前のスタックを選択します。
 - [削除] を選択します。
- [CloudFormation コンソール](#)を開きます。
 - hpc-networking という名前のスタックを選択します。
 - [削除] を選択します。

CloudFormation と PCS AWS の使用を開始する

AWS CloudFormation を使用して PCS AWS クラスターを作成できます。CloudFormation を使用すると、AWS インフラストラクチャのデプロイを予測どおりに繰り返し作成およびプロビジョニングできます。CloudFormation を使用して、多くの AWS サービスからリソースを自動的にプロビジョニングし、基盤となる AWS インフラストラクチャを作成および設定 AWS クラウドすることなく、信頼性、スケーラビリティ、費用対効果の高いアプリケーションを構築できます。CloudFormation では、テンプレートファイルを使用して、スタックと呼ばれる 1 つのユニットとしてリソースのコレクションを作成および削除できます。詳細については CloudFormation、「AWS CloudFormation ユーザーガイド」の「[とは CloudFormation](#)」を参照してください。の AWS PCS リソースタイプの詳細については CloudFormation、AWS CloudFormation「ユーザーガイド」の[AWS「PCS リソースタイプのリファレンス](#)」を参照してください。

トピック

- [CloudFormation を使用してサンプル PCS AWS クラスターを作成する](#)
- [で作成された AWS PCS クラスターに接続する CloudFormation](#)
- [で PCS AWS クラスターをクリーンアップする CloudFormation](#)
- [PCS AWS 用の CloudFormation テンプレートの一部](#)
- [CloudFormation サンプル PCS AWS クラスターを作成するための テンプレート](#)

CloudFormation を使用してサンプル PCS AWS クラスターを作成する

次の手順では、で CloudFormation テンプレート AWS マネジメントコンソール を使用してサンプル PCS AWS クラスターを作成します。詳細については CloudFormation、「AWS CloudFormation ユーザーガイド」の「[What is CloudFormation?](#)」を参照してください。の AWS PCS リソースタイプの詳細については CloudFormation、AWS CloudFormation「ユーザーガイド」の[AWS「PCS リソースタイプのリファレンス](#)」を参照してください。

サンプルクラスターを作成するには

1. AWS リージョン を選択して、でクラスターを作成します (リンクは テンプレートを使用して CloudFormation コンソールを開きます)。
 - [米国東部 \(バージニア北部\)](#) (us-east-1)

- [米国東部 \(オハイオ\) \(us-east-2\)](#)
 - [米国西部 \(オレゴン\) \(us-west-2\)](#)
 - [アジアパシフィック \(シンガポール\) \(ap-southeast-1\)](#)
 - [アジアパシフィック \(シドニー\) \(ap-southeast-2\)](#)
 - [アジアパシフィック \(東京\) \(ap-northeast-1\)](#)
 - [欧州 \(フランクフルト\) \(eu-central-1\)](#)
 - [欧州 \(アイルランド\) \(eu-west-1\)](#)
 - [欧州 \(ロンドン\) \(eu-west-2\)](#)
 - [欧州 \(ストックホルム\) \(eu-north-1\)](#)
 - [AWS GovCloud \(米国東部\) \(us-gov-east-1\)](#)
 - [AWS GovCloud \(米国西部\) \(us-gov-west-1\)](#)
2. スタック名を指定する にわかりやすい名前を入力します。これは CloudFormation スタックの名前です。テンプレートは、この値を PCS AWS クラスターの名前として使用します。
 3. パラメータの下:
 - a. SlurmVersion で、クラスターで使用する Slurm のバージョンを選択します。
 - b. NodeArchitecture で、x86 を選択して x86_64 互換インスタンスを使用するクラスターをデプロイするか、Graviton を選択して Arm64 インスタンスを使用します。
 - c. KeyName で、クラスターログインノードにアクセスする SSH キーペアを選択します。選択したキーペアの PEM ファイルがあることを確認します。
 - d. ClientIpcidr の場合は、CIDR 形式で IP 範囲を入力して、ログインノードへのアクセスを制御します。
-  **Warning**

のデフォルト値では、すべての IP アドレスからのアクセス 0.0.0.0/0 が許可されます。
- e. HpcRecipesS3Bucket と HpcRecipesBranch の値はデフォルト値のままにします。
4. 機能および変換:
 - a. チェックボックスをオンにして、 が CloudFormation IAM リソースを作成することを確認します。

- b. チェックボックスをオンにして、が CloudFormation カスタム名で IAM リソースを作成することを確認します。
 - c. チェックボックスをオンにして、新しいスタックCAPABILITY_AUTO_EXPANDを承認します。詳細については、AWS CloudFormation API リファレンスの[CreateStack](#)」を参照してください。
5. [スタックの作成] を選択してください。
 6. スタックのステータスをモニタリングします。スタックのステータスが になったら、クラスターに接続できますCREATE_COMPLETE。

で作成された AWS PCS クラスターに接続する CloudFormation

CloudFormation テンプレートから AWS PCS クラスターを作成したら、PCS AWS コンソール (内 AWS マネジメントコンソール) を使用してクラスターを管理できます。クラスターのログインノードの 1 つに接続して、クラスターの管理、ジョブの実行、データの管理を行うこともできます。CloudFormation スタックには、クラスターへの接続に使用できるリンクが用意されています。

クラスターに接続するには

1. [CloudFormation コンソール](#)を開きます。
2. 作成したスタックを選択します。
3. スタックの出力タブを選択します。

スタックには次のリンクがあります。

- PcsConsoleUrl — このリンクを選択すると、クラスターを選択した状態で AWS PCS コンソールが開きます。これを使用して、クラスター、ノードグループ、キューの設定を調べることができます。
- Ec2ConsoleUrl — このリンクを選択して Amazon EC2 コンソールを開き、クラスターのログインノードグループが管理するインスタンスを表示するようにフィルタリングします。

このビューから、インスタンスを選択し、Connect を選択できます。サンプルクラスターのインスタンスは、ウェブブラウザでのインバウンド SSH と AWS Systems Manager 接続をサポートしています。詳細については、「[PCS AWS クラスターに接続する](#)」を参照してください。

ログインインスタンスに接続したら、「」のチュートリアルに従います[PCS AWS でクラスター環境を調べる](#)。

で PCS AWS クラスターをクリーンアップする CloudFormation

CloudFormation を使用して AWS PCS クラスターを作成した場合は、[CloudFormation コンソール](#)を開き、スタックを削除して、クラスターおよび関連するすべてのリソースを削除できます。

Important

サンプルクラスターで、クラスターに追加のコンピューティングノードグループまたはキューを作成した場合 (サンプル CloudFormation テンプレートが作成した login および compute-1 グループ以外)、CloudFormation スタックを削除する前に [AWS PCS コンソール](#) または AWS CLI を使用してそれらのリソースを削除する必要があります。詳細については、「[PCS AWS でのクラスターの削除](#)」を参照してください。

PCS AWS 用の CloudFormation テンプレートの一部

CloudFormation テンプレートには、それぞれが特定の目的を果たす 1 つ以上のセクションがあります。は、テンプレート内の標準形式、構文、言語 CloudFormation を定義します。詳細については、「AWS CloudFormation ユーザーガイド」の「[CloudFormation テンプレートの使用](#)」を参照してください。

CloudFormation テンプレートは高度にカスタマイズ可能であるため、形式は異なる場合があります。CloudFormation テンプレートの必要な部分を理解して PCS AWS クラスターを作成するには、提供されているサンプルテンプレートを調べてサンプルクラスターを作成することをお勧めします。このトピックでは、そのサンプルテンプレートのセクションを簡単に説明します。

Important

このトピックのコードサンプルは完了していません。省略記号 ([...]) は、表示されない追加のコードがあることを示します。完全な YAML 形式の CloudFormation テンプレートをダウンロードするには、「」を参照してください [CloudFormation サンプル PCS AWS クラスターを作成するための テンプレート](#)。

目次

- [ヘッダー](#)
- [メタデータ](#)

- [パラメータ](#)
- [Mappings](#)
- [リソース](#)
- [アウトプット](#)

ヘッダー

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: AWS Parallel Computing Service "getting started" cluster
```

AWSTemplateFormatVersion は、テンプレートが準拠するテンプレート形式バージョンを識別します。詳細については、「AWS CloudFormation ユーザーガイド」の[CloudFormation テンプレート形式のバージョン構文](#)を参照してください。

Transform は、CloudFormation がテンプレートの処理に使用するマクロを指定します。詳細については、「AWS CloudFormation ユーザーガイド」の[CloudFormation テンプレート変換](#)セクションを参照してください。AWS::Serverless-2016-10-31 変換により、CloudFormation は AWS Serverless Application Model (AWS SAM) 構文で記述されたテンプレート进行处理できます。詳細については、「AWS CloudFormation ユーザーガイド」の[AWS::Serverless「変換」](#)を参照してください。

メタデータ

```
### Stack metadata
Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
      - Label:
          default: PCS Cluster configuration
        Parameters:
          - SlurmVersion
          - ManagedAccounting
          - AccountingPolicyEnforcement
      - Label:
          default: PCS ComputeNodeGroups configuration
        Parameters:
          - NodeArchitecture
          - KeyName
```

```
- ClientIpCidr
- Label:
  default: HPC Recipes configuration
Parameters:
  - HpcRecipesS3Bucket
  - HpcRecipesBranch
```

CloudFormation テンプレートの metadata セクションには、テンプレート自体に関する情報が表示されます。サンプルテンプレートは、PCS を使用する完全なハイパフォーマンズコンピューティング (HPC) AWS クラスターを作成します。サンプルテンプレートのメタデータセクションは、が対応するスタック CloudFormation を起動 (プロビジョニング) する方法を制御するパラメータを宣言します。アーキテクチャの選択 (NodeArchitecture)、Slurm バージョン (SlurmVersion)、アクセスコントロール (KeyName および) を制御するパラメータがあります ClientIpCidr。

パラメータ

このセクションでは、テンプレートのカスタムパラメータ Parameters を定義します。CloudFormation は、これらのパラメータ定義を使用して、このテンプレートからスタックを起動するときに操作するフォームを構築および検証します。

Parameters:

NodeArchitecture:

Type: String

Default: x86

AllowedValues:

- x86
- Graviton

Description: Processor architecture for the login and compute node instances

SlurmVersion:

Type: String

Default: 25.05

Description: Version of Slurm to use

AllowedValues:

- 24.11
- 25.05

ManagedAccounting:

Type: String

Default: 'disabled'

AllowedValues:

- 'enabled'
- 'disabled'

Description: Monitor cluster usage, manage access control, and enforce resource limits with Slurm accounting. Requires Slurm 24.11 or newer.

AccountingPolicyEnforcement:

Description: Specify which Slurm accounting policies to enforce

Type: String

Default: none

AllowedValues:

- none
- 'associations,limits,safe'

KeyName:

Description: SSH keypair to log in to the head node

Type: AWS::EC2::KeyPair::KeyName

AllowedPattern: ".+" # Required

ClientIpCidr:

Description: IP(s) allowed to access the login node over SSH. We recommend that you restrict it with your own IP/subnet (x.x.x.x/32 for your own ip or x.x.x.x/24 for range. Replace x.x.x.x with your own PUBLIC IP. You can get your public IP using tools such as <https://ifconfig.co/>)

Default: 127.0.0.1/32

Type: String

AllowedPattern: (\d{1,3})\.\(\d{1,3})\.\(\d{1,3})\.\(\d{1,3})/(\d{1,2})

ConstraintDescription: Value must be a valid IP or network range of the form x.x.x.x/x.

HpcRecipesS3Bucket:

Type: String

Default: aws-hpc-recipes

Description: HPC Recipes for AWS S3 bucket

AllowedValues:

- aws-hpc-recipes
- aws-hpc-recipes-dev

HpcRecipesBranch:

Type: String

Default: main

Description: HPC Recipes for AWS release branch

AllowedPattern: '^(?!.*\/\.git\$)(?!.*\/\.)(?!.*\\.\.)[a-zA-Z0-9-_\.\.]+\$'

Mappings

このセクションでは、特定の条件または依存関係に基づいて値を指定するキーと値のペアMappingsを定義します。

```
Mappings:
```

```
  Architecture:
```

```
    AmiArchParameter:
```

```
      Graviton: arm64
```

```
      x86: x86_64
```

```
    LoginNodeInstances:
```

```
      Graviton: c7g.xlarge
```

```
      x86: c6i.xlarge
```

```
    ComputeNodeInstances:
```

```
      Graviton: c7g.xlarge
```

```
      x86: c6i.xlarge
```

リソース

このセクションでは、スタックの一部としてプロビジョニングおよび設定する AWS リソースをResources宣言します。

```
Resources:
```

```
  [...]
```

テンプレートは、サンプルクラスターインフラストラクチャをレイヤーにプロビジョニングします。VPC 設定Networkingの場合は で始まります。ストレージは、共有ストレージEfsStorage用と高性能ストレージFSxLStorage用のデュアルシステムによって提供されます。コアクラスターはを通じて確立されますPCSCluster。

```
Networking:
```

```
  Type: AWS::CloudFormation::Stack
```

```
  Properties:
```

```
    Parameters:
```

```
      ProvisionSubnetsC: "False"
```

```
      TemplateURL: !Sub 'https://${HpcRecipesS3Bucket}.s3.amazonaws.com/  
${HpcRecipesBranch}/recipes/net/hpc_large_scale/assets/main.yaml'
```

```

EfsStorage:
  Type: AWS::CloudFormation::Stack
  Properties:
    Parameters:
      SubnetIds: !GetAtt [ Networking, Outputs.DefaultPrivateSubnet ]
      SubnetCount: 1
      VpcId: !GetAtt [ Networking, Outputs.VPC ]
      TemplateURL: !Sub 'https://${HpcRecipesS3Bucket}.s3.amazonaws.com/
${HpcRecipesBranch}/recipes/storage/efs_simple/assets/main.yaml'

FSxLStorage:
  Type: AWS::CloudFormation::Stack
  Properties:
    Parameters:
      PerUnitStorageThroughput: 125
      SubnetId: !GetAtt [ Networking, Outputs.DefaultPrivateSubnet ]
      VpcId: !GetAtt [ Networking, Outputs.VPC ]
      TemplateURL: !Sub 'https://${HpcRecipesS3Bucket}.s3.amazonaws.com/
${HpcRecipesBranch}/recipes/storage/fsx_lustre/assets/persistent.yaml'

[...]

# Cluster
PCSCluster:
  Type: AWS::PCS::Cluster
  Properties:
    Name: !Sub '${AWS::StackName}'
    Size: SMALL
    Scheduler:
      Type: SLURM
      Version: !Ref SlurmVersion
    Networking:
      SubnetIds:
        - !GetAtt [ Networking, Outputs.DefaultPrivateSubnet ]
      SecurityGroupIds:
        - !GetAtt [ PCSSecurityGroup, Outputs.ClusterSecurityGroupId ]

```

コンピューティングリソースの場合、テンプレートは2つのノードグループを作成します。1つのログインノードPCSClusterLogin用と最大4つのコンピューティングノードPCSClusterCompute用です。これらのノードグループは、アクセス許可とインスタンス設定PCSClusterProfileのために PCSLaunchTemplate でサポートされています。

```

# Compute Node groups
PCSIInstanceProfile:
  Type: AWS::CloudFormation::Stack
  Properties:
    Parameters:
      # We have to regionalize this in case CX use the template in more than one
      region. Otherwise,
      # the create action will fail since instance-role-${AWS::StackName} already
      exists!
      RoleName: !Sub '${AWS::StackName}-${AWS::Region}'
      TemplateURL: !Sub 'https://${HpcRecipesS3Bucket}.s3.amazonaws.com/
${HpcRecipesBranch}/recipes/pcs/getting_started/assets/pcs-iip-minimal.yaml'

PCSLaunchTemplate:
  Type: AWS::CloudFormation::Stack
  Properties:
    Parameters:
      VpcDefaultSecurityGroupId: !GetAtt [ Networking, Outputs.SecurityGroup ]
      ClusterSecurityGroupId: !GetAtt [ PCSSecurityGroup,
Outputs.ClusterSecurityGroupId ]
      SshSecurityGroupId: !GetAtt [ PCSSecurityGroup,
Outputs.InboundSshSecurityGroupId ]
      EfsFileSystemSecurityGroupId: !GetAtt [ EfsStorage, Outputs.SecurityGroupId ]
      FSxLustreFileSystemSecurityGroupId: !GetAtt [ FSxLStorage,
Outputs.FSxLustreSecurityGroupId ]
      SshKeyName: !Ref KeyName
      EfsFileSystemId: !GetAtt [ EfsStorage, Outputs.EFSFileSystemId ]
      FSxLustreFileSystemId: !GetAtt [ FSxLStorage, Outputs.FSxLustreFileSystemId ]
      FSxLustreFileSystemMountName: !GetAtt [ FSxLStorage,
Outputs.FSxLustreMountName ]
      TemplateURL: !Sub 'https://${HpcRecipesS3Bucket}.s3.amazonaws.com/
${HpcRecipesBranch}/recipes/pcs/getting_started/assets/cfn-pcs-lt-efs-fsx1.yaml'

# Compute Node groups - Login Nodes
PCSNODEGroupLogin:
  Type: AWS::PCS::ComputeNodeGroup
  Properties:
    ClusterId: !GetAtt [ PCSCluster, Id]
    Name: login
    ScalingConfiguration:
      MinInstanceCount: 1
      MaxInstanceCount: 1

```

```

    IamInstanceProfileArn: !GetAtt [ PCSInstanceProfile, Outputs.InstanceProfileArn ]
    CustomLaunchTemplate:
      TemplateId: !GetAtt [ PCSLaunchTemplate, Outputs.LoginLaunchTemplateId ]
      Version: 1
    SubnetIds:
      - !GetAtt [ Networking, Outputs.DefaultPublicSubnet ]
    AmiId: !GetAtt [ PcsSampleAmi, AmiId]
    InstanceConfigs:
      - InstanceType: !FindInMap [ Architecture, LoginNodeInstances, !Ref
NodeArchitecture ]

# Compute Node groups - Compute Nodes
PCSNodeGroupCompute:
  Type: AWS::PCS::ComputeNodeGroup
  Properties:
    ClusterId: !GetAtt [ PCSCluster, Id]
    Name: compute-1
    ScalingConfiguration:
      MinInstanceCount: 0
      MaxInstanceCount: 4
    IamInstanceProfileArn: !GetAtt [ PCSInstanceProfile, Outputs.InstanceProfileArn ]
    CustomLaunchTemplate:
      TemplateId: !GetAtt [ PCSLaunchTemplate, Outputs.ComputeLaunchTemplateId ]
      Version: 1
    SubnetIds:
      - !GetAtt [ Networking, Outputs.DefaultPrivateSubnet ]
    AmiId: !GetAtt [ PcsSampleAmi, AmiId]
    InstanceConfigs:
      - InstanceType: !FindInMap [ Architecture, ComputeNodeInstances, !Ref
NodeArchitecture ]

```

ジョブのスケジュールは、 を通じて処理されますPCSQueueCompute。

```

PCSQueueCompute:
  Type: AWS::PCS::Queue
  Properties:
    ClusterId: !GetAtt [ PCSCluster, Id]
    Name: demo
    ComputeNodeGroupConfigurations:
      - ComputeNodeGroupId: !GetAtt [ PCSNodeGroupCompute, Id]

```

AMI の選択は、PcsAMILookupFn Lambda 関数および関連リソースを介して自動的行われます。

```
PcsAMILookupRole:
  Type: AWS::IAM::Role
  [...]

PcsAMILookupFn:
  Type: AWS::Lambda::Function
  Properties:
    Runtime: python3.12
    Handler: index.handler
    Role: !GetAtt PcsAMILookupRole.Arn
    Code:
      [...]
    Timeout: 30
    MemorySize: 128

# Example of using the custom resource to look up an AMI
PcsSampleAmi:
  Type: Custom::AMILookup
  Properties:
    ServiceToken: !GetAtt PcsAMILookupFn.Arn
    OperatingSystem: 'amzn2'
    Architecture: !FindInMap [ Architecture, AmiArchParameter, !Ref
NodeArchitecture ]
    SlurmVersion: !Ref SlurmVersion
```

アウトプット

テンプレートはClusterId、、、PcsConsoleUrlおよび を介してクラスター識別および管理 URLs を出力しますEc2ConsoleUrl。

```
Outputs:
  ClusterId:
    Description: The Id of the PCS cluster
    Value: !GetAtt [ PCSCluster, Id ]

  PcsConsoleUrl:
    Description: URL to access the cluster in the PCS console
    Value: !Sub
      - https://${ConsoleDomain}/pcs/home?region=${AWS::Region}#/clusters/${ClusterId}
```

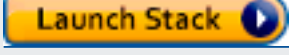
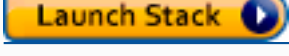
```

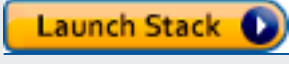
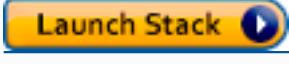
- { ConsoleDomain: !If [ GovCloud, 'console.amazonaws-us-gov.com', !If [ China,
'console.amazonaws.cn', !Sub '${AWS::Region}.console.aws.amazon.com']],
  ClusterId: !GetAtt [ PCSCluster, Id ]
}
Export:
  Name: !Sub ${AWS::StackName}-PcsConsoleUrl

Ec2ConsoleUrl:
  Description: URL to access instance(s) in the login node group via Session Manager
  Value: !Sub
    - https://${ConsoleDomain}/ec2/home?region=
    ${AWS::Region}#Instances:instanceState=running;tag:aws:pcs:compute-node-group-id=
    ${NodeGroupLoginId}
    - { ConsoleDomain: !If [ GovCloud, 'console.amazonaws-us-gov.com', !If [ China,
'console.amazonaws.cn', !Sub '${AWS::Region}.console.aws.amazon.com']],
      NodeGroupLoginId: !GetAtt [ PCSNodeGroupLogin, Id ]
    }
Export:
  Name: !Sub ${AWS::StackName}-Ec2ConsoleUrl

```

CloudFormation サンプル PCS AWS クラスターを作成するためのテンプレート

AWS リージョン 名前	AWS リージョン	ソースの表示	スタックを起動する
米国東部 (バージニア北部)	us-east-1	YAML のダウンロード	
米国東部 (オハイオ)	us-east-2	YAML のダウンロード	
米国西部 (オレゴン)	us-west-2	YAML のダウンロード	
アジアパシフィック (シンガポール)	ap-southeast-1	YAML のダウンロード	
アジアパシフィック (シドニー)	ap-southeast-2	YAML のダウンロード	

AWS リージョン 名前	AWS リージョン	ソースの表示	スタックを起動する
アジアパシフィック (東京)	ap-northeast-1	YAML のダウンロード	
欧州 (フランクフルト)	eu-central-1	YAML のダウンロード	
欧州 (アイルランド)	eu-west-1	YAML のダウンロード	
欧州 (ロンドン)	eu-west-2	YAML のダウンロード	
欧州 (ストックホルム)	eu-north-1	YAML のダウンロード	
AWS GovCloud (米国 東部)	us-gov-east-1	YAML のダウンロード	
AWS GovCloud (米国 西部)	us-gov-west-1	YAML のダウンロード	

AWS PCS クラスター

AWS PCS クラスターは、次のコンポーネントで構成されます。

- Slurm コントロールデーモン () などの HPC システムスケジューラソフトウェアのマネージドインスタンス `slurmctld`。
- HPC システムスケジューラと統合して Amazon EC2 インスタンスをプロビジョニングおよび管理するためのコンポーネント。
- HPC システムスケジューラと統合してログとメトリクスを Amazon CloudWatch に送信するコンポーネント。

これらのコンポーネントは、が管理するアカウントで実行されます AWS。これらは連携して、お客様のアカウントの Amazon EC2 インスタンスを管理します。AWS PCS は、Amazon VPC サブネットに Elastic Network Interface をプロビジョニングして、スケジューラソフトウェアから Amazon EC2 インスタンスへの接続を提供します (たとえば、バッチジョブのスケジュールをサポートし、ユーザーがスケジューラコマンドを実行してそれらのジョブを一覧表示および管理できるようにします)。

トピック

- [PCS AWS でのクラスターの作成](#)
- [PCS AWS でのクラスターの更新](#)
- [PCS AWS でのクラスターの削除](#)
- [PCS AWS のクラスターサイズ](#)
- [PCS AWS でのクラスターシークレットの使用](#)

PCS AWS でのクラスターの作成

このトピックでは、使用可能なオプションの概要と、AWS Parallel Computing Service (AWS PCS) でクラスターを作成するときに考慮すべき点について説明します。AWS PCS クラスターを初めて作成する場合は、に従うことをお勧めします [AWS Parallel Computing Service の使用を開始する](#)。このチュートリアルは、使用可能なすべてのオプションとシステムアーキテクチャを拡張することなく、動作する HPC システムを作成するのに役立ちます。

Note

クラスターを作成したら、インフラストラクチャを再構築することなく、多くの構成設定を変更できます。詳細については、「[PCS AWS でのクラスターの更新](#)」を参照してください。

Note

カスタム Slurm 設定を設定して、高度なスケジューリングポリシーとリソース管理を実装できます。詳細については、「[PCS でのカスタム Slurm AWS 設定の設定](#)」を参照してください。

前提条件

- [AWS PCS ネットワーク](#) 要件を満たす既存の VPC とサブネット。本番用にクラスターをデプロイする前に、VPC とサブネットの要件を十分に理解しておくことをお勧めします。VPC とサブネットを作成するには、「」を参照してください[PCS クラスターの VPC AWS の作成](#)。
- PCS リソースを作成および管理するためのアクセス許可を持つ [IAM プリンシパル](#)。AWS 詳細については、「[AWS 並列コンピューティングサービスの Identity and Access Management](#)」を参照してください。

PCS AWS クラスターを作成する

AWS マネジメントコンソール または を使用してクラスター AWS CLI を作成できます。

AWS マネジメントコンソール

クラスターを作成するには

1. <https://console.aws.amazon.com/pcs/home#/clusters> で AWS PCS コンソールを開き、クラスターの作成を選択します。
2. クラスター設定セクションで、次のフィールドに入力します。
 - クラスター名 – クラスターの名前。この名前には英数字 (大文字と小文字が区別されます) とハイフンのみを使用できます。アルファベット文字で始まり、40 文字を超えることはで

きません。名前は、クラスター AWS アカウント を作成する AWS リージョン および 内で一意である必要があります。

- スケジューラ – スケジューラとバージョンを選択します。詳細については、「[PCS の Slurm AWS バージョン](#)」を参照してください。
- コントローラーサイズ – コントローラーのサイズを選択します。これにより、AWS PCS クラスターで管理できる同時ジョブとコンピューティングノードの数を決定します。コントローラーのサイズは、クラスターの作成時にのみ設定できます。サイズ設定の詳細については、「」を参照してください[PCS AWS のクラスターサイズ](#)。

3. ネットワークセクションで、次のフィールドの値を選択します。

- ネットワークタイプ – クラスターの IP アドレスタイプを選択します。クラスターは IPv4 または IPv6 のいずれかを使用できますが、両方を使用することはできません。VPC とサブネットは同じネットワークアドレスタイプを使用する必要があります。各サブネットに使用する IP アドレスブロックには、少なくとも 1 つの使用可能なアドレスが必要です。は、各サブネットの一部のアドレス AWS を予約します。詳細については、「Amazon VPC ユーザーガイド」の「[サブネット CIDR ブロック](#)」を参照してください。
- VPC – PCS 要件を満たす既存の VPC AWS を選択します。詳細については、「[AWS PCS VPC とサブネットの要件と考慮事項](#)」を参照してください。クラスターを作成した後は、その VPC を変更することはできません。VPCs がリストされていない場合は、最初に VPC を作成する必要があります。
- サブネット – 選択した VPC で使用可能なすべてのサブネットが一覧表示されます。AWS PCS サブネット要件を満たすサブネットを選択します。詳細については、「[AWS PCS VPC とサブネットの要件と考慮事項](#)」を参照してください。スケジューラエンドポイントがパブリックインターネットに公開されないように、プライベートサブネットを選択することをお勧めします。
- セキュリティグループ – PCS AWS がクラスター用に作成するネットワークインターフェイスに関連付けるセキュリティグループ (複数可) を指定します。クラスターとそのコンピューティングノード間の通信を許可するセキュリティグループを少なくとも 1 つ選択する必要があります。セキュリティグループをすばやく作成して、選択した VPC AWS で必要な設定で PCS に作成させるか、既存のセキュリティグループを選択できます。詳細については、「[セキュリティグループの要件と考慮事項](#)」を参照してください。

4. (オプション) Slurm アカウンティング設定セクションで、Slurm アカウンティングを有効にし、アカウンティングパラメータを設定できます。詳細については、「[PCS での Slurm AWS アカウンティング](#)」を参照してください。

5. (オプション) Slurm 設定セクションで、パラメータ名と値のペアを追加して、追加の Slurm 設定を設定できます。サポートされているパラメータの完全なリストについては、「」を参照してください[PCS クラスターのカスタム Slurm AWS 設定](#)。
6. (オプション) タグで、PCS AWS クラスターにタグを追加します。
7. [クラスターを作成] を選択します。Status フィールドは、PCS AWS がクラスターを作成するCreating間に表示されます。この処理には数分かかることもあります。

Important

ごとに AWS リージョン 1 つのCreating状態にあるクラスターは 1 つだけです AWS アカウント。AWS PCS は、クラスターを作成しようとしたときに Creating状態のクラスターが既に存在する場合にエラーを返します。

AWS CLI

クラスターを作成するには

1. 下記のコマンドを使用して、クラスターを作成します。コマンドを実行する前に、次の置き換えを行います：
 - **region** を、など、クラスター AWS リージョン を作成する の ID に置き換えますus-east-1。
 - **#####** の部分は自分のクラスター名に置き換えます。この名前には英数字 (大文字と小文字が区別されます) とハイフンのみを使用できます。アルファベット文字で始まり、40 文字を超えることはできません。名前は、クラスターを作成する AWS リージョン および AWS アカウント 内で一意である必要があります。
 - **25.05** をサポートされているバージョンの Slurm に置き換えます。

Note

AWS PCS は現在、Slurm 25.05 および 24.11 をサポートしています。

- **SMALL** をサポートされている任意のクラスターサイズに置き換えます。これにより、PCS AWS クラスターで管理できる同時ジョブとコンピューティングノードの数を決定します。これは、クラスターの作成時にのみ設定できます。サイズ設定の詳細については、「」を参照してください[PCS AWS のクラスターサイズ](#)。

- の値を独自の値subnetIdsに置き換えます。スケジューラエンドポイントがパブリックインターネットに公開されないように、プライベートサブネットを選択することをお勧めします。
- PCS AWS securityGroupIdsがクラスター用に作成するネットワークインターフェイスに関連付ける を指定します。セキュリティグループは、クラスターと同じ VPC に存在する必要があります。クラスターとそのコンピューティングノード間の通信を許可するセキュリティグループを少なくとも 1 つ選択する必要があります。詳細については、「[セキュリティグループの要件と考慮事項](#)」を参照してください。

```
aws pcs create-cluster --region region \  
  --cluster-name my-cluster \  
  --scheduler type=SLURM,version=25.05 \  
  --size SMALL \  
  --networking subnetIds=subnet-ExampleId1,securityGroupIds=sg-ExampleId1
```

- IPv6 を使用するには、 を --networking設定networkType=IPV6に追加します。

```
--networking networkType=IPV6,subnetIds=subnet-ExampleId1,securityGroupIds=sg-ExampleId1
```

- 必要に応じて、オプションを追加して Slurm の動作--slurm-configurationをカスタマイズし、Slurm 設定オプションを指定できます。次の例では、スケールダウンアイドル時間を 60 分 (3600 秒) に設定し、Slurm アカウンティングを有効にして、 の値としてslurm.conf設定を指定しますslurmCustomSettings。詳細については、「[PCS の Slurm AWS アカウンティング](#)」を参照してください。

 Note

アカウンティングは Slurm 24.11 以降でサポートされています。

```
aws pcs create-cluster --region region \  
  --cluster-name my-cluster \  
  --scheduler type=SLURM,version=25.05 \  
  --size SMALL \  
  --networking subnetIds=subnet-ExampleId1,securityGroupIds=sg-ExampleId1
```

```
--slurm-configuration  
scaleDownIdleTimeInSeconds=3600,accounting='{mode=STANDARD}',slurmCustomSettings='{p
```

2. クラスターのプロビジョニングには数分かかる場合があります。クラスターのステータスのクエリを実行するには次のコマンドを使用します。クラスターのステータスフィールドがになるまで、キューまたはコンピューティングノードグループの作成に進まないでくださいACTIVE。

```
aws pcs get-cluster --region region --cluster-identifier my-cluster
```

Important

ごとに AWS リージョン 1 つのCreating状態にあるクラスターは 1 つだけです AWS アカウント。AWS PCS は、クラスターを作成しようとしたときに Creating状態のクラスターが既に存在する場合にエラーを返します。

クラスターに推奨される次のステップ

- コンピューティングノードグループを追加します。
- キューを追加します。
- ログ作成を有効化します。

PCS AWS でのクラスターの更新

AWS PCS では、UpdateCluster API またはコンソールを使用して、作成後にクラスター設定を更新できます。インフラストラクチャを再構築することなくクラスター設定を変更できるため、運用上のオーバーヘッドが軽減され、中断が最小限に抑えられます。

クラスター更新の利点

PCS AWS クラスターを更新すると、サービスを中断することなく、HPC インフラストラクチャを新しい要件に適応させることができます。設定の変更には、クラスターの再構築に 1 時間以上かかるのではなく、数分かかります。この機能は、最小限のダウンタイムを必要とする本番環境や、ワークロードパターンの変化に応じてクラスター設定を調整する必要があるチームにとって重要です。

サポートされている設定の変更

設定の 3 つの主要なカテゴリを変更できます。

- アカウンティング設定 - マネージドアカウンティングを有効または無効にし、保持設定を構成します。
- スケールダウン動作 - `scaleDownIdleTime` パラメータを調整します。これにより、PCS AWS が自動的にインスタンスを終了するまで、動的インスタンスがアイドル状態のままになる時間を制御します。
- Slurm カスタム設定 - Prolog、Epilog、SelectTypeParameters など、クラスターレベルで適用されるサポートされている Slurm 設定を変更します。

制限

クラスターの作成後に特定の設定を変更することはできません。具体的には次のとおりです。

- セキュリティグループ設定
- VPC サブネットの選択
- クラスターサイズ
- Slurm バージョン
- クラスター名

これらの設定はクラスターのアーキテクチャの基礎であり、変更するには新しいクラスターを作成する必要があります。

クラスター更新の前提条件

クラスターを更新する前に、次の条件が満たされていることを確認してください。

- クラスターは ACTIVE、UPDATE_FAILED、または SUSPENDED 状態である必要があります
- 関連するすべてのリソース (キュー、コンピューティングノードグループ) は ACTIVE 状態である必要があります
- UpdateCluster オペレーションには適切な IAM アクセス許可が必要です
- 他の更新オペレーションは進行中にできません

プロセスとジョブの影響を更新する

更新オペレーション中、クラスターコントローラーが一時的に到達不能になっても、コンピューティングノードは既存のジョブを引き続き実行します。ただし、この期間中、システムは新しいジョブの送信を受け入れたり、スケジュールを決定したりすることはできません。

コンソールと API インターフェイスの両方でクラスターの更新をモニタリングできます。クラスターは、更新中に次の状態に移行します。

- UPDATING - 更新中
- ACTIVE - 更新が正常に完了しました
- UPDATE_FAILED - 更新でエラーが発生しました

更新中の請求

PCS AWS クラスターの標準時間単位の料金は、更新オペレーション中も継続されます。アカウントिंगを無効にするようにクラスターを更新すると、アカウントिंग機能の請求は、クラスターが UPDATING 状態になるとすぐに停止します。アカウントिंगを有効にすると、クラスターが正常に更新を完了して ACTIVE 状態に戻るまで、請求は開始されません。

トピック

- [PCS AWS クラスターを更新する](#)
- [PCS AWS でのクラスターの更新に関するよくある質問](#)
- [PCS AWS クラスターの更新のトラブルシューティング](#)

PCS AWS クラスターを更新する

次のステップを使用して、クラスターのスケジューラ設定、アカウントिंग設定、および Slurm カスタム設定を変更します。詳細については、「[PCS クラスターのカスタム Slurm AWS 設定](#)」を参照してください。

前提条件

- クラスターは ACTIVE、UPDATE_FAILED、または SUSPENDED 状態である必要があります
- 関連するすべてのリソース (キュー、コンピューティングノードグループ) は ACTIVE 状態である必要があります
- 他の更新オペレーションは進行中にできません

手順

AWS マネジメントコンソール

1. <https://console.aws.amazon.com/pcs/> で AWS PCS コンソールを開きます。
2. ナビゲーションペインで [Clusters] (クラスター) を選択してください。
3. 更新するクラスターを選択します。
4. [編集] を選択します。
5. クラスターの編集ページで、必要な設定を変更します。
 - スケジューラ設定で、スケールダウンアイドル時間を更新して、自動終了までに動的インスタンスがアイドル状態のままになる時間を制御します。
 - 必要に応じて Prolog、Epilog、および Select-type パラメータ設定を変更します。
 - マネージドアカウンティングの保持時間を有効、無効に、または設定します。
 - 「追加のスケジューラ設定」で、Slurm カスタム設定を追加、編集、または削除します。サポートされているパラメータの詳細については、「」を参照してください [PCS クラスターのカスタム Slurm AWS 設定](#)。

Note

編集できないフィールドは読み取り専用として表示され、現在の値が表示されます。

6. 更新を選択して変更を送信します。
7. クラスターのステータスをモニタリングします。このステータスは、プロセス中に「更新中」と表示されます。更新が正常に完了すると、ステータスが変わります。

AWS CLI

1. ターミナルまたはコマンドプロンプトを開きます。
2. 次のコマンドを使用して、クラスターのステータスを確認します。

```
aws pcs get-cluster --cluster-identifier my-cluster
```

3. 次のいずれかの例を使用して、更新リクエストを送信します。

- マネージドアカウンティングを有効にするには:


```
aws pcs update-cluster --cluster-identifier my-cluster \  
--slurm-configuration 'accounting={mode=STANDARD}'
```

- Slurm Prolog 設定を更新するには:

```
aws pcs update-cluster --cluster-identifier my-cluster \  
--slurm-configuration \  
'SlurmCustomSettings=[{parameterName=Prolog,parameterValue="/path/to/  
prolog.sh"}]'
```

- スケールダウンアイドル時間を更新するには:

```
aws pcs update-cluster --cluster-identifier my-cluster \  
--slurm-configuration 'scaleDownIdleTimeInSeconds=300'
```

4. クラスターのステータスを確認して、更新の進行状況をモニタリングします。

```
aws pcs get-cluster --cluster-identifier my-cluster
```

更新リクエストが成功すると、コマンドはすべての変更を含むクラスターオブジェクトを返します。クラスターのステータスは、完了ACTIVEすると から UPDATING に変わります。

PCS AWS でのクラスターの更新に関するよくある質問

AWS PCS でのクラスター設定の更新に関する一般的な質問に対する回答を取得します。

どのような設定を変更できますか？

アカウントिंग設定 (マネージドアカウントINGの有効化/無効化)、スケールダウン動作 (scaleDownIdleTime パラメータ)、およびクラスターレベルで適用されるサポートされている Slurm カスタム設定を変更できます。セキュリティグループ、VPC サブネット、クラスターサイズ、Slurm バージョン、またはクラスター名を変更することはできません。

複数の更新をキューに入れることはできますか？

いいえ。クラスターが ACTIVE状態に戻るのを待ってから、別の更新を送信する必要があります。関連するすべてのリソース (キュー、コンピューティングノードグループ) も ACTIVE状態である必要があります。

クラスターの更新オペレーションをキャンセルできますか？

いいえ。進行中のクラスター更新オペレーションをキャンセルすることはできません。

クラスターの更新中にジョブを送信できますか？

クラスターの更新中にジョブを送信しないことをお勧めします。Slurm コントローラーは、更新プロセス中に使用できない場合があります。

クラスターの更新中もジョブは引き続き実行されますか？

はい。実行中のジョブは、更新プロセス中にクラスターコントローラーに短時間到達できない場合でも、コンピューティングノードで引き続き実行されます。ただし、コントローラーが再び使用可能になるまで、ジョブステータスは更新されない場合があります。

更新中の請求にはどのような影響がありますか？

標準時間単位の料金は、更新オペレーション中も継続されます。アカウンティングを無効にすると、クラスターが UPDATING 状態になると請求が停止します。アカウンティングを有効にすると、クラスターが正常に ACTIVE 状態に戻ったときに請求が開始されます。

PCS AWS クラスターの更新のトラブルシューティング

このトピックは、クラスター設定の更新時に発生する可能性がある一般的な問題を特定して解決するのに役立ちます。

更新がアカウンティング設定エラーで失敗する

一般的な原因

クラスターが UPDATE_FAILED 状態になり、エラーメッセージはアカウンティング設定の問題を示します。これは通常、アカウンティング設定が現在の Slurm バージョンと互換性がない場合、または無効な設定が含まれている場合に発生します。

解決方法

アカウンティング設定でクラスターの Slurm バージョンとの互換性を確認し、有効な設定パラメータを使用して修正された更新リクエストを送信します。

更新がカスタム設定エラーで失敗する

一般的な原因

クラスターが UPDATE_FAILED 状態になり、エラーメッセージは Slurm カスタム設定の問題を示します。これは、無効な Slurm パラメータ値またはサポートされていないパラメータの組み合わせを指定した場合に発生します。

解決方法

サポートされているパラメータに対して Slurm カスタム設定を検証し、有効なパラメータ値と組み合わせを使用して修正された更新リクエストを送信します。

更新リクエストを送信できません

一般的な原因

更新ボタンはコンソールで無効になっているか、API が 400 レベルのエラーを返します。これは、クラスターが適切な状態にない場合、関連するリソースがアクティブでない場合、または設定で検証に失敗した場合に発生します。

解決方法

クラスターと関連するすべてのリソースが ACTIVE 状態になるまで待つってから、設定で検証エラーを確認してから、更新リクエストを再送信します。

検証エラー

一般的な原因

コマンドは、すぐに 400 レベルの HTTP エラーと説明メッセージを返します。これは、無効なクラスター状態、リソース状態、または設定パラメータが原因で発生します。

解決方法

レスポンスに記載されている特定の検証エラーに対処し、更新オペレーションを再試行します。

PCS AWS でのクラスターの削除

このトピックでは、AWS PCS クラスターを削除する方法の概要を説明します。

PCS AWS クラスターを削除する際の考慮事項

- クラスターを削除する前に、クラスターに関連付けられているすべてのキューを削除する必要があります。詳細については、「[PCS AWS でのキューの削除](#)」を参照してください。
- クラスターに関連付けられているすべてのコンピューティングノードグループは、クラスターを削除する前に削除する必要があります。詳細については、「[PCS AWS でのコンピューティングノードグループの削除](#)」を参照してください。

クラスターを削除する

AWS マネジメントコンソール または を使用してクラスター AWS CLI を削除できます。

AWS マネジメントコンソール

クラスターを削除するには

1. [AWS PCS コンソール](#)を開きます。
2. 削除するクラスターを選択します。
3. [削除] を選択します。
4. クラスターのステータスフィールドには が表示されますDeleting。完了までに数分かかることがあります。

AWS CLI

クラスターを削除するには

1. クラスターを削除するには、次のコマンドを使用します。
 - *region-code* をクラスターがある に置き換え AWS リージョン ます。
 - *my-cluster* をクラスターの名前または ID に置き換えます。

```
aws pcs delete-cluster --region region-code --cluster-identifier my-cluster
```

2. クラスターの削除には数分かかる場合があります。次のコマンドを使用して、クラスターのステータスを確認できます。

```
aws pcs get-cluster --region region-code --cluster-identifier my-cluster
```

PCS AWS のクラスターサイズ

AWS PCS は、パッチ適用、ノードプロビジョニング、更新などの主要なタスクを自動化しながら、高可用性で安全なクラスターを提供します。

クラスターを作成するときは、次の 2 つの要素に基づいてクラスターのサイズを選択します。

- 管理するコンピューティングノードの数
- クラスターで実行される予定のアクティブジョブとキューに入っているジョブの数

Important

クラスターの作成後にクラスターサイズを変更することはできません。サイズを変更する必要がある場合は、新しいクラスターを作成する必要があります。

Slurm クラスターサイズ	マネージドインスタンスの数	アクティブなジョブとキューに入れられたジョブの数
Small	最大 32	最大 256
中	最大 512	最大 8192
大	最大 2048	最大 16384

例

- クラスターに最大 24 個のマネージドインスタンスがあり、最大 100 個のジョブを実行する場合は、Small を選択します。
- クラスターに最大 24 個のマネージドインスタンスがあり、最大 1000 個のジョブを実行する場合は、中を選択します。
- クラスターに最大 1000 個のマネージドインスタンスがあり、最大 100 個のジョブを実行する場合は、ラージ を選択します。

- クラスターに最大 1000 個のマネージドインスタンスがあり、最大 10,000 個のジョブを実行する場合は、**ラージ** を選択します。

PCS AWS でのクラスターシークレットの使用

クラスターの作成の一環として、AWS PCS はクラスターのジョブスケジューラに接続するために必要なクラスターシークレットを作成します。また、スケーリングイベントに応じて起動するインスタンスのセットを定義する AWS PCS コンピューティングノードグループを作成します。AWS PCS は、それらのコンピューティングノードグループによって起動されるインスタンスをクラスターシークレットで設定して、ジョブスケジューラに接続できるようにします。Slurm クライアントを手動で設定する場合があります。例としては、永続的なログインノードの構築や、ジョブ管理機能を備えたワークフローマネージャーの設定などがあります。

AWS PCS は、`pcs!`にプレフィックスが付いた[マネージドシークレットとしてクラスターシークレット](#)を保存します AWS Secrets Manager。シークレットのコストは、PCS AWS の使用料金に含まれています。クラスターシークレットを `ローテーション` AWS Secrets Manager して、セキュリティコンプライアンスを維持し、潜在的なセキュリティ侵害を修復できます。

トピック

- [AWS Secrets Manager を使用してクラスターシークレットを検索する](#)
- [PCS AWS を使用してクラスターシークレットを検索する](#)
- [Slurm クラスターシークレットを取得する](#)
- [PCS AWS でのクラスターシークレットの更新](#)

AWS Secrets Manager を使用してクラスターシークレットを検索する

AWS マネジメントコンソール

1. [\[Secrets Manager console\]](#) (Secrets Managerのコンソール) に移動します。
2. シークレットを選択し、`pcs!`プレフィックスを検索します。

Note

PCS クラスターシークレットには、`pcs!`が AWS PCS クラスター ID `aws-pcs-slurm-secret-cluster-id`である形式の名前があります。

AWS CLI

各 AWS PCS クラスターシークレットには もタグ付けされます `aws:pcs:cluster-id`。クラスターのシークレット ID は、次のコマンドで取得できます。コマンドを実行する前に、次の置換を行います。

- を `region` に置き換え AWS リージョン `us-east-1`、などの でクラスターを作成します。
- を、クラスターシークレットを検索する AWS PCS クラスターの ID `cluster-id` に置き換えます。

```
aws secretsmanager list-secrets \
  --region region \
  --filters Key=tag-key,Values=aws:pcs:cluster-id \
    Key=tag-value,Values=cluster-id
```

PCS AWS を使用してクラスターシークレットを検索する

を使用して AWS CLI、PCS クラスターシークレットの ARN AWS を検索できます。次のコマンドを入力し、次の置換を行います。

- を `region` に置き換え AWS リージョン `us-east-1`、などの でクラスターを作成します。
- をクラスターの名前または識別子 `my-cluster` に置き換えます。

```
aws pcs get-cluster --region region --cluster-identifier my-cluster
```

次の出力例は、`get-cluster` コマンドからのものです。シークレットを取得するには、`secretArn`と `secretVersion`一緒に使用できます。

```
{
  "cluster": {
    "name": "get-started",
    "id": "pcs_123456abcd",
    "arn": "arn:aws:pcs:us-east-1:111122223333:cluster/pcs_123456abcd",
    "status": "ACTIVE",
    "createdAt": "2024-12-17T21:03:52+00:00",
    "modifiedAt": "2024-12-17T21:03:52+00:00",
    "scheduler": {
```

```

        "type": "SLURM",
        "version": "25.05"
    },
    "size": "SMALL",
    "slurmConfiguration": {
        "authKey": {
            "secretArn": "arn:aws:secretsmanager:us-east-1:111122223333:secret:pcs!slurm-secret-pcs_123456abcd-a12ABC",
            "secretVersion": "ef232370-d3e7-434c-9a87-ec35c1987f75"
        }
    },
    "networking": {
        "subnetIds": [
            "subnet-0123456789abcdef0"
        ],
        "securityGroupIds": [
            "sg-0123456789abcdef0"
        ]
    },
    "endpoints": [
        {
            "type": "SLURMCTLD",
            "privateIpAddress": "10.3.149.220",
            "port": "6817"
        }
    ]
}

```

Slurm クラスターシークレットを取得する

Secrets Manager を使用して、Slurm クラスターシークレットの現在の base64 エンコードバージョンを取得できます。次の例では、 を使用します AWS CLI。コマンドを実行する前に、次の置換を行います。

- を *region* に置き換え AWS リージョン、 の、 などの でクラスターを作成します us-east-1。
- を PCS AWS クラスター secretArn の *secret-arn* に置き換えます。

```

aws secretsmanager get-secret-value \
  --region region \
  --secret-id 'secret-arn' \

```



```
--version-stage AWSCURRENT \  
--query 'SecretString' \  
--output text
```

Slurm クラスターシークレットの使用方法については、「」を参照してください[スタンドアロンインスタンス AWS を PCS ログインノードとして使用する](#)。

アクセス許可

IAM プリンシパルを使用して Slurm クラスターシークレットを取得します。IAM プリンシパルには、シークレットを読み取るアクセス許可が必要です。詳細については、「AWS Identity and Access Management ユーザーガイド」の[「ロールの用語と概念」](#)を参照してください。

次のサンプル IAM ポリシーは、サンプルクラスターシークレットへのアクセスを許可します。

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "AllowSecretValueRetrievalAndVersionListing",  
      "Effect": "Allow",  
      "Action": [  
        "secretsmanager:GetSecretValue",  
        "secretsmanager:ListSecretVersionIds"  
      ],  
      "Resource": "arn:aws:secretsmanager:us-east-1:012345678901:secret:pcs!  
slurm-secret-s3431v9rx2-FN7tJF"  
    }  
  ]  
}
```

PCS AWS でのクラスターシークレットの更新

AWS Secrets Manager マネージドローテーションを使用して、PCS AWS でクラスターシークレットをローテーションします。定期的なシークレットローテーションは、HPC 環境で強力なセキュリティ体制を維持するためのセキュリティのベストプラクティスです。この機能を使用すると、定期的な認証情報のローテーションを義務付ける HIPAA や FedRAMP などの業界のコンプライアンス基準を満たすことができます。

クラスターシークレットは、クラスターに参加するコンピューティングノードを認証し、Slurm REST API 認証の JWT キーとして使用するという 2 つの目的を果たします。ローテーションすると、両方の側面が同時に影響を受けます。

クラスターシークレットローテーションの仕組み

シークレットのローテーション中にクラスターの安定性を維持するために手動で準備します。

1. 準備 — すべてのコンピューティングノードグループを 0 容量にスケールし、ジョブが実行されていないことを確認します。
2. ローテーション – Secrets Manager コンソールまたは API を使用してローテーションを開始します。
3. モニタリング – CloudTrail イベントを通じて進行状況を追跡する
4. 復旧 — コンピューティングノードグループを必要な容量にスケールバックする

ローテーション中、クラスターは ACTIVE 状態のままになり、請求は通常どおり続行されます。このプロセスには通常数分かかります。

要件と制限

クラスターシークレットを更新する前に、以下の要件を満たす必要があります。

- クラスターは ACTIVE または UPDATE_FAILED 状態である必要があります
- IAM ロールには アクセス `secretsmanager:RotateSecret` 許可が必要です
- すべてのコンピューティングノードグループは 0 容量にスケールする必要があります
- ローテーション前にすべてのジョブを停止する

制限:

- 各ローテーションに必要な手動準備
- 既存の JWT トークンが無効になり、再発行が必要
- BYO ログインノードでは、ローテーション後に手動でシークレットを更新する必要があります

トピック

- [PCS AWS でクラスターシークレットをローテーションする](#)
- [PCS AWS でのクラスターシークレットローテーションに関するよくある質問](#)
- [PCS AWS でのクラスターシークレットローテーションのトラブルシューティング](#)

PCS AWS でクラスターシークレットをローテーションする

クラスターシークレットを更新してセキュリティ要件に準拠し、潜在的な侵害に対処します。このプロセスでは、クラスターをメンテナンスモードにする必要があります。

前提条件

- アクセス `secretsmanager:RotateSecret` 許可を持つ IAM ロール
- ACTIVE または UPDATE_FAILED 状態のクラスター

手順

1. 次回のメンテナンスウィンドウをクラスターユーザーに通知します。
2. すべてのコンピューティングノードグループを 0 容量にスケーリングして、クラスターをメンテナンスモードにします。
 - a. UpdateComputeNodeGroup API を使用して、すべてのコンピューティングノードグループに対して minInstanceCount と maxInstanceCount の両方を 0 に設定します。
 - b. すべてのノードが停止するまで待ちます。
 - c. オプション: 正常なジョブ処理のために容量を終了する前に、Slurm コマンドを使用してスケジューラキューをドレインします。
3. Secrets Manager を使用してローテーションを開始します。
 - コンソールメソッド:
 - Secrets Manager に移動し、クラスターシークレットを選択し、シークレットのローテーションを選択します。
 - API メソッド:
 - Secrets Manager rotate-secret API を使用します。
4. ローテーションの進行状況をモニタリングします。
 - a. CloudTrail イベントを通じて進行状況を追跡します。
 - b. Secrets Manager コンソールまたは secretsmanager:describeSecret API lastRotatedDate で確認します。
 - c. RotationSucceeded または RotationFailed CloudTrail イベントを待ちます。
5. ローテーションが成功したら、クラスターの容量を復元します。

- a. UpdateComputeNodeGroup API を使用して、ノードグループを希望の最小/最大容量にリセットします。
- b. PCS AWS マネージドログインノードの場合: 追加のアクションは必要ありません。
- c. BYO ログインノードの場合:
 - i. ログインノードに接続します。
 - ii. Secrets Manager の新しいシークレット/etc/slurm/slurm.keyで を更新します。
 - iii. Slurm Auth デーモンと Cred Kiosk デーモン (サック) を再起動します。

PCS AWS でのクラスターシークレットローテーションに関するよくある質問

PCS でのクラスターシークレットのローテーションに関する一般的な質問に対する回答を見つけ AWS ます。

クラスターシークレットとは

クラスターシークレットは、Slurm コントローラーと AWS PCS コンピューティングノード間の安全な通信を可能にする安全な認証情報です。また、Slurm REST API 認証の JSON ウェブトークン (JWT) キーとしても機能します。

クラスターシークレットと JWT キーの違いは何ですか？

AWS PCS では、クラスターシークレットと JWT キーは同じリソースであり、さまざまな目的に対応します。クラスターシークレットは Slurm 内部通信を認証し、JWT キーは REST API 認証用のトークンに署名します。ローテーションすると、両方の側面が同時に影響を受けます。

ローテーションにはどのくらいの時間がかかりますか？

ローテーションプロセスには通常数分かかります。クラスターは ACTIVE 状態のままで、ローテーション中は請求が通常どおり続行されます。

自動ローテーションをスケジュールできますか？

Secrets Manager でスケジュールされたローテーションを有効にできます。ただし、最初のリリースでは、各ローテーションの前に手動で準備 (ノードグループを 0 にスケーリング) する必要があります。

ローテーション後も既存の JWT トークンは機能しますか？

いいえ。既存の JWT トークンはローテーション後に無効になります。REST API クライアントの新しいトークンを発行します。

クラスターシークレットはどこにありますか？

クラスターシークレットは、Secrets Manager コンソールまたは PCS AWS コンソールから見つけることができます。詳細な手順については、[AWS Secrets Manager を使用してクラスターシークレットを検索する](#)「」および「」を参照してください[PCS AWS を使用してクラスターシークレットを検索する](#)。

ローテーションでノードグループを 0 にスケーリングする必要があるのはなぜですか？

ローテーションでは、シークレット更新プロセス中にクラスターの安定性を確保するために実行中のインスタンスは必要ありません。これにより、古いシークレットと新しいシークレット間の認証の競合が防止されます。

この機能はどのようなコンプライアンス要件をサポートしていますか？

この機能により、PCS AWS は HIPAA や FedRAMP などの業界のコンプライアンス標準を満たすことができます。これは、セキュリティコントロールの一部として定期的な認証情報のローテーションを義務付けています。

PCS AWS でのクラスターシークレットローテーションのトラブルシューティング

環境が適切に準備されていない場合、クラスターシークレットのローテーションは失敗します。最も一般的な原因は、クラスター内のアクティブなインスタンスです。失敗を防ぐには：

1. すべてのノードグループを 0 容量に設定します。
2. ノードが停止するのを待ちます。
3. クラスターが CREATE_FAILED、DELETE_FAILED、RESUMING、SUSPENDING または の状態にないことを確認します SUSPENDED。

ローテーションが失敗した場合：

- RotationFailed CloudTrail イベントが表示される
- クラスターシークレットは変更されません
- 詳細については、CloudTrail で RotationFailed イベントを確認してください。
- ローテーションを成功させるためのすべての準備ステップを完了する

AWS PCS コンピューティングノードグループ

PCS AWS コンピューティングノードグループは、ノード (Amazon EC2 インスタンス) の論理コレクションです。これらのノードは、コンピューティングジョブの実行や、HPC システムへのインタラクティブなシェルベースのアクセスの提供に使用できます。コンピューティングノードグループは、使用する Amazon EC2 インスタンスタイプ、実行するインスタンスの数、スポットインスタンスとオンデマンドインスタンスのどちらを使用するか、使用するサブネットとセキュリティグループ、起動時に各インスタンスを設定する方法など、ノードを作成するためのルールで構成されます。これらのルールが更新されると、AWS PCS はコンピューティングノードグループに関連付けられたリソースを一致させるように更新します。

トピック

- [PCS AWS でのコンピューティングノードグループの作成](#)
- [PCS AWS コンピューティングノードグループの更新](#)
- [PCS AWS でのコンピューティングノードグループの削除](#)
- [PCS AWS でコンピューティングノードグループの詳細を取得する](#)
- [PCS AWS でのコンピューティングノードグループインスタンスの検索](#)

PCS AWS でのコンピューティングノードグループの作成

このトピックでは、使用可能なオプションの概要と、AWS Parallel Computing Service (AWS PCS) でコンピューティングノードグループを作成するときに考慮すべき点について説明します。PCS でコンピューティングノードグループを初めて作成する場合は、AWS 「」のチュートリアルに従うことをお勧めします[AWS Parallel Computing Service の使用を開始する](#)。このチュートリアルは、使用可能なすべてのオプションとシステムアーキテクチャを拡張することなく、動作する HPC システムを作成するのに役立ちます。

Note

コンピューティングノードグループのカスタム Slurm 設定を設定して、リソース使用率とノードレベルの動作を制御できます。詳細については、「[PCS でのカスタム Slurm AWS 設定の設定](#)」を参照してください。

⚠ Important

AWS 現在、PCS では、IPv6 専用ネットワークで PCS を使用している場合でも、ローカルノード通信に IPv4 IPv6-only をサポートするカーネルが必要です。AWS 詳細については、「[PCS のカスタム Amazon AWS マシンイメージ \(AMIs\)](#)」を参照してください。

前提条件

- 必要な数の EC2 インスタンスを起動するのに十分なサービスクォータ AWS リージョン。を使用して[AWS マネジメントコンソール](#)、サービスクォータの引き上げを確認およびリクエストできます。
- PCS ネットワーク要件を満たす既存の VPC AWS とサブネット (複数可)。本番環境用にクラスターをデプロイする前に、これらの要件を十分に理解することをお勧めします。詳細については、「[AWS PCS VPC とサブネットの要件と考慮事項](#)」を参照してください。CloudFormation テンプレートを使用して VPC と subnets を作成することもできます。は CloudFormation テンプレートの HPC レシピ AWS を提供します。詳細については、GitHub [のaws-hpc-recipes](#)を参照してください。
- PCS RegisterComputeNodeGroupInstance API AWS アクションを呼び出すアクセス許可と、ノードグループインスタンスに必要な他の AWS リソースへのアクセス許可を持つ IAM インスタンスプロファイル。詳細については、「[AWS Parallel Computing Service の IAM インスタンスプロファイル](#)」を参照してください。
- ノードグループインスタンスの起動テンプレート。詳細については、「[PCS での Amazon EC2 AWS 起動テンプレートの使用](#)」を参照してください。
- Amazon EC2 スポットインスタンスを使用するコンピューティングノードグループを作成するには、に AWSServiceRoleForEC2Spot サービスにリンクされたロールが必要です AWS アカウント。詳細については、「[PCS AWS の Amazon EC2 スポットロール](#)」を参照してください。

PCS AWS でコンピューティングノードグループを作成する

コンピューティングノードグループは、AWS マネジメントコンソール または を使用して作成できます AWS CLI。

AWS マネジメントコンソール

コンソールを使用してコンピューティングノードグループを作成するには

1. [AWS PCS コンソール](#)を開きます。
2. コンピューティングノードグループを作成するクラスターを選択します。Compute ノードグループに移動し、作成を選択します。
3. Compute ノードグループのセットアップセクションで、ノードグループの名前を指定します。名前には、大文字と小文字を区別する英数字とハイフンのみを含めることができます。アルファベット文字で始まり、25 文字を超えることはできません。名前はクラスター内で一意である必要があります。
4. コンピューティング設定で、次の値を入力または選択します。
 - a. EC2 起動テンプレート – このノードグループに使用するカスタム起動テンプレートを選択します。起動テンプレートを使用して、サブネット、セキュリティグループ、モニタリング設定、インスタンスレベルのストレージなどのネットワーク設定をカスタマイズできます。起動テンプレートを準備していない場合は、[PCS での Amazon EC2 AWS 起動テンプレートの使用](#)「」を参照して作成方法を確認してください。

Important

AWS PCS は、コンピューティングノードグループごとにマネージド起動テンプレートを作成します。これらは という名前です pcs-*identifier*-do-not-delete。コンピューティングノードグループを作成または更新するときにこれらを選択しないでください。そうしないと、ノードグループが正しく機能しません。

- b. EC2 起動テンプレートのバージョン – カスタム起動テンプレートのバージョンを選択する必要があります。後でバージョンを変更する場合は、コンピューティングノードグループを更新して、起動テンプレートの変更を検出する必要があります。詳細については、「[PCS AWS コンピューティングノードグループの更新](#)」を参照してください。
- c. AMI ID – 起動テンプレートに AMI ID が含まれていない場合、または起動テンプレートの値を上書きする場合は、ここで AMI ID を指定します。ノードグループに使用される AMI は PCS AWS と互換性がある必要があることに注意してください。が提供するサンプル AMI を選択することもできます AWS。このトピックの詳細については、「」を参照してください [PCS 用の Amazon AWS マシンイメージ \(AMIs\)](#)。

- d. IAM インスタンスプロファイル – ノードグループのインスタンスプロファイルを選択します。インスタンスプロファイルは、リソースとサービスに安全にアクセスする AWS アクセス許可をインスタンスに付与します。準備が整っていない場合は、基本プロファイルの作成を選択して、最小ポリシーで PCS AWS に作成させるか、「」を参照してください[AWS Parallel Computing Service の IAM インスタンスプロファイル](#)。
 - e. サブネット – PCS クラスターがデプロイされている VPC 内の 1 AWS つ以上のサブネットを選択します。複数のサブネットを選択すると、ノード間で EFA 通信が利用できなくなり、異なるサブネット内のノード間の通信のレイテンシーが増加する可能性があります。ここで指定するサブネットが、EC2 起動テンプレートで定義したサブネットと一致することを確認します。
 - f. インスタンス – ノードグループ内のスケーリングリクエストを満たすインスタンスタイプを 1 つ以上選択します。すべてのインスタンスタイプには、同じプロセッサアーキテクチャ (x86_64 または arm64) と vCPUs の数が必要です。インスタンスに GPUs がある場合、すべてのインスタンスタイプに同じ数の GPUs が必要です。
 - g. スケーリング設定 – ノードグループのインスタンスの最小数と最大数を指定します。実行中のノード数が一定である静的設定、またはノードの最大数まで実行できる動の設定のいずれかを定義できます。静的設定の場合は、最小値と最大値をゼロより大きい同じに設定します。動の設定の場合、最小インスタンスを 0 に設定し、最大インスタンスを 0 より大きい数に設定します。AWS PCS は、静的インスタンスと動のインスタンスが混在するコンピューティングノードグループをサポートしていません。
5. (オプション) 追加設定で、以下を指定します。
- a. 購入オプション – オンデマンドインスタンス、スポットインスタンス、または既存のキャパシティブロックを選択します。オンデマンドキャパシティ予約 (ODCR) を使用する場合は、オンデマンド も選択します。詳細については、「[PCS での ODCRs AWS の使用](#)」を参照してください。キャパシティブロックを選択して、ML 予約に既存の Amazon EC2 キャパシティブロックを使用します。詳細については、「[PCS での ML での Amazon EC2 AWS キャパシティブロックの使用](#)」を参照してください。
 - b. 配分戦略 – スポット購入オプションを選択した場合、ノードグループでインスタンスを起動するときにスポットキャパシティプールを選択する方法を指定できます。詳細については、「Amazon Elastic Compute Cloud ユーザーガイド」の「[スポットインスタンスの配分戦略](#)」を参照してください。このオプションは、オンデマンド購入オプションを選択した場合は効果がありません。
6. (オプション) Slurm カスタム設定セクションで、パラメータ名と値のペアを追加して、追加の Slurm 設定を設定できます。サポートされているパラメータの完全なリストについては、

「」を参照してください[PCS コンピューティングノードグループのカスタム Slurm AWS 設定](#)。

7. (オプション) タグ で、コンピューティングノードグループにタグを追加します。
8. コンピューティングノードグループの作成 を選択します。Status フィールドは、PCS AWS がノードグループをプロビジョニングCreatingしている間に表示されます。これには数分間かかる場合があります。

推奨される次のステップ

- PCS AWS のキューにノードグループを追加して、ジョブを処理できるようにします。

AWS CLI

を使用してコンピューティングノードグループを作成するには AWS CLI

次のコマンドを使用してキューを作成します。コマンドを実行する前に、次の置き換えを行います：

1. **region** を、 などのクラスター AWS リージョン を作成する の ID に置き換えますus-east-1。
2. **my-cluster** をクラスターの名前またはに置き換えclusterIdます。
3. **my-node-group** をコンピューティングノードグループの名前に置き換えます。この名前には英数字 (大文字と小文字が区別されます) とハイフンのみを使用できます。アルファベット文字で始まり、25 文字を超えることはできません。名前はクラスター内で一意である必要があります。
4. **subnet-ExampleID1** をクラスター VPC の 1 つ以上のサブネット IDs に置き換えます。
5. **lt-ExampleID1** をカスタム起動テンプレートの ID に置き換えます。準備が整っていない場合は、[PCS での Amazon EC2 AWS 起動テンプレートの使用](#)「」を参照して作成方法を確認してください。

Important

AWS PCS は、コンピューティングノードグループごとにマネージド起動テンプレートを作成します。これらは という名前ですpcs-**identifier**-do-not-delete。コンピューティングノードグループを作成または更新するときにこれらを選択しないでください。そうしないと、ノードグループが正しく機能しません。

6. `launch-template-version` を特定の起動テンプレートバージョンに置き換えます。AWS PCS は、ノードグループをその特定のバージョンの起動テンプレートに関連付けます。
7. `arn:InstanceProfile` を IAM インスタンスプロファイルの ARN に置き換えます。準備が整っていない場合は、[PCS での Amazon EC2 AWS 起動テンプレートの使用](#)「」のガイドを参照してください。
8. `min-instances` と `max-instances` を整数値に置き換えます。実行中のノード数が一定である静的設定、またはノードの最大数まで実行できる動的設定のいずれかを定義できます。静的設定の場合は、最小値と最大値をゼロより大きい同じに設定します。動的設定の場合は、最小インスタンスを 0 に設定し、最大インスタンスを 0 より大きい数に設定します。AWS PCS は、静的インスタンスと動的インスタンスが混在するコンピューティングノードグループをサポートしていません。
9. `t3.large` を別のインスタンスタイプに置き換えます。instanceType 設定のリストを指定することで、インスタンスタイプを追加できます。たとえば、`--instance-configs instanceType=c6i.16xlarge instanceType=c6a.16xlarge` です。すべてのインスタンスタイプには、同じプロセッサアーキテクチャ (x86_64 または arm64) と vCPUs の数が必要です。インスタンスに GPUs がある場合、すべてのインスタンスタイプに同じ数の GPUs が必要です。

```
aws pcs create-compute-node-group --region region \  
  --cluster-identifier my-cluster \  
  --compute-node-group-name my-node-group \  
  --subnet-ids subnet-ExampleID1 \  
  --custom-launch-template id=lt-ExampleID1,version='launch-template-version' \  
  --iam-instance-profile-arn=arn:InstanceProfile \  
  --scaling-config minInstanceCount=min-instances,maxInstanceCount=max-instance \  
  --instance-configs instanceType=t3.large
```

Example- カスタム Slurm 設定を使用したコンピューティングノードグループの作成

```
aws pcs create-compute-node-group --region region \  
  --cluster-identifier my-cluster \  
  --compute-node-group-name my-node-group \  
  --subnet-ids subnet-ExampleID1 \  
  --custom-launch-template id=lt-ExampleID1,version='launch-template-version' \  
  --iam-instance-profile-arn=arn:InstanceProfile \  
  --scaling-config minInstanceCount=min-instances,maxInstanceCount=max-instance \  
  --instance-configs instanceType=t3.large \  
  --slurm-config slurm-config
```

```
--slurm-configuration \  
'slurmCustomSettings=[{parameterName=Features,parameterValue="gpu,nvme"}]'
```

詳細については、「[PCS コンピューティングノードグループのカスタム Slurm AWS 設定](#)」を参照してください。

create-compute-node-group コマンドに追加できるオプションの設定がいくつかあります。

- カスタム起動テンプレートに AMI への参照が含まれていない --amiId か、その値を上書きするかを指定できます。ノードグループに使用される AMI は PCS AWS と互換性がある必要がありますことに注意してください。が提供するサンプル AMI を選択することもできます AWS。このトピックの詳細については、「」を参照してください [PCS 用の Amazon AWS マシンイメージ \(AMIs\)](#)。
- を使用して --purchase-option、PCS AWS がコンピューティングノードグループの EC2 インスタンスを購入する方法を選択します。オンデマンドがデフォルトです。
 - ONDEMAND – オンデマンドインスタンスを使用します。オンデマンドキャパシティ予約 (ODCR) を使用する場合は、このオプションも選択します。詳細については、「[PCS での ODCRs AWS の使用](#)」を参照してください。
 - SPOT – 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を選択した場合は、--allocation-strategy を使用して、ノードグループでインスタンスを起動するときに AWS PCS がスポットキャパシティプールを選択する方法を定義することもできます。詳細については、「Amazon Elastic Compute Cloud ユーザーガイド」の「[スポットインスタンスの配分戦略](#)」を参照してください。
 - CAPACITY_BLOCK – ML 予約に既存の Amazon EC2 キャパシティブロックを使用します。詳細については、「[PCS での ML での Amazon EC2 AWS キャパシティブロックの使用](#)」を参照してください。
- を使用して、ノードグループ内のノード Slurm の設定オプションを指定できます --slurm-configuration。重み (スケジューリングの優先度) と実際のメモリを設定できます。重みの低いノードは優先度が高く、単位は任意です。詳細については、Slurm ドキュメントの「[重み](#)」を参照してください。実メモリは、ノードグループ内のノードの実メモリのサイズ (GB 単位) を設定します。これは、Slurm 設定の PCS AWS のクラスター CR_CPU_Memory のオプションと組み合わせて使用することを目的としています。詳細については、Slurm ドキュメントの [RealMemory](#)」を参照してください。

⚠ Important

コンピューティングノードグループの作成には数分かかる場合があります。

次のコマンドを使用して、ノードグループのステータスをクエリできます。ステータスが `ACTIVE` に達するまで、ノードグループをキューに関連付けることはできません。

```
aws pcs get-compute-node-group --region region \  
    --cluster-identifier my-cluster \  
    --compute-node-group-identifier my-node-group
```

PCS AWS コンピューティングノードグループの更新

このトピックでは、使用可能なオプションの概要と、AWS PCS コンピューティングノードグループを更新するときに考慮すべき点について説明します。Slurm のカスタム設定の詳細については、「」を参照してください [PCS コンピューティングノードグループのカスタム Slurm AWS 設定](#)。

AWS PCS コンピューティングノードグループを更新するためのオプション

AWS PCS コンピューティングノードグループを更新すると、AWS PCS によって起動されるインスタンスのプロパティと、それらのインスタンスの起動方法のルールを変更できます。たとえば、ノードグループインスタンスの AMI を、別のソフトウェアがインストールされている別のインスタンスに置き換えることができます。または、セキュリティグループを更新して、インバウンドまたはアウトバウンドのネットワーク接続を変更できます。スケーリング設定と優先購入オプションを変更することもできます。

次のノードグループ設定は、作成後に変更することはできません。

- 名前
- インスタンス

PCS AWS コンピューティングノードグループを更新する際の考慮事項

コンピューティングノードグループは、ジョブの処理、インタラクティブなシェルアクセスの提供、およびその他のタスクに使用される EC2 インスタンスを定義します。多くの場合、1 つ以上の PCS

AWS キューに関連付けられます。コンピューティングノードグループを更新して動作 (またはそのノードの動作) を変更するときは、次の点を考慮してください。

- コンピューティングノードグループプロパティの変更は、コンピューティングノードグループのステータスが更新からアクティブに変わると有効になります。新しいインスタンスは、更新されたプロパティで起動します。
- 特定のノードの設定に影響を与えない更新は、実行中のノードには影響しません。たとえば、サブネットの追加や配分戦略の変更などです。
- コンピューティングノードグループの起動テンプレートを更新する場合は、新しいバージョンを使用するようにコンピューティングノードグループを更新する必要があります。
- コンピューティングノードグループのノードからセキュリティグループを追加または削除するには、その起動テンプレートを編集し、コンピューティングノードグループを更新します。新しいインスタンスは、更新されたセキュリティグループのセットで起動します。
- コンピューティングノードグループが使用するセキュリティグループを直接編集すると、実行中のインスタンスと将来のインスタンスにすぐに反映されます。
- コンピューティングノードグループで使用される IAM インスタンスプロフィールにアクセス許可を追加または削除すると、実行中のインスタンスと将来のインスタンスにすぐに反映されます。
- コンピューティングノードグループのインスタンスで使用される AMI を変更するには、コンピューティングノードグループ (またはその起動テンプレート) を更新して新しい AMI を使用し、PCS AWS がインスタンスを置き換えるのを待ちます。
- AWS PCS は、ノードグループの更新オペレーション後にノードグループ内の既存のインスタンスを置き換えます。ノードで実行中のジョブがある場合、それらのジョブは AWS PCS がノードを置き換える前に完了できます。インタラクティブユーザープロセス (ログインノードインスタンスなど) は終了します。ノードグループのステータスは、PCS AWS がインスタンスを置き換え対象としてマークActiveするに 戻りますが、実際の置き換えはインスタンスがアイドル状態のときに発生します。
- コンピューティングノードグループで許可されるインスタンスの最大数を減らすと、AWS PCS は新しい最大値を満たすために Slurm からノードを削除します。AWS PCS は、削除された Slurm ノードに関連付けられた実行中のインスタンスを終了します。削除されたノードで実行中のジョブは失敗し、キューに戻ります。
- AWS PCS は、コンピューティングノードグループごとにマネージド起動テンプレートを作成します。これらは という名前です `pcs-identifier-do-not-delete`。コンピューティングノードグループを作成または更新するときに選択しないでください。選択しないと、ノードグループが正しく機能しません。

- 購入オプションにスポットを使用するようにコンピューティングノードグループを更新する場合は、アカウントに `AWSServiceRoleForEC2Spot` サービスにリンクされたロールが必要です。詳細については、「[PCS AWS の Amazon EC2 スポットロール](#)」を参照してください。

AWS PCS コンピューティングノードグループを更新するには

ノードグループは、AWS マネジメントコンソールまたは AWS CLI を使用して更新できます。

AWS マネジメントコンソール

コンピューティングノードグループを更新するには

- で AWS PCS コンソールを開きます。 <https://console.aws.amazon.com/pcs/home#/clusters>
- コンピューティングノードグループを更新するクラスターを選択します。
- Compute ノードグループに移動し、更新するノードグループに移動し、編集を選択します。
- コンピューティング設定、追加設定、Slurmカスタマイズ設定セクションで、以下以外の値を更新します。

- インスタンス – コンピューティングノードグループのインスタンスを変更することはできません。

Slurm のカスタム設定の詳細については、「」を参照してください [PCS コンピューティングノードグループのカスタム Slurm AWS 設定](#)。

- [更新] を選択します。ステータスフィールドには、変更が適用されている間の更新が表示されます。

Important

コンピューティングノードグループの更新には数分かかる場合があります。

AWS CLI

コンピューティングノードグループを更新するには

1. 次のコマンドを使用して、コンピューティングノードグループを更新します。コマンドを実行する前に、次の置き換えを行います：
 - a. *region-code* を、クラスターを作成する AWS リージョンに置き換えます。
 - b. *my-node-group* をコンピューティングノードグループの名前または `computeNodeGroupId` に置き換えます。
 - c. *my-cluster* をクラスターの名前または `clusterId` に置き換えます。

```
aws pcs update-compute-node-group --region region-code \  
  --cluster-identifier my-cluster \  
  --compute-node-group-identifier my-node-group
```

Example- カスタム Slurm 設定でコンピューティングノードグループを更新する

```
aws pcs update-compute-node-group --region region-code \  
  --cluster-identifier my-cluster \  
  --compute-node-group-identifier my-node-group \  
  --slurm-configuration \  
  'slurmCustomSettings=[{parameterName=Features,parameterValue="gpu, nvme"}]'
```

詳細については、「[PCS コンピューティングノードグループのカスタム Slurm AWS 設定](#)」を参照してください。

2. 以外のノードグループパラメータを更新します `--instance-configs`。たとえば、新しい AMI ID を設定するには、*my-custom-ami-id* が選択した AMI に置き換えられる `--amiId my-custom-ami-id` 場所に を渡します。

 Important

コンピューティングノードグループの更新には数分かかる場合があります。

次のコマンドを使用して、ノードグループのステータスをクエリできます。


```
aws pcs get-compute-node-group --region region-code \  
  --cluster-identifier my-cluster \  
  --compute-node-group-identifier my-node-group
```

PCS AWS でのコンピューティングノードグループの削除

このトピックでは、使用可能なオプションの概要と、PCS AWS でコンピューティングノードグループを削除するときに考慮すべき点について説明します。

コンピューティングノードグループを削除する際の考慮事項

コンピューティングノードグループは、ジョブの処理、インタラクティブなシェルアクセスの提供、およびその他のタスクに使用される EC2 インスタンスを定義します。多くの場合、1 つ以上の PCS AWS キューに関連付けられます。コンピューティングノードグループを削除する前に、次の点を考慮してください。

- コンピューティングノードグループによって起動された EC2 インスタンスはすべて終了します。これにより、これらのインスタンスで実行されているジョブがキャンセルされ、実行中のインタラクティブプロセスが終了します。
- 削除する前に、すべてのキューからコンピューティングノードグループの関連付けを解除する必要があります。詳細については、「[PCS AWS キューの更新](#)」を参照してください。

コンピューティングノードグループを削除する

AWS マネジメントコンソール または `aws` を使用して、コンピューティングノードグループ AWS CLI を削除できます。

AWS マネジメントコンソール

コンピューティングノードグループを削除するには

- [AWS PCS コンソール](#)を開きます。
- コンピューティングノードグループのクラスターを選択します。
- Compute ノードグループに移動し、削除するコンピューティングノードグループを選択します。
- [削除] を選択します。

5. Status フィールドには が表示されますDeleting。完了までに数分かかることがあります。

 Note

スケジューラにネイティブなコマンドを使用して、コンピューティングノードグループが削除されたことを確認できます。たとえば、Slurm queueには `sinfo` または `scall` を使用します。

AWS CLI

コンピューティングノードグループを削除するには

- コンピューティングノードグループを削除するには、次のコマンドを使用します。
 - `region-code` をクラスターがある に置き換え AWS リージョン ます。
 - `my-node-group` をコンピューティングノードグループの名前または ID に置き換えます。
 - `my-cluster` をクラスターの名前または ID に置き換えます。

```
aws pcs delete-compute-node-group --region region-code \  
    --compute-node-group-identifier my-node-group \  
    --cluster-identifier my-cluster
```

コンピューティングノードグループの削除には数分かかる場合があります。

 Note

スケジューラにネイティブなコマンドを使用して、コンピューティングノードグループが削除されたことを確認できます。たとえば、Slurm queueには `sinfo` または `scall` を使用します。

PCS AWS でコンピューティングノードグループの詳細を取得する

AWS マネジメントコンソール または `aws` を使用して、コンピューティングノードグループ ID、Amazon リソースネーム (ARN)、Amazon マシンイメージ (AMI) ID など、コンピューティング

ノードグループの詳細 AWS CLI を取得できます。これらの詳細は、多くの場合、PCS API AWS アクションと設定に必要な値です。

AWS マネジメントコンソール

コンピューティングノードグループの詳細を取得するには

1. [AWS PCS コンソール](#)を開きます。
2. クラスターを選択します。
3. Compute ノードグループを選択します。
4. リストからコンピューティングノードグループを選択します。

AWS CLI

コンピューティングノードグループの詳細を取得するには

1. [ListClusters](#) API アクションを使用して、クラスター名または ID を検索します。

```
aws pcs list-clusters
```

出力例:

```
{
  "clusters": [
    {
      "name": "get-started-cfn",
      "id": "pcs_abc1234567",
      "arn": "arn:aws:pcs:us-east-1:111122223333:cluster/pcs_abc1234567",
      "createdAt": "2025-04-01T20:11:22+00:00",
      "modifiedAt": "2025-04-01T20:11:22+00:00",
      "status": "ACTIVE"
    }
  ]
}
```

2. [ListComputeNodeGroups](#) API アクションを使用して、クラスター内のコンピューティングノードグループを一覧表示します。

```
aws pcs list-compute-node-groups --cluster-identifier cluster-name-or-id
```

呼び出しの例:

```
aws pcs list-compute-node-groups --cluster-identifier get-started-cfn
```

出力例:

```
{
  "computeNodeGroups": [
    {
      "name": "compute-1",
      "id": "pcs_abc123abc1",
      "arn": "arn:aws:pcs:us-east-1:111122223333:cluster/pcs_abc1234567/computenodegroup/pcs_abc123abc1",
      "clusterId": "pcs_abc1234567",
      "createdAt": "2025-04-01T20:19:25+00:00",
      "modifiedAt": "2025-04-01T20:19:25+00:00",
      "status": "ACTIVE"
    },
    {
      "name": "login",
      "id": "pcs_abc456abc7",
      "arn": "arn:aws:pcs:us-east-1:111122223333:cluster/pcs_abc1234567/computenodegroup/pcs_abc456abc7",
      "clusterId": "pcs_abc1234567",
      "createdAt": "2025-04-01T20:19:31+00:00",
      "modifiedAt": "2025-04-01T20:19:31+00:00",
      "status": "ACTIVE"
    }
  ]
}
```

3. [GetComputeNodeGroup](#) API アクションを使用して、コンピューティングノードグループの追加の詳細を取得します。

```
aws pcs get-compute-node-group --cluster-identifier cluster-name-or-id --compute-node-group-identifier compute-node-group-name-or-id
```

呼び出しの例:

```
aws pcs get-compute-node-group --cluster-identifier get-started-cfn --compute-node-group-identifier compute-1
```

出力例:

```
{
  "computeNodeGroup": {
    "name": "compute-1",
    "id": "pcs_abc123abc1",
    "arn": "arn:aws:pcs:us-east-1:111122223333:cluster/pcs_abc1234567/computenodegroup/pcs_abc123abc1",
    "clusterId": "pcs_abc1234567",
    "createdAt": "2025-04-01T20:19:25+00:00",
    "modifiedAt": "2025-04-01T20:19:25+00:00",
    "status": "ACTIVE",
    "amiId": "ami-0123456789abcdef0",
    "subnetIds": [
      "subnet-abc012345789abc12"
    ],
    "purchaseOption": "ONDEMAND",
    "customLaunchTemplate": {
      "id": "lt-012345abcdef01234",
      "version": "1"
    },
    "iamInstanceProfileArn": "arn:aws:iam::111122223333:instance-profile/AWSPCS-get-started-cfn-us-east-1",
    "scalingConfiguration": {
      "minInstanceCount": 0,
      "maxInstanceCount": 4
    },
    "instanceConfigs": [
      {
        "instanceType": "c6i.xlarge"
      }
    ]
  }
}
```

PCS AWS でのコンピューティングノードグループインスタンスの検索

各 PCS AWS コンピューティングノードグループは、共有設定で EC2 インスタンスを起動できます。EC2 タグを使用して、AWS マネジメントコンソール または でコンピューティングノードグループのインスタンスを検索できます AWS CLI。

AWS マネジメントコンソール

コンピューティングノードグループインスタンスを検索するには

1. [AWS PCS コンソール](#)を開きます。
2. クラスターを選択します。
3. Compute ノードグループを選択します。
4. 作成したログインノードグループの ID を見つけます。
5. [EC2 コンソール](#)に移動し、インスタンスを選択します。
6. 次のタグを持つインスタンスを検索します。*node-group-id* をコンピューティングノードグループの ID (名前ではありません) に置き換えます。

```
aws:pcs:compute-node-group-id=node-group-id
```

7. (オプション) 検索フィールドのインスタンス状態の値を変更して、設定中または最近終了したインスタンスを検索できます。
8. タグ付けされたインスタンスのリストで、各インスタンスのインスタンス ID と IP アドレスを見つけてます。

AWS CLI

ノードグループインスタンスを検索するには、次のコマンドを使用します。コマンドを実行する前に、次の置換を行います。

- をクラスター AWS リージョン の *region-code* に置き換えます。例: us-east-1
- をコンピューティングノードグループの ID (名前ではなく) *node-group-id* に置き換えます。コンピューティングノードグループの ID を確認するには、「」を参照してください[PCS AWS でコンピューティングノードグループの詳細を取得する](#)。

- を pending や などの他のインスタンス状態 running に置き換え terminated で、他の状態の EC2 インスタンスを見つけます。

```
aws ec2 describe-instances \
  --region region-code --filters \
    "Name=tag:aws:pcs:compute-node-group-id,Values=node-group-id" \
    "Name=instance-state-name,Values=running" \
  --query 'Reservations[*].Instances[*].\
    {InstanceID:InstanceId,State:State.Name,PublicIP:PublicIpAddress,PrivateIP:PrivateIpAddress}'
```

このコマンドにより、以下のような出力が返されます。インスタンスがプライベートサブネットにある null 場合、 の値は PublicIP です。

```
[
  [
    {
      "InstanceID": "i-0123456789abcdefa",
      "State": "running",
      "PublicIP": "18.189.32.188",
      "PrivateIP": "10.0.0.1"
    }
  ]
]
```

Note

多数のインスタンス describe-instances を返すことが予想される場合は、複数のページにオプションを使用する必要があります。詳細については、Amazon Elastic Compute Cloud API リファレンスの「[DescribeInstances](#)」を参照してください。

PCS での Amazon EC2 AWS 起動テンプレートの使用

Amazon EC2 では、起動テンプレートに一連の設定を保存できるため、インスタンスを起動するときに個別に指定する必要はありません。AWS PCS には、コンピューティングノードグループを設定する柔軟な方法として起動テンプレートが組み込まれています。ノードグループを作成するときは、起動テンプレートを指定します。AWS PCS はそこから派生した起動テンプレートを作成します。このテンプレートには、サービスで確実に機能するための変換が含まれています。

カスタム起動テンプレートを記述する際のオプションと考慮事項を理解すると、PCS AWS で使用するオプションと考慮事項を記述するのに役立ちます。起動テンプレートの詳細については、「Amazon EC2 ユーザーガイド」の「[起動テンプレートからインスタンスを起動する](#)」を参照してください。

トピック

- [PCS AWS の起動テンプレートの概要](#)
- [基本的な起動テンプレートを作成する](#)
- [AWS PCS の Amazon EC2 ユーザーデータの使用](#)
- [PCS AWS でのキャパシティ予約](#)
- [便利な起動テンプレートパラメータ](#)

PCS AWS の起動テンプレートの概要

EC2 起動テンプレートに含めることができる [パラメータは 30 以上あり](#)、インスタンスの設定方法の多くの側面を制御します。ほとんどののは PCS AWS と完全に互換性がありますが、いくつかの例外があります。

EC2 Launch テンプレートの次のパラメータは、サービスによって直接管理する必要があるため、PCS AWS によって無視されます。

- インスタンスタイプ/インスタンスタイプの属性を指定する (InstanceRequirements) – AWS PCS は、属性ベースのインスタンス選択をサポートしていません。
- インスタンスタイプ (InstanceType) – ノードグループを作成するときにインスタンスタイプを指定します。
- 詳細/IAM インスタンスプロファイル (IamInstanceProfile) – ノードグループを作成または更新するときに指定します。

- 詳細/API 終了の無効化 (DisableApiTermination) – AWS PCS は、起動するノードグループインスタンスのライフサイクルを制御する必要があります。
- 詳細/API 停止の無効化 (DisableApiStop) – AWS PCS は、起動するノードグループインスタンスのライフサイクルを制御する必要があります。
- 詳細/停止 – 休止動作 (HibernationOptions) – AWS PCS はインスタンスの休止をサポートしていません。
- 詳細/Elastic GPU (ElasticGpuSpecifications) – Amazon Elastic Graphics は 2024 年 1 月 8 日にサポート終了しました。
- 高度な詳細/Elastic Inference (ElasticInferenceAccelerators) – Amazon Elastic Inference は、新規のお客様にはご利用いただけません。
- AAdvanced/CPU オプションを指定する/コアあたりのスレッド数 (ThreadsPerCore) – AWS PCS は、コアあたりのスレッド数を 1 に設定します。

これらのパラメータには、AWS PCS との互換性をサポートする特別な要件があります。

- ユーザーデータ (UserData) – これはマルチパートエンコードされている必要があります。
「[AWS PCS の Amazon EC2 ユーザーデータの使用](#)」を参照してください。
- Application and OS Images(ImageId) – これを含めることができます。ただし、ノードグループを作成または更新するときに AMI ID を指定すると、起動テンプレートの値が上書きされます。指定する AMI は PCS AWS と互換性がある必要があります。詳細については、「」を参照してください。[PCS 用の Amazon AWS マシンイメージ \(AMIs\)](#)。
- ネットワーク設定/ファイアウォール (セキュリティグループ)(SecurityGroups) – セキュリティグループ名のリストを AWS PCS 起動テンプレートで設定することはできません。起動テンプレートでネットワークインターフェイスを定義しない限り、セキュリティグループ IDs (SecurityGroupIds) のリストを設定できます。次に、インターフェイスごとにセキュリティグループ IDs を指定する必要があります。詳細については、「[PCS AWS のセキュリティグループ](#)」を参照してください。
- ネットワーク設定/高度なネットワーク設定 (NetworkInterfaces) – 単一のネットワークカードで EC2 インスタンスを使用し、特別なネットワーク設定を必要としない場合、AWS PCS はインスタンスネットワークを設定できます。複数のネットワークカードを設定するか、インスタンスで Elastic Fabric Adapter を有効にするには、を使用しますNetworkInterfaces。各ネットワークインターフェイスには、のセキュリティグループ IDs のリストが必要ですGroups。詳細については、「[PCS AWS の複数のネットワークインターフェイス](#)」を参照してください。
- 詳細/キャパシティ予約 (CapacityReservationSpecification) – これは設定できますが、AWS PCS CapacityReservationIdを使用する際に特定の を参照することはできません。ただ

し、キャパシティ予約グループを参照できます。このグループには、1 つ以上のキャパシティ予約が含まれます。詳細については、「[PCS AWS でのキャパシティ予約](#)」を参照してください。

基本的な起動テンプレートを作成する

AWS マネジメントコンソール または を使用して起動テンプレートを作成できます AWS CLI。

AWS マネジメントコンソール

起動テンプレートを作成するには

1. [Amazon EC2 コンソール](#)を開き、起動テンプレートを選択します。
2. [起動テンプレートの作成] を選択してください。
3. 起動テンプレートの名前と説明に、起動テンプレート名の一意の固有の名前を入力します。
4. 「キーペア名 (ログイン)」の「キーペア名」で、PCS によって管理される EC2 インスタンスへのログインに使用される SSH AWS キーペアを選択します。これはオプションですが推奨されます。
5. ネットワーク設定、次にファイアウォール (セキュリティグループ) で、ネットワークインターフェイスにアタッチするセキュリティグループを選択します。起動テンプレート内のすべてのセキュリティグループは、PCS クラスタ VPC AWS からのものである必要があります。少なくとも、以下を選択します。
 - AWS PCS クラスタとの通信を許可するセキュリティグループ
 - AWS PCS によって起動された EC2 インスタンス間の通信を許可するセキュリティグループ
 - (オプション) インタラクティブインスタンスへのインバウンド SSH アクセスを許可するセキュリティグループ
 - (オプション) コンピューティングノードがインターネットに発信接続できるようにするセキュリティグループ
 - (オプション) 共有ファイルシステムやデータベースサーバーなどのネットワークリソースへのアクセスを許可するセキュリティグループ (複数可) 。
6. 新しい起動テンプレート ID は、Amazon EC2 コンソールの起動テンプレートからアクセスできます。起動テンプレート ID の形式は です `lt-0123456789abcdef01`。

推奨される次のステップ

- 新しい起動テンプレートを使用して、PCS AWS コンピューティングノードグループを作成または更新します。

AWS CLI

起動テンプレートを作成するには

次のコマンドを使用して起動テンプレートを作成します。

- コマンドを実行する前に、次の置き換えを行います：
 - `region-code` を PCS を使用している AWS に置き換え AWS リージョン ます。
 - `my-launch-template-name` をテンプレートの名前に置き換えます。AWS リージョン 使用している AWS アカウント と に一意である必要があります。
 - `my-ssh-key-name` を任意の SSH キーの名前に置き換えます。
 - `sg-ExampleID1` と `sg-ExampleID2` を、EC2 インスタンスとスケジューラ間の通信と EC2 インスタンス間の通信を許可するセキュリティグループ IDs に置き換えます。このトラフィックをすべて有効にするセキュリティグループが 1 つしかない場合は、`sg-ExampleID2` とその前のカンマ文字を削除できます。セキュリティグループ IDs をさらに追加することもできます。起動テンプレートに含めるすべてのセキュリティグループは、PCS クラスター VPC AWS からのものである必要があります。

```
aws ec2 create-launch-template --region region-code \  
  --launch-template-name my-template-name \  
  --launch-template-data '{"KeyName":"my-ssh-key-name","SecurityGroupIds":  
  ["sg-ExampleID1", "sg-ExampleID2"]}'
```

AWS CLI は次のようなテキストを出力します。起動テンプレート ID は にありますLaunchTemplateId。

```
{  
  "LaunchTemplate": {  
    "LatestVersionNumber": 1,  
    "LaunchTemplateId": "lt-0123456789abcdef01",  
    "LaunchTemplateName": "my-launch-template-name",
```

```
    "DefaultVersionNumber": 1,  
    "CreatedBy": "arn:aws:iam::123456789012:user/Bob",  
    "CreateTime": "2019-04-30T18:16:06.000Z"  
  }  
}
```

推奨される次のステップ

- 新しい起動テンプレートを使用して、PCS AWS コンピューティングノードグループを作成または更新します。

AWS PCS の Amazon EC2 ユーザーデータの使用

インスタンスの起動時にcloud-init実行される起動テンプレートに EC2 ユーザーデータを指定できます。コンテンツタイプのユーザーデータブロックは、インスタンスが AWS PCS API に登録される前にcloud-config実行されますが、コンテンツタイプのユーザーデータブロックは、登録の完了後に実行されますが、Slurm デーモンの起動前にtext/x-shellscript実行されます。コンテンツタイプの詳細については[cloud-init](#) のドキュメントを参照してください。

ユーザーデータは、次のような一般的な設定シナリオを実行できます。

- [ユーザーまたはグループを含める](#)
- [パッケージのインストール](#)
- [パーティションとファイルシステムの作成](#)
- ネットワークファイルシステムのマウント

起動テンプレートのユーザーデータは、[MIME マルチパートアーカイブ](#)形式である必要があります。これは、ユーザーデータがノードグループ内のノードの設定に必要な他の AWS PCS ユーザーデータとマージされるためです。複数のユーザーデータブロックと単一の MIME マルチパートファイルを組み合わせることができます。

MIME マルチパートファイルには次のコンポーネントが含まれます。

- コンテンツの種類およびパート境界の宣言: Content-Type: multipart/mixed; boundary=="=BOUNDARY=="
- MIME バージョンの宣言: MIME-Version: 1.0
- 次のコンポーネントを含む 1 つ以上のユーザーデータブロック:

- ユーザーデータブロックの始まりを示す開始境界: `--==BOUNDARY==`。この境界の前の行は空白にしておく必要があります。
- ブロックのコンテンツタイプ宣言: `Content-Type: text/cloud-config; charset="us-ascii"`または `Content-Type: text/x-shellscript; charset="us-ascii"`。コンテンツタイプ宣言の後の行は空白にしておく必要があります。
- ユーザーデータのコンテンツ (例えば、シェルコマンドや `cloud-config` ディレクティブのリスト)。
- MIME マルチパートファイルの終わりを示す、終了境界: `--==BOUNDARY==--`。この境界の前の行は空白にしておく必要があります。

Note

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

```
{
  "LaunchTemplateName": "base64-user-data",
  "LaunchTemplateData": {
    "UserData":
"ewogICAgIkxhdW5jaFR1bXBsYXR1TmFtZSI6ICJpbmNyZWZzZS1jb250YWluZXItZm9sdW..."
  }
}
```

例

- [例: パッケージリポジトリからソフトウェアをインストールする](#)
- [例: S3 バケットからスクリプトを実行する](#)
- [例: グローバル環境変数の設定](#)
- [PCS AWS でのネットワークファイルシステムの使用](#)
- [例: EFS ファイルシステムを共有ホームディレクトリとして使用する](#)

例: パッケージリポジトリから PCS AWS 用のソフトウェアをインストールする

このスクリプトを起動テンプレート"userData"の の値として指定します。詳細については、「[AWS PCS の Amazon EC2 ユーザーデータの使用](#)」を参照してください。

このスクリプトは、cloud-config を使用して、起動時にノードグループインスタンスにソフトウェアパッケージをインストールします。詳細については、cloud-init ドキュメントの「[ユーザーデータ形式](#)」を参照してください。この例では、curlと をインストールしますllvm。

Note

インスタンスは、設定されたパッケージリポジトリに接続できる必要があります。

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

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

packages:
- python3-devel
- rust
- golang

--MYBOUNDARY--
```

例: S3 バケットから AWS PCS の追加スクリプトを実行する

このスクリプトを起動テンプレート"userData"の の値として指定します。詳細については、「[AWS PCS の Amazon EC2 ユーザーデータの使用](#)」を参照してください。

次のユーザーデータスクリプトは、cloud-config を使用して S3 バケットからスクリプトをインポートし、起動時にノードグループインスタンスで実行します。詳細については、cloud-init ドキュメントの「[ユーザーデータ形式](#)」を参照してください。

次の値を独自の詳細に置き換えます。

- **amzn-s3-demo-bucket** – アカウントが読み取ることができる S3 バケットの名前。

- **object-key** – インポートするスクリプトの S3 オブジェクトキー。これには、スクリプトの名前とバケットのフォルダ構造内の場所が含まれます。例えば、scripts/script.sh。詳細については、[「Amazon Simple Storage Service ユーザーガイド」の「フォルダを使用して Amazon S3 コンソールでオブジェクトを整理する」](#)を参照してください。
- **shell** – など、スクリプトの実行に使用する Linux シェルbash。

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

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

runcmd:
- aws s3 cp s3://amzn-s3-demo-bucket/object-key /tmp/script.sh
- /usr/bin/shell /tmp/script.sh

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

ノードグループの IAM インスタンスプロファイルは、バケットにアクセスする必要があります。次の IAM ポリシーは、上記のユーザーデータスクリプトのバケットの例です。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket",
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ]
    }
  ]
}
```

例: PCS AWS のグローバル環境変数を設定する

このスクリプトを起動テンプレート"userData"の の値として指定します。詳細については、「[AWS PCS の Amazon EC2 ユーザーデータの使用](#)」を参照してください。

次の例では/etc/profile.d、 を使用してノードグループインスタンスにグローバル変数を設定します。

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

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

#!/bin/bash
touch /etc/profile.d/awspcs-userdata-vars.sh
echo MY_GLOBAL_VAR1=100 >> /etc/profile.d/awspcs-userdata-vars.sh
echo MY_GLOBAL_VAR2=abc >> /etc/profile.d/awspcs-userdata-vars.sh

--MYBOUNDARY--
```

例: AWS PCS の共有ホームディレクトリとして EFS ファイルシステムを使用する

このスクリプトを起動テンプレート"userData"の の値として指定します。詳細については、「[AWS PCS の Amazon EC2 ユーザーデータの使用](#)」を参照してください。

この例では、 の EFS マウントの例を拡張[PCS AWS でのネットワークファイルシステムの使用](#)して、共有ホームディレクトリを実装します。/home の内容は、EFS ファイルシステムがマウントされる前にバックアップされます。その後、マウントが完了すると、コンテンツは共有ストレージ上の所定の場所にすばやくコピーされます。

このスクリプトの次の値を独自の詳細に置き換えます。

- */mount-point-directory* – EFS ファイルシステムをマウントするインスタンス上のパス。
- *filesystem-id* – EFS ファイルシステムのファイルシステム ID。

```
MIME-Version: 1.0
```



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

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

packages:
  - amazon-efs-utils

runcmd:
  - mkdir -p /tmp/home
  - rsync -a /home/ /tmp/home
  - echo "filesystem-id:/ /mount-point-directory efs tls,_netdev" >> /etc/fstab
  - mount -a -t efs defaults
  - rsync -a --ignore-existing /tmp/home/ /home
  - rm -rf /tmp/home/

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

例: パスワードレス SSH の有効化

共有ホームディレクトリの例に基づいて を構築し、SSH キーを使用してクラスターインスタンス間に SSH 接続を実装できます。共有ホームファイルシステムを使用するユーザーごとに、次のようなスクリプトを実行します。

```
#!/bin/bash

mkdir -p $HOME/.ssh && chmod 700 $HOME/.ssh
touch $HOME/.ssh/authorized_keys
chmod 600 $HOME/.ssh/authorized_keys

if [ ! -f "$HOME/.ssh/id_rsa" ]; then
  ssh-keygen -t rsa -b 4096 -f $HOME/.ssh/id_rsa -N ""
  cat ~/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
fi
```

Note

インスタンスは、クラスターノード間の SSH 接続を許可するセキュリティグループを使用する必要があります。

PCS AWS でのキャパシティ予約

オンデマンドキャパシティ予約または Amazon EC2 Capacity Blocks for ML を使用して、特定のアベイラビリティゾーンと特定の期間に Amazon EC2 キャパシティを予約し、必要なコンピューティングキャパシティを必要なときに利用できるようにすることができます。

オンデマンドキャパシティ予約 (ODCRs) を使用すると、特定のアベイラビリティゾーンの Amazon EC2 インスタンスのコンピューティングキャパシティを任意の期間予約できます。長期契約や前払いなしで、いつでも予約を作成およびキャンセルできます。ODCRsは、要件の変化に応じて変更できる柔軟なキャパシティ予約が必要な場合に最適です。詳細については、「Amazon Elastic Compute Cloud ユーザーガイド」の「[オンデマンドキャパシティ予約](#)」を参照してください。

ML 用 Amazon EC2 キャパシティブロックを使用すると、GPU ベースの高速コンピューティングインスタンスを将来の使用のために最大 8 週間前に予約できます。1~64 インスタンスのブロックは、1 日から 6 か月間予約できます。キャパシティブロックは、特定の時間に GPU 容量への保証付きアクセスを必要とする機械学習ワークロードに最適です。詳細については、「[Amazon Elastic Compute Cloud ユーザーガイド](#)」の「[ML のキャパシティブロック](#)」を参照してください。

トピック

- [PCS での ODCRs AWS の使用](#)
- [PCS での ML での Amazon EC2 AWS キャパシティブロックの使用](#)

PCS での ODCRs AWS の使用

PCS AWS がリザーブドインスタンスを使用する方法を選択できます。オープン ODCR を作成すると、AWS PCS またはアカウント内の他のプロセスによって起動された一致するインスタンスは、予約に対してカウントされます。ターゲット ODCR では、特定の予約 ID で起動されたインスタンスのみが予約に対してカウントされます。時間的制約のあるワークロードでは、ターゲット ODCRs がより一般的です。

PCS AWS コンピューティングノードグループを起動テンプレートに追加することで、ターゲット ODCR を使用するように設定できます。これを行う手順は次のとおりです。

1. [Amazon EC2 キャパシティ予約の作成ユーザーガイド](#) を使用して、ターゲットのオンデマンドキャパシティ予約 (ODCR) を作成します。
2. ODCR を起動テンプレートに関連付けます。そのためには、次の 2 つの方法があります。

- a. 直接 ODCR の関連付け: 起動テンプレートで ODCR ID を直接参照します。このアプローチは厳密な容量制御を提供し、インスタンスのバックファイルをサポートしていません (コンピューティングノードグループが ODCR で利用可能なインスタンスよりも多くのインスタンスをリクエストした場合、追加のインスタンスは起動されません)。
 - b. キャパシティ予約グループの関連付け: ODCR をキャパシティ予約グループに追加し、起動テンプレートでグループを参照します。このアプローチはインスタンスのバックファイルをサポートし、予約容量を超えた場合に AWS PCS が追加のオンデマンドインスタンスを起動できるようにします。
3. 起動テンプレートを使用するように PCS AWS コンピューティングノードグループを作成または更新します。詳細については、[AWS 「PCS Compute Node Groups ユーザーガイド」](#) を参照してください。
- コンピューティングノードグループの `purchaseOption` を に設定します ONDEMAND。

例: ターゲット ODCR で hpc6a.48xlarge インスタンスを予約して使用する

このコマンド例では、32 個の hpc6a.48xlarge インスタンスのターゲット ODCR を作成します。プレイacementグループでリザーブドインスタンスを起動するには、コマンド `--placement-group-arn` に を追加します。 `--end-date` および を使用して終了日を定義できます。そうしないと `--end-date-type`、予約は手動で終了するまで続行されます。

```
aws ec2 create-capacity-reservation \
  --instance-type hpc6a.48xlarge \
  --instance-platform Linux/UNIX \
  --availability-zone us-east-2a \
  --instance-count 32 \
  --instance-match-criteria targeted
```

このコマンドの結果は、新しい ODCR の ARN になります。ODCR ID は、ARN から、`"arn:aws:ec2:us-east-2:123456789012:capacity-reservation/ODCR-ID"` または [Amazon EC2 DescribeCapacityReservations](#) を使用して取得できます。

ODCR の直接関連付け: ODCR ID を起動テンプレートに追加します。ODCR ID を参照する起動テンプレートの例を次に示します。

```
{
  "CapacityReservationSpecification": {
```

```

    "CapacityReservationTarget": {
      "CapacityReservationId": "cr-1234567890abcdef1"
    }
  }
}

```

キャパシティ予約グループの関連付け: キャパシティ予約グループを作成し、そのグループを起動テンプレートに追加します。次のコマンドは、 という名前のキャパシティ予約グループを作成しますEXAMPLE-CR-GROUP。

```

aws resource-groups create-group \
  --name EXAMPLE-CR-GROUP \
  --configuration \
    '{"Type": "AWS::EC2::CapacityReservationPool"}' \
    '{"Type": "AWS::ResourceGroups::Generic", "Parameters": [{"Name": "allowed-resource-types", "Values": ["AWS::EC2::CapacityReservation"]}]}

```

次のコマンドは、キャパシティ予約グループに ODCR を追加します。

```

aws resource-groups group-resources --group EXAMPLE-CR-GROUP \
  --resource-arns arn:aws:ec2:us-east-2:123456789012:capacity-reservation/cr-1234567890abcdef1

```

ODCR を作成してキャパシティ予約グループに追加すると、起動テンプレートに追加して PCS AWS コンピューティングノードグループに接続できるようになりました。キャパシティ予約グループを参照する起動テンプレートの例を次に示します。

```

{
  "CapacityReservationSpecification": {
    "CapacityReservationResourceGroupArn": "arn:aws:resource-groups:us-east-2:123456789012:group/EXAMPLE-CR-GROUP"
  }
}

```

最後に、PCS AWS コンピューティングノードグループを作成または更新して hpc6a.48xlarge インスタンスを使用し、ODCR を参照する起動テンプレートを使用します。静的ノードグループの場合、最小インスタンス数と最大インスタンス数を予約のサイズ (32) に設定します。動的ノードグループの場合、最小インスタンス数を 0 に設定し、最大インスタンス数を希望のインスタンスサイズに設定します。

この例は、1つのコンピューティングノードグループにプロビジョニングされた1つの ODCR の簡単な実装です。ただし、AWS PCS は他の多くの設計をサポートしています。たとえば、複数のコンピューティングノードグループ間で大きな ODCR またはキャパシティ予約グループを分割できます。または、別の AWS アカウントが作成して共有した ODCRs を使用することもできます。

詳細については、Amazon Elastic Compute Cloud ユーザーガイドの [「ML の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」](#) を参照してください。

PCS での ML での Amazon EC2 AWS キャパシティブロックの使用

Amazon EC2 Capacity Blocks for ML は、短期間のワークロードをサポートするために、特定の日時範囲内で GPU ベースの高速コンピューティングインスタンスを事前に予約するための Amazon EC2 購入オプションです。キャパシティブロック内で実行されるインスタンスは、Amazon EC2 UltraClusters 内に自動的に互いに近く配置され、低レイテンシーでペタビットスケールのノンブッキングネットワークングを実現します。詳細については、[「Amazon Elastic Compute Cloud ユーザーガイド」](#)の [「ML のキャパシティブロック」](#) を参照してください。

起動テンプレートを使用して、コンピューティングノードグループのインスタンスを起動するときに PCS AWS にキャパシティブロックを使用させることができます。

Note

AWS PCS は、Slurm バージョン 24.05 以降、キャパシティブロックのサポートを導入しました。

制限事項

- AWS PCS は、P5en, P5e, P5P4d インスタンスファミリーのキャパシティブロックのみをサポートします。
- コンピューティングノードグループは、一度に1つのキャパシティブロックにのみ関連付けることができます。
- コンピューティングノードグループを、複数のキャパシティブロックを組み合わせたキャパシティ予約グループに関連付けることはできません。
- PCS で使用するキャパシティブロックは、scheduledまたは AWS active状態である必要があります。などの他の状態でキャパシティブロックを使用することはできません payment-failed。詳細については、「Amazon Elastic Compute Cloud ユーザーガイド」の [「キャパシティブロックの表示」](#) を参照してください。

キャパシティブロックの有効期限

キャパシティブロックは、特定の日付と時刻の範囲に制限されます。キャパシティブロックの有効期限が切れた場合:

- その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ノードグループは引き続き存在し、同じキューに関連付けられます。
- コンピューティングノードグループ内のすべてのインスタンスが終了し、Slurm 設定に基づいてアクティブなジョブが失敗する可能性があります。
- AWS PCS はコンピューティングノードグループで新しいインスタンスを起動できません。
- キューに入れられたジョブまたは新しく送信されたジョブはすべて、別のコンピューティングノードグループがキューにアタッチされるか、新しいキャパシティブロックを指定する新しい起動テンプレートを使用するようにコンピューティングノードグループを更新するまで、保留中の状態のままになります。

キャパシティブロックを使用するように AWS PCS コンピューティングノードグループを設定する

キャパシティブロックをコンピューティングノードグループに関連付けるには

1. キャパシティブロックを指定する PCS AWS 用の Amazon EC2 起動テンプレートを作成します。PCS の起動テンプレートの作成の詳細については、AWS 「」を参照してください[PCS での Amazon EC2 AWS 起動テンプレートの使用](#)。

起動テンプレートには以下が含まれている必要があります。

- MarketType の値は に設定InstanceMarketOptionsする必要がありますcapacity-block。
 - 有効な CapacityReservationSpecificationを持つ CapacityReservationId
 - 購入したキャパシティブロックのインスタンスタイプInstanceTypeに一致する有効な。
2. 起動テンプレートを使用するコンピューティングノードグループを作成します。詳細については、「[PCS AWS でのコンピューティングノードグループの作成](#)」を参照してください。既存のコンピューティングノードグループを更新して、起動テンプレートを使用することもできます。詳細については、「[PCS AWS コンピューティングノードグループの更新](#)」を参照してください。

コンピューティングノードグループを作成または更新するとき :

- コンピューティングノードグループを作成または更新するために使用する IAM ID には、次のアクセス許可が必要です。

```
ec2:DescribeCapacityReservations
```

詳細については、「[PCS AWS の最小アクセス許可](#)」を参照してください。

- キャパシティブロックは scheduled または active 状態である必要があります。
- コンピューティングノードグループの purchaseOption を に設定しますCAPACITY_BLOCK。
- maxInstanceCount コンピューティングノードグループのは、キャパシティブロックのサイズを超えることはできません。
- コンピューティングノードグループのアベイラビリティーゾーンは、コンピューティングノードグループのサブネットアベイラビリティーゾーンの 1 つと一致する必要があります。

Important

更新時にコンピューティングノードグループのインスタンスタイプを変更することはできません。キャパシティブロックは、コンピューティングノードグループと同じインスタンスタイプでのみ使用できます。別のインスタンスタイプのキャパシティブロックを使用する場合は、新しいコンピューティングノードグループを作成する必要があります。

PCS AWS でのキャパシティブロックの使用に関するよくある質問

キャパシティブロックに対して支払いを行ったところ、すぐに PCS AWS で使用しようとしたのですが、コンピューティングノードグループの作成に失敗しました。何が起きたのか。

キャパシティブロックが scheduled または active 状態ではない可能性があります。キャパシティブロックが scheduled または になったら、もう一度試してください active。

PCS AWS でキャパシティブロックを使用していて、有効期限が切れる前に拡張機能を購入しました。PCS AWS で引き続き使用するにはどうすればよいですか？

AWS PCS でキャパシティブロックを引き続き使用するために何もする必要はありません。拡張機能の支払いが成功すると、キャパシティブロックの終了日が更新されます。キャパシティブロックの有効期限が切れない限り、コンピューティングノードグループは引き続き動作します。

拡張機能の支払いが失敗した場合、キャパシティブロックは残りactive、コンピューティングノードグループはキャパシティブロックが元の終了日に期限切れになるまで動作します。

キャパシティブロックの有効期限が切れると、キューに入れられたジョブと実行中のジョブはどうなりますか？

キャパシティブロックの有効期限が切れる前に開始されなかったキューに入れられたジョブは、別のコンピューティングノードグループをキューにアタッチするか、コンピューティングノードグループを新しいキャパシティブロックで更新するまで保留中のままになります。ジョブは引き続きキューに送信できます。Slurm 設定はアクティブなジョブに影響します。デフォルトでは、アクティブなジョブは自動的に再キューに入れられますが、エラーが発生したり、失敗したりする可能性があります。

キャパシティブロックの有効期限が切れました。何かすべきですか？

何もする必要はありません。Amazon EC2 コンソールで EC2 キャパシティ予約のステータスを確認できます。キャパシティブロックの有効期限が切れても、その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ノードグループは引き続き存在し、同じキューを処理します。コンピューティングノードグループには、ジョブを実行するインスタンスがありません。コンピューティングノードグループを削除するか、キューとの関連付けを解除して、ユーザーが実行されないジョブを送信しないようにできます。

AWS PCS コンピューティングノードグループで新しいキャパシティブロックを使用します。どうすればよいですか？

新しいキャパシティブロックを使用するには、新しいコンピューティングノードグループを作成することをお勧めします。詳細については、「[キャパシティブロックを使用するように AWS PCS コンピューティングノードグループを設定する](#)」を参照してください。

クラスターとサービス間で 1 つのキャパシティブロックを共有するにはどうすればよいですか？

キャパシティブロックは、複数のクラスターとサービスに分割できます。たとえば、PCS-Cluster-1 に 20 個のノードを持つ 64 個の p5.48xlarge インスタンス、PCS-Cluster-2 に 16 個のノード、および他のサービスの残りのノードを持つキャパシティブロックを分割するには、PCS-Cluster-1 の場合は minInstanceCount との両方 maxInstanceCount を 20、PCS-Cluster-2 の場合は 16 に設定します。PCS-Cluster-1

1 つ以上のキャパシティブロックまたは 1 つのコンピューティングノードグループの合計キャパシティを使用できますか？

いいえ。1 つのコンピューティングノードグループに関連付けることができるキャパシティブロックは 1 つだけです。AWS PCS は、複数のキャパシティブロックを組み合わせたキャパシティ予約グループをサポートしていません。

キャパシティブロックがいつ開始または期限切れになるかを知るにはどうすればよいですか？

AWS PCS とは無関係に、Amazon EC2 はキャパシティブロック予約が開始されると EventBridge を介して Capacity Block Reservation Delivered イベントを送信し、キャパシティブロック予約の有効期限が切れる 40 分前に Capacity Block Reservation Expiration Warning イベントを送信します。詳細については、[「Amazon Elastic Compute Cloud ユーザーガイド」の EventBridge を使用したキャパシティブロックのモニタリング](#)」を参照してください。

Slurm はキャパシティブロックの状態をどのように追跡しますか？

を実行して sinfo、PCS AWS がキャパシティブロックを使用する方法を理解できます。次の出力例では、キューはキャパシティ active ブロックから 4 つのインスタンスを実行するコンピュティングノードグループに関連付けられています。ノードは Slurm idle 状態です (使用可能で、どのジョブにもまだ割り当てられていません)。

```
$ sinfo
PARTITION AVAIL TIMELIMIT NODES STATE NODELIST
fanout up infinite 4 idle node-fanout-[1-4]
```

ノードが代わりに maint 状態になっている場合は、scontrol show res を実行して、この状態を制御する Slurm 予約の詳細を表示できます。次の出力例では、キャパシティブロックは将来の開始日 scheduled になります。

```
$ scontrol show res

ReservationName=node-fanout-scheduled StartTime=2025-10-14T13:09:17
EndTime=2025-10-14T13:11:17 Duration=00:02:00
  Nodes=node-fanout-[1-4] NodeCnt=4 CoreCnt=16 Features=(null) PartitionName=(null)
Flags=MAINT,SPEC_NODES
  TRES=cpu=16

  Users=root Groups=(null) Accounts=(null) Licenses=(null) State=ACTIVE
BurstBuffer=(null)
  MaxStartDelay=(null)

  Comment=node-fanout Scheduled
```

キャパシティブロックが共有されているために、キャパシティの起動中に発生するエラーが かどうかを確認するにはどうすればよいですか？

Amazon EC2 コンソールでキャパシティ予約をチェックして、キャパシティブロックからアクティブにプロビジョニングされているインスタンスの数を確認します。各インスタンスのタグをチェックして、どのサービスまたはクラスターがそれを使用しているかを確認します。たとえば、AWS PCS のすべてのインスタンスには AWS、インスタンスが属 `aws:pcs:cluster-id = pcs_l0mizqyk5o | aws:pcs:compute-node-group-id = pcs_ic7onkmfqk` するクラスターとコンピューティングノードグループを示すなどの PCS タグがあります。その後、キャパシティブロックが最大キャパシティにあるかどうかを確認できます。

`scontrol show nodes` を使用して、AWS PCS クラスターのキャパシティブロックノードがトリガーしているかどうかを確認します `ReservationCapacityExceeded`。

```
[root@ip-172-16-10-54 ~]# scontrol show nodes test-node-8-gamma-cb-2
NodeName=test-8-gamma-cb-2 CoresPerSocket=1
CPUAlloc=0 CPUEfctv=8 CPUTot=8 CPULoad=0.00
AvailableFeatures=test-8-gamma-cb,gpu
ActiveFeatures=test-8-gamma-cb,gpu
Gres=gpu:H100:1
NodeAddr=test-8-gamma-cb-2 NodeHostName=test-8-gamma-cb-2
RealMemory=249036 AllocMem=0 FreeMem=N/A Sockets=8 Boards=1
State=IDLE+CLOUD+POWERING_DOWN ThreadsPerCore=1 TmpDisk=0 Weight=1 Owner=N/A
MCS_label=N/A
Partitions=my-q
BootTime=None SlurmdStartTime=None
LastBusyTime=Unknown ResumeAfterTime=None
CfgTRES=cpu=8,mem=249036M,billing=8
AllocTRES=
CurrentWatts=0 AveWatts=0
Reason=Failed to launch backing instance (Error Code:
ReservationCapacityExceeded) [root@2025-08-28T15:15:33]
```

複数のコンピューティングノードグループが同じキューにアタッチされている場合、キャパシティブロックバックされたインスタンスでジョブを強制的に実行するにはどうすればよいですか？

Slurm の機能と制約を使用して、特定のノードセットにジョブをロックできます。コンピューティングノードグループごとに Slurm の重みを設定しないことをお勧めします。これは、maint 状態ではないノードでのみ機能するためです。

便利な起動テンプレートパラメータ

このセクションでは、AWS PCS で広く役立つ起動テンプレートパラメータについて説明します。

詳細な CloudWatch モニタリングを有効にする

起動テンプレートパラメータを使用して、短い間隔で CloudWatch メトリクスの収集を有効にできます。

AWS マネジメントコンソール

起動テンプレートを作成または編集するためのコンソールページで、このオプションは詳細セクションにあります。詳細 CloudWatch モニタリングを有効に設定します。

YAML

```
Monitoring:
  Enabled: True
```

JSON

```
{"Monitoring": {"Enabled": "True"}}
```

詳細については、Linux [インスタンス用 Amazon Elastic Compute Cloud ユーザーガイドのインスタンスの詳細な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。

インスタンスメタデータサービスバージョン 2 (IMDS v2)

EC2 インスタンスで IMDS v2 を使用すると、セキュリティが大幅に強化され、AWS 環境内のインスタンスメタデータへのアクセスに関連する潜在的なリスクを軽減できます。

AWS マネジメントコンソール

起動テンプレートを作成または編集するためのコンソールページで、このオプションは詳細セクションにあります。メタデータを有効、メタデータバージョンを V2 のみ (トークンが必要)、メタデータレスポンスホップ制限を 4 に設定します。

YAML

```
MetadataOptions:
```

```
HttpEndpoint: enabled
HttpTokens: required
HttpPutResponseHopLimit: 4
```

JSON

```
{
  "MetadataOptions": {
    "HttpEndpoint": "enabled",
    "HttpPutResponseHopLimit": 4,
    "HttpTokens": "required"
  }
}
```

.

AWS PCS キュー

AWS PCS キューは、スケジューラのワークキューのネイティブ実装に対する軽量な抽象化です。Slurm の場合、PCS AWS キューは Slurm パーティションと同等です。

ユーザーは、1 つ以上のコンピューティングノードグループによって提供されるノードで実行するようにスケジュールされるまで、ジョブが存在するキューに送信します。PCS AWS クラスターには、複数のジョブキューを含めることができます。例えば、優先度の高いジョブには Amazon EC2 オンデマンドインスタンスを使用するキューを作成し、優先度の低いジョブには Amazon EC2 スポットインスタンスを使用する別のキューを作成できます。

トピック

- [PCS AWS でのキューの作成](#)
- [PCS AWS キューの更新](#)
- [PCS AWS でのキューの削除](#)

PCS AWS でのキューの作成

このトピックでは、使用可能なオプションの概要と、PCS AWS でキューを作成するときに考慮すべき点について説明します。

Note

キューにカスタム Slurm 設定を設定して、パーティション固有のスケジューリングポリシーとリソース管理を実装できます。詳細については、「[PCS でのカスタム Slurm AWS 設定の設定](#)」を参照してください。

前提条件

- AWS PCS クラスター - キューは、特定の PCS AWS クラスターに関連してのみ作成できます。
- 1 つ以上の AWS PCS コンピューティングノードグループ - キューは少なくとも 1 つの PCS AWS コンピューティングノードグループに関連付ける必要があります。

PCS AWS でキューを作成するには

キューは、AWS マネジメントコンソール または を使用して作成できます AWS CLI。

AWS マネジメントコンソール

コンソールを使用してキューを作成するには

1. [AWS PCS コンソール](#)を開きます。
2. キューのクラスターを選択します。キューに移動し、キューの作成を選択します。
3. キュー設定セクションで、次の値を指定します。
 - a. キュー名 – キューの名前。この名前には英数字 (大文字と小文字が区別されます) とハイフンのみを使用できます。アルファベット文字で始まり、25 文字を超えることはできません。名前はクラスター内で一意である必要があります。
 - b. コンピューティングノードグループ – このキューを処理するコンピューティングノードグループを 1 つ以上選択します。コンピューティングノードグループは、複数のキューに関連付けることができます。
4. (オプション) 追加スケジューラ設定セクションで、パラメータ名と値のペアを追加して、追加の Slurm 設定を設定できます。サポートされているパラメータの完全なリストについては、「」を参照してください[PCS キューのカスタム Slurm AWS 設定](#)。
5. (オプション) タグで、PCS AWS キューにタグを追加します。
6. [キューの作成]を選択します。Status フィールドには、PCS AWS がキューを作成している間に作成中と表示されます。キューの作成には数分かかる場合があります。

推奨される次のステップ

- 新しいキューにジョブを送信します。

AWS CLI

を使用してキューを作成するには AWS CLI

キューを作成するには、次のコマンドを使用します。次の置換を行います。

1. *region-code* をクラスターの AWS リージョンに置き換えます。例えば、us-east-1。

2. *my-queue* をキューの名前に置き換えます。この名前には英数字 (大文字と小文字が区別されます) とハイフンのみを使用できます。アルファベット文字で始まり、25 文字を超えることはできません。名前はクラスター内で一意である必要があります。
3. *my-cluster* をクラスターの名前または ID に置き換えます。
4. *compute-node-group-id* を、キューを処理するコンピューティングノードグループの ID に置き換えます。例えば、pcs_abcdef12345。

 Note

キューを作成するときは、名前ではなく、コンピューティングノードグループの ID を指定する必要があります。

```
aws pcs create-queue --region region-code \  
  --queue-name my-queue \  
  --cluster-identifier my-cluster \  
  --compute-node-group-configurations \  
    computeNodeGroupId=compute-node-group-id
```

Example- カスタム Slurm 設定でキューを作成する

```
aws pcs create-queue --region region-code \  
  --queue-name my-queue \  
  --cluster-identifier my-cluster \  
  --compute-node-group-configurations \  
    computeNodeGroupId=compute-node-group-id \  
  --slurm-configuration \  
    'slurmCustomSettings=[{parameterName=Default,parameterValue=YES}]'
```

詳細については、「[PCS キューのカスタム Slurm AWS 設定](#)」を参照してください。

キューの作成には数分かかる場合があります。次のコマンドを使用して、キューのステータスをクエリできます。ステータスが `ACTIVE` に達するまで、ジョブをキューに送信することはできません。

```
aws pcs get-queue --region region-code \  
  --cluster-identifier my-cluster \  
  --queue-identifier my-queue
```

推奨される次のステップ

- 新しいキューにジョブを送信する

PCS AWS キューの更新

このトピックでは、使用可能なオプションの概要と、PCS AWS キューを更新するときに考慮すべき点について説明します。Slurm のカスタム設定の詳細については、「」を参照してください[PCS キューのカスタム Slurm AWS 設定](#)。

PCS AWS キューを更新する際の考慮事項

キューの更新は実行中のジョブには影響しませんが、キューの更新中にクラスターが新しいジョブを受け入れることができない場合があります。

PCS AWS キューを更新するには

AWS マネジメントコンソール または を使用してキュー AWS CLI を更新できます。

AWS マネジメントコンソール

キューを更新するには

1. で PCS AWS コンソールを開きます。 <https://console.aws.amazon.com/pcs/home#/clusters>
2. キューを更新するクラスターを選択します。
3. キューに移動し、更新するキューに移動し、編集を選択します。
4. キュー設定セクションで、次のいずれかの値を更新します。
 - ノードグループ – キューとの関連付けからコンピューティングノードグループを追加または削除します。
 - 追加のスケジューラ設定 – キューのカスタム Slurm 設定を追加、変更、または削除します。詳細については、「[PCS キューのカスタム Slurm AWS 設定](#)」を参照してください。
 - タグ – キューのタグを追加または削除します。
5. [更新] を選択します。ステータスフィールドには、変更が適用されている間の更新が表示されます。

 Important

キューの更新には数分かかる場合があります。

AWS CLI

キューを更新するには

1. 次のコマンドを使用してキューを更新します。コマンドを実行する前に、次の置き換えを行います：
 - a. *region-code* AWS リージョン を、クラスターを作成する に置き換えます。
 - b. *my-queue* をキューの名前または に置き換えcomputeNodeIdます。
 - c. *my-cluster* をクラスターの名前または に置き換えclusterIdます。
 - d. コンピューティングノードグループの関連付けを変更するには、 の更新されたリストを指定します--compute-node-group-configurations。
 - たとえば、2 番目のコンピューティングノードグループ を追加するにはcomputeNodeGroupExampleID2:

```
--compute-node-group-configurations  
computeNodeId=computeNodeGroupExampleID1,computeNodeGroupId=computeNodeGroupExampleID2
```

```
aws pcs update-queue --region region-code \  
  --queue-identifier my-queue \  
  --cluster-identifier my-cluster \  
  --compute-node-group-configurations \  
  computeNodeId=computeNodeGroupExampleID1
```

Example– カスタム Slurm 設定でキューを更新する

```
aws pcs update-queue --region region-code \  
  --queue-identifier my-queue \  
  --cluster-identifier my-cluster \  
  --slurm-configuration \  
  'slurmCustomSettings=[{parameterName=Default,parameterValue=YES}]'
```

詳細については、「[PCS キューのカスタム Slurm AWS 設定](#)」を参照してください。

2. キューの更新には数分かかる場合があります。次のコマンドを使用して、キューのステータスをクエリできます。ステータスが `ACTIVE` に達するまで、ジョブをキューに送信することはできません。

```
aws pcs get-queue --region region-code \  
  --cluster-identifier my-cluster \  
  --queue-identifier my-queue
```

推奨される次のステップ

- 更新されたキューにジョブを送信します。

PCS AWS でのキューの削除

このトピックでは、PCS AWS で キューを削除する方法の概要を説明します。

キューを削除する際の考慮事項

- キューで実行中のジョブがある場合、キューが削除されるとスケジューラによってジョブが終了します。キュー内の保留中のジョブはキャンセルされます。キュー内のジョブが終了するのを待つか、スケジューラのネイティブコマンド (Slurm `scancel` の場合など) を使用して手動で停止/キャンセルすることを検討してください。

キューの削除

AWS マネジメントコンソール または `aws` を使用してキュー AWS CLI を削除できます。

AWS マネジメントコンソール

キューを削除するには

1. [AWS PCS コンソール](#)を開きます。
2. キューのクラスターを選択します。
3. キューに移動し、削除するキューを選択します。
4. [削除] を選択します。

5. Status フィールドには が表示されますDeleting。完了までに数分かかることがあります。

 Note

スケジューラにネイティブなコマンドを使用して、キューが削除されたことを確認できます。たとえば、Slurm queueには `sinfo` または `scancel` を使用します。

AWS CLI

キューを削除するには

- キューを削除するには、次のコマンドを使用します。
 - `region-code` をクラスターがある AWS リージョン に置き換え ます。
 - `my-queue` をキューの名前または ID に置き換えます。
 - `my-cluster` をクラスターの名前または ID に置き換えます。

```
aws pcs delete-queue --region region-code \  
    --queue-identifier my-queue \  
    --cluster-identifier my-cluster
```

キューの削除には数分かかる場合があります。

 Note

スケジューラにネイティブなコマンドを使用して、キューが削除されたことを確認できます。たとえば、Slurm queueには `sinfo` または `scancel` を使用します。

AWS PCS ログインノード

AWS PCS クラスターでは通常、インタラクティブなアクセスとジョブ管理をサポートするために、少なくとも 1 つのログインノードが必要です。これを実現する方法は、ログインノード機能用に設定された静的 PCS AWS コンピューティングノードグループを使用することです。ログインノードとして機能するスタンドアロン EC2 インスタンスを設定することもできます。

トピック

- [PCS AWS コンピューティングノードグループを使用してログインノードを提供する](#)
- [スタンドアロンインスタンス AWS を PCS ログインノードとして使用する](#)
- [スタンドアロンログインノードを PCS AWS の複数のクラスターに接続する](#)

PCS AWS コンピューティングノードグループを使用してログインノードを提供する

このトピックでは、推奨される設定オプションの概要と、AWS PCS コンピューティングノードグループを使用してクラスターへの永続的でインタラクティブなアクセスを提供するときに考慮すべき点について説明します。

ログインノードの AWS PCS コンピューティングノードグループの作成

運用上、これは通常のコンピューティングノードグループの作成と大きく変わりません。ただし、いくつかの主要な設定選択肢があります。

- コンピューティングノードグループ内の少なくとも 1 つの EC2 インスタンスの静的スケーリング設定を設定します。
- インスタンスが再利用されないように、オンデマンド購入オプションを選択します (複数可)。
- ログインなど、コンピューティングノードグループの情報名を選択します。
- ログインノードインスタンス (複数可) に VPC の外部からアクセスできるようにする場合は、パブリックサブネットの使用を検討してください。
- SSH アクセスを許可する場合、起動テンプレートには、選択した IP アドレスに SSH ポートを公開するセキュリティグループが必要です。
- IAM インスタンスプロファイルには、エンドユーザーに付与させたい AWS アクセス許可のみが必要です。詳細については、「[AWS Parallel Computing Service の IAM インスタンスプロファイル](#)」を参照してください。

- AWS Systems Manager Session Manager がログインインスタンスを管理することを許可することを検討してください。
- インスタンスの AWS 認証情報へのアクセスを管理者ユーザーのみに制限することを検討してください。
- ログインノード (複数可) が継続的に実行されるため、通常のコンピューティングノードグループよりも安価なインスタンスタイプを選択します。
- 他のコンピューティングノードグループと同じ (または派生的な) AMI を使用して、すべてのインスタンスに同じソフトウェアがインストールされていることを確認します。AMIs「」を参照してください。 [PCS 用の Amazon AWS マシンイメージ \(AMIs\)](#)
- コンピューティングインスタンスと同じネットワークファイルシステム (Amazon EFS、Amazon FSx for Lustre など) マウントをログインノードに設定します。詳細については、「[PCS AWS でのネットワークファイルシステムの使用](#)」を参照してください。

ログインノードにアクセスする

新しいコンピューティングノードグループが ACTIVE ステータスになったら、作成した EC2 インスタンス (複数可) を見つけてログインできます。詳細については、「[PCS AWS でのコンピューティングノードグループインスタンスの検索](#)」を参照してください。

ログインノードの PCS AWS コンピューティングノードグループの更新

UpdateComputeNodeGroup を使用してログインノードグループを更新できます。ノードグループの更新プロセスの一環として、実行中のインスタンスが置き換えられます。これにより、インスタンス上のアクティブなユーザーセッションまたはプロセスが中断されることに注意してください。実行中またはキューに入れられた Slurm ジョブは影響を受けません。詳細については、「[PCS AWS コンピューティングノードグループの更新](#)」を参照してください。

コンピューティングノードグループで使用される起動テンプレートを編集することもできます。UpdateComputeNodeGroup を使用して、更新された起動テンプレートをコンピューティングノードグループに適用する必要があります。コンピューティングノードグループで起動された新しい EC2 インスタンスは、更新された起動テンプレートを使用します。詳細については、「[PCS での Amazon EC2 AWS 起動テンプレートの使用](#)」を参照してください。

ログインノードの AWS PCS コンピューティングノードグループの削除

AWS PCS のコンピューティングノードグループの削除メカニズムを使用して、ログインノードグループを更新できます。実行中のインスタンスは、ノードグループの削除の一部として終了します。

これにより、インスタンス上のアクティブなユーザーセッションまたはプロセスが中断されることに注意してください。実行中またはキューに入れられた Slurm ジョブは影響を受けません。詳細については、「[PCS AWS でのコンピューティングノードグループの削除](#)」を参照してください。

スタンドアロンインスタンス AWS を PCS ログインノードとして使用する

独立した EC2 インスタンスを設定して、PCS AWS クラスターの Slurm スケジューラとやり取りできます。これは、PCS クラスターと連携するが、PCS AWS 管理の外部で動作するログインノード、ワークステーション、または専用ワークフロー AWS 管理ホストを作成するのに役立ちます。これを行うには、各スタンドアロンインスタンスが以下を行う必要があります。

1. 互換性のある Slurm ソフトウェアバージョンがインストールされていること。
2. AWS PCS クラスターの Slurmctld エンドポイントに接続できる。
3. Slurm Auth デーモンと Cred Kiosk デーモン (sackd) を AWS PCS クラスターのエンドポイントとシークレットで適切に設定します。詳細については、Slurm ドキュメントの「[sackd](#)」を参照してください。

このチュートリアルは、PCS AWS クラスターに接続する独立したインスタンスを設定するのに役立ちます。

目次

- [ステップ 1 – ターゲット PCS AWS クラスターのアドレスとシークレットを取得する](#)
- [ステップ 2 – EC2 インスタンスを起動する](#)
- [ステップ 3 – インスタンスに Slurm をインストールする](#)
- [ステップ 4 – クラスターシークレットを取得して保存する](#)
- [ステップ 5 – PCS AWS クラスターへの接続を設定する](#)
- [ステップ 6 – \(オプション\) 接続をテストする](#)

ステップ 1 – ターゲット PCS AWS クラスターのアドレスとシークレットを取得する

次のコマンド AWS CLI で を使用して、ターゲット AWS PCS クラスターの詳細を取得します。コマンドを実行する前に、次の置き換えを行います：

- *region-code* を、ターゲットクラスター AWS リージョン が実行されている に置き換えます。
- *cluster-ident* をターゲットクラスターの名前または識別子に置き換えます。

```
aws pcs get-cluster --region region-code --cluster-identifier cluster-ident
```

コマンドは、この例のような出力を返します。

```
{
  "cluster": {
    "name": "get-started",
    "id": "pcs_123456abcd",
    "arn": "arn:aws:pcs:us-east-1:111122223333:cluster/pcs_123456abcd",
    "status": "ACTIVE",
    "createdAt": "2024-12-17T21:03:52+00:00",
    "modifiedAt": "2024-12-17T21:03:52+00:00",
    "scheduler": {
      "type": "SLURM",
      "version": "25.05"
    },
    "size": "SMALL",
    "slurmConfiguration": {
      "authKey": {
        "secretArn": "arn:aws:secretsmanager:us-east-1:111122223333:secret:pcs!slurm-secret-pcs_123456abcd-a12ABC",
        "secretVersion": "ef232370-d3e7-434c-9a87-ec35c1987f75"
      }
    },
    "networking": {
      "subnetIds": [
        "subnet-0123456789abcdef0"
      ],
      "securityGroupIds": [
        "sg-0123456789abcdef0"
      ]
    },
    "endpoints": [
      {
        "type": "SLURMCTLD",
        "privateIpAddress": "10.3.149.220",
        "port": "6817"
      }
    ]
  }
}
```

```
}  
}
```

このサンプルでは、クラスター Slurm コントローラーエンドポイントの IP アドレスは 10.3.149.220、ポート で実行されています 6817。secretArn は、後のステップでクラスターシークレットを取得するために使用します。IP アドレスとポートは、後のステップで sackd サービスの設定に使用されます。

ステップ 2 – EC2 インスタンスを起動する

EC2 インスタンスを起動するには

1. [Amazon EC2 コンソール](#)を開きます。
2. ナビゲーションペインで、[Instances] (インスタンス) を選択し、[Launch Instances] (インスタンスの起動) を選択して、新しいインスタンス起動ウィザードを開きます。
3. (オプション) 名前とタグセクションで、 などのインスタンスの名前を指定します PCS-LoginNode。指定した名前はリソースタグとしてインスタンスに割り当てられます (Name=PCS-LoginNode)。
4. アプリケーションと OS イメージセクションで、PCS でサポートされているオペレーティングシステムの 1 つに AMI AWS を選択します。詳細については、「[サポートされるオペレーティングシステム](#)」を参照してください。
5. [インスタンスタイプ] セクションで、サポートされているインスタンスタイプを選択してください。詳細については、「[サポートされるインスタンスタイプ](#)」を参照してください。
6. キーペアセクションで、インスタンスに使用する SSH キーペアを選択します。
7. [ネットワーク設定] セクションで以下の操作を行います。
 - [編集] を選択します。
 - i. PCS クラスターの VPC AWS を選択します。
 - ii. [ファイアウォール (セキュリティグループ)] で、[既存のセキュリティグループを選択する] を選択してください。
 - A. インスタンスとターゲット PCS AWS クラスターの Slurm コントローラー間のトラフィックを許可するセキュリティグループを選択します。詳細については、「[セキュリティグループの要件と考慮事項](#)」を参照してください。
 - B. (オプション) インスタンスへのインバウンド SSH アクセスを許可するセキュリティグループを選択します。

8. ストレージセクションで、必要に応じてストレージボリュームを設定します。アプリケーションとライブラリをインストールしてユースケースを有効にするのに十分なスペースを設定してください。
9. Advanced で、クラスターシークレットへのアクセスを許可する IAM ロールを選択します。詳細については、「[Slurm クラスターシークレットを取得する](#)」を参照してください。
10. 概要ペインで、インスタンスの起動を選択します。

ステップ 3 – インスタンスに Slurm をインストールする

インスタンスが起動してアクティブになったら、任意のメカニズムを使用してインスタンスに接続します。が提供する Slurm インストーラ AWS を使用して、インスタンスに Slurm をインストールします。詳細については、「[Slurm インストーラ](#)」を参照してください。

Slurm インストーラをダウンロードして解凍し、`installer.sh`スクリプトを使用して Slurm をインストールします。詳細については、「[ステップ 3 – Slurm をインストールする](#)」を参照してください。

ステップ 4 – クラスターシークレットを取得して保存する

これらの手順には が必要です AWS CLI。詳細については、「[バージョン 2 ユーザーガイド](#)」の「[の最新バージョンのインストールまたは更新 AWS CLI](#)」AWS Command Line Interface」を参照してください。

次のコマンドを使用してクラスターシークレットを保存します。

- Slurm の設定ディレクトリを作成します。

```
sudo mkdir -p /etc/slurm
```

- クラスターシークレットを取得、デコード、保存します。このコマンドを実行する前に、`region-code` をターゲットクラスターが実行されているリージョンに置き換え、`secret-arn` を[ステップ 1](#)で`secretArn`取得した の値に置き換えます。

```
aws secretsmanager get-secret-value \  
  --region region-code \  
  --secret-id 'secret-arn' \  
  --version-stage AWSCURRENT \  
  --query 'SecretString' \  
  --output text | base64 -d | sudo tee /etc/slurm/slurm.key
```

⚠ Warning

マルチユーザー環境では、インスタンスにアクセスできるすべてのユーザーがインスタンスメタデータサービス (IMDS) にアクセスできる場合、クラスターシークレットを取得できる可能性があります。これにより、他のユーザーを偽装できるようになります。IMDS へのアクセスをルートユーザーまたは管理ユーザーのみに制限することを検討してください。または、インスタンスプロファイルに依存しない別のメカニズムを使用してシークレットを取得および設定することを検討してください。

- Slurm キーファイルに所有権とアクセス許可を設定します。

```
sudo chmod 0600 /etc/slurm/slurm.key
sudo chown slurm:slurm /etc/slurm/slurm.key
```

i Note

Slurm キーは、sackd サービスが実行されるユーザーとグループによって所有されている必要があります。

ステップ 5 – PCS AWS クラスターへの接続を設定する

AWS PCS クラスターへの接続を確立するには、以下の手順に従ってシステムサービス sackd として起動します。

i Note

Slurm 25.05 以降を使用している場合は、代わりにスクリプトを使用してログインノードをセットアップし、複数のクラスターに接続できます。詳細については、「[スタンドアロンログインノードを PCS AWS の複数のクラスターに接続する](#)」を参照してください。

1. 次のコマンドを使用して、sackd サービスの環境ファイルをセットアップします。コマンドを実行する前に、*ip-address* と *port* を [ステップ 1](#) でエンドポイントから取得した値に置き換えます。

```
sudo echo "SACKD_OPTIONS='--conf-server=ip-address:port'" > /etc/sysconfig/sackd
```

2. sackd プロセスを管理するためのsystemdサービスファイルを作成します。

```
sudo cat << EOF > /etc/systemd/system/sackd.service
[Unit]
Description=Slurm auth and cred kiosk daemon
After=network-online.target remote-fs.target
Wants=network-online.target
ConditionPathExists=/etc/sysconfig/sackd

[Service]
Type=notify
EnvironmentFile=/etc/sysconfig/sackd
User=slurm
Group=slurm
RuntimeDirectory=slurm
RuntimeDirectoryMode=0755
ExecStart=/opt/aws/pcs/scheduler/slurm-25.05/sbin/sackd --systemd \${SACKD_OPTIONS}
ExecReload=/bin/kill -HUP \${MAINPID}
KillMode=process
LimitNOFILE=131072
LimitMEMLOCK=infinity
LimitSTACK=infinity

[Install]
WantedBy=multi-user.target
EOF
```

3. sackd サービスファイルの所有権を設定します。

```
sudo chown root:root /etc/systemd/system/sackd.service && \
sudo chmod 0644 /etc/systemd/system/sackd.service
```

4. sackd サービスを有効にします。

```
sudo systemctl daemon-reload && sudo systemctl enable sackd
```

5. sackd サービスを開始します。

```
sudo systemctl start sackd
```

ステップ 6 – (オプション) 接続をテストする

sackd サービスが実行されていることを確認します。サンプル出力を次に示します。エラーがある場合は、一般的にここに表示されます。

```
[root@ip-10-3-27-112 ~]# systemctl status sackd
[x] sackd.service - Slurm auth and cred kiosk daemon
   Loaded: loaded (/etc/systemd/system/sackd.service; enabled; vendor preset: disabled)
   Active: active (running) since Tue 2024-12-17 16:34:55 UTC; 8s ago
     Main PID: 9985 (sackd)
        CGroup: /system.slice/sackd.service
                ##9985 /opt/aws/pcs/scheduler/slurm-25.05/sbin/sackd --systemd --conf-
server=10.3.149.220:6817

Dec 17 16:34:55 ip-10-3-27-112.ec2.internal systemd[1]: Starting Slurm auth and cred
kiosk daemon...
Dec 17 16:34:55 ip-10-3-27-112.ec2.internal systemd[1]: Started Slurm auth and cred
kiosk daemon.
Dec 17 16:34:55 ip-10-3-27-112.ec2.internal sackd[9985]: sackd: running
```

クラスターへの接続が、sinfoやなどのSlurmクライアントコマンドを使用して動作していることを確認します。queueからの出力例を次に示します。

```
[root@ip-10-3-27-112 ~]# /opt/aws/pcs/scheduler/slurm-25.05/bin/sinfo
PARTITION AVAIL TIMELIMIT NODES STATE NODELIST
all up infinite 4 idle~ compute-[1-4]
```

ジョブを送信することもできます。たとえば、この例のようなコマンドは、クラスター内の1つのノードでインタラクティブジョブを起動します。

```
/opt/aws/pcs/scheduler/slurm-25.05/bin/srun --nodes=1 -p all --pty bash -i
```

スタンドアロンログインノードを PCS AWS の複数のクラスターに接続する

このpcs-multi-cluster-login-configure.shスクリプトは、単一のスタンドアロンログインノードで複数のSlurm sackd デーモンを自動的に設定する方法を提供します。これにより、ログインノードは複数のクラスターと通信できるようになります。このスクリプトは、次のオペレーションを自動化します。

- PCS API AWS アクションを使用してクラスター情報を取得する
- base64 でエンコードされた Slurm 認証キーの入力を求めるプロンプト
- クラスター認証キーを使用して Slurm JWKS ファイルを作成します。
- クラスターエンドポイントとポートを使用してsackdサービスを設定します。
- クラスター固有のデーモンsystemdのサービスファイルを作成しますsackd。
- クラスター環境設定のアクティベーションスクリプトを生成します。
- sackd サービスを有効にして開始する

 Note

このスクリプトには Slurm バージョン 25.05 以降が必要です。

Slurm はすでにインスタンスにインストールされている必要があります (手動プロセスの[ステップ 3](#)に相当)。インスタンスは、ターゲットクラスターのエンドポイントに到達できる必要があります。このスクリプトは、手動設定プロセスの[ステップ 4](#)と[ステップ 5](#)の同等のオペレーションを実行します。クラスター情報を自動的に取得し、sackdサービスを設定し、必要なsystemdサービスファイルを作成し、ユーザーがクラスターの操作のためにシェル環境を設定するために使用できるアクティベーションスクリプトを作成します。

トピック

- [PCS マルチクラスターログインノード設定スクリプト AWS の前提条件](#)
- [AWS PCS マルチクラスターログインノード設定スクリプトコード](#)
- [AWS PCS マルチクラスターログインノード設定スクリプトの使用](#)

PCS マルチクラスターログインノード設定スクリプト AWS の前提条件

システム要件

- systemd をサポートする Linux OS
- システム設定のルート権限

必要なコマンドとパッケージ

- `bash` – シェルインタープリタ (バージョン 4.0 以降)
- `curl` – AWS IMDS v2 メタデータの取得用
- `jq` – AWS API レスポンスを解析するための JSON プロセッサ
- `aws` – PCS API AWS アクションを実行するための AWS CLI v2 と Secrets Manager アクセスのための v2
- `systemctl` – `systemd` サービス管理
- `find` – ファイルシステム検索ユーティリティ
- `grep` – テキストパターンマッチング
- `sed` – テキスト操作のストリームエディタ
- `sort` – テキストソートユーティリティ
- `tail` – ファイルの最終行を表示します。
- `mkdir` – ディレクトリの作成
- `chmod` – ファイルのアクセス許可を変更します。
- `chown` – ファイルの所有権を変更します。
- `ldconfig` – 動的リンカー設定

AWS 要件

- Slurm AWS バージョン 25.05 以降を実行する PCS クラスタ
- AWS 設定された認証情報 (IAM ロール、認証情報ファイル、または環境変数経由)
- 以下のアクセス許可:
 - `pcs:GetCluster`
 - `secretsmanager:GetSecretValue` (代替シークレットを使用する場合)

システムユーザーとグループ

- `slurm` ユーザーとグループはシステムに存在する必要があります

Slurm のインストール

- Slurm は、PCS AWS Slurm インストーラパッケージと同じ場所にインストールする必要があります。

```
/opt/aws/pcs/scheduler/slurm-version
```

AWS PCS マルチクラスターログインノード設定スクリプトコード

次のソースコードを次の名前のファイルに保存します。

```
pcs-multi-cluster-login-configure.sh
```

スクリプトソースコード

```
#!/bin/bash
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.

# AWS PCS Multi-Cluster Standalone Login Node Configuration Script
#
# This script configures AWS Parallel Computing Service (PCS) multi-cluster stand alone
# login nodes
# by setting up the Slurm authentication and credential kiosk daemon (sackd)
# for connecting to remote PCS clusters.
#
# Prerequisites:
# - AWS CLI configured with appropriate permissions
# - Slurm version 25.05 or later
# - Root privileges for system configuration
# - Network connectivity to AWS PCS endpoints

set -eo pipefail

# Function to display usage
usage() {
    echo "Usage: $0 --cluster-identifier <cluster-identifier> [--endpoint-url
<endpoint-url>]"
    echo "    $0 -h|--help"
}
```

```

# Function to display help
help() {
    echo "AWS PCS Multi-Cluster Standalone Login Node Configuration Script"
    echo "=====
    echo
    echo "This script configures multi-cluster standalone login node for AWS Parallel
Computing Service (PCS)"
    echo "by setting up the Slurm authentication and credential kiosk daemon (sackd)."
    echo
    usage
    echo
    echo "Options:"
    echo "  --cluster-identifier <id>      AWS PCS cluster identifier (required)"
    echo "  --endpoint-url <url>           Custom PCS endpoint URL (optional)"
    echo "  -h, --help                     Show this help message"
    echo
    echo "Examples:"
    echo "  $0 --cluster-identifier my-pcs-cluster"
    echo
    echo "Note: This script requires root privileges and Slurm version 25.05 or later."
}

# Function to retrieve authentication key
get_auth_key() {
    if [ "$ALTERNATE_SECRET_RETRIEVAL" = "true" ]; then
        echo "Retrieving authentication key from AWS Secrets Manager..." >&2
        local auth_key_arn=$(echo "$CLUSTER_INFO" | jq -r
'.cluster.slurmConfiguration.authKey.secretArn')
        local auth_key_version=$(echo "$CLUSTER_INFO" | jq -r
'.cluster.slurmConfiguration.authKey.secretVersion')

        if [ "$auth_key_arn" = "null" ] || [ "$auth_key_version" = "null" ]; then
            echo "Error: Auth key information not found in cluster configuration" >&2
            exit 1
        fi

        if ! aws secretsmanager get-secret-value --secret-id "$auth_key_arn" --version-
id "$auth_key_version" --query SecretString --output text --region "$REGION" 2>/dev/
null; then
            echo "Error: Failed to retrieve auth key from Secrets Manager" >&2
            exit 1
        fi
    else
        echo "Please enter the base64-encoded Slurm authentication key:" >&2
    fi
}

```



```

        echo -n "Base64 of the Slurm secret key: " >&2
        local key
        read -rs key
        echo >&2
        echo "$key"
    fi
}

# Function to get next available SACKD port
get_next_sackd_port() {
    local exclude_file="$1"
    local port=6918
    local used_ports=()

    # Get all currently used SACKD ports into an array
    while IFS= read -r line; do
        used_ports+=("$line")
    done < <(find /etc/sysconfig -name "sackd-pcs-*" ! -path "$exclude_file" \
        -exec grep SACKD_PORT= '{}' ';' 2>/dev/null | \
        sed 's/.*SACKD_PORT=//' | sort -n)

    # Loop through used ports to find first available port
    for used_port in "${used_ports[@]"; do
        if [ "$port" -lt "$used_port" ]; then
            break
        elif [ "$port" -eq "$used_port" ]; then
            ((port++))
        fi
    done

    echo "$port"
}

# Function to configure cluster
configure_cluster() {
    mkdir -p /etc/slurm
    SLURM_JWKS_FILE="/etc/slurm/slurm-${CLUSTER_NAME}.jwks"
    echo '{"keys":
[{"alg":"HS256","kty":"oct","kid":"key-'"${CLUSTER_ID}"'", "k":'"${BASE64_SLURM_KEY}"'"]}]'
    | jq -c '.' > "${SLURM_JWKS_FILE}"

    chmod 0600 "$SLURM_JWKS_FILE"
    chown slurm:slurm "$SLURM_JWKS_FILE"
}

```

```

SLURM_INSTALL_PATH="/opt/aws/pcs/scheduler/slurm-${SLURM_VERSION}"

SACKD_RUNTIME_DIRECTORY="/run/slurm-${CLUSTER_NAME}"
mkdir -p "${SACKD_RUNTIME_DIRECTORY}"
chown slurm:slurm "${SACKD_RUNTIME_DIRECTORY}"

mkdir -p /etc/sysconfig
SACKD_SERVICE_NAME="sackd-pcs-${CLUSTER_NAME}"
SACKD_SERVICE_ENV="/etc/sysconfig/${SACKD_SERVICE_NAME}"
SACKD_PORT=$(get_next_sackd_port "${SACKD_SERVICE_ENV}")
cat > "${SACKD_SERVICE_ENV}" << EOF
SACKD_OPTIONS='--conf-server=$ENDPOINTS'
SLURM_SACK_JWKS='$SLURM_JWKS_FILE'
RUNTIME_DIRECTORY='$SACKD_RUNTIME_DIRECTORY'
SACKD_PORT=$SACKD_PORT
EOF

SACKD_SERVICE_PATH="/etc/systemd/system/${SACKD_SERVICE_NAME}.service"

cat << EOF > "${SACKD_SERVICE_PATH}"
[Unit]
Description=Slurm auth and cred kiosk daemon
After=network-online.target remote-fs.target
Wants=network-online.target
ConditionPathExists=${SACKD_SERVICE_ENV}

[Service]
Type=notify
EnvironmentFile=${SACKD_SERVICE_ENV}
User=slurm
Group=slurm
RuntimeDirectory=slurm-${CLUSTER_NAME}
RuntimeDirectoryMode=0755
ExecStart=${SLURM_INSTALL_PATH}/sbin/sackd --systemd \${SACKD_OPTIONS}
ExecReload=/bin/kill -HUP \${MAINPID}
KillMode=process
LimitNOFILE=131072
LimitMEMLOCK=infinity
LimitSTACK=infinity

[Install]
WantedBy=multi-user.target
EOF

```

```

chown root:root "$SACKD_SERVICE_PATH"
chmod 0644 "$SACKD_SERVICE_PATH"
systemctl daemon-reload && systemctl enable "$SACKD_SERVICE_NAME"
systemctl restart "$SACKD_SERVICE_NAME"

ACTIVATE_SCRIPT="activate-pcs-${CLUSTER_NAME}"
cat > "$ACTIVATE_SCRIPT" << EOF
# Activate script for Slurm cluster ${CLUSTER_NAME}

# Add Slurm paths
export PATH="${SLURM_INSTALL_PATH}/bin:\$PATH"
export MANPATH="${SLURM_INSTALL_PATH}/share/man:\$MANPATH"
export LD_LIBRARY_PATH="${SLURM_INSTALL_PATH}/lib:\$LD_LIBRARY_PATH"
ldconfig

# Set Slurm configuration
export SLURM_CONF="/run/slurm-${CLUSTER_NAME}/conf/slurm.conf"
export PCS_CLUSTER_NAME="${CLUSTER_NAME}"
export PCS_CLUSTER_IDENTIFIER="${CLUSTER_IDENTIFIER}"
export PCS_CLUSTER_ID="${CLUSTER_ID}"

echo "Activated PCS cluster environment: ${CLUSTER_NAME}"

# Deactivate function
function deactivate-pcs-${CLUSTER_NAME}() {
    export PATH="\$(echo "\$PATH" | sed -e "s|${SLURM_INSTALL_PATH}/bin:||g" -e "s|:${SLURM_INSTALL_PATH}/bin\||g" -e "s|^${SLURM_INSTALL_PATH}/bin\$||")"
    export MANPATH="\$(echo "\$MANPATH" | sed -e "s|${SLURM_INSTALL_PATH}/share/man:||g" -e "s|:${SLURM_INSTALL_PATH}/share/man\||g" -e "s|^${SLURM_INSTALL_PATH}/share/man\$||")"
    export LD_LIBRARY_PATH="\$(echo "\$LD_LIBRARY_PATH" | sed -e "s|${SLURM_INSTALL_PATH}/lib:||g" -e "s|:${SLURM_INSTALL_PATH}/lib\||g" -e "s|^${SLURM_INSTALL_PATH}/lib\$||")"
    unset SLURM_CONF
    unset PCS_CLUSTER_NAME
    unset PCS_CLUSTER_IDENTIFIER
    unset PCS_CLUSTER_ID
    unset -f deactivate-pcs-${CLUSTER_NAME}
    ldconfig
    echo "Deactivated PCS cluster environment: ${CLUSTER_NAME}"
}

export -f deactivate-pcs-${CLUSTER_NAME}

```

```
EOF
}

# Main function
main() {
    # Parse arguments
    CLUSTER_IDENTIFIER=""
    PCS_ENDPOINT_URL=""

    while [ "$1" != "" ]; do
        case $1 in
            --cluster-identifier)
                shift
                CLUSTER_IDENTIFIER="$1"
                ;;
            --endpoint-url)
                shift
                PCS_ENDPOINT_URL="--endpoint-url $1"
                ;;
            -h|--help)
                help
                exit 0
                ;;
            *)
                echo "Invalid argument: $1" >&2
                usage >&2
                exit 1
                ;;
        esac
        shift
    done

    # Validate required arguments
    if [ -z "$CLUSTER_IDENTIFIER" ]; then
        echo "Error: --cluster-identifier is required" >&2
        usage >&2
        exit 1
    fi

    # Validate running as root
    if [ "$EUID" -ne 0 ]; then
        echo "Error: This script must be run as root" >&2
        exit 1
    fi
}
```

```

# Validate required commands are available
for cmd in aws jq curl; do
    if ! command -v "$cmd" &> /dev/null; then
        echo "Error: Required command '$cmd' not found" >&2
        exit 1
    fi
done

# Get the region name from IMDS v2 with error handling (try IPv6 first, fallback to
IPv4)
echo "Retrieving AWS region from instance metadata..."
# Try IPv6 IMDS endpoint first (fd00:ec2::254) with fast timeout (1s connect, 2s
total)
# If IPv6 fails, fallback to IPv4 IMDS endpoint (169.254.169.254)
IMDS_ENDPOINT="http://[fd00:ec2::254]"
if ! TOKEN=$(curl -s -X PUT "${IMDS_ENDPOINT}/latest/api/token" -H "X-aws-ec2-
metadata-token-ttl-seconds: 21600" --connect-timeout 1 --max-time 2 >/dev/null); then
    IMDS_ENDPOINT="http://169.254.169.254"
    if ! TOKEN=$(curl -s -X PUT "${IMDS_ENDPOINT}/latest/api/token" -H "X-aws-ec2-
metadata-token-ttl-seconds: 21600" --max-time 5); then
        echo "Error: Failed to retrieve IMDS token. Ensure this script is running
on an EC2 instance." >&2
        exit 1
    fi
fi

if ! REGION=$(curl -s -H "X-aws-ec2-metadata-token: $TOKEN" "${IMDS_ENDPOINT}/
latest/dynamic/instance-identity/document" --max-time 5 | jq -r '.region'); then
    echo "Error: Failed to retrieve AWS region from instance metadata" >&2
    exit 1
fi

echo "Detected AWS region: $REGION"

# Retrieve cluster information from AWS PCS
echo "Retrieving cluster information for: $CLUSTER_IDENTIFIER"
# shellcheck disable=SC2086
if ! CLUSTER_INFO=$(aws pcs get-cluster --region "$REGION" --cluster-identifier
"$CLUSTER_IDENTIFIER" $PCS_ENDPOINT_URL >/dev/null); then
    echo "Error: Failed to retrieve cluster information. Check cluster identifier
and AWS permissions." >&2
    exit 1
fi

```

```

CLUSTER_ID=$(echo "$CLUSTER_INFO" | jq -r '.cluster.id')
CLUSTER_NAME=$(echo "$CLUSTER_INFO" | jq -r '.cluster.name')
SLURM_VERSION=$(echo "$CLUSTER_INFO" | jq -r '.cluster.scheduler.version')
SLURM_VERSION=${SLURM_VERSION#Slurm_}

# Check if Slurm version is >= 25.05
# shellcheck disable=SC2072
if [[ "$SLURM_VERSION" < "25.05" ]]; then
    echo "Error: This script requires Slurm version 25.05 or later. Found version:
$SLURM_VERSION" >&2
    exit 1
fi

ENDPOINTS=$(echo "$CLUSTER_INFO" | jq -r '.cluster.endpoints[] | select(.type
== "SLURMCTLD") | (if .privateIpAddress != "" then .privateIpAddress else "["
+ .ipv6Address + "]" end) + ":" + .port' | tr '\n' ',' | sed 's/,,$//')

# Get BASE64_SLURM_KEY
BASE64_SLURM_KEY=$(get_auth_key)

if [ -z "$BASE64_SLURM_KEY" ]; then
    echo "Error: base64 Slurm key cannot be empty" >&2
    exit 1
fi

configure_cluster

# Final configuration summary
echo "======"
echo "Configuration completed successfully!"
echo "======"
echo "Cluster Name: $CLUSTER_NAME"
echo "Cluster ID: $CLUSTER_ID"
echo "Slurm Version: $SLURM_VERSION"
echo "Service Name: $SACKD_SERVICE_NAME"
echo "SACKD Port: $SACKD_PORT"
echo
echo "To activate this cluster environment, run:"
echo "  source ./${ACTIVATE_SCRIPT}"
echo
echo "To deactivate this cluster environment, run:"
echo "  deactivate-pcs-${CLUSTER_NAME}"
echo

```

```
echo "To check service status:"
echo "  systemctl status $SACKD_SERVICE_NAME"
echo
echo "To view service logs:"
echo "  journalctl -u $SACKD_SERVICE_NAME -f"
}

# Exit if being sourced for testing
[[ "${BASH_SOURCE[0]}" != "${0}" ]] && return

# Execute main function
main "$@"
```

AWS PCS マルチクラスターログインノード設定スクリプトの使用

スクリプトの実行

設定スクリプトを実行するには

1. [スクリプトの内容を](#) という名前のファイルに保存します。

```
pcs-multi-cluster-login-configure.sh
```

2. 実行可能にします。

```
chmod +x pcs-multi-cluster-login-configure.sh
```

3. スクリプトを実行します。

```
./pcs-multi-cluster-login-configure.sh --cluster-identifier cluster-name
```

クラスターインタラクション環境

設定が成功すると、スクリプトは現在のディレクトリにクラスター固有のアクティベーションスクリプトを生成します。スクリプトの名前は `activate-pcs-cluster-name`。アクティベーションスクリプトは、ターゲットクラスターとやり取りするために必要な環境変数とパスを設定します。

クラスター環境をアクティブ化するには

- `source` コマンドを使用してアクティベーションスクリプトを実行する

```
source ./activate-pcs-cluster-name
```

Example

```
# Activate cluster environment for cluster 'my-cluster'
source ./activate-pcs-my-cluster

# Now you can use Slurm commands
sinfo
squeue
sbatch my-job.sh
```

アクティベーションスクリプトの動作

- クラスターの設定を指すようにSLURM_CONF環境変数を設定します。
- クラスターの Slurm バイナリを含めるPATHように を更新します。
- その他の必要な Slurm 環境変数 (MANPATH、) を設定しますLD_LIBRARY_PATH。
- PCS AWS クラスター識別変数を設定します。
- ターゲット PCS AWS クラスターとのシームレスなインタラクションを有効にします。

クラスター環境を非アクティブ化するには

- 非アクティブ化コマンドを実行します。

```
deactivate-pcs-cluster-name
```

Example

```
# After activating a cluster
source ./activate-pcs-my-cluster

# Work with the cluster
sinfo

# Deactivate when done
deactivate-pcs-my-cluster
```


無効コマンドの動作

- 元のPATH環境変数を復元します。
- クラスター固有の Slurm 環境変数の設定を解除します。
- シェル環境をアクティブ化前の状態に戻します。

Note

アクティベーションはセッション固有であり、クラスターを操作するシェルセッションで取得する必要があります。

AWS PCS ネットワーク

PCS AWS クラスターは Amazon VPC に作成されます。この章には、クラスターのスケジューラとノードのネットワークに関する以下のトピックが含まれています。

インスタンスを起動するサブネットを選択する場合を除き、EC2 起動テンプレートを使用して PCS AWS コンピューティングノードグループのネットワークを設定する必要があります。起動テンプレートの詳細については、「[PCS での Amazon EC2 AWS 起動テンプレートの使用](#)」を参照してください。

トピック

- [AWS PCS VPC とサブネットの要件と考慮事項](#)
- [PCS クラスターの VPC AWS の作成](#)
- [PCS AWS のセキュリティグループ](#)
- [PCS AWS の複数のネットワークインターフェイス](#)
- [PCS の EC2 AWS インスタンスのプレイacementグループ](#)
- [PCS での Elastic Fabric Adapter \(EFA\) AWS の使用](#)

AWS PCS VPC とサブネットの要件と考慮事項

AWS PCS クラスターを作成するときは、VPC にサブネットを指定します。このトピックでは、クラスターで使用する VPC とサブネット (複数可) に関する AWS PCS 固有の要件と考慮事項の概要を説明します。PCS で使用する VPC がない場合は、AWS が提供する CloudFormation テンプレートを使用して VPC AWS を作成できます。VPCs 「Amazon VPC ユーザーガイド」の「[仮想プライベートクラウド \(VPC\)](#)」を参照してください。

VPC の要件と考慮事項

クラスターを作成する際には、指定する VPC が次の要件と考慮事項を満たす必要があります。

- VPC には、クラスター、ノード、および作成するその他のクラスターリソースで利用できる十分な数の IP アドレスが必要です。詳細については、「Amazon VPC ユーザーガイド」の「[VPC とサブネットの IP アドレス指定](#)」を参照してください。
- クラスターが IPv6 を使用している場合:
 - IPv6 CIDR ブロックと VPC の関連付け 詳細については、「Amazon VPC ユーザーガイド」の「[VPC を作成する](#)」を参照してください。

⚠ Important

IPv4 と IPv6 の両方を使用して VPC を設定できますが、クラスターには 1 つのネットワークタイプしか選択できません。

- サブネットの IPv6 アドレスの自動割り当てを有効にします。
- 詳細については、以下を参照してください。
 - [での IPv6 AWS](#)
 - [AWS での IPv6 アドレス指定とスケーラブルなアドレス指定プランの設計について](#)
- VPC には、DNS ホスト名と DNS 解決のサポートが必要です。そうしないと、ノードはカスタマークラスターを登録できません。詳細については、「Amazon VPC ユーザーガイド」の「[VPC の DNS 属性](#)」を参照してください。
- VPC が AWS PCS API に連絡できるようにする AWS PrivateLink には、を使用して VPC エンドポイントが必要になる場合があります。詳細については、「Amazon [VPC ユーザーガイド](#)」の「[を使用して VPC を サービスに接続する AWS PrivateLink](#)」を参照してください。

⚠ Important

AWS PCS は、専用インスタンステナンシーを持つ VPC をサポートしていません。PCS に使用する VPC AWS は、default インスタンステナンシーを使用する必要があります。既存の VPC のインスタンステナンシーを変更できます。詳細については、「Amazon Elastic Compute Cloud ユーザーガイド」の「VPC [のインスタンステナンシーを変更する](#)」を参照してください。

サブネットの要件と考慮事項

Slurm クラスターを作成すると、AWS PCS は指定したサブネットに [Elastic Network Interface \(ENI\)](#) を作成します。このネットワークインターフェイスにより、スケジューラコントローラーとカスタマー VPC 間の通信が可能になります。ネットワークインターフェイスを使用すると、Slurm はアカウントにデプロイされたコンポーネントと通信することもできます。クラスターのサブネットは、作成時にのみ指定できます。

クラスターのサブネットの要件

クラスターの作成時に指定する [サブネット](#) は、次の要件を満たしている必要があります。

- サブネットには、PCS で使用する IP アドレスが少なくとも 1 AWS つ必要です。
- クラスターが IPv6 を使用している場合、クラスター内のすべてのサブネットで IPv6 を使用する必要があります。

Important

PCS サンプル AMIs AWS と複数のネットワークインターフェイスで設定されたコンピューティングノードグループは、サブネットが IPv6 を使用するようのみ設定されている場合、現在機能しません。代わりに、デュアルスタックサブネット (IPv4 および IPv6) または IPv4-only サブネットを使用します。詳細については、「[PCS でサンプル Amazon マシンイメージ \(AMIs\) AWS を使用する](#)」を参照してください。

- サブネットは AWS Outposts AWS Wavelength、または AWS ローカルゾーンに存在できません。
- サブネットはパブリックでもプライベートでもかまいません。可能であれば、プライベートサブネットを指定することをお勧めします。パブリックサブネットは、[インターネットゲートウェイ](#)へのルートを含むルートテーブルを持つサブネットです。プライベートサブネットは、インターネットゲートウェイへのルートを含まないルートテーブルを持つサブネットです。

ノードのサブネットの要件

ノードやその他のクラスターリソースは、PCS クラスターの作成時に指定したサブネットと、同じ VPC AWS 内の他のサブネットにデプロイできます。

ノードとクラスターリソースをデプロイするサブネットは、次の要件を満たしている必要があります。

- サブネットに、すべてのノードとクラスターリソースをデプロイするための十分な IP アドレスがあることを確認する必要があります。
- クラスターが IPv4 を使用していて、ノードをパブリックサブネットにデプロイする場合、そのサブネットは IPv4 パブリックアドレスを自動的に割り当てる必要があります。

Note

パブリックサブネット内のインスタンスは、パブリック IP アドレスからのトラフィックを許可するインバウンドルールを持つセキュリティグループを使用する必要があります。

特定の送信元アドレスの制限がない限り、これは IPv4 送信元アドレスが 0.0.0.0/0 または IPv6 送信元アドレスが ::/0 であることを意味します。

- ノードをデプロイするサブネットがプライベートサブネットで、そのルートテーブルにネットワークアドレス変換 (NAT) デバイス (IPv4) へのルートが含まれていない場合は、を使用して VPC エンドポイント AWS PrivateLink をカスタマー VPC に追加します。VPC エンドポイントは、ノードが接続するすべての AWS サービスに必要です。必要なエンドポイントは、PCS AWS がノードに RegisterComputeNodeGroupInstance API アクションの呼び出しを許可するだけです。詳細については、AWS PCS API リファレンスの [RegisterComputeNodeGroupInstance](#) を参照してください。
- パブリックサブネットまたはプライベートサブネットのステータスは PCS AWS には影響しません。必要なエンドポイントに到達できる必要があります。

PCS クラスターの VPC AWS の作成

AWS Parallel Computing Service (AWS PCS) 内でクラスターの Amazon Virtual Private Cloud (Amazon VPC) を作成できます。

Amazon VPC を使用して、定義した仮想ネットワークで VPC リソースを起動します。この仮想ネットワークは、お客様自身のデータセンターで運用されている従来のネットワークによく似ています。ただし、Amazon Web Services のスケーラブルなインフラストラクチャを使用できるというメリットがあります。本番稼働用 VPC クラスターをデプロイする前に、Amazon VPC サービスを十分に理解しておくことをお勧めします。詳細については、「作成者ビジュアルモード」の「[Amazon VPC とは](#)」を参照してください。Amazon VPC ユーザーガイド。

PCS クラスター、ノード、およびサポートリソース (ファイルシステムやディレクトリサービスなど) は、Amazon VPC 内にデプロイされます。PCS で既存の Amazon VPC を使用する場合は、[AWS PCS VPC とサブネットの要件と考慮事項](#)「」で説明されている要件を満たしている必要があります。このトピックでは、AWS が提供する CloudFormation テンプレートを使用して PCS 要件を満たす VPC を作成する方法について説明します。テンプレートをデプロイすると、テンプレートによって作成されたリソースを表示して、作成されたリソースとそれらのリソースの設定を正確に把握できます。

前提条件

PCS 用の Amazon VPC を作成するには、Amazon VPC リソースを作成するために必要な IAM アクセス許可が必要です。これらのリソースは、VPC、サブネット、セキュリティグループ、ルート

テーブルとルート、およびインターネットと NAT ゲートウェイです。詳細については、「[Amazon VPC ユーザーガイド](#)」の「[パブリックサブネットを使用して VPC を作成する](#)」を参照してください。Amazon EC2 の完全なリストを確認するには、「サービス認可リファレンス」の[Amazon EC2 のアクション、リソース、および条件キー](#)」を参照してください。

Amazon VPC を作成する

PCS を使用する の適切な URL をコピーして貼り付け、AWS リージョン VPC を作成します。CloudFormation テンプレートをダウンロードして、自分で [CloudFormation コンソール](#) にアップロードすることもできます。

- 米国東部 (バージニア北部) (us-east-1)

```
https://console.aws.amazon.com/cloudformation/home?region=us-east-1#/stacks/create/review?stackName=hpc-networking&templateURL=https://aws-hpc-recipes.s3.us-east-1.amazonaws.com/main/recipes/net/hpc_large_scale/assets/main.yaml
```

- 米国東部 (オハイオ) (us-east-2)

```
https://console.aws.amazon.com/cloudformation/home?region=us-east-2#/stacks/create/review?stackName=hpc-networking&templateURL=https://aws-hpc-recipes.s3.us-east-1.amazonaws.com/main/recipes/net/hpc_large_scale/assets/main.yaml
```

- 米国西部 (オレゴン) (us-west-2)

```
https://console.aws.amazon.com/cloudformation/home?region=us-west-2#/stacks/create/review?stackName=hpc-networking&templateURL=https://aws-hpc-recipes.s3.us-east-1.amazonaws.com/main/recipes/net/hpc_large_scale/assets/main.yaml
```

- テンプレートのみ

```
https://aws-hpc-recipes.s3.us-east-1.amazonaws.com/main/recipes/net/hpc_large_scale/assets/main.yaml
```

PCS 用の Amazon VPC を作成するには

1. [CloudFormation コンソール](#) でテンプレートを開きます。

Note

これらはテンプレートにあらかじめ入力されているため、デフォルト値のままにしておくだけで済みます。

2. 「スタック名を指定」に「」と入力しますhpc-networking。
3. パラメータに、次の詳細を入力します。
 - a. VPC で CidrBlock と入力します。 10.3.0.0/16
 - b. サブネット A の下:
 - i. 次に、CidrPublicSubnetA と入力します。 10.3.0.0/20
 - ii. 次に、CidrPrivateSubnetA と入力します。 10.3.128.0/20
 - c. サブネット B の下:
 - i. 次に、CidrPublicSubnetB と入力します。 10.3.16.0/20
 - ii. 次に、CidrPrivateSubnetA と入力します。 10.3.144.0/20
 - d. サブネット C の下:
 - i. ProvisionSubnetsC で、 を選択しますTrue。

Note

アベイラビリティゾーンが 3 つ未満のリージョンで VPC を作成する場合、このオプションを に設定すると無視されますTrue。

- ii. 次に、CidrPublicSubnetB と入力します。 10.3.32.0/20
 - iii. 次に、CidrPrivateSubnetA と入力します。 10.3.160.0/20
4. 「機能」のチェックボックスをオンにして、AWS CloudFormation が IAM リソースを作成する可能性があることを確認します。

CloudFormation スタックのステータスをモニタリングします。に達するとCREATE_COMPLETE、VPC リソースを使用できるようになります。

Note

CloudFormation テンプレートが作成したすべてのリソースを表示するには、[CloudFormation コンソール](#)を開きます。hpc-networking スタックを選択し、[リソース] タブを選択します。

PCS AWS のセキュリティグループ

Amazon EC2 のセキュリティグループは、インスタンスへのインバウンドトラフィックとアウトバウンドトラフィックを制御する仮想ファイアウォールとして機能します。AWS PCS コンピューティングノードグループの起動テンプレートを使用して、インスタンスにセキュリティグループを追加または削除します。起動テンプレートにネットワークインターフェイスが含まれていない場合は、SecurityGroupIds を使用してセキュリティグループのリストを指定します。起動テンプレートでネットワークインターフェイスを定義する場合は、Groups パラメータを使用して各ネットワークインターフェイスにセキュリティグループを割り当てる必要があります。起動テンプレートの詳細については、「[PCS での Amazon EC2 AWS 起動テンプレートの使用](#)」を参照してください。

Note

起動テンプレートのセキュリティグループ設定の変更は、コンピューティングノードグループの更新後に起動された新しいインスタンスにのみ影響します。

セキュリティグループの要件と考慮事項

AWS PCS は、クラスターの作成時に指定したサブネットにクロスアカウント [Elastic Network Interface \(ENI\)](#) を作成します。これにより、AWS PCS によって起動された EC2 インスタンスと通信するためのパス AWS である、 が管理するアカウントで実行されている HPC スケジューラが提供されます。スケジューラ ENI とクラスター EC2 インスタンス間の双方向通信を許可する、その ENI のセキュリティグループを指定する必要があります。

これを実現する簡単な方法は、グループ内のすべてのメンバー間のすべてのポートで TCP/IP トラフィックを許可する、許可された自己参照セキュリティグループを作成することです。これは、クラスターとノードグループ EC2 インスタンスの両方にアタッチできます。

許可されたセキュリティグループ設定の例

IPv4

ルールタイプ	プロトコル	ポート	送信元	目的地
インバウンド	すべて	すべて	自分	
アウトバウンド	すべて	すべて		0.0.0.0/0
アウトバウンド	すべて	すべて		自分

IPv6

ルールタイプ	プロトコル	ポート	送信元	目的地
インバウンド	すべて	すべて	自分	
アウトバウンド	すべて	すべて		::/0
アウトバウンド	すべて	すべて		自分

これらのルールにより、すべてのトラフィックが Slurm コントローラーとノード間で自由に流れ、すべてのアウトバウンドトラフィックが任意の宛先に流れ、[EFA トラフィック](#)が有効になります。

制限付きセキュリティグループ設定の例

クラスターとそのコンピューティングノード間のオープンポートを制限することもできます。Slurm スケジューラの場合、クラスターにアタッチされたセキュリティグループは次のポートを許可する必要があります。

- 6817 – EC2 インスタンス `slurmctld` から へのインバウンド接続を有効にする
- 6818 – EC2 インスタンスで `slurmd` 実行されている から `slurmctld` へのアウトバウンド接続を有効にする

コンピューティングノードにアタッチされたセキュリティグループは、次のポートを許可する必要があります。

- 6817 – EC2 インスタンス `slurmctld` から へのアウトバウンド接続を有効にします。
- 6818 – ノードグループインスタンス `slurmd` の `slurmctld` と `slurmd` との間のインバウンドおよびアウトバウンド接続を有効にする
- 60001 ~ 63000 — サポートするノードグループインスタンス間のインバウンド接続とアウトバウンド接続 `srun`
- ノードグループインスタンス間の EFA トラフィック。詳細については、Linux インスタンス用ユーザーガイドの「[EFA 対応セキュリティグループの準備](#)」を参照してください。
- ワークロードに必要なその他のノード間トラフィック

PCS AWS の複数のネットワークインターフェイス

一部の EC2 インスタンスには複数のネットワークカードがあります。これにより、100 Gbps を超える帯域幅機能やパケット処理の改善など、より高いネットワークパフォーマンスを実現できます。複数のネットワークカードを持つインスタンスの詳細については、「Amazon [Elastic Compute Cloud ユーザーガイド](#)」の「[Elastic Network Interface](#)」を参照してください。

EC2 起動テンプレートにネットワークインターフェイスを追加して、PCS AWS コンピューティングノードグループのインスタンス用に追加のネットワークカードを設定します。以下は、`hpc7a.96xlarge` インスタンスにある など、2 つのネットワークカードを有効にする起動テンプレートの例です。次の詳細情報に注意してください。

- 各ネットワークインターフェイスのサブネットは、起動テンプレートを使用する PCS AWS コンピューティングノードグループを設定するときに選択したサブネットと同じである必要があります。
- SSH や HTTPS トラフィックなどの定期的なネットワーク通信が発生するプライマリネットワークデバイスは、`DeviceIndex` を に設定することで確立されます⁰。他のネットワークインターフェイスのは `DeviceIndex` です¹。プライマリネットワークインターフェイスは 1 つだけです。他のすべてのインターフェイスはセカンダリです。
- すべてのネットワークインターフェイスには一意の が必要です `NetworkCardIndex`。起動テンプレートで定義されているように、順番に番号を付けることをお勧めします。
- 各ネットワークインターフェイスのセキュリティグループは、 を使用して設定されます `Groups`。この例では、インバウンド SSH セキュリティグループ (`sg-SshSecurityGroupId`) がプライマリネットワークインターフェイスに追加され、クラスター内通信を有効にするセキュリティグループ () も追加されます `sg-ClusterSecurityGroupId`。最後に、インターネット

(sg-*InternetOutboundSecurityGroupId*) へのアウトバウンド接続を許可するセキュリティグループが、プライマリインターフェイスとセカンダリインターフェイスの両方に追加されます。

```
{
  "NetworkInterfaces": [
    {
      "DeviceIndex": 0,
      "NetworkCardIndex": 0,
      "SubnetId": "subnet-SubnetId",
      "Groups": [
        "sg-SshSecurityGroupId",
        "sg-ClusterSecurityGroupId",
        "sg-InternetOutboundSecurityGroupId"
      ]
    },
    {
      "DeviceIndex": 1,
      "NetworkCardIndex": 1,
      "SubnetId": "subnet-SubnetId",
      "Groups": ["sg-InternetOutboundSecurityGroupId"]
    }
  ]
}
```

PCS の EC2 AWS インスタンスのプレイスメントグループ

プレイスメントグループを使用して EC2 インスタンスの配置に影響を与え、それらで実行されるワークロードのニーズに合わせることができます。

プレイスメントグループタイプ

- クラスター – インスタンスをアベイラビリティーゾーンに密接にパックして、低レイテンシーの通信を最適化します。
- パーティション – 論理パーティションにインスタンスを分散して、回復力を最大化します。
- スプレッド – 少数のインスタンスが個別のハードウェアで起動することを厳密に強制します。これは回復力にも役立ちます。

詳細については、[Amazon EC2 インスタンスのプレイスメントグループ](#)」を参照してください。

Elastic Fabric Adapter (EFA) を使用するように AWS PCS コンピューティングノードグループを設定するときは、クラスタープレイスメントグループを含めることをお勧めします。

EFA で動作するクラスタープレイスメントグループを作成するには

1. コンピューティングノードグループのタイプクラスターを使用してプレイスメントグループを作成します。
 - 次の AWS CLI コマンドを使用します。

```
aws ec2 create-placement-group --strategy cluster --group-name PLACEMENT-GROUP-NAME
```

- CloudFormation テンプレートを使用してプレイスメントグループを作成することもできます。詳細については、「AWS CloudFormation ユーザーガイド」の「[CloudFormation テンプレートの使用](#)」を参照してください。次の URL からテンプレートをダウンロードし、[CloudFormation コンソール](#)にアップロードします。

```
https://aws-hpc-recipes.s3.amazonaws.com/main/recipes/pcs/enable_efa/assets/efa-placement-group.yaml
```

2. PCS コンピューティングノードグループの EC2 AWS 起動テンプレートにプレイスメントグループを含めます。

PCS での Elastic Fabric Adapter (EFA) AWS の使用

Elastic Fabric Adapter (EFA) は、EC2 インスタンスにアタッチしてハイパフォーマンスコンピューティング (HPC) と機械学習アプリケーションを高速化できる、からの AWS 高性能な高度なネットワーク相互接続です。EFA を使用して AWS PCS クラスターで実行されているアプリケーションを有効にするには、次のように EFA を使用するように AWS PCS コンピューティングノードグループインスタンスを設定する必要があります。

Note

PCS 互換 AMI AWS に EFA をインストールする – PCS コンピューティングノードグループで使用される AMI AWS には、EFA ドライバーがインストールされ、ロードされている必要があります。EFA ソフトウェアがインストールされたカスタム AMI を構築する方法については、「」を参照してください[PCS のカスタム Amazon AWS マシンイメージ \(AMIs\)](#)。

目次

- [EFA 対応 EC2 インスタンスを特定する](#)
- [EFA 通信をサポートするセキュリティグループを作成する](#)
- [\(オプション\) プレイメントグループを作成する](#)
- [EC2 起動テンプレートを作成または更新する](#)
- [EFA のコンピューティングノードグループを作成または更新する](#)
- [\(オプション\) EFA をテストする](#)
- [\(オプション\) CloudFormation テンプレートを使用して EFA 対応の起動テンプレートを作成する](#)

EFA 対応 EC2 インスタンスを特定する

EFA を使用するには、PCS AWS コンピューティンググループに許可されるすべてのインスタンスタイプが EFA をサポートし、同じ数の vCPUs (および必要に応じて GPUs を持っている必要があります。EFA 対応インスタンスのリストについては、[Amazon EC2 の HPC および ML ワークロード用の Elastic Fabric Adapter](#)」を参照してください。を使用して AWS CLI、EFA をサポートするインスタンスタイプのリストを表示することもできます。*region-code* を、などの AWS PCS AWS リージョン を使用する に置き換えます us-east-1。

```
aws ec2 describe-instance-types \
  --region region-code \
  --filters Name=network-info.efa-supported,Values=true \
  --query "InstanceTypes[*].[InstanceType]" \
  --output text | sort
```

Note

使用可能なネットワークインターフェイスの数を決定する – 一部の EC2 インスタンスには複数のネットワークカードがあります。これにより、複数の EFAs を持つことができます。詳細については、「[PCS AWS の複数のネットワークインターフェイス](#)」を参照してください。

EFA 通信をサポートするセキュリティグループを作成する

AWS CLI

次の AWS CLI コマンドを使用して、EFA をサポートするセキュリティグループを作成できます。コマンドはセキュリティグループ ID を出力します。次の置換を行います。

- *region-code* – など、PCS AWS を使用する を指定します AWS リージョン us-east-1。
- *vpc-id* – PCS に使用する VPC AWS の ID を指定します。
- *efa-group-name* – セキュリティグループに選択した名前を指定します。

```
aws ec2 create-security-group \  
  --group-name efa-group-name \  
  --description "Security group to enable EFA traffic" \  
  --vpc-id vpc-id \  
  --region region-code
```

次のコマンドを使用して、インバウンドおよびアウトバウンドのセキュリティグループルールをアタッチします。次の置換を行います。

- *efa-secgroup-id* – 先ほど作成した EFA セキュリティグループの ID を指定します。

```
aws ec2 authorize-security-group-ingress \  
  --group-id efa-secgroup-id \  
  --protocol -1 \  
  --source-group efa-secgroup-id  
  
aws ec2 authorize-security-group-egress \  
  --group-id efa-secgroup-id \  
  --protocol -1 \  
  --source-group efa-secgroup-id
```

CloudFormation template

CloudFormation テンプレートを使用して、EFA をサポートするセキュリティグループを作成できます。次の URL からテンプレートをダウンロードし、[AWS CloudFormation コンソール](#)にアップロードします。

```
https://aws-hpc-recipes.s3.amazonaws.com/main/recipes/pcs/enable_efa/assets/efa-sg.yaml
```

AWS CloudFormation コンソールでテンプレートを開いた状態で、次のオプションを入力します。

- スタック名を指定する
 - スタック名に、 などの名前を入力しますefa-sg-stack。
- パラメータの下
 - SecurityGroupName で、 などの名前を入力しますefa-sg。
 - VPC で、 PCS を使用する VPC AWS を選択します。

CloudFormation スタックの作成を完了し、そのステータスをモニタリングします。CREATE_COMPLETE EFA セキュリティグループが使用可能になると、そのグループを使用できます。

(オプション) プレイACEMENTグループを作成する

EFA を使用するすべてのインスタンスをクラスタープレイACEMENTグループで起動して、それらの間の物理的な距離を最小限に抑えることをお勧めします。EFA を使用する予定のコンピューティングノードグループごとにプレイACEMENTグループを作成します。コンピューティングノードグループのプレイACEMENTグループを作成するには[PCS の EC2 AWS インスタンスのプレイACEMENTグループ](#)、「」を参照してください。

EC2 起動テンプレートを作成または更新する

EFA ネットワークインターフェイスは、PCS コンピューティングノードグループの EC2 AWS 起動テンプレートで設定されます。複数のネットワークカードがある場合は、複数の EFAsを設定できます。EFA セキュリティグループとオプションのプレイACEMENTグループも起動テンプレートに含まれます。

hpc7a.96xlarge など、2 つのネットワークカードを持つインスタンスの起動テンプレートの例を次に示します。インスタンスは、クラスタープレイACEMENTグループ subnet-*SubnetID1* の で起動されますpg-*PlacementGroupId1*。

セキュリティグループは、特に各 EFA インターフェイスに追加する必要があります。すべての EFA には、EFA トラフィック () を有効にするセキュリティグループが必要ですsg-*EfaSecGroupId*。

他のセキュリティグループ、特に SSH や HTTPS などの通常のトラフィックを処理するセキュリティグループは、プライマリネットワークインターフェイス (DeviceIndexの で指定) にのみアタッチする必要があります。ネットワークインターフェイスが定義されている起動テンプレートは、SecurityGroupIdsパラメータを使用したセキュリティグループの設定をサポートしています。設定するネットワークインターフェイスGroupsごとに の値を設定する必要があります。

```
{
  "Placement": {
    "GroupId": "pg-PlacementGroupId1"
  },
  "NetworkInterfaces": [
    {
      "DeviceIndex": 0,
      "InterfaceType": "efa",
      "NetworkCardIndex": 0,
      "SubnetId": "subnet-SubnetId1",
      "Groups": [
        "sg-SecurityGroupId1",
        "sg-EfaSecGroupId"
      ]
    },
    {
      "DeviceIndex": 1,
      "InterfaceType": "efa",
      "NetworkCardIndex": 1,
      "SubnetId": "subnet-SubnetId1"
      "Groups": ["sg-EfaSecGroupId"]
    }
  ]
}
```

EFA のコンピューティングノードグループを作成または更新する

AWS PCS コンピューティングノードグループには、同じ数の vCPUs プロセッサアーキテクチャ、EFA サポートを持つインスタンスが含まれている必要があります。EFA ソフトウェアがインストールされている AMI を使用し、EFA 対応ネットワークインターフェイスを設定する起動テンプレートを使用するようにコンピューティングノードグループを設定します。

(オプション) EFA をテストする

EFA ソフトウェアのインストールに含まれている `fi_pingpong` プログラムを実行することで、コンピューティングノードグループ内の 2 つのノード間の EFA 対応通信をデモンストレーションできます。このテストが成功すると、EFA が正しく設定されている可能性があります。

開始するには、コンピューティングノードグループに 2 つの実行中のインスタンスが必要です。コンピューティングノードグループが静的容量を使用している場合、使用可能なインスタンスが既にあるはずです。動的容量を使用するコンピューティングノードグループの場合、`salloc` コマンドを使用して 2 つのノードを起動できます。という名前のキュー `hpc7g` に関連付けられた という名前の動的ノードグループを持つクラスターの例を次に示します `all`。

```
% salloc --nodes 2 -p all
salloc: Granted job allocation 6
salloc: Waiting for resource configuration
... a few minutes pass ...
salloc: Nodes hpc7g-[1-2] are ready for job
```

を使用して、割り当てられた 2 つのノードの IP アドレスを確認します `scontrol`。次の例では、アドレスは `10.3.140.69` 用 `hpc7g-1`、`10.3.132.211` 用です `hpc7g-2`。

```
% scontrol show nodes hpc7g-[1-2]
NodeName=hpc7g-1 Arch=aarch64 CoresPerSocket=1
  CPUAlloc=0 CPUEfctv=64 CPUTot=64 CPULoad=0.00
  AvailableFeatures=hpc7g
  ActiveFeatures=hpc7g
  Gres=(null)
  NodeAddr=10.3.140.69 NodeHostName=ip-10-3-140-69 Version=25.05.4
  OS=Linux 5.10.218-208.862.amzn2.aarch64 #1 SMP Tue Jun 4 16:52:10 UTC 2024
  RealMemory=124518 AllocMem=0 FreeMem=110763 Sockets=64 Boards=1
  State=IDLE+CLOUD ThreadsPerCore=1 TmpDisk=0 Weight=1 Owner=N/A MCS_label=N/A
  Partitions=efa
  BootTime=2024-07-02T19:00:09 SlurmdStartTime=2024-07-08T19:33:25
  LastBusyTime=2024-07-08T19:33:25 ResumeAfterTime=None
  CfgTRES=cpu=64,mem=124518M,billing=64
  AllocTRES=
  CapWatts=n/a
  CurrentWatts=0 AveWatts=0
  ExtSensorsJoules=n/a ExtSensorsWatts=0 ExtSensorsTemp=n/a
  Reason=Maintain Minimum Number Of Instances [root@2024-07-02T18:59:00]
  InstanceId=i-04927897a9ce3c143 InstanceType=hpc7g.16xlarge
```

```

NodeName=hpc7g-2 Arch=aarch64 CoresPerSocket=1
CPUAlloc=0 CPUEfctv=64 CPUTot=64 CPULoad=0.00
AvailableFeatures=hpc7g
ActiveFeatures=hpc7g
Gres=(null)
NodeAddr=10.3.132.211 NodeHostName=ip-10-3-132-211 Version=25.05.4
OS=Linux 5.10.218-208.862.amzn2.aarch64 #1 SMP Tue Jun 4 16:52:10 UTC 2024
RealMemory=124518 AllocMem=0 FreeMem=110759 Sockets=64 Boards=1
State=IDLE+CLOUD ThreadsPerCore=1 TmpDisk=0 Weight=1 Owner=N/A MCS_label=N/A
Partitions=efa
BootTime=2024-07-02T19:00:09 SlurmdStartTime=2024-07-08T19:33:25
LastBusyTime=2024-07-08T19:33:25 ResumeAfterTime=None
CfgTRES=cpu=64,mem=124518M,billing=64
AllocTRES=
CapWatts=n/a
CurrentWatts=0 AveWatts=0
ExtSensorsJoules=n/a ExtSensorsWatts=0 ExtSensorsTemp=n/a
Reason=Maintain Minimum Number Of Instances [root@2024-07-02T18:59:00]
InstanceId=i-0a2c82623cb1393a7 InstanceType=hpc7g.16xlarge

```

SSH (または SSMhpc7g-1) を使用して、いずれかのノード (この例では) に接続します。これは内部 IP アドレスであるため、SSH を使用する場合はログインノードのいずれかから接続する必要があります。また、インスタンスは、コンピューティングノードグループの起動テンプレートを使用して SSH キーで設定する必要があることに注意してください。

```
% ssh ec2-user@10.3.140.69
```

次に、サーバーモードで を起動 `fi_pingpong` します。

```
/opt/amazon/efa/bin/fi_pingpong -p efa
```

2 番目のインスタンス () に接続します `hpc7g-2`。

```
% ssh ec2-user@10.3.132.211
```

`fi_pingpong` クライアントモードで を実行し、 のサーバーに接続します `hpc7g-1`。以下の例のような出力が表示されます。

```
% /opt/amazon/efa/bin/fi_pingpong -p efa 10.3.140.69
```

bytes	#sent	#ack	total	time	MB/sec	usec/xfer	Mxfers/sec

64	10	=10	1.2k	0.00s	3.08	20.75	0.05
256	10	=10	5k	0.00s	21.24	12.05	0.08
1k	10	=10	20k	0.00s	82.91	12.35	0.08
4k	10	=10	80k	0.00s	311.48	13.15	0.08

```
[error] util/pingpong.c:1876: fi_close (-22) fid 0
```

(オプション) CloudFormation テンプレートを使用して EFA 対応の起動テンプレートを作成する

EFA の設定にはいくつかの依存関係があるため、コンピューティングノードグループの設定に使用できる CloudFormation テンプレートが用意されています。最大 4 つのネットワークカードを持つインスタンスをサポートします。複数のネットワークカードを持つインスタンスの詳細については、「Amazon [Elastic Compute Cloud ユーザーガイド](#)」の「[Elastic Network Interface](#)」を参照してください。

次の URL から CloudFormation テンプレートをダウンロードし、PCS AWS リージョン を使用するの CloudFormation AWS コンソールにアップロードします。

```
https://aws-hpc-recipes.s3.amazonaws.com/main/recipes/pcs/enable_efa/assets/pcs-lt-efa.yaml
```

CloudFormation コンソールでテンプレートを開いた状態で、次の値を入力します。テンプレートはいくつかのデフォルトのパラメータ値を提供することに注意してください。デフォルト値のままにしておくことができます。

- スタック名を指定する
 - スタック名にわかりやすい名前を入力します。など、PCS AWS コンピューティングノードグループに選択する名前を組み込むことをお勧めします **`NODEGROUPNAME`**-efa-lt。
- パラメータの下
 - NumberOfNetworkCards で、ノードグループに含まれるインスタンス内のネットワークカードの数を選択します。
 - VpcId で、PCS クラスターがデプロイされている VPC AWS を選択します。
 - NodeGroupSubnetId で、EFA 対応インスタンスを起動するクラスター VPC 内のサブネットを選択します。
 - PlacementGroupName で、フィールドを空白のままにして、ノードグループの新しいクラスタープレイスメントグループを作成します。使用する既存のプレイスメントグループがある場合は、ここにその名前を入力します。

- ClusterSecurityGroupId で、クラスター内の他のインスタンスと AWS PCS API へのアクセスを許可するために使用するセキュリティグループを選択します。多くのお客様は、クラスター VPC からデフォルトのセキュリティグループを選択します。
- SshSecurityGroupId で、クラスター内のノードへのインバウンド SSH アクセスを許可するために使用するセキュリティグループの ID を指定します。
- SshKeyName で、クラスター内のノードにアクセスするための SSH キーペアを選択します。
- LaunchTemplateName には、などの起動テンプレートのわかりやすい名前を入力します **`NODEGROUPNAME`**-efa-1t。名前は、PCS AWS リージョン を使用する AWS アカウント の AWS に一意である必要があります。
- 機能の下
 - が IAM リソースを作成する AWS CloudFormation 可能性があることを確認する のチェックボックスをオンにします。

CloudFormation スタックのステータスをモニタリングします。起動テンプレートに到達する CREATE_COMPLETE と、使用する準備が整います。「」で前述したように、PCS AWS コンピューティングノードグループで使います [EFA のコンピューティングノードグループを作成または更新する](#)。

PCS AWS でのネットワークファイルシステムの使用

AWS Parallel Computing Service (AWS PCS) コンピューティングノードグループで起動されたノードにネットワークファイルシステムをアタッチして、データとファイルを書き込んでアクセスできる永続的な場所を提供できます。[Amazon Elastic File System](#) (Amazon EFS)、[Amazon FSx for Lustre](#)、[Amazon FSx for NetApp ONTAP](#)、[Amazon FSx FSx for OpenZFS](#)、[Amazon File Cache](#) などの AWS のサービスが提供するファイルシステムを使用できます。NFS サーバーなどのセルフマネージドファイルシステムを使用することもできます。

このトピックでは、AWS PCS でネットワークファイルシステムを使用する際の考慮事項と例について説明します。

ネットワークファイルシステムの使用に関する考慮事項

さまざまなファイルシステムの実装の詳細は異なりますが、いくつかの一般的な考慮事項があります。

- 関連するファイルシステムソフトウェアをインスタンスにインストールする必要があります。例えば、Amazon FSx for Lustre を使用するには、適切なLustreパッケージが存在する必要があります。これは、コンピューティングノードグループ AMI に含めるか、インスタンスの起動時に実行されるスクリプトを使用することで実現できます。
- 共有ネットワークファイルシステムとコンピューティングノードグループインスタンスの間にネットワークルートが必要です。
- 共有ネットワークファイルシステムとコンピューティングノードグループインスタンスの両方のセキュリティグループルールでは、関連するポートへの接続を許可する必要があります。
- ファイルシステムにアクセスするリソース全体で一貫したPOSIXユーザーとグループの名前空間を維持する必要があります。そうしないと、PCS クラスターで実行されるジョブやインタラクティブプロセスでアクセス許可エラーが発生する可能性があります。
- ファイルシステムのマウントは、EC2起動テンプレートを使用して行われます。ネットワークファイルシステムのマウント時にエラーやタイムアウトが発生すると、インスタンスがジョブを実行できなくなる可能性があります。これにより、予期しないコストが発生する可能性があります。起動テンプレートのデバッグの詳細については、「」を参照してください[PCS での Amazon EC2 AWS 起動テンプレートの使用](#)。

ネットワークマウントの例

Amazon EFS、Amazon FSx for Lustre、Amazon FSx for NetApp ONTAP、Amazon FSx for OpenZFS、Amazon File Cache を使用してファイルシステムを作成できます。以下の関連セクションを展開して、各ネットワークマウントの例を確認してください。

Amazon EFS

ファイルシステムのセットアップ

Amazon EFS ファイルシステムを作成します。PCS コンピューティングノードグループインスタンスを起動する各アベイラビリティーゾーンにマウントターゲットがあることを確認します。また、各マウントターゲットが、PCS コンピューティングノードグループインスタンスからのインバウンドアクセスとアウトバウンドアクセス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ことを確認します。詳細については、Amazon Elastic File Systemユーザーガイド」の[「マウントターゲットとセキュリティグループ」](#)を参照してください。

起動テンプレート

ファイルシステムのセットアップから、コンピューティングノードグループに使用する起動テンプレートにセキュリティグループ (複数可) を追加します。

Amazon EFS ファイルシステムをマウントするcloud-configメカニズムを使用するユーザーデータを含めます。このスクリプトの次の値を独自の詳細に置き換えます。

- *mount-point-directory* – Amazon EFS をマウントする各インスタンスのパス
- *filesystem-id* – EFS ファイルシステムのファイルシステム ID

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

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

packages:
  - amazon-efs-utils

runcmd:
  - mkdir -p /mount-point-directory
  - echo "filesystem-id:/ mount-point-directory efs tls,_netdev" >> /etc/fstab
```

```
- mount -a -t efs defaults

---MYBOUNDARY---
```

Amazon FSx for Lustre

ファイルシステムのセットアップ

PCS を使用する VPC AWS に FSx for Lustre ファイルシステムを作成します。ゾーン間転送を最小限に抑えるには、PCS コンピューティングノードグループインスタンスの大部分を起動するのと同じアベイラビリティゾーンのサブネットに をデプロイします。ファイルシステムが、PCS コンピューティングノードグループインスタンスからのインバウンドアクセスとアウトバウンドアクセス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ことを確認します。セキュリティグループの詳細については、[「Amazon FSx for Lustre ユーザーガイド」の「Amazon VPC によるファイルシステムのアクセスコントロール」](#)を参照してください。 FSx

起動テンプレート

が FSx for Lustre ファイルシステムのマウントcloud-configに使用するユーザーデータを含めます。このスクリプトの次の値を独自の詳細に置き換えます。

- *mount-point-directory* – FSx for Lustre をマウントするインスタンスのパス
- *filesystem-id* – FSx for Lustre ファイルシステムのファイルシステム ID
- *mount-name* – FSx for Lustre ファイルシステムのマウント名
- *region-code* – AWS リージョン FSx for Lustre ファイルシステムがデプロイされている (PCS AWS システムと同じである必要があります)
- (オプション) *latest* – FSx for Lustre でLustreサポートされている の任意のバージョン

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

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

runcmd:
- amazon-linux-extras install -y lustre=latest
- mkdir -p /mount-point-directory
- mount -t lustre filesystem-id.fsx.region-code.amazonaws.com@tcp:/mount-name /mount-point-directory
```

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

Amazon FSx for NetApp ONTAP

ファイルシステムのセットアップ

AWS PCS を使用する VPC に Amazon FSx for NetApp ONTAP ファイルシステムを作成します。ゾーン間転送を最小限に抑えるには、PCS AWS コンピューティングノードグループインスタンスの大部分を起動するのと同じアベイラビリティゾーンのサブネットに をデプロイします。ファイルシステムが、AWS PCS コンピューティングノードグループインスタンスからのインバウンドアクセスとアウトバウンドアクセス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ことを確認します。セキュリティグループの詳細については、FSx for ONTAP ユーザーガイド」の「[Amazon VPC によるファイルシステムのアクセスコントロール](#)」を参照してください。

起動テンプレート

が FSx for ONTAP ファイルシステムのルートボリュームをマウントcloud-configするために使用するユーザーデータを含めます。このスクリプトの次の値を独自の詳細に置き換えます。

- *mount-point-directory* – FSx for ONTAP ボリュームをマウントするインスタンスのパス
- *svm-id* – FSx for ONTAP ファイルシステムの SVM ID
- *filesystem-id* – FSx for ONTAP ファイルシステムのファイルシステム ID
- *region-code* – AWS リージョン FSx for ONTAP ファイルシステムがデプロイされている (PCS AWS システムと同じである必要があります)
- *volume-name* – FSx for ONTAP ボリューム名

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

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

runcmd:
- mkdir -p /mount-point-directory
- mount -t nfs svm-id.filesystem-id.fsx.region-code.amazonaws.com:/volume-name /mount-point-directory
```



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

Amazon FSx for OpenZFS

ファイルシステムのセットアップ

PCS を使用する VPC AWS に FSx for OpenZFS ファイルシステムを作成します。ゾーン間転送を最小限に抑えるには、PCS AWS コンピューティングノードグループインスタンスの大部分を起動すると同じアベイラビリティゾーンのサブネットに をデプロイします。ファイルシステムが、AWS PCS コンピューティングノードグループインスタンスからのインバウンドアクセスとアウトバウンドアクセス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ことを確認します。セキュリティグループの詳細については、FSx for OpenZFS ユーザーガイド」の「[Amazon VPC によるファイルシステムアクセスの管理](#)」を参照してください。

起動テンプレート

が FSx for OpenZFS ファイルシステムのルートボリュームをマウントcloud-configするために使用するユーザーデータを含めます。このスクリプトの次の値を独自の詳細に置き換えます。

- *mount-point-directory* – FSx for OpenZFS 共有をマウントするインスタンスのパス
- *filesystem-id* – FSx for OpenZFS ファイルシステムのファイルシステム ID
- *region-code* – AWS リージョン FSx for OpenZFS ファイルシステムがデプロイされている (PCS AWS システムと同じである必要があります)

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

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

runcmd:
- mkdir -p /mount-point-directory
- mount -t nfs -o noatime,nfsvers=4.2,rsync,rsync=1048576,wsync=1048576 filesystem-id.fsx.region-code.amazonaws.com:/fsx/ /mount-point-directory

--==MYBOUNDARY==
```

Amazon File Cache

ファイルシステムのセットアップ

AWS PCS を使用する VPC に [Amazon ファイルキャッシュ](#) を作成します。ゾーン間転送を最小限に抑えるには、PCS コンピューティングノードグループインスタンスの大部分を起動するのと同じアベイラビリティゾーン内のサブネットを選択します。ファイルキャッシュが、PCS インスタンスとファイルキャッシュ間のポート 988 でのインバウンドトラフィックとアウトバウンドトラフィックを許可するセキュリティグループに関連付けられていることを確認します。セキュリティグループの詳細については、「[Amazon File Cache User Guide](#)」の「[Cache access control with Amazon VPC](#)」を参照してください。

起動テンプレート

ファイルシステムのセットアップから、コンピューティングノードグループに使用する起動テンプレートにセキュリティグループ (複数可) を追加します。

が Amazon File Cache のマウント cloud-config に使用するユーザーデータを含めます。このスクリプトの次の値を独自の詳細に置き換えます。

- *mount-point-directory* – FSx for Lustre をマウントするインスタンスのパス
- *cache-dns-name* – ファイルキャッシュのドメインネームシステム (DNS) 名
- *mount-name* – ファイルキャッシュのマウント名

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

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

runcmd:
- amazon-linux-extras install -y lustre=2.12
- mkdir -p /mount-point-directory
- mount -t lustre -o relatime,flock cache-dns-name@tcp://mount-name /mount-point-directory

--MYBOUNDARY--
```

PCS 用の Amazon AWS マシンイメージ (AMIs)

AWS PCS は、お客様が提供する AMIs と連携するため、クラスター内のノードにあるソフトウェアと設定に高い柔軟性がもたらされます。PCS を試す場合は、AWS が提供するサンプル AMI を使用し、が維持できます AWS。本稼働環境で PCS AWS を使用している場合は、独自の AMIs を構築することをお勧めします。このトピックでは、サンプル AMIs を検出して使用方法と、カスタマイズされた独自の AMIs を構築して使用方法について説明します。

トピック

- [PCS でサンプル Amazon マシンイメージ \(AMIs\) AWS を使用する](#)
- [PCS のカスタム Amazon AWS マシンイメージ \(AMIs\)](#)
- [PCS 用のカスタム AMIs AWS を構築するためのソフトウェアインストーラ](#)
- [PCS サンプル AWS AMIs リリースノート](#)

PCS でサンプル Amazon マシンイメージ (AMIs) AWS を使用する

AWS には、[PCS AMIs](#) AWS が用意されています。

Important

サンプル AMIs はデモンストレーション用であり、本番稼働用ワークロードには推奨されません。

Important

PCS サンプル AMIs AWS と複数のネットワークインターフェイスで設定されたコンピューティングノードグループは、サブネットが IPv6 を使用するようにより設定されている場合、現在機能しません。代わりに、デュアルスタックサブネット (IPv4 および IPv6) または IPv4-only サブネットを使用します。

現在の AWS PCS サンプル AMIs

AWS マネジメントコンソール

AWS PCS サンプル AMIs には、次の命名規則があります。

```
aws-pcs-sample_ami-OS-architecture-scheduler-scheduler-major-version
```

使用できる値

- *OS* – amzn2
- ##### – x86_64 または arm64
- ス##### – slurm
- ス*scheduler-major-version* – 25.05

AWS PCS サンプル AMIs を検索するには

1. [Amazon EC2 コンソール](#)を開きます。
2. AMIs に移動します。
3. 「パブリックイメージ」を選択します。
4. 属性またはタグで AMI を検索する で、テンプレート化された名前を使用して AMI を検索します。

例

- Arm64 インスタンスでの Slurm 25.05 のサンプル AMI

```
aws-pcs-sample_ami-amzn2-arm64-slurm-25.05
```

- x86 インスタンスでの Slurm 25.05 のサンプル AMI

```
aws-pcs-sample_ami-amzn2-x86_64-slurm-25.05
```

Note

複数の AMIs がある場合は、最新のタイムスタンプで AMI を使用します。

5. コンピューティングノードグループを作成または更新するときは、AMI ID を使用します。

AWS CLI

最新の AWS PCS サンプル AMI は、以下のコマンドで確認できます。*region-code* を、などの AWS PCS AWS リージョン 使用する に置き換えます us-east-1。

- x86_64

```
aws ec2 describe-images --region region-code --owners amazon \
--filters 'Name=name,Values=aws-pcs-sample_ami-amzn2-x86_64-slurm-25.05*' \
          'Name=state,Values=available' \
--query 'sort_by(Images, &CreationDate)[-1].[Name,ImageId]' --output text
```

- Arm64

```
aws ec2 describe-images --region region-code --owners amazon \
--filters 'Name=name,Values=aws-pcs-sample_ami-amzn2-arm64-slurm-25.05*' \
          'Name=state,Values=available' \
--query 'sort_by(Images, &CreationDate)[-1].[Name,ImageId]' --output text
```

コンピューティングノードグループを作成または更新するときは、AMI ID を使用します。

AWS PCS サンプル AMIs の詳細

AWS PCS サンプル AMIs 「」を参照してください[PCS サンプル AWS AMIs リリースノート](#)。

PCS と互換性のある独自の AMIs AWS を構築する

PCS で動作する独自の AMIs 「」を参照してください[PCS のカスタム Amazon AWS マシンイメージ \(AMIs\)](#)。AWS

PCS のカスタム Amazon AWS マシンイメージ (AMIs)

AWS PCS は、サービスに持ち込む Amazon マシンイメージ (AMI) と連携するように設計されています。これらの AMIs には、PCS エージェントと互換性のあるバージョンの Slurm AWS が正しくインストールおよび設定されている限り、任意のソフトウェアと設定をインストールできます。カス

タム AMI に PCS AWS ソフトウェアをインストールするには、AWS が提供するインストーラを使用する必要があります。AWS カスタム AMI に Slurm をインストールするには、が提供するインストーラを使用することをお勧めしますが、必要に応じて Slurm を独自にインストールできます (推奨されません)。

Note

カスタム AMI を構築せずに AWS PCS を試す場合は、が提供するサンプル AMI を使用できます AWS。詳細については、「[PCS でサンプル Amazon マシンイメージ \(AMIs\) AWS を使用する](#)」を参照してください。

Important

AWS 現在、PCS では、IPv6 専用ネットワークで PCS を使用している場合でも、ローカルノード通信に IPv4 IPv6-only をサポートするカーネルが必要です。AWS

このチュートリアルは、HPC および AI/ML ワークロードを強化するために PCS コンピューティングノードグループで利用できる AMI を作成するのに役立ちます。

トピック

- [ステップ 1 – 一時インスタンスを起動する](#)
- [ステップ 2 – AWS PCS エージェントをインストールする](#)
- [ステップ 3 – Slurm をインストールする](#)
- [ステップ 4 – \(オプション\) 追加のドライバー、ライブラリ、アプリケーションソフトウェアをインストールする](#)
- [ステップ 5 – PCS と互換性のある AMI AWS を作成する](#)
- [ステップ 6 – PCS コンピューティングノードグループでカスタム AMI AWS を使用する](#)
- [ステップ 7 – 一時インスタンスを終了する](#)

ステップ 1 – 一時インスタンスを起動する

PCS AWS ソフトウェアと Slurm スケジューラのインストールと設定に使用できる一時インスタンスを起動します。このインスタンスを使用して、PCS と互換性のある AMI AWS を作成します。

一時インスタンスを起動するには

1. [Amazon EC2 コンソール](#)を開きます。
2. ナビゲーションペインで、インスタンスを選択し、インスタンスの起動を選択して新しいインスタンス起動ウィザードを開きます。
3. (オプション) 名前とタグセクションで、 などのインスタンスの名前を指定します PCS-AMI-instance。指定した名前はリソースタグとしてインスタンスに割り当てられます (Name=PCS-AMI-instance)。
4. [Application and OS Images] (アプリケーションと OS イメージ) セクションで、[サポートされるオペレーティングシステム](#)を選択してください。
5. [Instance type] (インスタンスタイプ) セクションで、[サポートされているインスタンスタイプ](#)を選択してください。
6. [Key pair] (キーペア) セクションで、インスタンスに使用するキーペアを選択してください。
7. [ネットワーク設定] セクションで以下の操作を行います。
 - ファイアウォール (セキュリティグループ) で、既存のセキュリティグループを選択し、インスタンスへのインバウンド SSH アクセスを許可するセキュリティグループを選択します。
8. [Storage] (ストレージ) セクションで、必要に応じてボリュームを設定します。独自のアプリケーションとライブラリをインストールするのに十分なスペースを設定してください。
9. [Summary] (サマリー) パネルで、[Launch instance] (インスタンスの起動) を選択してください。

ステップ 2 – AWS PCS エージェントをインストールする

Slurm で使用する PCS AWS によって起動されたインスタンスを設定するエージェントをインストールします。PCS エージェントの詳細については、AWS 「」を参照してください [AWS PCS エージェントバージョン](#)。

AWS PCS エージェントをインストールするには

1. 起動したインスタンスに接続します。詳細については、「Linux インスタンスへの接続」を参照してください。
2. (オプション) すべてのソフトウェアパッケージが最新であることを確認するには、インスタンスでクイックソフトウェア更新を実行します。このプロセスには数分かかることがあります。

- Amazon Linux 2、Amazon Linux 2023、RHEL 9、RHEL 8、Rocky Linux 9、Rocky Linux 8

```
sudo yum update -y
```

- Ubuntu 22.04 および Ubuntu 24.04

```
sudo apt-get update && sudo apt-get upgrade -y
```

3. インスタンスを再起動して、そのインスタンスに再接続します。
4. AWS PCS エージェントのインストールファイルをダウンロードします。インストールファイルは、圧縮された tarball (.tar.gz) ファイルにパッケージ化されます。次のコマンドを使用して、安定している最新バージョンをダウンロードします。region は、 などの一時インスタンスを起動 AWS リージョン した に置き換えます us-east-1。

```
curl https://aws-pcs-repo-region.s3.region.amazonaws.com/aws-pcs-agent/aws-pcs-agent-v1.3.2-1.tar.gz -o aws-pcs-agent-v1.3.2-1.tar.gz
```

latest 前のコマンドでバージョン番号を に置き換えることで、最新バージョンを取得することもできます (例: aws-pcs-agent-v1-latest.tar.gz)。

Note

これは、AWS PCS エージェントソフトウェアの今後のリリースで変更される可能性があります。

5. (オプション) PCS AWS ソフトウェア tarball の信頼性と整合性を検証します。ソフトウェアパブリッシャーの ID を検証し、発行後にファイルの改変や破損がないことを確認するために、これを行うことをお勧めします。
 - a. PCS AWS のパブリック GPG キーをダウンロードし、キーリングにインポートします。region は、一時インスタンスを起動 AWS リージョン した に置き換えます。コマンドはキーの値を返します。キー値を記録します。次のステップで使用します。

```
wget https://aws-pcs-repo-public-keys-region.s3.region.amazonaws.com/aws-pcs-public-key.pub && \
    gpg --import aws-pcs-public-key.pub
```

- b. 次のコマンドを実行して、GPG キーのフィンガープリントを確認します。


```
gpg --fingerprint 7EEF030EDDF5C21C
```

コマンドは、以下と同じフィンガープリントを返します。

```
1C24 32C1 862F 64D1 F90A 239A 7EEF 030E DDF5 C21C
```

Important

フィンガープリントが一致しない場合は、AWS PCS エージェントのインストールスクリプトを実行しないでください。[AWS Support](#) にお問い合わせください。

- c. 署名ファイルをダウンロードし、PCS AWS ソフトウェア tarball ファイルの署名を確認します。**region** を、などの一時インスタンスを起動 AWS リージョンした に置き換えますus-east-1。

```
wget https://aws-pcs-repo-region.s3.region.amazonaws.com/aws-pcs-agent/aws-pcs-agent-v1.3.2-1.tar.gz.sig && \
gpg --verify ./aws-pcs-agent-v1.3.2-1.tar.gz.sig
```

出力は次の例のようになります:

```
gpg: assuming signed data in './aws-pcs-agent-v1.3.2-1.tar.gz'
gpg: Signature made Thu 06 Nov 2025 11:10:36 AM CET using RSA key ID ECC0AE5C
gpg: Good signature from "AWS PCS Packages (AWS PCS Packages)"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 1C24 32C1 862F 64D1 F90A 239A 7EEF 030E DDF5 C21C
Subkey fingerprint: B7E1 8788 3517 6A74 C3D5 EAF5 6088 136D ECC0 AE5C
```

結果に が含まれGood signature、フィンガープリントが前のステップで返されたフィンガープリントと一致する場合は、次のステップに進みます。

Important

フィンガープリントが一致しない場合は、AWS PCS ソフトウェアのインストールスクリプトを実行しないでください。[AWS Support](#) にお問い合わせください。

6. 圧縮ファイルから .tar.gz ファイルを抽出し、抽出されたディレクトリに移動します。

```
tar -xf aws-pcs-agent-v1.3.2-1.tar.gz && \  
cd aws-pcs-agent
```

7. PCS AWS ソフトウェアをインストールします。

```
sudo ./installer.sh
```

8. PCS AWS ソフトウェアバージョンファイルをチェックして、正常にインストールされたことを確認します。

```
cat /opt/aws/pcs/version
```

出力は次の例のようになります:

```
AGENT_INSTALL_DATE='Fri Dec 13 12:28:43 UTC 2024'  
AGENT_VERSION='1.3.2'  
AGENT_RELEASE='1'
```

ステップ 3 – Slurm をインストールする

PCS と互換性のあるバージョンの Slurm AWS をインストールします。詳細については、「[PCS の Slurm AWS バージョン](#)」を参照してください。

Note

以前のバージョンの Slurm ソフトウェアがインストールされている AMI がある場合は、次の手順を実行して新しいバージョンの Slurm をインストールする必要があります。AWS PCS エージェントは、クラスターの作成時に設定された Slurm バージョンに従って、実行時に正しいバージョンの Slurm バイナリを有効にします。

Slurm をインストールするには

1. PCS AWS ソフトウェアをインストールしたのと同じ一時インスタンスに接続します。
2. Slurm インストーラソフトウェアをダウンロードします。Slurm インストーラは、圧縮された tarball (.tar.gz) ファイルにパッケージ化されています。次のコマンドを使用して、安定して

いる最新バージョンをダウンロードします。region を、 などの一時インスタンス AWS リージョン の に置き換えます us-east-1。

```
curl https://aws-pcs-repo-region.s3.region.amazonaws.com/aws-pcs-slurm/aws-pcs-slurm-25.05-installer-25.05.4-1.tar.gz \
-o aws-pcs-slurm-25.05-installer-25.05.4-1.tar.gz
```

latest 前のコマンドでバージョン番号を に置き換えることで、最新バージョンを取得することもできます (例: aws-pcs-slurm-25.05-installer-latest.tar.gz)。チェックサムで使用可能なバージョンの完全なリストについては、「」を参照してください[PCS の Slurm AWS バージョン](#)。

 Note

これは、Slurm インストーラソフトウェアの今後のリリースで変更される可能性があります。

3. (オプション) Slurm インストーラ tarball の信頼性と整合性を検証します。ソフトウェアパブリッシャーの ID を検証し、発行後にファイルの改変や破損がないことを確認するために、これを行うことをお勧めします。
 - a. PCS AWS のパブリック GPG キーをダウンロードし、キーリングにインポートします。region は、一時インスタンスを起動 AWS リージョン した に置き換えます。コマンドはキーの値を返します。キー値を記録します。次のステップで使用します。

```
wget https://aws-pcs-repo-public-keys-region.s3.region.amazonaws.com/aws-pcs-public-key.pub && \
gpg --import aws-pcs-public-key.pub
```

- b. 次のコマンドを実行して、GPG キーのフィンガープリントを確認します。

```
gpg --fingerprint 7EEF030EDDF5C21C
```

コマンドは、以下と同じフィンガープリントを返します。

```
1C24 32C1 862F 64D1 F90A 239A 7EEF 030E DDF5 C21C
```

⚠ Important

フィンガープリントが一致しない場合は、Slurm インストールスクリプトを実行しないでください。[AWS Support](#) にお問い合わせください。

- c. 署名ファイルをダウンロードし、Slurm インストーラ tarball ファイルの署名を確認します。**region** を、などの一時インスタンスを起動 AWS リージョン **us-east-1** に置き換えます。

```
wget https://aws-pcs-repo-region.s3.region.amazonaws.com/aws-pcs-slurm/aws-pcs-slurm-25.05-installer-25.05.4-1.tar.gz.sig && \
gpg --verify ./aws-pcs-slurm-25.05-installer-25.05.4-1.tar.gz.sig
```

出力は次の例ようになります:

```
gpg: assuming signed data in './aws-pcs-slurm-25.05-installer-25.05.4-1.tar.gz'
gpg: Signature made Fri 24 Oct 2025 05:05:11 PM UTC using RSA key ID ECC0AE5C
gpg: Good signature from "AWS PCS Packages (AWS PCS Packages)"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 1C24 32C1 862F 64D1 F90A 239A 7EEF 030E DDF5 C21C
Subkey fingerprint: B7E1 8788 3517 6A74 C3D5 EAF5 6088 136D ECC0 AE5C
```

結果に **Good signature**、フィンガープリントが前のステップで返されたフィンガープリントと一致する場合は、次のステップに進みます。

⚠ Important

フィンガープリントが一致しない場合は、Slurm インストールスクリプトを実行しないでください。[AWS Support](#) にお問い合わせください。

4. 圧縮された .tar.gz ファイルからファイルを展開し、展開されたディレクトリに移動します。

```
tar -xf aws-pcs-slurm-25.05-installer-25.05.4-1.tar.gz && \
cd aws-pcs-slurm-25.05-installer
```

5. Slurm をインストールします。インストーラは、Slurm とその依存関係をダウンロード、コンパイル、インストールします。選択した一時インスタンスの仕様に応じて、数分かかります。

```
sudo ./installer.sh -y
```

6. スケジューラのバージョンファイルをチェックして、インストールを確認します。

```
cat /opt/aws/pcs/scheduler/slurm-25.05/version
```

出力は次の例のようになります:

```
SLURM_INSTALL_DATE='Mon Nov 3 14:23:38 UTC 2025'  
SLURM_VERSION='25.05.4'  
PCS_SLURM_RELEASE='1'
```

ステップ 4 – (オプション) 追加のドライバー、ライブラリ、アプリケーションソフトウェアをインストールする

一時インスタンスに追加のドライバー、ライブラリ、アプリケーションソフトウェアをインストールします。インストール手順は、特定のアプリケーションとライブラリによって異なります。AWS PCS 用のカスタム AMI を以前に構築したことがない場合は、まず PCS ソフトウェアと Slurm がインストールされている AMI AWS を構築してテストし、最初の成功を確認したら独自のソフトウェアと設定を段階的に追加することをお勧めします。

例

- Elastic Fabric Adapter (EFA) ソフトウェア。詳細については、[「Amazon Elastic Compute Cloud ユーザーガイド」の「Get started with EFA and MPI for HPC workloads on Amazon EC2」](#)を参照してください。
- Amazon Elastic File System (Amazon EFS) クライアント。詳細については、[「Amazon Elastic File System ユーザーガイド」の「Amazon EFS クライアントを手動でインストールする」](#)を参照してください。Amazon Elastic File System
- Amazon FSx for Lustre と Amazon File Cache を使用する Lustre クライアント。詳細については、FSx for [Lustre ユーザーガイドの「Lustre クライアントのインストール」](#)を参照してください。FSx
- Amazon CloudWatch Logs and Metrics を使用する Amazon CloudWatch エージェント。詳細については、「Amazon [CloudWatch ユーザーガイド](#)」の「[CloudWatch エージェント](#)のインストール」を参照してください。Amazon CloudWatch

- AWS Neuron、trn* および inf* インスタンスタイプを使用します。詳細については、[AWS Neuron のドキュメント](#)を参照してください。
- p* または g* インスタンスタイプを使用する NVIDIA ドライバー、CUDA、および DCGM。

ステップ 5 – PCS と互換性のある AMI AWS を作成する

必要なソフトウェアコンポーネントをインストールしたら、再利用して PCS コンピューティング ノードグループでインスタンスを起動できる AMI AWS を作成します。

Important

AWS 現在、PCS では、IPv6 専用ネットワークで PCS を使用している場合でも、ローカル ノード通信に IPv4 IPv6-only をサポートするカーネルが必要です。AWS

一時インスタンスから AMI を作成するには

1. [Amazon EC2 コンソール](#)を開きます。
2. ナビゲーションペインで、[インスタンス] を選択してください。
3. 作成した一時インスタンスを選択します。アクション、イメージ、イメージの作成 を選択します。
4. [イメージの作成] で、次を行います。
 - a. [イメージ名] に、の分かりやすい AMI 名を入力してください。
 - b. (オプション) [イメージの説明] に、AMI の簡単な説明を入力してください。
 - c. [イメージを作成] を選択してください。
5. ナビゲーションペインで [AMI] を選択してください。
6. リストで作成した AMI を見つけます。ステータスが Pending から Available に変わるのを待ってから、PCS AWS コンピューティングノードグループで使します。

ステップ 6 – PCS コンピューティングノードグループでカスタム AMI AWS を使用する

カスタム AMI は、新規または既存の PCS AWS コンピューティングノードグループで使用できます。

⚠ Important

AWS 現在、PCS では、IPv6 専用ネットワークで PCS を使用している場合でも、ローカルノード通信に IPv4 IPv6-only をサポートするカーネルが必要です。AWS

New compute node group

カスタム AMI を使用するには

1. [AWS PCS コンソール](#)を開きます。
2. ナビゲーションペインで [Clusters] (クラスター) を選択してください。
3. カスタム AMI を使用するクラスターを選択し、コンピューティングノードグループを選択します。
4. 新しいコンピューティングノードグループを作成します。詳細については、「[PCS AWS でのコンピューティングノードグループの作成](#)」を参照してください。AMI ID で、使用するカスタム AMI の名前または ID を検索します。コンピューティングノードグループの設定を終了し、コンピューティングノードグループの作成を選択します。
5. (オプション) AMI がインスタンスの起動をサポートしていることを確認します。コンピューティングノードグループでインスタンスを起動します。これを行うには、1 つの静的インスタンスを持つようにコンピューティングノードグループを設定するか、コンピューティングノードグループを使用するキューにジョブを送信できます。
 - a. インスタンスに新しいコンピューティングノードグループ ID がタグ付けされるまで、Amazon EC2 コンソールを確認します。詳細については、「」を参照してください [PCS AWS でのコンピューティングノードグループインスタンスの検索](#)。
 - b. インスタンスの起動が表示され、ブートストラッププロセスが完了したら、想定される AMI を使用していることを確認します。これを行うには、インスタンスを選択し、詳細で AMI ID を検査します。これは、コンピューティングノードグループ設定で設定した AMI と一致する必要があります。
 - c. (オプション) コンピューティングノードグループのスケーリング設定を任意の値に更新します。

Existing compute node group

カスタム AMI を使用するには

1. [AWS PCS コンソール](#)を開きます。
2. ナビゲーションペインで [Clusters] (クラスター) を選択してください。
3. カスタム AMI を使用するクラスターを選択し、コンピューティングノードグループを選択します。
4. 設定するノードグループを選択し、編集を選択します。AMI ID で、使用するカスタム AMI の名前または ID を検索します。コンピューティングノードグループの設定を終了し、更新を選択します。コンピューティングノードグループで起動された新しいインスタンスは、更新された AMI ID を使用します。既存のインスタンスは、PCS によって置き換えられるまで、古い AMI AWS を引き続き使用します。詳細については、「[PCS AWS コンピューティングノードグループの更新](#)」を参照してください。
5. (オプション) AMI がインスタンスの起動をサポートしていることを確認します。コンピューティングノードグループでインスタンスを起動します。これを行うには、1 つの静的インスタンスを持つようにコンピューティングノードグループを設定するか、コンピューティングノードグループを使用するキューにジョブを送信できます。
 - a. インスタンスに新しいコンピューティングノードグループ ID がタグ付けされるまで、Amazon EC2 コンソールを確認します。詳細については、「」を参照してください [PCS AWS でのコンピューティングノードグループインスタンスの検索](#)。
 - b. インスタンスの起動が表示され、ブートストラッププロセスが完了したら、想定される AMI を使用していることを確認します。これを行うには、インスタンスを選択し、詳細で AMI ID を検査します。これは、コンピューティングノードグループ設定で設定した AMI と一致する必要があります。
 - c. (オプション) コンピューティングノードグループのスケーリング設定を任意の値に更新します。

ステップ 7 – 一時インスタンスを終了する

AMI が AWS PCS で意図したとおりに動作することを確認したら、一時インスタンスを終了して料金の発生を停止できます。

一時インスタンスを終了するには

1. [Amazon EC2 コンソール](#)を開きます。

2. ナビゲーションペインで、[インスタンス] を選択してください。
3. 作成した一時インスタンスを選択し、アクション、インスタンスの状態、インスタンスの終了を選択します。
4. 確認を求められたら、終了を選択します。

PCS 用のカスタム AMIs AWS を構築するためのソフトウェアインストーラ

AWS は、インスタンスに PCS AWS ソフトウェアをインストールできるダウンロード可能なファイルを提供します。は、関連するバージョンの Slurm とその依存関係をダウンロード、コンパイル、およびインストールできるソフトウェア AWS も提供します。これらの手順を使用して、PCS で使用するカスタム AMIs AWS を構築することも、独自の方法を使用することもできます。

目次

- [AWS PCS エージェントソフトウェアインストーラ](#)
- [Slurm インストーラ](#)
- [サポートされるオペレーティングシステム](#)
- [サポートされるインスタンスタイプ](#)
- [サポートされている Slurm バージョン](#)
- [チェックサムを使用してインストーラを検証する](#)

AWS PCS エージェントソフトウェアインストーラ

AWS PCS エージェントソフトウェアインストーラは、インスタンスブートストラッププロセス中に PCS AWS と連携するようにインスタンスを設定します。カスタム AMI に AWS PCS エージェントをインストールするには、AWS が提供するインストーラを使用する必要があります。

PCS エージェントソフトウェアの詳細については、AWS 「」を参照してください[AWS PCS エージェントバージョン](#)。

Slurm インストーラ

Slurm インストーラは、関連するバージョンの Slurm とその依存関係をダウンロード、コンパイル、インストールします。Slurm インストーラを使用して、PCS 用のカスタム AMIs AWS を構築できます。Slurm インストーラが提供するソフトウェア設定と整合性がある場合は、独自のメカニズム

を使用することもできます。Slurm の AWS PCS サポートの詳細については、「」を参照してください [PCS の Slurm AWS バージョン](#)。

AWSが提供するソフトウェアは、以下をインストールします。

- リクエストされたメジャーバージョンとメンテナンスバージョン (現在のバージョン 25.05.x) の [Slurm - ライセンス GPL 2](#)
 - Slurm は を `--sysconfdir` に設定して構築されます `/etc/slurm`
 - Slurm は オプション `--enable-pam` と を使用して構築されています。 `--without-munge`
 - Slurm は オプションで構築されています `--sharedstatedir=/run/slurm/`
 - Slurm は PMIX と JWT サポートで構築されています
 - Slurm が にインストールされている `/opt/aws/pcs/schedulers/slurm-25.05`
- [OpenPMIX](#) (バージョン 4.2.6) – [ライセンス](#)
 - OpenPMIX は のサブディレクトリとしてインストールされます。 `/opt/aws/pcs/scheduler/`
- [libjwt](#) (バージョン 1.17.0) – [ライセンス MPL-2.0](#)
 - libjwt は のサブディレクトリとしてインストールされます `/opt/aws/pcs/scheduler/`

AWSが提供するソフトウェアは、システム設定を次のように変更します。

- ビルドによって作成された Slurm systemd ファイルは、ファイル名 `/etc/systemd/system/` でコピーされます `slurmd-25.05.service`。
- 存在しない場合、Slurm ユーザーとグループ (`slurm:slurm`) は の UID/GID で作成されます 401。
- フォルダ `/etc/aws/pcs/scheduler/slurm-25.05/plugstack.conf.d/` が作成され、[SPANK プラグインを使用して AWS PCS で Slurm 機能を拡張する](#) 設定が保存されます。
- Amazon Linux 2 および Rocky Linux 9 では、インストールによって EPEL リポジトリが追加され、Slurm またはその依存関係を構築するために必要なソフトウェアがインストールされます。
- RHEL9 では、インストールによって `codeready-builder-for-rhel-9-rhui-rpms` と `epel-release-latest-9` が Slurm またはその依存関係を構築するために必要なソフトウェア `fedoraproject` をインストールできるようになります。

サポートされるオペレーティングシステム

「[PCS AWS でサポートされているオペレーティングシステム](#)」を参照してください。

Note

AWS Deep Learning AMIs Amazon Linux 2 および Ubuntu 22.04 に基づく (DLAMI) バージョンは、PCS AWS ソフトウェアおよび Slurm インストーラと互換性がある必要があります。詳細については、AWS Deep Learning AMIs デベロッパーガイドの「[DLAMI の選択](#)」を参照してください。

サポートされるインスタンスタイプ

AWS PCS ソフトウェアと Slurm インストーラは、サポートされているオペレーティングシステムのいずれかを実行できるよりも、任意の x86_64 または arm64 インスタンスタイプをサポートします。

サポートされている Slurm バージョン

「[PCS の Slurm AWS バージョン](#)」を参照してください。

チェックサムを使用してインストーラを検証する

SHA256 チェックサムを使用して、インストーラ tarball (.tar.gz) ファイルを検証できます。ソフトウェアパブリッシャーの ID を確認し、発行後にアプリケーションの変更または破損がないことを確認するために、この操作を行うことをお勧めします。

tarball を検証するには

SHA256 チェックサムに sha256sum ユーティリティを使用し、tarball ファイル名を指定します。SHA256 tarball ファイルを保存したディレクトリから コマンドを実行する必要があります。

- SHA256

```
$ sha256sum tarball_filename.tar.gz
```

コマンドは、次の形式でチェックサム値を返す必要があります。

```
checksum_value tarball_filename.tar.gz
```

コマンドによって返されるチェックサム値と、次の表に示すチェックサム値を比較します。チェックサムが一致した場合は、インストールスクリプトを安全に実行できます。

⚠ Important

チェックサムが一致しない場合は、インストールスクリプトを実行しないでください。[サポート](#) に連絡する。

たとえば、次のコマンドは Slurm 25.05.4-1 tarball の SHA256 チェックサムを生成します。

```
$ sha256sum aws-pcs-slurm-25.05-installer-25.05.4-1.tar.gz
```

出力例:

```
3b0f93bce441d4f4f6935175f2c1e81cd961cb923adb416fa6689f5592047a7d aws-pcs-slurm-25.05-installer-25.05.4-1.tar.gz
```

次の表に、インストーラの最新バージョンのチェックサムを示します。*us-east-1* を PCS AWS を使用する AWS リージョン に置き換えます。

AWS PCS エージェント

Installer (インストーラ)	URL のダウンロード	SHA256 チェックサム
AWS PCS エージェント 1.3.2-1	<code>https://aws-pcs-repo-<i>us-east-1</i>.s3.<i>us-east-1</i>.amazonaws.com/aws-pcs-agent/aws-pcs-agent-v1.3.2-1.tar.gz</code>	06b32a952a1c849e34 42e35c28ac2e4d6962 b09286cad748f3c83d 561b52ec6f
AWS PCS エージェント 1.3.1-1	<code>https://aws-pcs-repo-<i>us-east-1</i>.s3.<i>us-east-1</i>.amazonaws.com/aws-pcs-agent/aws-pcs-agent-v1.3.1-1.tar.gz</code>	5b7f1eb7b3a86bd2d3 31b5cb0138d868dc94 52da34b480becd86af 892c7e8d19

Installer (インストーラ)	URL のダウンロード	SHA256 チェックサム
	-agent-v1.3.1-1.tar.gz	
AWS PCS エージェント 1.3.0-1	https://aws-pcs-repo-us-east-1.s3.us-east-1.amazonaws.com/aws-pcs-agent/aws-pcs-agent-v1.3.0-1.tar.gz	eadc9b65c3db248bdd e2a6c41814dfb1b972 39f24ad55e03d8526d d9ab4a8d16
AWS PCS エージェント 1.2.2-1	https://aws-pcs-repo-us-east-1.s3.us-east-1.amazonaws.com/aws-pcs-agent/aws-pcs-agent-v1.2.2-1.tar.gz	fd7b6ea5442db75d72 3fc4971781ce6ae511 baa21b87c4286fc1df 8127b282b8
AWS PCS エージェント 1.2.1-1	https://aws-pcs-repo-us-east-1.s3.us-east-1.amazonaws.com/aws-pcs-agent/aws-pcs-agent-v1.2.1-1.tar.gz	2b784643ca01ccca1b aa64fbfb34bb41efe8 bdca69470998b74ce3 962bc271d4
AWS PCS エージェント 1.2.0-1	https://aws-pcs-repo-us-east-1.s3.us-east-1.amazonaws.com/aws-pcs-agent/aws-pcs-agent-v1.2.0-1.tar.gz	470db8c4fc9e50277b 6317f98584b6b547e7 3523043e34f018eeca e767846805

Installer (インストーラ)	URL のダウンロード	SHA256 チェックサム
AWS PCS エージェント 1.1.1-1	<code>https://aws-pcs-repo- <i>us-east-1</i> .s3.<i>us-east-1</i> .amazonaws.com/aws-pcs-agent/aws-pcs-agent-v1.1.1-1.tar.gz</code>	<code>bef078bf60a6d8ecde2e6c49cd34d088703f02550279e3bf483d57a235334dc6</code>
AWS PCS エージェント 1.1.0-1	<code>https://aws-pcs-repo- <i>us-east-1</i> .s3.<i>us-east-1</i> .amazonaws.com/aws-pcs-agent/aws-pcs-agent-v1.1.0-1.tar.gz</code>	<code>594c32194c71bccc5d66e5213213ae38dd2c6d2f9a950bb01accea0bbab0873a</code>
AWS PCS エージェント 1.0.1-1	<code>https://aws-pcs-repo- <i>us-east-1</i> .s3.<i>us-east-1</i> .amazonaws.com/aws-pcs-agent/aws-pcs-agent-v1.0.1-1.tar.gz</code>	<code>04e22264019837e3f42d8346daf5886eaaced21571742eb505ea8911786bcb2</code>
AWS PCS エージェント 1.0.0-1	<code>https://aws-pcs-repo- <i>us-east-1</i> .s3.<i>us-east-1</i> .amazonaws.com/aws-pcs-agent/aws-pcs-agent-v1.0.0-1.tar.gz</code>	<code>d2d3d68d00c685435c38af471d7e2492dde5ce9eb222d7b6ef0042144b134ce0</code>

Slurm インストーラ

Installer (インストーラ)	URL のダウンロード	SHA256 チェックサム
スラム 25.05.4-1	<code>https://aws-pcs-repo- <i>us-east-1</i> .s3.<i>us-east-1</i> .amazonaws.com/aws-pcs-slurm/aws-pcs</code>	<code>3b0f93bce441d4f4f6935175f2c1e81cd961</code>

Installer (インストーラ)	URL のダウンロード	SHA256 チェックサム
	-slurm-25.05-installer-25.05.4-1.tar.gz	cb923adb416fa6689f5592047a7d
スラム 25.05.3-1	<a href="https://aws-pcs-repo-<i>us-east-1</i>.s3.<i>us-east-1</i>.amazonaws.com/aws-pcs-slurm/aws-pcs-slurm-25.05-installer-25.05.3-1.tar.gz">https://aws-pcs-repo-<i>us-east-1</i>.s3.<i>us-east-1</i>.amazonaws.com/aws-pcs-slurm/aws-pcs-slurm-25.05-installer-25.05.3-1.tar.gz	851bb5815b6700ceb30cc4a3fda204ca8ce362c14528c339908983255a936cf0
スラム 24.11.6-2	<a href="https://aws-pcs-repo-<i>us-east-1</i>.s3.<i>us-east-1</i>.amazonaws.com/aws-pcs-slurm/aws-pcs-slurm-24.11-installer-24.11.6-2.tar.gz">https://aws-pcs-repo-<i>us-east-1</i>.s3.<i>us-east-1</i>.amazonaws.com/aws-pcs-slurm/aws-pcs-slurm-24.11-installer-24.11.6-2.tar.gz	f17cd78e0bc6b9c818b794d9d2685cceabdc73f4fbb12f7566ae5b86a5abc32b
スラム 24.11.6-1	<a href="https://aws-pcs-repo-<i>us-east-1</i>.s3.<i>us-east-1</i>.amazonaws.com/aws-pcs-slurm/aws-pcs-slurm-24.11-installer-24.11.6-1.tar.gz">https://aws-pcs-repo-<i>us-east-1</i>.s3.<i>us-east-1</i>.amazonaws.com/aws-pcs-slurm/aws-pcs-slurm-24.11-installer-24.11.6-1.tar.gz	225de9fc18206f5f65f412effe1fd457614ac97ee9822b3ff804a452b0fae522
スラム 24.11.5-1	<a href="https://aws-pcs-repo-<i>us-east-1</i>.s3.<i>us-east-1</i>.amazonaws.com/aws-pcs-slurm/aws-pcs-slurm-24.11-installer-24.11.5-1.tar.gz">https://aws-pcs-repo-<i>us-east-1</i>.s3.<i>us-east-1</i>.amazonaws.com/aws-pcs-slurm/aws-pcs-slurm-24.11-installer-24.11.5-1.tar.gz	593efe4d66bef2f3e46d5a382fb5a32f7a3ca2510bcf1b3c85739f4f951810d5

Installer (インストーラ)	URL のダウンロード	SHA256 チェックサム
スラム 24.05.8-2	<code>https://aws-pcs-repo- <i>us-east-1</i> .s3.<i>us-east-1</i> .amazonaws.com/aws-pcs-slurm/aws-pcs-slurm-24.05-installer-24.05.8-2.tar.gz</code>	<code>c494b0b55c319a4c2f3faf668c759d46c32c4c7aa94ae97d94128328fe95364b</code>
スラム 24.05.8-1	<code>https://aws-pcs-repo- <i>us-east-1</i> .s3.<i>us-east-1</i> .amazonaws.com/aws-pcs-slurm/aws-pcs-slurm-24.05-installer-24.05.8-1.tar.gz</code>	<code>210a43b376af082bbad640b2032655885790c5dab0e6489cc327c7310a375849</code>
スラム 24.05.7-1	<code>https://aws-pcs-repo- <i>us-east-1</i> .s3.<i>us-east-1</i> .amazonaws.com/aws-pcs-slurm/aws-pcs-slurm-24.05-installer-24.05.7-1.tar.gz</code>	<code>0b5ed7c81195de2628c78f37c79e63fc4ae99132ca6b019b53a0d68792ee82c5</code>
スラム 24.05.5-2	<code>https://aws-pcs-repo- <i>us-east-1</i> .s3.<i>us-east-1</i> .amazonaws.com/aws-pcs-slurm/aws-pcs-slurm-24.05-installer-24.05.5-2.tar.gz</code>	<code>7cc8d8294f2fbff95fe0602cf9e21e02003b5d96c0730e0a18c6aa04c7a4967b</code>
Slurm 23.11.10-4 (非推奨)	<code>https://aws-pcs-repo- <i>us-east-1</i> .s3.<i>us-east-1</i> .amazonaws.com/aws-pcs-slurm/aws-pcs-slurm-23.11-installer-23.11.10-4.tar.gz</code>	<code>bb2d8c919c69dba38d14358f49c7f0427564c5dd4af85a1c9eca2c57ceeae29a</code>

Installer (インストーラ)	URL のダウンロード	SHA256 チェックサム
Slurm 23.11.10-3 (非推奨)	<code>https://aws-pcs-repo- <i>us-east-1</i> .s3.<i>us-east-1</i> .amazonaws.com/aws-pcs-slurm/aws-pcs-slurm-23.11-installer-23.11.10-3.tar.gz</code>	<code>488a10ee0fbd57ec0e0ff7ea708a9e3038fafdc025c6bb391c75c2e2a7852a00</code>
Slurm 23.11.10-2 (非推奨)	<code>https://aws-pcs-repo- <i>us-east-1</i> .s3.<i>us-east-1</i> .amazonaws.com/aws-pcs-slurm/aws-pcs-slurm-23.11-installer-23.11.10-2.tar.gz</code>	<code>0bbe85423305c05987931168caf98da08a34c25f9eec0690e8e74de0b7bc8752</code>
Slurm 23.11.10-1 (廃止)	<code>https://aws-pcs-repo- <i>us-east-1</i> .s3.<i>us-east-1</i> .amazonaws.com/aws-pcs-slurm/aws-pcs-slurm-23.11-installer-23.11.10-1.tar.gz</code>	<code>27e8faa9980e92cdfd8cfdc71f937777f0934552ce61e33dac4ecf5a20321e44</code>
Slurm 23.11.9-1 (廃止)	<code>https://aws-pcs-repo- <i>us-east-1</i> .s3.<i>us-east-1</i> .amazonaws.com/aws-pcs-slurm/aws-pcs-slurm-23.11-installer-23.11.9-1.tar.gz</code>	<code>1de7d919c8632fe8e2806611bed4fde1005a4fadc795412456e935c7bba2a9b8</code>

PCS サンプル AWS AMIsリリースノート

サポートされている最新のメジャーバージョンのスケジューラの AMIs は、セキュリティアップデートと重要なバグ修正を受け取ります。これらの増分セキュリティパッチは、公式リリースノートには含まれていません。

Important

古いスケジューラバージョンに関連するサンプル AMIs はサポートされておらず、更新を受け取りません。

Important

サンプル AMIs はデモンストレーション用であり、本番稼働用ワークロードには推奨されません。

目次

- [AWS x86_64 AMIs \(Amazon Linux 2\)](#)
- [AWS Arm64 の PCS サンプル AMIs \(Amazon Linux 2\)](#)

AWS x86_64 AMIs (Amazon Linux 2)

Slurm 25.05

AMI 名

- `aws-pcs-sample_ami-amzn2-x86_64-slurm-25.05`

サポートされる EC2 インスタンス

- 64 ビット x86 プロセッサを搭載したすべてのインスタンス。互換性のあるインスタンスを検索するには、Amazon EC2 コンソールに移動します。インスタンスタイプを選択し、アーキテクチャ = x86_64 を検索します。

AMI のコンテンツ

- サポートされている AWS サービス: AWS PCS
- オペレーティングシステム: Amazon Linux 2
- コンピューティングアーキテクチャ: x86_64
- EBS ボリュームタイプ: gp2
- EFA インストーラ: 1.43.1
- GDRCopy: 2.5.1
- NVIDIA ドライバー: 550.127.08
- NVIDIA CUDA: 12.4.1_550.54.15

Slurm 24.11

Note

AWS PCS は Slurm 24.11 以降のアカウントティングをサポートしています。詳細については、「[PCS での Slurm AWS アカウントティング](#)」を参照してください。

AMI 名

- aws-pcs-sample_ami-amzn2-x86_64-slurm-24.11

サポートされる EC2 インスタンス

- 64 ビット x86 プロセッサを搭載したすべてのインスタンス。互換性のあるインスタンスを検索するには、[Amazon EC2 コンソール](#)に移動します。インスタンスタイプを選択し、 を検索しますArchitectures=x86_64。

AMI のコンテンツ

- サポートされている AWS サービス: AWS PCS
- オペレーティングシステム: Amazon Linux 2
- コンピューティングアーキテクチャ: x86_64
- EBS ボリュームタイプ: gp2

- EFA インストーラ: 1.33.0
- GDRCopy: 2.4
- NVIDIA ドライバー: 550.127.08
- NVIDIA CUDA: 12.4.1_550.54.15

Slurm 24.05

AMI 名

- aws-pcs-sample_ami-amzn2-x86_64-slurm-24.05

サポートされる EC2 インスタンス

- 64 ビット x86 プロセッサを搭載したすべてのインスタンス。互換性のあるインスタンスを検索するには、[Amazon EC2 コンソール](#)に移動します。インスタンスタイプを選択し、 を検索しますArchitectures=x86_64。

AMI のコンテンツ

- サポートされている AWS サービス: AWS PCS
- オペレーティングシステム: Amazon Linux 2
- コンピューティングアーキテクチャ: x86_64
- EBS ボリュームタイプ: gp2
- EFA インストーラ: 1.33.0
- GDRCopy: 2.4
- NVIDIA ドライバー: 550.127.08
- NVIDIA CUDA: 12.4.1_550.54.15

Slurm 23.11

AMI 名

- aws-pcs-sample_ami-amzn2-x86_64-slurm-23.11

サポートされる EC2 インスタンス

- 64 ビット x86 プロセッサを搭載したすべてのインスタンス。互換性のあるインスタンスを検索するには、[Amazon EC2 コンソール](#)に移動します。インスタンスタイプを選択し、 を検索しますArchitectures=x86_64。

AMI のコンテンツ

- サポートされている AWS サービス: AWS PCS
- オペレーティングシステム: Amazon Linux 2
- コンピューティングアーキテクチャ: x86_64
- EBS ボリュームタイプ: gp2
- EFA インストーラ: 1.33.0
- GDRCopy: 2.4
- NVIDIA ドライバー: 550.127.08
- NVIDIA CUDA: 12.4.1_550.54.15

AWS Arm64 の PCS サンプル AMIs (Amazon Linux 2)

Slurm 25.05

AMI 名

- aws-pcs-sample_ami-amzn2-arm64-slurm-25.05

サポートされる EC2 インスタンス

- 64 ビット Arm プロセッサを使用するすべてのインスタンス。互換性のあるインスタンスを検索するには、Amazon EC2 コンソールに移動します。インスタンスタイプを選択し、アーキテクチャ=arm64 を検索します。

AMI のコンテンツ

- サポートされている AWS サービス: AWS PCS
- オペレーティングシステム: Amazon Linux 2

- コンピューティングアーキテクチャ: arm64
- EBS ボリュームタイプ: gp2
- EFA インストーラ: 1.43.1
- GDRCopy: 2.5.1
- NVIDIA ドライバー: 550.127.08
- NVIDIA CUDA: 12.4.1_550.54.15

Slurm 24.11

 Note

AWS PCS は Slurm 24.11 以降のアカウントティングをサポートしています。詳細については、「[PCS での Slurm AWS アカウントティング](#)」を参照してください。

AMI 名

- aws-pcs-sample_ami-amzn2-arm64-slurm-24.11

サポートされる EC2 インスタンス

- 64 ビット Arm プロセッサを使用するすべてのインスタンス。互換性のあるインスタンスを検索するには、[Amazon EC2 コンソール](#)に移動します。インスタンスタイプを選択し、 を検索しますArchitectures=arm64。

AMI のコンテンツ

- サポートされている AWS サービス: AWS PCS
- オペレーティングシステム: Amazon Linux 2
- コンピューティングアーキテクチャ: arm64
- EBS ボリュームタイプ: gp2
- EFA インストーラ: 1.33.0
- GDRCopy: 2.4
- NVIDIA ドライバー: 550.127.08

- NVIDIA CUDA: 12.4.1_550.54.15

Slurm 24.05

AMI 名

- aws-pcs-sample_ami-amzn2-arm64-slurm-24.05

サポートされる EC2 インスタンス

- 64 ビット Arm プロセッサを使用するすべてのインスタンス。互換性のあるインスタンスを検索するには、[Amazon EC2 コンソール](#)に移動します。インスタンスタイプを選択し、 を検索しますArchitectures=arm64。

AMI のコンテンツ

- サポートされている AWS サービス: AWS PCS
- オペレーティングシステム: Amazon Linux 2
- コンピューティングアーキテクチャ: arm64
- EBS ボリュームタイプ: gp2
- EFA インストーラ: 1.33.0
- GDRCopy: 2.4
- NVIDIA ドライバー: 550.127.08
- NVIDIA CUDA: 12.4.1_550.54.15

Slurm 23.11

AMI 名

- aws-pcs-sample_ami-amzn2-arm64-slurm-23.11

サポートされる EC2 インスタンス

- 64 ビット Arm プロセッサを使用するすべてのインスタンス。互換性のあるインスタンスを検索するには、[Amazon EC2 コンソール](#)に移動します。インスタンスタイプを選択し、 を検索しますArchitectures=arm64。

AMI のコンテンツ

- サポートされている AWS サービス: AWS PCS
- オペレーティングシステム: Amazon Linux 2
- コンピューティングアーキテクチャ: arm64
- EBS ボリュームタイプ: gp2
- EFA インストーラ: 1.33.0
- GDRCopy: 2.4
- NVIDIA ドライバー: 550.127.08
- NVIDIA CUDA: 12.4.1_550.54.15

PCS AWS でサポートされているオペレーティングシステム

AWS PCS は、コンピューティングノードグループ用に設定された Amazon マシンイメージ (AMI) を使用して、そのコンピューティングノードグループで EC2 インスタンスを起動します。AMI は、EC2 インスタンスが使用するオペレーティングシステムを決定します。PCS サンプル AMIs AWS でオペレーティングシステムを変更することはできません。別のオペレーティングシステムを使用する場合は、カスタム AMI を作成する必要があります。詳細については、「[PCS 用の Amazon AWS マシンイメージ \(AMIs\)](#)」を参照してください。

サポートされるオペレーティングシステム

- Amazon Linux 2

これは PCS サンプル AWS AMIs。

Important

サンプル AMIs はデモンストレーション用であり、本番稼働用ワークロードには推奨されません。Amazon Linux 2 を使用する予定であっても、本番ワークロード用のカスタム AMI を作成して使用する必要があります。

- Amazon Linux 2023
- RedHat Enterprise Linux 9 (RHEL 9)

RHEL のすべて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ションシステムよりも高いです。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 8 (RHEL 8)
- Rocky Linux 9

[公式の Rocky Linux 9 AMIs](#) をカスタム AMI のベースとして使用できます。ベース AMI に最新のカーネルがない場合、カスタム AMI ビルドが失敗する可能性があります。

カーネルをアップグレードするには

1. 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。

2. インスタンスに SSH で接続し、コマンドを実行します。

```
sudo yum -y update
```

3. インスタンスからイメージを作成します。このイメージは、カスタム AMI ParentImage として指定します。

- Rocky Linux 8
- Ubuntu 22.04

Ubuntu 22.04 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。代わりに ED25519 キーを生成して使用することをお勧めします。

- Ubuntu 24.04

AWS PCS エージェントバージョン

AWS PCS エージェントソフトウェアは、PCS が起動する EC2 AWS インスタンスを Slurm で使用するように設定します。エージェントは、クラスターのコンピューティングノードグループを作成するときに指定する Amazon マシンイメージ (AMI) に含めます。これらのコンピューティングノードグループで起動される EC2 インスタンスは、指定された AMI とそれに含まれる AWS PCS エージェントソフトウェアを使用します。AWS PCS エージェントは、EC2 インスタンスが自身をクラスターの一部として登録できるようにします。最新の AWS PCS エージェントソフトウェアを使用するには、カスタム AMIs を更新する必要があります。詳細については、「[PCS のカスタム Amazon AWS マシンイメージ \(AMIs\)](#)」の [ステップ 2 – AWS PCS エージェントをインストールする](#) を参照してください。

AWS PCS エージェントバージョン	リリース日	リリースノート
v1.3.2-1	2026 年 3 月 10 日	• RHEL 8.10 または Rocky Linux 8.10 を実行しているコンピューティングノードが、それらのオペレーティングシステムの curl SigV4 バックポートの障害によりブートストラップに失敗する問題を修正しました。
v1.3.1-1	2025 年 11 月 7 日	• 使用可能な場合は「smt/control」sysfs パラメータを使用して、ハイパースレッディングの無効化を改善しました。 • PCS エージェントがハイパースレッディングを無効にしようとしたときに CPU が起動中にロックされた場合の潜在的な競合状態を修正しました。

AWS PCS エージェントバージョン	リリース日	リリースノート
		Slurm コンピューティングノードの フィールドInstanceId と InstanceType フィールドにそれぞれタイムスタンプとハイフンが入力される問題を修正しました。
v1.3.0-1	2025 年 11 月 3 日	新しいオペレーティングシステムのサポートが追加されました: Amazon Linux 2023、Ubuntu 24、RHEL 8、Rocky 8。
v1.2.2-1	2025 年 10 月 16 日	- IPv6 エンドポイントへのインスタンスメタデータクエリを許可します。IPv4 - カーネルが CPU ID 範囲として兄弟スレッドを返した場合、ハイパースレッディングが無効にならない問題を修正しました。 - ハイパースレッディングが正常に無効になったときに、ログに誤った失敗メッセージを生成する問題を修正しました。

AWS PCS エージェントバージョン	リリース日	リリースノート
v1.2.1-1	2025 年 6 月 19 日	• コントローラーが使用できない場合、AWS PCS エージェントは最大 30 分間スラムを開始しようとします。 • RegisterComputeNodeGroupInstance へのレスポンスに SLURMDBD エンドポイントが含まれている場合、誤ったスラム設定が発生する問題を修正しました。
v1.2.0-1	2025 年 3 月 7 日	• で IPv6 のサポートを有効にしましたslurmd.conf。
v1.1.1-1	2024 年 12 月 13 日	• RegisterComputeNodeGroupInstance への呼び出しで誤った Slurm バージョンが報告される問題を修正しました。 • のカスタムスクリプト/opt/aws/pcs/etc/bootstrap_hooks/ が実行された場合にインスタンスメタデータが正しく取得されない問題を修正しました。
v1.1.0-1	2024 年 12 月 6 日	• ブートストラップステップの前に /opt/aws/pcs/etc/bootstrap_hooks/ 実行されるカスタムスクリプトを有効にしました。

AWS PCS エージェントバージョン	リリース日	リリースノート
v1.0.1-1	2024 年 10 月 22 日	GPU 対応インスタンスで <code>slurmd</code> を起動したときに NVIDIA デバイスが動作しない問題を修正しました。
v1.0.0-1	2024 年 8 月 28 日	初回リリース。

PCS の Slurm AWS スケジューラ

Slurm は Linux クラスター用に設計されたオープンソースのワークロードマネージャーで、HPC ワークロードのジョブスケジューリング、リソース割り当て、ジョブモニタリング機能を提供します。AWS PCS は、クラスターワークロードを管理する Slurm スケジューラをサポートしています。

トピック

- [PCS の Slurm AWS バージョン](#)
- [PCS での Slurm AWS アカウンティング](#)
- [PCS の Slurm AWS REST API](#)
- [PCS での Slurm を使用したコンピューティングノード AWS の再起動](#)
- [PCS でのカスタム Slurm AWS 設定の設定](#)
- [SPANK プラグインを使用して AWS PCS で Slurm 機能を拡張する](#)
- [Slurm CLI フィルタープラグインを使用して PCS AWS でのジョブ送信をカスタマイズする](#)

PCS の Slurm AWS バージョン

SchedMD は、新機能、最適化、セキュリティパッチを使用して Slurm を継続的に強化します。SchedMD は定期的に新しいメジャーバージョンをリリースhttps://slurm.schedmd.com/upgrades.html#release_cycleし、いつでも最大 3 つのバージョンをサポートする予定です。AWS PCS は、Slurm コントローラーをパッチバージョンで自動的に更新するように設計されています。

SchedMD が特定のメジャーバージョンの[サポート](#)を終了すると、AWS PCS はそのバージョンをサポート終了 (EOL) として指定します。EOL の後、そのバージョンで新しいクラスターを作成することはできませんが、既存のクラスターは、サポートが保証されずに最大 12 か月間実行し続けることができます。AWS PCS は、Slurm メジャーバージョンが EOL に近い場合に事前通知を送信し、クラスターを新しいサポートされているバージョンにアップグレードするタイミングをお客様が把握できるようにします。

サポートされている最新の Slurm バージョンを使用してクラスターをデプロイし、最新の進歩と改善にアクセスすることをお勧めします。

PCS でサポートされている Slurm AWS バージョン

次の表は、サポートされている Slurm バージョンと、各バージョンの重要な日付と情報を示しています。

Slurm バージョン	SchedMD リリース日	AWS PCS リリース日	AWS PCS EOL の日付	互換性のある最小 AWS PCS エージェントバージョン	サポートされている AWS PCS サンプル AMIs
25.05	5/29/2025	10/16/2025	5/31/2027	1.0.0-1	- aws-pcs-sample_ami-amzn2-x86_64-slurm-25.05 - aws-pcs-sample_ami-amzn2-arm64-slurm-25.05
24.11	11/29/2024	5/14/2025	5/31/2026	1.0.0-1	- aws-pcs-sample_ami-amzn2-x86_64-slurm-24.11 - aws-pcs-sample_ami

Slurm バージョン	SchedMD リリース日	AWS PCS リリース日	AWS PCS EOL の日付	互換性のある最小 AWS PCS エージェントバージョン	サポートされている AWS PCS サンプル AMIs
					-amzn2-arm64-slurm-24.11

PCS でサポートされていない Slurm AWS バージョン

次の表は、PCS でサポートされていない Slurm AWS バージョンを示しています。

Slurm バージョン	SchedMD リリース日	AWS PCS リリース日	AWS PCS EOL の日付		
24.05	5/30/2024	12/18/2024	11/30/2025		
23.11	11/21/2023	8/28/2024	5/31/2025		

PCS での Slurm AWS バージョンのリリースノート

このトピックでは、AWS PCS で現在サポートされている Slurm バージョンごとの重要な変更について説明します。クラスターをアップグレードするときは、古いバージョンと新しいバージョンの変更を確認することをお勧めします。

Slurm 25.05

PCS AWS で実装される変更

- Slurm `requeue_on_resume_failure SchedulerParameter` がデフォルトで有効になりました。
- 「`stderr`」は、Slurm 25.05 で無効になっているため、`LogTimeFormat` のオプションとして削除されました。
- AWS PCS はマルチクラスターのサック設定をサポートしています。ログインノードは複数のクラスターにアクセスできます。

Slurm 25.05 の詳細については、以下の出版物を参照してください。

- SchedMD リリースのお知らせ: <https://www.schedmd.com/slurm-version-25-05-0-is-now-available/>
- SchedMD リリースノート: https://github.com/SchedMD/slurm/blob/slurm-25-05-0-1/RELEASE_NOTES.md

Slurm 24.11

PCS AWS で実装される変更

- AWS PCS は Slurm アカウンティングをサポートしています。詳細については、「[PCS での Slurm AWS アカウンティング](#)」を参照してください。

Slurm 24.11 の詳細については、以下の出版物を参照してください。

- [SchedMD リリースのお知らせ](#)
- [SchedMD リリースノート](#)

Slurm 24.05

PCS AWS で実装される変更

- 新しい Slurm Step Manager モジュールが PCS AWS でデフォルトで有効になりました。このモジュールでは、ステップ管理を中央コントローラーからコンピューティングノードにオフロードすることで、ステップ使用率が高い環境でのシステム同時実行を大幅に改善することで、大きなメリットが得られます。この設定をサポートし、分離PrologとEpilogプロセスの実行を改善するために、新しいプロログフラグ (Contain、Alloc) が有効になっています。
- コントローラーからコンピューティングノードへの階層通信を有効にすると、Slurm ノード内通信を最適化できるため、スケーラビリティとパフォーマンスが向上します。さらに、ルーティング設定では、プラグインのデフォルトのルーティングアルゴリズムではなく、コントローラーからの通信にパーティションノードリストを使用するようになり、システムの耐障害性が向上しました。
- 新しいハッシュプラグインは、前の `HashPlugin=hash/sha3` を置き換えます `hash/k12 plugin`。これは、PCS AWS クラスターでデフォルトで有効になりました。
- Slurm コントローラーログに、へのすべてのインバウンドリモートプロシージャコール (RPC) の拡張監査機能が含まれるようになりました `slurmctld`。ログには、接続処理前の送信元アドレス、認証されたユーザー、および RPC タイプが含まれます。

Slurm 24.05 の詳細については、以下の出版物を参照してください。

- [SchedMD リリースのお知らせ](#)
- [SchedMD リリースノート](#)

Slurm 23.11

PCS で変更できる Slurm AWS 設定

- のSuspendTimeデフォルトは 60です。PCS AWS `scaleDownIdleTimeInSeconds`設定パラメータを使用して設定します。詳細については、AWS PCS API リファレンスClusterSlurmConfigurationのデータ型の [scaleDownIdleTimeInSeconds](#)パラメータを参照してください。
- MaxJobCount および MaxArraySizeは、クラスター用に選択したサイズに基づいています。詳細については、PCS CreateCluster API AWS リファレンスの API アクションの [size](#)パラメータを参照してください。
- Slurm SelectTypeParameters 設定のデフォルトは CR_CPUです。クラスターの作成時に設定slurmCustomSettingsするための値として指定できます。詳細については、「PCS CreateCluster API AWS リファレンス」の「API アクションの [slurmCustomSettings](#)パラメータ」および[SlurmCustomSetting](#)」を参照してください。
- クラスターレベルで Prologと Epilog を設定できます。クラスターの作成時に設定slurmCustomSettingsするための値として指定できます。詳細については、AWS 「PCS API リファレンス」の[CreateCluster](#)「」と[SlurmCustomSetting](#)」を参照してください。
- コンピューティングノードグループレベルで Weightと RealMemory を設定できます。コンピューティングノードグループの作成時に設定slurmCustomSettingsするための値として指定できます。詳細については、AWS 「PCS API リファレンス」の[CreateComputeNodeGroup](#)「」と[SlurmCustomSetting](#)」を参照してください。

PCS の Slurm AWS バージョンに関するよくある質問

AWS PCS は、複数の Slurm バージョンのサポートを維持します。新しい Slurm バージョンが導入されると、AWS PCS は、そのバージョンが SchedMD からサポート終了 (EOS) に達するまでテクニカルサポートとセキュリティパッチを提供します。AWS PCS は AWS 、用語と一致するように Slurm バージョンの EOS 日をサポート終了 (EOL) と呼びます。

PCS AWS は Slurm バージョンをどのくらいの期間サポートしていますか？

AWS Slurm バージョンの PCS サポートは、SchedMD のメジャーバージョンのサポートサイクルと一致しています。AWS PCS は、現在のバージョンと最新の 2 つのメジャーバージョンをサポートしています。SchedMD が新しいメジャーバージョンをリリースすると、AWS PCS はサポートされている最も古いバージョンのサポートを終了します。AWS PCS はできるだけ早く Slurm の新しいメジャーバージョンをリリースしますが、SchedMD のリリースと PCS AWS での可用性の間に遅延が生じる可能性があります。

クラスターで新しい Slurm パッチバージョンのリリースを取得するにはどうすればよいですか？

バグとセキュリティ修正に対処するために、AWS PCS は内部サービス所有アカウントで実行されるクラスターコントローラーにパッチを自動的に適用するように設計されています。の EC2 インスタンスにパッチをインストールするには AWS アカウント、コンピューティングノードグループの Amazon マシンイメージ (AMI) を更新し、更新された AMI を使用するようにコンピューティングノードグループを更新します。詳細については、「[PCS のカスタム Amazon AWS マシンイメージ \(AMIs\)](#)」を参照してください。

 Note

Slurm コントローラーは、更新中は利用できません。実行中のジョブは影響を受けません。クラスターのコントローラーが使用できなくなる前に送信されたジョブは、コントローラーが使用可能になるまで保持されます。

今後の Slurm バージョン EOL イベントについてどのように通知されますか？

EOL 日の 6 か月前に E メールメッセージを送信します。EOL の 1 週間前に最終 E メールメッセージとともに、EOL の 1 か月前に E メールメッセージを送信します。EOL 日以降、EOL Slurm バージョンで AWS PCS クラスターを実行しているお客様に 12 か月間毎月 E メールメッセージを送信します。そのバージョンでセキュリティの脆弱性が特定された場合、EOL Slurm バージョンを持つクラスターを停止することがあります。

クラスターが使用している Slurm バージョンが EOL Slurm バージョンを実行しているかどうかを確認するにはどうすればよいですか？

EOL Slurm バージョンで実行中のクラスターがあることを通知する E メールメッセージを送信します。EOL Slurm バージョンを持つクラスターの詳細を含む AWS Health Dashboard アラートにアラートを投稿します。AWS PCS コンソールを使用して、EOL Slurm バージョンのクラスターを特定することもできます。

Slurm バージョンが EOL に近い場合、または EOL を超えている場合はどうすればよいですか？

サポートされている新しいバージョンの Slurm を使用して新しいクラスターを作成し、コンピューティングノードグループ AMIs で Slurm バージョンを更新します。AMIs および実行中の EC2 インスタンスの Slurm バージョンは、クラスターの Slurm バージョンより 2 バージョンを超えることはできません。詳細については、「[PCS のカスタム Amazon AWS マシンイメージ \(AMIs\)](#)」を参照してください。

EOL 日までに Slurm の新しいバージョンに切り替えられない場合どうなりますか？

EOL Slurm バージョンで新しいクラスターを作成することはできません。既存のクラスターは、AWS サポートなしで最大 12 か月間運用でき、運用を維持するために即時のアクションは必要ありません。EOL 日以降、サポート、セキュリティ更新、可用性は保証されません。セキュリティ上の理由から、クラスターを停止することがあります。PCS クラスターのセキュリティとサポートを維持するために、サポートされている Slurm AWS バージョンを使用することを強くお勧めします。

EOL Slurm バージョンでクラスターを操作する場合のリスクは何ですか？

EOL Slurm バージョンを使用するクラスターには、重大なセキュリティと運用上のリスクがあります。SchedMD のアクティブなモニタリングがない場合、セキュリティの脆弱性は検出されず、対処されないままになる可能性があります。重大な脆弱性が検出された場合、クラスターが直ちに停止される可能性があります。

クラスターが中断されると、ジョブ、クラスターコンピューティング、ストレージ、ネットワークリソースはどうなりますか？

PCS AWS によって管理されるすべてのリソースは終了します。これには、Slurm コントローラー、コンピューティングノードグループ、EC2 インスタンスが含まれます。コンピューティングインスタンスで実行されているジョブは直ちに終了し、クラスターは中断状態になります。外部ファイルシステムなどのカスタマーマネージドリソースはそのまま残ります。PCS コンソールと API AWS アクションを使用して、クラスターの設定にアクセスできます。

中断されたクラスターを再起動して残りのジョブを再開できますか？

いいえ。中断されたクラスターを再起動することはできません。中断されたクラスターの設定を使用して、サポートされている Slurm バージョンで新しいクラスターを作成できます。残りのジョブを外部ファイルシステムに保存した場合は、実行できます。

12 か月の猶予期間を超える延長をリクエストできますか？

いいえ。12 か月の猶予期間を超えてクラスターを実行する拡張機能をリクエストすることはできません。サポートされている Slurm バージョンへの切り替えに役立つ延長時間を提供します。クラス

ターオペレーションの中断を避けるため、Slurm バージョンが EOL に達する前に切り替えることをお勧めします。

PCS での Slurm AWS アカウンティング

新しい AWS PCS クラスターでアカウンティングを有効にして、クラスターの使用状況のモニタリング、リソース制限の適用、特定のキューまたはコンピューティングノードグループへのきめ細かなアクセスコントロールの管理を行うことができます。AWS PCS はクラスターのアカウンティングデータベースを作成および管理するため、独自のアカウンティングデータベースを作成および管理する必要はありません。AWS PCS は Slurm のアカウンティング機能を使用します。Slurm のアカウンティング機能の詳細については、[Slurm ドキュメントの SchedMD を参照してください](#)。

アカウンティングを使用するには、新しいクラスターを作成し、オプションでアカウンティングパラメータを設定するときには有効にします。クラスターのステータスが Active でコンピューティングノードグループがある場合は、ログインノードの Linux シェルに接続して、Slurm sacct コマンドでジョブデータを表示するなどのアカウンティング機能を実行できます。

Note

アカウンティングは Slurm 24.11 以降でサポートされています。

AWS PCS console

クラスターの作成ページで、有効なバージョンの Slurm (バージョン 24.11 以降) を選択する必要があります。スケジューラ設定で、アカウンティングを有効にします。

AWS PCS API

CreateCluster API アクションの呼び出しでaccounting設定を指定します。accounting オブジェクトで、modeを に設定しますSTANDARD。詳細については、AWS PCS API リファレンスの[CreateCluster](#) and [Accounting](#)」を参照してください。

次の例では AWS CLI、 を使用して CreateCluster API アクションを呼び出します。パラメータ値の部分文字列は、アカウンティングaccounting='{mode=STANDARD}' を有効にします。

```
aws pcs create-cluster --cluster-name cluster-name \  
                      --scheduler type=SLURM,version=24.11 \  
                      --size SMALL \  
                      --networking subnetIds=cluster-subnet-  
id,securityGroupIds=cluster-security-group-id \  
                      --accounting mode=STANDARD
```

```
--slurm-configuration  
scaleDownIdleTimeInSeconds=180,accounting='{mode=STANDARD}',slurmCustomSettings='[{parameter
```

Important

アカウントリングを有効にすると、追加料金が発生します。詳細については、[AWS「PCSの料金」ページ](#)を参照してください。

アカウントリング設定の変更

インフラストラクチャを再構築することなく、既存のクラスターのアカウントリングを有効または無効にできます。詳細については、「[PCS AWS でのクラスターの更新](#)」を参照してください。

アカウントリングを無効にすると、クラスターが UPDATING 状態になるとすぐにアカウントリング機能の請求が停止します。アカウントリングを有効にすると、クラスターが正常に ACTIVE 状態に戻ったときに請求が開始されます。

PCS での Slurm AWS アカウントリングの主な概念

以下の概念は PCS AWS に固有のものであり、PCS が Slurm AWS アカウントリングを実装する方法を制御します。

アカウントリングデータベース

AWS PCS は、AWS アカウント AWS が所有する で作成されたデータベースにアカウントリングデータを保存します。にアクセスできません `slurmdbd.conf`。

デフォルトのパーシステンス時間

この AWS PCS 設定は、すべてのアカウントリングレコードタイプ (ジョブ、イベント、予約、ステップ、停止、トランザクション、使用状況データ) の保持期間 (日数) を指定します。たとえば、値が 30 の場合、AWS PCS はアカウントリングレコードを 30 日間保持します。この値は、クラスターの作成時に指定します。値を指定しない場合、AWS PCS はデータベースにアカウントリングレコードを無期限に保持します。

AWS PCS console

クラスターを作成するステップの一部として、デフォルトのパーシステンス時間を指定します。クラスターの作成ページで、有効なバージョンの Slurm (バージョン 24.11 以降) を選択し、アカウン

ティングを有効にする必要があります。スケジューラ設定で、デフォルトのパージ時間 (日数) の整数値を指定します。

AWS PCS API

CreateCluster API アクションの呼び出しで指定するaccounting情報defaultPurgeTimeInDaysの一部としてを指定します。詳細については、AWS PCS API リファレンスの[CreateCluster](#) and [Accounting](#)」を参照してください。

Note

AWS PCS API を使用してクラスターを作成する場合、のデフォルト値defaultPurgeTimeInDaysは -1であり、有効な値ではありません。

アカウンティングポリシーの適用

この設定により、Slurm がクラスターのジョブ送信ルール、リソース制限、およびアカウンティングポリシーを厳密に適用する方法が決まります。この設定は、クラスターの slurm.confファイルのAccountingStorageEnforceパラメータに対応します。強制オプションの任意の組み合わせを選択できます。オプションを選択しない場合、クラスターのジョブに適用されるアカウンティング制約はありません。AWS PCS は次のオプションをサポートしています。

- 関連付け — job-to-accountマッピング
- 制限 — リソースの制約
- QoS — サービス品質要件
- セーフモード — 制限内の完了を保証
- nosteps — ステップアカウンティングを無効にする
- nojobs — ジョブアカウンティングを無効にする

これらのオプションの詳細については、[SchedMD の Slurm ドキュメント](#)を参照してください。

AWS PCS console

オプションは、クラスターを作成する手順の一部として設定します。クラスターの作成ページで、有効なバージョンの Slurm (バージョン 24.11 以降) を選択し、アカウンティングを有効にする必要があります。スケジューラ設定のアカウンティングポリシーの適用ドロップダウンリストから、必要なオプションを選択します。

AWS PCS API

Slurm では、これらのオプションはクラスターの `slurm.conf` ファイルで設定されます。PCS クラスター `slurm.conf` の AWS に直接アクセスすることはできません。代わりに、クラスターを作成するときに `CreateCluster` API アクション `SlurmCustomSettings` に を指定します。詳細については、AWS PCS API リファレンスの [CreateCluster](#) を参照してください。

既存の PCS AWS クラスターのアカウントिंग設定を取得する

Slurm アカウントिंग設定は、クラスターの Slurm 設定に含まれています。

AWS PCS console

1. ナビゲーションペインからクラスターを選択します。
2. リストからクラスター名を選択します。
3. 設定タブで、Slurm 設定でアカウントिंग設定を見つけます。

AWS PCS API

`GetCluster` API アクションを使用して、クラスター設定を取得します。アカウントिंग設定は、 にあります `slurmConfiguration.mode` と の値の設定 `defaultPurgeTimeInDays` は の下にあります `accounting`。選択したアカウントिंगポリシーの適用オプションは の下にあります `slurmCustomSettings`。詳細については、AWS 「PCS API リファレンス」の [GetCluster](#) を参照してください。

PCS の Slurm AWS REST API

AWS PCS は、 を通じて Slurm のネイティブ REST API のマネージドサポートを提供し `slurmrestd`、プログラムによるクラスター操作のための HTTP インターフェイスを提供します。クラスターへの直接シェルアクセスを必要とせずに、標準 HTTP リクエストを通じてジョブの送信、クラスターステータスのモニタリング、リソースの管理を行うことができます。

一般的なユースケース

Slurm REST API は、さまざまな統合シナリオをサポートしています。

- ウェブアプリケーション統合: ジョブを直接送信および管理するためのカスタムフロントエンドとウェブアプリケーションを構築します。

- Jupyter Notebook Integration: 開発ワークフローを離れることなく、ノートブック環境からジョブを送信できます。
- パートナーソリューションの統合: サードパーティーの HPC ツールとワークフローマネージャーを PCS AWS クラスターに接続します。
- プログラムによるクラスター管理: ジョブの送信、モニタリング、リソース管理のワークフローを自動化します。
- Research Computing Workflows: API 主導のジョブ管理を必要とする学術的および企業の研究環境をサポートします。

要件と制限

Slurm REST API を使用する前に、以下の詳細を確認してください。

- クラスターは Slurm バージョン 25.05 以降を使用する必要があります。
- API エンドポイントには、クラスターの VPC 内のプライベート IP アドレスを介してのみアクセスできます。
- クラスターセキュリティグループは、ポート 6820 で HTTP トラフィックを許可する必要があります。
- 認証には、特定のユーザー ID クレームを持つ JWT トークンが必要です。

現在の制限は次のとおりです。

- によって生成されたトークン `scontrol token` はサポートされていません。
- `X-SLURM-USER-NAME` ヘッダー偽装は使用できません。
- 一部の機能では、Slurm アカウンティングを有効にする必要があります。
- Slurm CLI フィルタープラグインメカニズムと互換性がありません。
- REST API エンドポイントへの接続は TLS で暗号化されません。

トピック

- [PCS での Slurm REST API AWS の有効化](#)
- [AWS PCS での Slurm REST API を使用した認証](#)
- [PCS でのジョブ管理に Slurm REST API AWS を使用する](#)
- [PCS での Slurm REST API AWS のよくある質問](#)

PCS での Slurm REST API AWS の有効化

Slurm REST API を有効にして、プログラムによるジョブ管理とモニタリングのためにクラスターの HTTP インターフェイスにアクセスします。この機能は、クラスターの作成時に有効にすることも、要件を満たす既存のクラスターを更新することもできます。

前提条件

Slurm REST API を有効にする前に、以下を確認してください。

- クラスターバージョン: Slurm バージョン 25.05 以降。
- セキュリティグループ: 目的のソースからのポート 6820 での HTTP トラフィックを許可するルール。

手順

新しいクラスターで Slurm REST API を有効にするには

AWS マネジメントコンソール

1. <https://console.aws.amazon.com/pcs/> で AWS PCS コンソールを開きます。
2. [クラスターを作成] を選択します。
3. クラスターの詳細で、Slurm バージョン 25.05 以降を選択します。
4. 必要に応じて、他のクラスター設定を構成します。
5. スケジューラ設定セクションで、REST API を Enabled に設定します。
6. 目的のソースからのポート 6820 での HTTP トラフィックを許可するようにクラスターセキュリティグループを設定します。
7. クラスター作成プロセスを完了します。

AWS CLI

1. クラスターの作成時に Slurm REST 設定を追加します。

```
aws pcs create-cluster --region region \  
  --cluster-name my-cluster \  
  --scheduler type=SLURM, version=25.05 \  
  --size SMALL \  
  --networking subnetIds=subnet-ExampleId1,securityGroupIds=sg-ExampleId1 \  
  --slurm-rest-api-enabled
```

```
--slurm-configuration slurmRest='{mode=STANDARD}'
```

2. 目的のソースからのポート 6820 での HTTP トラフィックを許可するようにクラスターセキュリティグループを設定します。

既存のクラスターで Slurm REST API を有効にするには

AWS マネジメントコンソール

1. <https://console.aws.amazon.com/pcs/> で AWS PCS コンソールを開きます。
2. リストからクラスターを選択します。
3. クラスターがクラスターの詳細で Slurm バージョン 25.05 以降を使用していることを確認します。
4. クラスターの編集 を選択します。
5. スケジューラ設定セクションで、REST API を Enabled に設定します。
6. クラスターの更新を選択して変更を適用します。
7. 目的のソースからのポート 6820 での HTTP トラフィックを許可するようにクラスターセキュリティグループを設定します。

AWS CLI

1. この例のように、Slurm REST 設定でクラスターを更新します。

```
aws pcs update-cluster --cluster-identifier my-cluster \  
  --slurm-configuration 'slurmRest={mode=STANDARD}'
```

2. 目的のソースからのポート 6820 での HTTP トラフィックを許可するようにクラスターセキュリティグループを設定します。

を有効にするとうなるか

REST API を有効にすると、AWS PCS は自動的に次の操作を行います。

- JWT 署名キーを生成し、AWS Secrets Manager に保存します。
- VPC `https://<clusterPrivateIpAddress>:6820` 内の にある API エンドポイントを公開します。
- クラスター設定を更新して、REST API エンドポイントの詳細を表示します。

ジョブ管理とクラスターオペレーションに REST API を認証して使用できるようになりました。

AWS PCS での Slurm REST API を使用した認証

AWS PCS の Slurm REST API は、JSON ウェブトークン (JWT) 認証を使用して、クラスターリソースへの安全なアクセスを確保します。AWS PCS は AWS Secrets Manager に保存されているマネージド署名キーを提供します。これは、特定のユーザー ID クレームを含む JWT トークンを生成するために使用されます。

前提条件

Slurm REST API で認証する前に、以下を確認してください。

- クラスター設定: Slurm 25.05 以降と REST API が有効になっている AWS PCS クラスター。
- AWS アクセス許可: JWT 署名キーの AWS Secrets Manager へのアクセス。
- ユーザー情報: クラスターアカウントのユーザー名、POSIX ユーザー ID、および 1 つ以上の POSIX グループ IDs。
- ネットワークアクセス: ポート 6820 を許可するセキュリティグループとのクラスターの VPC 内の接続。

手順

Slurm REST API エンドポイントアドレスを取得するには

AWS マネジメントコンソール

1. <https://console.aws.amazon.com/pcs/> で AWS PCS コンソールを開きます。
2. リストからクラスターを選択します。
3. クラスター設定の詳細で、エンドポイントセクションを見つけます。
4. Slurm REST API (slurmrestd) のプライベート IP アドレスとポートを書き留めます。
5. API コールを行うには、適切にフォーマットされた HTTP リクエストをこのアドレスに送信します。

AWS CLI

1. でクラスターのステータスをクエリします `aws pcs get-cluster`。レスポンスの `endpoints` フィールドで `SLURMRESTD` エンドポイントを探します。以下がその例です。

```
"endpoints": [  
  {  
    "type": "SLURMCTLD",  
    "privateIpAddress": "192.0.2.1",  
    "port": "6817"  
  },  
  {  
    "type": "SLURMRESTD",  
    "privateIpAddress": "192.0.2.1",  
    "port": "6820"  
  }  
]
```

2. API コールを行うには、適切にフォーマットされた HTTP リクエストを に送信します。
`http://<privateIpAddress>:<port>/`

JWT 署名キーを取得するには

1. <https://console.aws.amazon.com/pcs/> で AWS PCS コンソールを開きます。
2. リストからクラスターを選択します。
3. クラスター設定の詳細で、スケジューラ認証セクションを見つけます。
4. JSON ウェブトークン (JWT) キーの ARN とバージョンを書き留めます。
5. を使用して Secrets Manager から署名キー AWS CLI を取得します。

```
aws secretsmanager get-secret-value --secret-  
id arn:aws:secretsmanager:region:account:secret:name --version-id version
```

JWT トークンを生成するには

1. 次の必須クレームを使用して JWT を作成します。
 - exp – JWT の 1970 年以降の有効期限の秒単位
 - iat – 1970 年以降の現在の秒単位の時刻
 - sub – 認証のユーザー名
 - uid – POSIX ユーザー ID
 - gid – POSIX グループ ID

- `id` – 追加の POSIX ID プロパティ
 - `gecos` – ユーザーコメントフィールド。多くの場合、人間が読める名前を保存するために使用されます。
 - `dir` – ユーザーのホームディレクトリ
 - `shell` – ユーザーのデフォルトシェル
 - `gids` – ユーザーが属する追加の POSIX グループ IDs のリスト
2. Secrets Manager から取得した署名キーを使用して JWT に署名します。
 3. トークンの適切な有効期限を設定します。

 Note

`sun` クレームの代わりに、次のいずれかを指定できます。

- `username`
- `userclaimfield` の を介して定義するカスタムフィールド名 `AuthAltParameters`
`Slurm custom settings`
- `id` クレーム内の `name` フィールド

API リクエストを認証するには

1. 次のいずれかの方法を使用して、JWT トークンを HTTP リクエストに含めます。
 - ベアラートークン – `Authorization: Bearer <jwt>` ヘッダーの追加
 - Slurm ヘッダー – `X-SLURM-USER-TOKEN: <jwt>` ヘッダーの追加
2. REST API エンドポイントに HTTP リクエストを行います。

`curl` と `Authorized: Bearer` ヘッダーを使用して `/ping` API にアクセスする例を次に示します。

```
curl -X GET -H "Authorization: Bearer <jwt>" \  
http://<privateIpAddress>:6820/slurm/v0.0.43/ping
```

JWT 生成の例

PCS AWS クラスター JWT 署名キーを取得し、ローカルファイルとして保存します。aws-region、secret-arn、シークレットバージョンの値を、クラスターに適した値に置き換えます。

```
#!/bin/bash
SECRET_KEY=$(aws secretsmanager get-secret-value \
  --region aws-region \
  --secret-id secret-arn \
  --version-stage secret-version \
  --query 'SecretString' \
  --output text)
echo "$SECRET_KEY" | base64 --decode > jwt.key
```

この Python の例は、署名キーを使用して JWT トークンを生成する方法を示しています。

```
#!/usr/bin/env python3

import sys
import os
import pprint
import json
import time
from datetime import datetime, timedelta, timezone
from jwt import JWT
from jwt.jwa import HS256
from jwt.jwk import jwk_from_dict
from jwt.utils import b64decode, b64encode
if len(sys.argv) != 3:
    sys.exit("Usage: gen_jwt.py [jwt_key_file] [expiration_time_seconds]")
SIGNING_KEY = sys.argv[1]
EXPIRATION_TIME = int(sys.argv[2])
with open(SIGNING_KEY, "rb") as f:
    priv_key = f.read()
signing_key = jwk_from_dict({
    'kty': 'oct',
    'k': b64encode(priv_key)
})
message = {
    "exp": int(time.time() + EXPIRATION_TIME),
    "iat": int(time.time()),
    "sub": "ec2-user",
    "uid": 1000,
```



```
"gid": 1000,
"id": {
    "gecos": "EC2 User",
    "dir": "/home/ec2-user",
    "gids": [1000],
    "shell": "/bin/bash"
}
}
a = JWT()
compact_jws = a.encode(message, signing_key, alg='HS256')
print(compact_jws)
```

スクリプトは JWT を画面に出力します。

```
abcdefghijklmnopjwttoken...
```

PCS でのジョブ管理に Slurm REST API AWS を使用する

Slurm REST API の概要

Slurm REST API は、HTTP リクエストを通じてクラスター管理関数へのプログラムによるアクセスを提供します。これらの主要な特性を理解することは、PCS で API AWS を効果的に使用するのに役立ちます。

- アクセスプロトコル: API は、クラスターのプライベートネットワーク内の通信に HTTP (HTTPS ではなく) を使用します。
- 接続の詳細: クラスターのプライベート IP アドレスと `slurmrestd` ポート (通常は 6820) を使用して API にアクセスします。完全なベース URL 形式は `http://<privateIpAddress>:6820`。
- API バージョニング: API バージョンは Slurm のインストールに対応します。Slurm 25.05 の場合は、バージョン v0.0.43 を使用します。バージョン番号は Slurm リリースごとに変わります。現在サポートされている API バージョンは、[Slurm リリースノート](#)に記載されています。
- URL 構造: Slurm REST API の URL 構造は `http://<privateIpAddress>:<port>/<api-version>/<endpoint>`。REST API エンドポイントの使用方法の詳細については、[Slurm ドキュメント](#)を参照してください。

前提条件

Slurm REST API を使用する前に、以下を確認してください。

- クラスター設定: Slurm 25.05 以降と REST API が有効になっている AWS PCS クラスター。
- 認証: 適切なユーザー ID クレームを持つ有効な JWT トークン。
- ネットワークアクセス: ポート 6820 を許可するセキュリティグループとのクラスターの VPC 内の接続。

手順

REST API を使用してジョブを送信するには

1. 必要なパラメータを使用してジョブ送信リクエストを作成します。

```
{
  "job": {
    "name": "my-job",
    "partition": "compute",
    "nodes": 1,
    "tasks": 1,
    "script": "#!/bin/bash\nnecho 'Hello from Slurm REST API'"
  }
}
```

2. HTTP POST リクエストを使用してジョブを送信します。

```
curl -X POST \
  -H "Authorization: Bearer <jwt>" \
  -H "Content-Type: application/json" \
  -d '<job-json>' \
  https://<privateIpAddress>:6820/slurm/v0.0.43/job/submit
```

3. モニタリングの目的でレスポンスで返されるジョブ ID を書き留めます。

ジョブのステータスをモニタリングするには

1. 特定のジョブに関する情報を取得します。

```
curl -X GET -H "Authorization: Bearer <jwt>" \
  https://<privateIpAddress>:6820/slurm/v0.0.43/job/<job-id>
```

2. 認証されたユーザーのすべてのジョブを一覧表示します。

```
curl -X GET -H "Authorization: Bearer <jwt>" \  
https://<privateIpAddress>:6820/slurm/v0.0.43/jobs
```

ジョブをキャンセルする方法

- DELETE リクエストを送信して、特定のジョブをキャンセルします。

```
curl -X DELETE -H "Authorization: Bearer <jwt>" \  
https://<privateIpAddress>:6820/slurm/v0.0.43/job/<job-id>
```

PCS での Slurm REST API AWS のよくある質問

このセクションでは、PCS の Slurm REST API に関するよくある質問 AWS に回答します。

Slurm REST API とは

Slurm REST API は、Slurm ワークロードマネージャーをプログラムで操作できる HTTP インターフェイスです。GET、POST、DELETE などの標準 HTTP メソッドを使用して、クラスターへのコマンドラインアクセスを必要とせずに、ジョブの送信、クラスターのステータスのモニタリング、リソースの管理を行うことができます。

によって生成されたトークンを使用できますか **scontrol token**?

いいえ。標準 **scontrol token** 出力は PCS AWS と互換性がありません。PCS Slurm REST API には、username(sun)、POSIX ユーザー ID()、グループ IDs(uid) を含む特定の ID クレームを含む強化 JWT トークンが必要です。標準 Slurm トークンにはこれらの必須クレームがないため、API によって拒否されます。

VPC の外部から API にアクセスできますか？

いいえ。REST API エンドポイントには、Slurm コントローラーのプライベート IP アドレスを使用して VPC 内からのみアクセスできます。外部アクセスを有効にするには、VPC Link、API Gateway を使用して Application Load Balancer などの AWS サービスを実装するか、安全な接続のために VPC ピアリングまたは VPN 接続を確立します。

API が HTTPS の代わりに HTTP を使用するのなぜですか？

Slurm REST API は、クラスターのプライベートネットワーク内の内部エンドポイントとなることを目的としています。暗号化を必要とする本番環境のデプロイでは、API ゲートウェイ、ロー

ドバランサー、リバースプロキシなどを通じて、アーキテクチャのより高いレベルで SSL/TLS 終了を実装できます。

REST API へのアクセスを制御するにはどうすればよいですか？

Slurm コントローラーのポート 6820 へのアクセスを制限するようにクラスターのセキュリティグループルールを設定します。インバウンドルールを設定して、信頼できる IP 範囲または VPC 内の特定のソースからの接続のみを許可し、API エンドポイントへの不正アクセスをブロックします。

JWT 署名キーをローテーションするにはどうすればよいですか？

アクティブなインスタンスなしでクラスターをメンテナンスモードにして、AWS Secrets Manager を介してキーローテーションを開始します。ローテーションが完了したら、キューを再度有効にします。既存の JWT トークンはすべて無効になり、Secrets Manager の新しい署名キーを使用して再生成する必要があります。

REST API を使用するには、Slurm アカウンティングを有効にする必要がありますか？

いいえ。ジョブの送信やモニタリングなどの基本的な REST API オペレーションに Slurm アカウンティングは必要ありません。ただし、/slurmdb エンドポイント全体では、アカウンティングをアクティブにする必要があります。

PCS AWS REST API では、どのようなサードパーティー製ツールを使用できますか？

Slurm Exporter for Prometheus、SlurmWeb、標準 Slurm REST API AWS 形式に従うカスタムアプリケーションなど、既存の Slurm REST API クライアントの多くは PCS と連携する必要があります。ただし、認証 `scontrol token` に依存するツールは、PCS JWT AWS の要件に合わせて変更する必要があります。

REST API の使用には追加料金がかかりますか？

いいえ。Slurm REST API 機能を有効化または使用しても追加料金はかかりません。通常どおり、基盤となるクラスターリソースに対してのみ料金が発生します。

REST API のトラブルシューティング方法を教えてください。

- ネットワーク接続の問題

API エンドポイントに到達できない場合、クラスターコントローラーに HTTP リクエストを行うと、接続タイムアウトまたは「接続拒否」エラーが表示されます。

対処方法: クライアントが同じ VPC にあるか、適切なネットワークルーティングがあることを確認し、セキュリティグループがソース IP またはサブネットからの HTTP トラフィックをポート 6820 で許可していることを確認します。

- Slurm REST 認証の問題

JWT トークンが無効、期限切れ、または不適切に署名されている場合、API リクエストはレスポンスのエラーフィールドに「プロトコル認証エラー」を返します。

エラーメッセージの例:

```
{
  "errors": [
    {
      "description": "Batch job submission failed",
      "error_number": 1007,
      "error": "Protocol authentication error",
      "source": "slurm_submit_batch_job()"
    }
  ]
}
```

対処方法: JWT トークンが適切にフォーマットされ、有効期限が切れておらず、Secrets Manager の正しいキーで署名されていることを確認します。トークンが適切に形成され、必要なクレームが含まれていること、および正しい認証ヘッダー形式を使用していることを確認します。

- 送信後にジョブの実行に失敗する

JWT トークンが有効で、誤った内部構造またはコンテンツが含まれている場合、ジョブは理由コードを使用して一時停止 (PD) 状態になった可能性があります。JobAdminHead.scontrol show job *<job-id>* を使用してジョブを検査する - JobState=PENDING, Reason=JobHeldAdmin、および *SystemComment=slurm_cred_create failure, holding job.*

対処方法: 根本原因は JWT の誤った値である可能性があります。PCS ドキュメントに従って、トークンが適切に構造化され、必要なクレームが含まれていることを確認します。

- 作業ディレクトリのアクセス許可の問題

JWT で指定されたユーザー ID にジョブの作業ディレクトリへの書き込みアクセス許可がない場合、アクセスできないディレクトリでを使用する場合と同様に、ジョブはアクセス許可エラー `sbatch --chdir` で失敗します。

対処方法: JWT トークンで指定されたユーザーに、ジョブの作業ディレクトリに対する適切なアクセス許可があることを確認します。

PCS での Slurm を使用したコンピューティングノード AWS の再起動

AWS PCS は Slurm のネイティブ `scontrol reboot` コマンドをサポートします。EC2 インスタンスを置き換えずにコンピューティングノードを再起動するには、このコマンドを使用します。その他の再起動方法 (Amazon EC2 コンソール、AWS CLI、自動パッチ、またはシステムメンテナンス) では、PCS AWS は EC2 インスタンスを異常と見なして置き換えます。

Slurm の再起動の利点

Slurm を再起動すると、クラスターのメンテナンスにいくつかの利点があります。

- キャパシティーを維持する – キャパシティーに制約のある EC2 インスタンスを他のお客様に失わないようにします。
- コストの削減 – 不要なインスタンス交換サイクルとアイドルノードの継続的な請求を排除します。
- 復旧の迅速化 – インスタンスの置き換えと比較してプロビジョニングの遅延はありません。
- 運用上の柔軟性 – メモリリークをクリアし、一時ファイルを削除し、ノードを劣化状態から復旧します。

Slurm の再起動を使用するタイミング

一般的な運用メンテナンスシナリオでは、Slurm の再起動を使用します。

- トラブルシューティング – 特に GPU ノードのパフォーマンスの問題や応答しないプロセスを解決します。
- リソースのクリーンアップ – メモリリーク、の一時ファイル/tmp、またはジョブのパフォーマンスに影響するプロセスのスタックをクリアします。
- 復旧 – 完全なノード交換が必要になる前に、ハング状態またはデグレード状態からノードを復旧します。

制限事項

- 再起動コマンドを実行できるのは、Slurm 管理者ユーザー (ルートユーザー) のみです。
- 再起動のサポートは `scontrol reboot` のみに制限されています。

- RebootProgram 設定はサポートされていません。
- コンソールインターフェイスなし – コマンドラインのみ。

トピック

- [PCS で Slurm を使用してコンピューティングノードを再起動 AWS する](#)
- [PCS で保留中の再起動をキャンセル AWS する](#)
- [PCS での Slurm AWS の再起動に関するよくある質問](#)
- [PCS での Slurm AWS の再起動に関する問題のトラブルシューティング](#)

PCS で Slurm を使用してコンピューティングノードを再起動 AWS する

Slurm のネイティブ再起動コマンドを使用して、EC2 インスタンス容量を失うことなく、パフォーマンスの問題を解決したり、リソースの問題をクリアしたり、パフォーマンスが低下している状態から回復したりできます。

前提条件

- Slurm 管理者権限 (ルートユーザーアクセス)
- PCS AWS クラスター内のログインノードへのアクセス

手順

1. EC2 コンソールを使用してログインノードに接続します。
 - a. EC2 コンソールで、[Instances] を選択します。
 - b. ログインノードインスタンスを選択します。
 - c. 接続 を選択します。
2. `sinfo` または `scontrol` を使用して、ターゲットコンピューティングノード名を特定します。`scontrol show node`。

```
sinfo
# or
scontrol show node
```

3. 次のいずれかのオプションを使用して、再起動コマンドを実行します。

⚠ Warning

`scontrol reboot` コマンド `nextstate=DOWN` で 使用しないでください。このパラメータはノードを異常としてマークし、インスタンスの置き換えをトリガーします。

- 基本的な再起動 (ノードがアイドル状態になるまで待機):

```
scontrol reboot nodename
```

- 即時再起動 (ジョブが完了するとノードがドレインされ再起動されます)。

```
scontrol reboot ASAP nodename
```

- 理由を指定して再起動します。

```
scontrol reboot ASAP reason="troubleshooting" nodename
```

- 再開状態で再起動します。

```
scontrol reboot ASAP nextstate=RESUME nodename
```

4. を使用して再起動の進行状況をモニタリングします `scontrol show node`。

```
scontrol show node nodename
```

5. 再起動の完了後にノードがサービスに戻ることを確認します。

PCS で保留中の再起動をキャンセル AWS する

保留中の再起動をキャンセルして、問題が解決された場合や再起動が不要になった場合の不要なダウンタイムを回避します。

前提条件

- Slurm 管理者権限
- ノードの再起動は保留中である必要があります (「再起動発行済み」ステータスを表示)
- コマンド実行用のログインノードへのアクセス

手順

1. ログインノードに接続します。
2. を使用して、ノードの再起動が保留中であることを確認します `scontrol show node`。

```
scontrol show node nodename
```

ノードステータスで「reboot issued」を探します。

3. `cancel` コマンドを実行します。

```
scontrol cancel_reboot nodename
```

4. 再起動のキャンセルとノードのステータスが正常に戻ることを確認します。

```
scontrol show node nodename
```

PCS での Slurm AWS の再起動に関するよくある質問

AWS PCS での Slurm 再起動の使用に関する一般的な質問に対する回答を示します。

Slurm 再起動サポートとは

ネイティブ Slurm `scontrol reboot` コマンドのサポート。このコマンドを使用して、自動インスタンス交換なしでコンピューティングノードを再起動します。これにより、EC2 インスタンスの容量が保持され、運用コストが削減されます。

Slurm 再起動コマンドを使用できるユーザー

再起動コマンドを実行できるのは、Slurm 管理者ユーザー (ルートユーザー) のみです。通常のユーザーは `scontrol reboot`、ノードに影響を与えずに Slurm からアクセス許可拒否エラーを受け取ります。

再起動中に実行中のジョブはどうなりますか？

デフォルトでは、再起動が発生する前にジョブは正常に完了します。ASAP オプションを使用すると、新しいジョブを防ぐためにノードがドレインされ、現在のジョブが終了した後に再起動されます。ジョブは、即時再起動のためにキャンセルまたは再キューに入れることができます。

これは EC2 コンソールの再起動とどのように異なりますか？

Slurm の再起動は EC2 インスタンスを保持し、置き換えを回避します。一方、EC2 コンソールの再起動は、再起動プロセス中にヘルスチェックが失敗したためにインスタンスを置き換えるためのトリガー PCS をトリガーします。

カスタム再起動スクリプトを設定できますか？

いいえ。RebootProgram 設定は、最初のリリースではサポートされていません。この機能は、カスタムスクリプトをサポートせずに標準の Slurm 再起動動作を使用します。

Slurm の再起動にはどのくらいの時間がかかりますか？

再起動時間は、インスタンスタイプ、顧客の起動プロセス、AMI 設定、ジョブを最初に完了する必要があるかどうかによって異なります。このプロセスには、ジョブの完了、物理的な再起動、ヘルスチェック、スラムデーモン登録の待機が含まれます。

再起動の履歴は確認できますか？

再起動イベントは Slurm ログ (slurmctld と slurmd) に記録され、CloudWatch を介してモニタリングできます。ノードステータスの reason フィールドには、プロセス中の再起動の理由が表示されます。

再起動中にノードが停止した場合はどうなりますか？

ノードが ResumeTimeout 内で再起動プロセスを完了しない場合、DOWN とマークされます。CloudWatch ログにエラーがないか確認し、ネットワーク接続を検証し、スラムログを調べます。問題が解決しない場合は、AWS サポートにお問い合わせください。

複数のノードを一度に再起動できますか？

はい。再起動コマンドで複数のノードを指定できます。

```
scontrol reboot ASAP node1,node2,node3
```

ジョブの完了を待たずにノードを再起動するにはどうすればよいですか？

マルチノードジョブに影響する問題のあるノード、パフォーマンスの大幅な低下、GPU の動作の不安定化などの問題に直面した場合に、ノードをすぐに再起動するには、次の 2 つのオプションがあります。

- キャンセルと再起動 – まず、を使用して影響を受けるジョブをキャンセルしてから `scancel <job_id>`、を使用して即時再起動を開始します `scontrol reboot ASAP <nodename>`。実行中のジョブは終了し、ノードの復旧後に再送信する必要があります。

- ドレインと再キュー (影響なし) – まずドレインを開始し、で再起動してから `scontrol reboot ASAP <nodename>`、を使用して影響を受けるジョブを再キューに入れ、`scontrol requeue <job_id>`。これにより、ジョブはキャンセルされるのではなく、保留状態に戻ります。

nextstate=DOWN を指定するとどうなりますか？

を指定するとnextstate=DOWN、再起動後にノードが異常としてマークされ、インスタンスの置き換えがトリガーされます。インスタンスの置き換えを回避するには、nextstate を指定したり、を使用しないでくださいnextstate=RESUME。

その他のリソース

- 基本的な再起動手順については、「」を参照してください[PCS で Slurm を使用してコンピューティングノードを再起動 AWS する](#)。
- 再起動の問題のトラブルシューティングについては、「」を参照してください[PCS での Slurm AWS の再起動に関する問題のトラブルシューティング](#)。
- Slurm の再起動に関するドキュメントについては、「[Slurm scontrol documentation](#)」を参照してください。

PCS での Slurm AWS の再起動に関する問題のトラブルシューティング

ノードの再起動の問題が発生した場合は、まず を使用してノードのステータスを確認します `scontrol show node nodename`。次に、Slurm (slurmd と slurmd) とシステムログの両方の CloudWatch ログを調べて、潜在的なエラーを特定します。

基本的なトラブルシューティングを行うには、ネットワーク接続を確認し、セキュリティグループの設定を確認し、再起動後に必要なサービスがすべて実行されていることを確認します。基本的なトラブルシューティング手順後も問題が解決しない場合は、AWS サポートにお問い合わせください。サポートに連絡するときは、関連するログの抜粋、ノードのステータス情報、再起動試行のタイムラインを提供して、解決プロセスを迅速化します。

その他のリソース

- CloudWatch を使用した AWS PCS インスタンスのモニタリングについては、[Amazon CloudWatch AWS を使用した PCS インスタンスのモニタリング](#)」を参照してください。
- 一般的なトラブルシューティングヘルプについては、「[AWS Parallel Computing Service の問題のトラブルシューティング](#)」を参照してください。

- Slurm のドキュメントについては、[「Slurm トラブルシューティングガイド」](#)を参照してください。

PCS でのカスタム Slurm AWS 設定の設定

カスタム Slurm 設定を使用して、クラスター、キュー、コンピューティングノードグループのリソース全体で追加の Slurm パラメータを設定します。このリリースでは、キューリソースでの Slurm 設定のサポートが追加され、パーティション固有の動作をきめ細かく制御できるようになりました。

カスタム Slurm 設定の利点

カスタム Slurm 設定では、PCS AWS ベースの HPC 環境を詳細に制御できます。quality-of-service 設定とプリエンプションポリシーを使用して、詳細なアカウンティングを実装し、アクセスコントロールを適用し、ワークロードの実行を最適化できます。これらの機能により、重要なジョブは効率的なクラスター使用率を維持しながら、必要なリソースを受け取ることができます。GPU アクセラレーションワークロードの管理、公平配分スケジューリングの実装、ジョブのライフサイクルの制御のいずれであっても、カスタム設定は HPC インフラストラクチャを運用要件と研究目標に合わせるのに役立ちます。

カスタム設定の構成

カスタム Slurm 設定は、リソースの作成時に AWS コンソール、CLI、または SDKs を介して設定することも、更新オペレーションを通じて後で変更することもできます。

AWS マネジメントコンソール

任意のリソースタイプ (クラスター、キュー、またはコンピューティングノードグループ) の作成または編集ページの追加のスケジューラ設定に移動します。

新しい設定を追加するには

1. 新しい設定の追加 を選択します。
2. ドロップダウンからパラメータ名を選択します (簡単なパラメータの説明が含まれます)。
3. 対応する値を指定します。

カスタム設定の設定を解除するには

1. 関連するパラメータと値のペアの横にある削除を選択します。
2. リソースを作成または更新します。

AWS CLI

カスタム設定をプログラムで管理するには、作成または更新オペレーションで `SlurmCustomSettings` フィールドを使用します。

Example- クラスターの Prolog パラメータの更新

```
aws pcs update-cluster --cluster-identifier my-cluster \  
--slurm-configuration \  
'SlurmCustomSettings=[{parameterName=Prolog,parameterValue="/path/to/prolog.sh"}]'
```

Example- クラスター Default の にキューを設定する

```
aws pcs update-queue \  
--cluster-identifier my-cluster \  
--queue-identifier my-queue \  
--slurm-configuration \  
'SlurmCustomSettings=[{parameterName=Default,parameterValue=YES}]'
```

Example- コンピューティングノードグループ Features でのカスタム設定

```
aws pcs update-compute-node-group \  
--cluster-identifier my-cluster \  
--compute-node-group-identifier my-cng-1 \  
--slurm-configuration \  
'SlurmCustomSettings=[{parameterName=Features,parameterValue="gpu,nvme"}]'
```

検証とエラー処理

AWS PCS は、カスタム Slurm 設定の多層検証プロセスを実装します。作成オペレーションと更新オペレーションの両方で、以下を含む同期検証を実行します。

- フィールドレベルのチェック: 正しいデータ型、許容値、形式要件について個々の設定を検証します。たとえば、時間値が正しい Slurm 形式であることを確認し、ブール値は受け入れられた Slurm ブール表現を使用します。
- コンテキスト対応の検証: 一部の設定は、より広範な設定コンテキストに対してチェックされます。たとえば、特定のパラメータは Slurm アカウンティングが有効になっている場合にのみ有効です。
- 設定間の整合性: 相互に排他的なオプションと一緒に設定されておらず、相互依存する設定が正しく設定されていることを確認します。

検証が失敗すると、特定のエラーコード (InvalidInput など)、問題を説明する明確なエラーメッセージ、無効なフィールドのリスト、およびそれぞれのエラーの詳細 ValidationException を含む が表示されます。

この初回検証中に多くの問題が発見されますが、設定間の複雑なやり取りは、設定を適用するときのみ明らかになる場合があります。このような場合、オペレーションは失敗し、有益なエラーメッセージが表示され、部分的な変更はロールバックされます。

制限事項

AWS PCS は、サービスのセキュリティと運用の安定性を保護するための許可リストアプローチを実装しています。サービスアカウントのセキュリティを侵害したり、マネージドサービス機能を妨害したりする可能性のある設定は制限されます。ただし、お客様のニーズを継続的に評価し、お客様からのフィードバックに基づいて追加の設定のサポートを追加できます。

トピック

- [PCS クラスターのカスタム Slurm AWS 設定](#)
- [PCS コンピューティングノードグループのカスタム Slurm AWS 設定](#)
- [PCS キューのカスタム Slurm AWS 設定](#)
- [PCS でのカスタム Slurm AWS 設定のトラブルシューティング](#)

PCS クラスターのカスタム Slurm AWS 設定

クラスターレベルでは、次のカスタム Slurm 設定がサポートされています。

- [AccountingStorageEnforce](#)

 Important

AWS PCS は、 のオプションの一部をサポートしています AccountingStorageEnforce。詳細については、「[PCS での Slurm AWS アカウンティング](#)」を参照してください。

- [AccountingStorageTRES](#)
- [AccountingStoreFlags](#)
- [DefMemPerCPU](#)
- [Epilog](#)
- [EnforcePartLimits](#)
- [FairShareDampeningFactor](#)
- [HealthCheckInterval](#)
- [HealthCheckNodeState](#)
- [HealthCheckProgram](#)
- [JobRequeue](#)
- [LaunchParameters](#)
- [Licenses](#)
- [MinJobAge](#)

 Note

AWS PCS は、 の最小値 5 秒をサポートします MinJobAge。

- [OverTimeLimit](#)
- [PreemptExemptTime](#)
- [PreemptMode](#)
- [PreemptParameters](#)
- [PreemptType](#)
- [PriorityCalcPeriod](#)
- [PriorityDecayHalfLife](#)
- [PriorityFavorSmall](#)

- [PriorityFlags](#)
- [PriorityMaxAge](#)
- [PriorityUsageResetPeriod](#)
- [PriorityWeightAge](#)
- [PriorityWeightAssoc](#)
- [PriorityWeightFairshare](#)
- [PriorityWeightJobSize](#)
- [PriorityWeightPartition](#)
- [PriorityWeightQOS](#)
- [PriorityWeightTRES](#)
- [PrivateData](#)
- [Prolog](#)
- [PrologFlags](#)
- [PropagatePrioProcess](#)
- [PropagateResourceLimits](#)
- [PropagateResourceLimitsExcept](#)
- [RequeueExit](#)
- [RequeueExitHold](#)
- [SchedulerParameters](#)
- [SelectTypeParameters](#)
- [SrunPortRange](#)
- [TaskEpilog](#)
- [TaskPluginParam](#)
- [TaskProlog](#)
- [UnkillableStepProgram](#)
- [UnkillableStepTimeout](#)

PCS コンピューティングノードグループのカスタム Slurm AWS 設定

コンピューティングノードグループレベルでは、次のカスタム Slurm 設定がサポートされています。

- [CpuSpecList](#)
- [Features](#)
- [MemSpecLimit](#)
- [RealMemory](#)
- [Weight](#)

PCS キューのカスタム Slurm AWS 設定

キューレベルでは、次のカスタム Slurm 設定がサポートされています。

- [AllowAccounts](#)
- [AllowQoS](#)
- [Default](#)
- [DefaultTime](#)
- [DenyAccounts](#)
- [DenyQoS](#)
- [ExclusiveUser](#)
- [GraceTime](#)
- [MaxTime](#)
- [OverSubscribe](#)
- [OverTimeLimit](#)
- [PreemptMode](#)
- [PriorityJobFactor](#)
- [PriorityTier](#)
- [QOS](#)
- [TRESBillingWeights](#)

PCS でのカスタム Slurm AWS 設定のトラブルシューティング

Slurm カスタム設定で AWS PCS リソースを作成または更新するときにエラーが発生した場合は、ログ記録を使用して問題を診断して解決できます。

互換性のない Slurm カスタム設定のトラブルシューティング

問題: クラスター、コンピューティングノードグループ、またはキューオペレーションを実行すると、次のようなエラーメッセージが表示されます。

```
{OPERATION} failed. The Slurm custom settings of the cluster might be incompatible.  
Check the settings and try again.
```

このエラーは、次のオペレーションで発生する可能性があります。

- CreateCluster
- CreateComputeNodeGroup
- UpdateComputeNodeGroup
- CreateQueue
- UpdateQueue

解決策: ログ記録を有効にして特定の問題を理解し、互換性のない設定のトラブルシューティングを行います。

互換性のない Slurm カスタム設定をトラブルシューティングするには

1. クラスターがまだ存在しない場合は作成するか、既存のクラスターがログ記録を有効にできる状態であることを確認します。
2. クラスターのログ記録を有効にします。詳細な手順については、「[PCS AWS のログ記録とモニタリング](#)」を参照してください。

Note

ログ記録は、クラスターの作成時に有効にできます。

3. ログを確認して、非互換性の原因となっている特定の Slurm 設定の問題を特定します。
4. ログ情報に基づいて互換性のないカスタム設定を修正し、オペレーションを再試行します。

サポートされている Slurm カスタム設定の詳細については、以下を参照してください。

- [PCS クラスターのカスタム Slurm AWS 設定](#)

- [PCS コンピューティングノードグループのカスタム Slurm AWS 設定](#)
- [PCS キューのカスタム Slurm AWS 設定](#)

SPANK プラグインを使用して AWS PCS で Slurm 機能を拡張する

SPANK (Slurm Plug-in Architecture for Node and job Kontrol) プラグインを使用して、PCS クラスターでのジョブの起動と実行中に Slurm AWS の動作を拡張および変更します。SPANK プラグインは、ジョブの起動ステージを傍受および変更するための汎用インターフェイスを提供します。

コンピューティングノード AMI に SPANK プラグインをインストールし、ワークロード要件に合わせて Slurm クラスターの動作をカスタマイズするように設定します。SPANK の詳細については、SchedMD ウェブサイトの [SPANK ドキュメント](#) を参照してください。

目次

- [PCS に SPANK AWS プラグインをインストールする](#)
- [PCS で SPANK AWS プラグインを設定する](#)
- [PCS での SPANK AWS プラグインに関するよくある質問](#)

PCS に SPANK AWS プラグインをインストールする

プラグインのドキュメントに従って、AMI に SPANK プラグインをインストールします。

クラスター上の特定の Slurm バージョンの SPANK プラグインをコンパイルします。PCS が提供する Slurm AWS インストーラは、Slurm を `/opt/aws/pcs/scheduler/slurm-version` に保存します。プラグインをコンパイルするときは、Slurm バージョンを指定します。

次の例は、一部のプラグインの Slurm バージョンを指定する方法を示しています。

```
export CFLAGS="-I/opt/aws/pcs/scheduler/slurm-version/include"
```

AMI に複数の Slurm バージョンがある場合は、バージョンごとにプラグインをコンパイルします。コンパイルされたプラグインをバージョン管理されたフォルダに保存します。

次の例は、一部のプラグインの送信先フォルダを指定する方法を示しています。

```
export DESTDIR="your-preferred-versioned-path"
```

 Important

プラグインには異なる変数が必要になる場合があります。インストールするプラグインの公式ドキュメントを参照してください。

PCS で SPANK AWS プラグインを設定する

デフォルトでは、設定ファイルを に保存します `/etc/aws/pcs/scheduler/slurm-version/plugstack.conf.d/`。

SPANK 設定を別の場所に保存するには、デフォルトのディレクトリの設定ファイルに場所を追加します。

次の例は、他のディレクトリの設定ファイルを含める方法を示しています。

```
# content of /etc/aws/pcs/scheduler/slurm-version/any-filename.conf
include path-to-your-configuration-folder/*.conf
include path-to-a-second-configuration-folder/*.conf
```

各設定を専用ファイルまたは共通ファイルに保存します。複数の設定ファイルを使用できます。

次の例は、サンプル設定ファイルを示しています。

```
# content of path-to-your-or-default-config-folder/filename-1.conf
required path-to-plugin-1 arguments
optional path-to-plugin-2 arguments
```

```
# content of path-to-your-or-default-config-folder/filename-2.conf
required path-to-plugin-3 arguments
```

プラグインの設定方法の詳細については、SchedMD ウェブサイトの [SPANK 設定ドキュメント](#) を参照してください。

 Important

プラグイン設定への不正な変更を防ぐため、フォルダのアクセス許可を設定します。

Note

AWS PCS は SPANK プラグインを管理しません。プラグインに関連するエラーが発生した場合は、コンピューティングノードのエラーログを確認してください。

Note

Slurm は、SPANK 設定をロードするときに、次のようなエラーを誤ってログに記録します。

```
error: "Include" failed in file /etc/slurm/plugstack.conf line 3
```

このエラーは無視して構いません。SPANK プラグインの動作には影響しません。

PCS での SPANK AWS プラグインに関するよくある質問

このセクションでは、PCS クラスターへの SPANK AWS プラグインのインストールと設定に関する一般的な質問について説明します。

ログインノードとコンピューティングノードの両方に SPANK プラグインをインストールする必要がありますか？

一部の SPANK プラグインはすべてのノードにインストールする必要はありませんが、互換性を高めるために、すべてのノードにすべての SPANK プラグインをインストールすることをお勧めします。

SPANK プラグインを本番環境で使用するには、どのような追加設定が必要ですか？

例に示す基本的なインストールと設定以外にも、本稼働デプロイでは通常、追加のセットアップが必要です。Pyxis などのコンテナベースのプラグインでは、Enroot の環境変数の設定、PMI (プロセス管理インターフェイス) の有効化、コンテナランタイムのアクセス許可の設定が必要になる場合があります。本番環境のデプロイ要件の詳細については、特定のプラグインのドキュメントを参照してください。

SPANK プラグインの問題をトラブルシューティングする方法を教えてください。

AWS PCS は SPANK プラグインを管理しません。コンピューティングノードのエラーログを調べて、問題をトラブルシューティングします。

Slurm CLI フィルタープラグインを使用して PCS AWS でのジョブ送信をカスタマイズする

AWS PCS は、Slurm CLI Filter Plugins をサポートし、ログインノードとコンピューティングノードでジョブ送信パラメータを検証および変更するカスタム Lua スクリプトを実行します。CLI フィルタープラグインの詳細については、SchedMD ウェブサイトの [cli_filter Plugin API ドキュメント](#) を参照してください。

要件

CLI フィルタープラグインには、Slurm バージョン 24.11 以降と、すべてのログインノードとコンピューティングノードにデプロイされた Lua スクリプトが必要です。

Important

Slurm バージョン 24.11 および 25.05 の場合、CLI フィルタープラグインは AWS PCS Slurm インストーラ (バージョン 24.11.6-2+ または 25.05.4-1+) を使用して Slurm をインストールする必要があります。Slurm のインストールの詳細については、「」を参照してください [ステップ 3 – Slurm をインストールする](#)。

制限とセキュリティ上の考慮事項

- セキュリティの適用 – CLI フィルタープラグインは、どのユーザーでも簡単にバイパスでき、セキュリティクリティカルなポリシーには使用しないでください。ユーザーは、ジョブの送信中に が無効になっているカスタム設定を指定することで、CLI フィルタープラグイン `CLIFilterPlugins` を無効にすることができます。
- Lua 実装のみ – Lua スクリプト実装がサポートされています。C 実装はサポートされていません。

トピック

- [PCS クラスタで Slurm CLI AWS フィルタープラグインを設定する](#)
- [Amazon S3 を使用して PCS に CLI AWS フィルタプラグインスクリプトをデプロイする](#)
- [Slurm Job Submit プラグインスクリプトを変換して PCS で CLI Filter Plugin AWS を使用する](#)
- [PCS の Slurm CLI AWS フィルタープラグインに関するよくある質問](#)
- [PCS での Slurm CLI AWS フィルタープラグインの問題のトラブルシューティング](#)

PCS クラスターで Slurm CLI AWS フィルタープラグインを設定する

新しい PCS クラスターを作成するときに CLI AWS フィルタープラグインを設定します。クラスターを再作成することなく、Update API またはコンソールを使用して、既存のクラスターで CLI Filter Plugins を有効または無効にできます。

前提条件

CLI フィルタープラグインを設定する前に、以下のタスクを完了してください。

- CLI Filter Plugin API を実装する Lua スクリプトを記述してテストする
- Lua スクリプトに正確に名前を付ける `cli_filter.lua`
- スクリプトをすべてのクラスターインスタンス (AMI、S3、またはファイルシステム) にデプロイする方法を選択する
- Slurm バージョン 24.11 以降を使用していることを確認します。

新しいクラスターで CLI フィルタープラグインを有効にする

AWS PCS console

1. <https://console.aws.amazon.com/pcs/> で AWS PCS コンソールを開きます。
2. ナビゲーションペインで [Clusters] (クラスター) を選択してください。
3. [クラスターを作成] を選択します。
4. 有効なバージョンの Slurm (バージョン 24.11 以降) を選択します。
5. スケジューラ設定で、追加のスケジューラ設定を展開します。
6. パラメータ名を `cliFilterPlugins` に設定し、パラメータ値を `cli_filter.lua` に設定して、新しい Slurm カスタム設定を追加します。
7. 残りのクラスター設定を完了し、[クラスターを作成] を選択します。

AWS PCS API

CreateCluster API アクションの呼び出しで `slurmCustomSettings` 設定を指定します。を `parameterName` に `cliFilterPlugins` に設定し、`parameterValue` を `cli_filter.lua` に設定します。詳細については、AWS PCS API リファレンスの [CreateCluster](#) を参照してください。

次の例では AWS CLI、を使用して CreateCluster API アクションを呼び出します。カスタム設定は CLI フィルタープラグイン `CliFilterPlugins=cli_filter/lua` を有効にします。

```
aws pcs create-cluster --cluster-name cluster-name \  
--scheduler type=SLURM,version=24.11 \  
--size SMALL \  
--networking subnetIds=cluster-subnet-id,securityGroupIds=cluster-security-group-id \  
--slurm-configuration \  
'slurmCustomSettings=[{parameterName=CliFilterPlugins,parameterValue="cli_filter/  
lua"}]'
```

CLI フィルタープラグインスクリプトをデプロイする

CLI フィルタープラグインスクリプトをクラスターにデプロイするには

1. コンピューティングノードグループで使用するすべての AMIs に、PCS Slurm インストーラを介して AWS Slurm がインストールされていることを確認します。

Note

すべてのコンピューティングノードグループに AWS PCS サンプル AMI を使用する場合は、このステップをスキップします。Slurm は既にインストールされています。

2. クラスター内のすべてのインスタンス `/etc/aws/pcs/scheduler/slurm-<version>/cli_filter.lua` に `cli_filter.lua` スクリプトを にデプロイします。

たとえば、Slurm バージョン 24.11 の場合:

```
/etc/aws/pcs/scheduler/slurm-24.11/cli_filter.lua
```

3. 準備AMIs を使用して、すべてのログインノードとコンピューティングノードを起動します。
4. ジョブの送信をテストして、CLI フィルタープラグインが正しく実行されることを確認します。

既存のクラスターで CLI フィルタープラグインを有効または無効にする

インフラストラクチャを再構築することなく、既存のクラスターで CLI フィルタープラグインを有効または無効にできます。詳細については、「[PCS AWS でのクラスターの更新](#)」を参照してください。

AWS PCS console

1. <https://console.aws.amazon.com/pcs/> で AWS PCS コンソールを開きます。
2. ナビゲーションペインで [Clusters] (クラスター) を選択してください。
3. 更新するクラスターを選択します。
4. [アクションを編集] を選択します。
5. クラスターの編集ページの「追加のスケジューラ設定」を参照してください。
 - CLI フィルタープラグインを有効にするには: パラメータ名を `cli_filter_plugins` に設定し、パラメータ値を `cli_filter/lua` に設定して、新しい Slurm カスタム設定を追加します。
 - CLI フィルタープラグインを無効にするには: 既存の `CliFilterPlugins` 設定を削除します。
6. クラスターの更新を選択して変更を送信します。
7. クラスターのステータスをモニタリングします。これは、プロセス中に「更新」と表示され、更新が完了すると「アクティブ」と表示されます。

AWS PCS API

UpdateCluster API アクションを使用して、CLI フィルタープラグインを有効または無効にします。詳細については、AWS「PCS API リファレンス」の [UpdateCluster](#) を参照してください。

既存のクラスターで CLI フィルタープラグインを有効にするには:

```
aws pcs update-cluster --cluster-identifier my-cluster \
--slurm-configuration \
'slurmCustomSettings=[{parameterName=CliFilterPlugins,parameterValue="cli_filter/
lua"}]'
```

既存のクラスターで CLI フィルタープラグインを無効にするには:

```
aws pcs update-cluster --cluster-identifier my-cluster \
--slurm-configuration \
'slurmCustomSettings=[]'
```

予想される結果

設定が完了したら、次の操作を行います。

- クラスターは CLI フィルタープラグインを有効にして作成されます
- ジョブの送信により、Slurm コントローラーに到達する前にカスタム検証ロジックがトリガーされます。
- 非準拠ジョブはカスタムエラーメッセージで拒否されます
- 準拠ジョブは Slurm スケジューラを介して正常に進行します

トラブルシューティング

CLI フィルタープラグインスクリプトがノードにありません

症状: プラグインのロードエラーにより、ジョブの送信がすぐに失敗します。

考えられる原因: スクリプトがすべてのインスタンスにデプロイされていないか、ファイルパスまたは名前が正しくありません。

解決策: スクリプトが、正確なファイル名を持つすべてのログインノードとコンピューティングノードに正しいパスに存在することを確認します `cli_filter.lua`。

無効な CLI フィルタープラグイン設定

症状: クラスターの作成が検証エラーで失敗します。

考えられる原因: `CliFilterPlugins` パラメータが `cli_filter/lua` 形式に設定されていません。

解決策: `cli_filter/lua` で正確なパラメータ値を使用します `slurmCustomSettings`。

Amazon S3 を使用して PCS に CLI AWS フィルタプラグインスクリプトをデプロイする

AMIs を再構築せずにライブクラスターでジョブ送信ロジックを更新する場合は、S3 を使用して CLI フィルタプラグインスクリプトをデプロイします。このアプローチでは、ユーザーデータを使用してインスタンスの起動中に S3 からスクリプトをダウンロードします。

前提条件

S3 を使用してスクリプトをデプロイする前に、以下のタスクを完了してください。

- CLI Filter Plugin Lua スクリプトを使用して S3 バケットを作成する
- S3 バケットへの読み取りアクセス権を持つ IAM インスタンスプロファイルを設定する
- インターネットなしで直接アクセスできるように S3 VPC Gateway エンドポイントを設定する
- S3 からダウンロードするユーザーデータスクリプトを準備する

S3 を使用して CLI フィルタープラグインスクリプトをデプロイするには

1. cli_filter.lua スクリプトを S3 バケットにアップロードします。
2. バケットの S3 読み取りアクセス許可を使用して IAM インスタンスプロファイルを設定します。
3. 起動テンプレートの user-data にシェルコードを追加して、スクリプトをダウンロードします。

```
aws s3 cp s3://my-bucket/cli_filter.lua /etc/aws/pcs/scheduler/slurm-24.11/  
cli_filter.lua  
chmod 644 /etc/aws/pcs/scheduler/slurm-24.11/cli_filter.lua
```

4. 更新された起動テンプレートを使用してコンピューティングノードグループをデプロイします。
5. ジョブの送信をテストして、スクリプトの機能を検証します。

予想される結果

S3 デプロイが完了したら:

- CLI フィルタープラグインスクリプトは、起動時にすべてのインスタンスに自動的にダウンロードされます
- S3 のスクリプト更新は、新しく起動されたインスタンスに反映されます。
- ジョブ送信ポリシーはクラスター全体で一貫して適用されます。

トラブルシューティング

S3 アクセスが拒否されました

症状: インスタンスの起動が失敗するか、スクリプトがダウンロードされません。

考えられる原因: IAM アクセス許可または S3 VPC エンドポイントがない。

解決策: IAM インスタンスプロファイルに `s3:GetObject` アクセス許可があり、S3 VPC エンドポイントが設定されていることを確認します。

Slurm Job Submit プラグインスクリプトを変換して PCS で CLI Filter Plugin AWS を使用する

他の Slurm 環境から移行するときに、既存の Job Submit Plugin Lua スクリプトを CLI Filter Plugin に変換します。翻訳プロセスでは、関数名とフィールドアクセスパターンを更新して CLI Filter Plugin API と連携します。

前提条件

スクリプトを翻訳する前に、以下のタスクを完了してください。

- 既存の Job Submit Plugin Lua スクリプトを確認する
- ジョブ送信 API と CLI フィルタープラグイン APIs の違いを理解する
- Slurm CLI Filter Plugin ドキュメントにアクセスする

ジョブ送信プラグインスクリプトを CLI フィルタープラグインに変換するには

1. 既存の Job Submit Plugin スクリプト関数 (`slurm_job_submit`、) を確認します `slurm_job_modify`。
2. 同等の CLI フィルタープラグイン関数を特定します。
 - `slurm_job_submit` が `slurm_cli_pre_submit` になります
 - デフォルトのパラメータ設定 `slurm_cli_setup_defaults` に を追加する
 - 送信後のアクション `slurm_cli_post_submit` に を追加する
3. ジョブ検証ロジックを `job_desc` フィールドから `options` 配列アクセスに変換します。
 - `job_desc.account` が `options["account"]` になります
 - `job_desc.partition` が `options["partition"]` になります
 - `job_desc.features` が `options["constraint"]` になります
4. ログ記録呼び出しを から `slurm.log_user()` に更新します `slurm.log_error()`。
5. 開発クラスターで翻訳されたスクリプトをテストします。

6. 標準の CLI フィルタープラグインのデプロイプロセスに従って、本番稼働用クラスターにデプロイします。

予想される結果

翻訳が完了したら、次の操作を行います。

- 翻訳されたスクリプトは同等のジョブ送信検証を提供します
- ユーザーには、元のジョブ送信プラグインと同様のエラーメッセージとプロンプトが表示されます。
- ジョブ送信ポリシーは PCS AWS への移行中に維持されます

トラブルシューティング

スクリプト翻訳エラー

症状: ジョブの送信は Lua 実行エラーで失敗します。

考えられる原因: 翻訳されたスクリプトのフィールドアクセスまたは関数呼び出しが正しくない。

解決策: CLI Filter Plugin API ドキュメントを確認し、ジョブ送信インターフェイスと CLI Filter インターフェイス間のフィールドマッピングを比較します。

PCS の Slurm CLI AWS フィルタープラグインに関するよくある質問

CLI フィルタープラグインに関するよくある質問を確認してください。

CLI フィルタープラグインとジョブ送信プラグインの違いは何ですか？

CLI フィルタープラグインは、ジョブ送信がコントローラーに到達する前にログインノードとコンピューティングノードでクライアント側を実行し、ジョブ送信プラグインはジョブ送信後にコントローラーでサーバー側を実行します。CLI フィルタープラグインはユーザーがバイパスできますが、コントローラーロックを保持しません。一方、ジョブ送信は安全ですが、実行中のクラスターのパフォーマンスに影響を与える可能性があります。

PCS AWS は Slurm ジョブ送信プラグインをサポートしていますか？

いいえ。ジョブ送信プラグインは PCS AWS ではサポートされていません。ジョブの送信の検証と変更には、代わりに CLI フィルタープラグインを使用します。

セキュリティの適用に CLI フィルタープラグインを使用できますか？

いいえ。CLI フィルタープラグインは、特定のユーザーがバイパスできるため、セキュリティの適用に頼るべきではありません。セキュリティクリティカルなポリシーではなく、ユーザーエクスペリエンスの向上、デフォルトのパラメータ設定、ポリシーガイダンスに使用します。

ログインノードだけでなく、スクリプトがすべてのコンピューティングノードに存在する必要があるのはなぜですか？

などの Slurm コマンドは、コンピューティングノードのジョブスクリプト内で実行 `srun` できます。これにより、CLI フィルタプラグインの実行もトリガーされます。スクリプトは、Slurm コマンドが実行されるすべての場所で利用できる必要があります。

ライブクラスターで CLI フィルタープラグインスクリプトを変更できますか？

はい。S3 またはファイルシステムのデプロイアプローチを使用する場合。新しいインスタンスは更新されたスクリプトを取得しますが、既存のインスタンスでは、スクリプトを手動で更新するか、選択したデプロイ方法を使用して更新する必要があります。

コンピューティングノードグループごとに異なる CLI Filter Plugin スクリプトを使用できますか？

はい。ただし、これは推奨されません。異なるコンピューティングノードグループに異なるロジックを持つスクリプトを提供できますが、相互依存関係を管理し、重複するロジックを防ぐ責任があります。ほとんどのお客様は、クラスター全体で 1 セットのロジックを提供します。

Lua の代わりに CLI Filter Plugin を C 実装で使用することはできますか？

C 実装はサポートされていません。PCS では Lua AWS スクリプトの実装のみがサポートされています。SchedMD では、CLI フィルタープラグインを実装するときに使いやすくするために、C よりも Lua を使用することをお勧めします。

既存のクラスターで CLI フィルタープラグインをオンまたはオフにできますか？

はい。クラスターを再作成せずに Update API を使用して、既存のクラスターで CLI Filter Plugin を有効または無効にできます。

PCS での Slurm CLI AWS フィルタープラグインの問題のトラブルシューティング

このトラブルシューティング情報を使用して、一般的な CLI フィルタープラグインの問題を解決します。

プラグインのロードエラーによりジョブの送信がすぐに失敗する

症状: ジョブを送信すると、ユーザーは CLI フィルタプラグインの欠落または失敗に関するエラーメッセージを受け取ります。

考えられる原因:

- CLI フィルタプラグインスクリプトが 1 つ以上のノードにありません
- スクリプトファイル名が正しくない (正確に `cli_filter.lua` である必要があります)
- 間違ったディレクトリパスにデプロイされたスクリプト
- スクリプトのファイルアクセス許可が正しくない

解決策:

- スクリプトがすべてのログインノードとコンピューティングノード `/etc/aws/pcs/scheduler/slurm-<version>/cli_filter.lua` に存在することを確認する
- スクリプトファイル名が正確であることを確認する `cli_filter.lua`
- スクリプトに読み取り可能なアクセス許可があることを確認する (644 など)
- フルクラスターにデプロイする前に、単一のログインノードでスクリプトのデプロイをテストする

クラスターの作成が CLI フィルタープラグインの検証エラーで失敗する

症状: クラスターの作成が失敗し、無効な `CliFilterPlugins` パラメータに関するエラーが発生します。

考えられる原因:

- のパラメータ値の形式が正しくない `slurmCustomSettings`
- パラメータ名または値のタイプミス

解決策:

- 正確なパラメータ名を使用します。 `CliFilterPlugins`
- 正確なパラメータ値を使用します。 `cli_filter/lua`

- `slurmCustomSettings` 配列で JSON 構文を検証する

CLI フィルタープラグインスクリプトは実行されますが、ジョブの検証は期待どおりに機能しません

症状: ジョブは正常に送信されますが、カスタム検証ロジックではトリガーされず、予期しない結果も生成されません。

考えられる原因:

- Lua スクリプト構文エラー
- フィールドアクセスパターンが正しくない (CLI フィルタープラグインの代わりにジョブ送信プラグイン構文を使用)
- 検証条件のロジックエラー

解決策:

- Lua スクリプトで構文エラーを確認する
- フィールドアクセスが `job_desc.field_name` ではなく `options["field_name"]` 形式を使用することを確認する
- スクリプト実行フローをデバッグするためのログ記録ステートメントの追加
- シンプルな検証ケースを最初に使用してスクリプトロジックをテストする

S3 スクリプトのデプロイが失敗する

症状: インスタンスは起動しますが、CLI Filter Plugin スクリプトは S3 からダウンロードされません。

考えられる原因:

- IAM インスタンスプロファイルに S3 読み取りアクセス許可がない
- S3 VPC エンドポイントが設定されていません
- ユーザーデータの S3 バケットまたはオブジェクトパスが正しくない

解決策:

- IAM インスタンスプロファイルにバケットに対する `s3:GetObject` アクセス許可があることを確認する
- 直接アクセス用に S3 VPC Gateway エンドポイントを設定する
- ユーザーデータスクリプトで S3 バケット名とオブジェクトパスを確認する
- インスタンスユーザーデータログで S3 ダウンロードエラーを確認する

AWS 並列コンピューティングサービスのセキュリティ

のクラウドセキュリティが最優先事項 AWS です。お客様は AWS 、最もセキュリティの影響を受けやすい組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャからメリットを得られます。

セキュリティは、AWS お客様とお客様の間の責任共有です。[責任共有モデル](#)ではこれをクラウドのセキュリティおよびクラウド内のセキュリティと説明しています。

- クラウドのセキュリティ – AWS は、で AWS サービスを実行するインフラストラクチャを保護する責任を担います AWS クラウド。は、お客様が安全に使用できるサービス AWS も提供します。サードパーティーの監査者は、[AWS コンプライアンスプログラム](#)コンプライアンスプログラムの一環として、当社のセキュリティの有効性を定期的にテストおよび検証。AWS Parallel Computing Service に適用されるコンプライアンスプログラムの詳細については、「[コンプライアンスプログラムAWS による対象範囲内のサービスコンプライアンスプログラム](#)」を参照してください。
- クラウド内のセキュリティ – お客様の責任は、使用する AWS サービスによって決まります。また、ユーザーは、データの機密性、会社の要件、適用される法律や規制など、その他の要因についても責任を負います。

このドキュメントは、PCS AWS を使用する際の責任共有モデルの適用方法を理解するのに役立ちます。以下のトピックでは、セキュリティとコンプライアンスの目的を達成するように AWS PCS を設定する方法について説明します。また、PCS リソースのモニタリングや保護に役立つ他の AWS AWS サービスの使用方法についても説明します。

トピック

- [AWS Parallel Computing Service でのデータ保護](#)
- [インターフェイスエンドポイント \(AWS PrivateLink\) AWS Parallel Computing Service を使用したへのアクセス](#)
- [AWS 並列コンピューティングサービスの Identity and Access Management](#)
- [AWS Parallel Computing Service のコンプライアンス検証](#)
- [AWS Parallel Computing Service の耐障害性](#)
- [AWS 並列コンピューティングサービスのインフラストラクチャセキュリティ](#)
- [AWS Parallel Computing Service での脆弱性分析と管理](#)
- [サービス間の混乱した代理の防止](#)

- [AWS Parallel Computing Service のセキュリティのベストプラクティス](#)

AWS Parallel Computing Service でのデータ保護

責任 AWS [共有モデル](#)、AWS Parallel Computing Service でのデータ保護に適用されます。このモデルで説明されているように、AWS はすべての 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 アイデンティティセンターまたは 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 は必須ですが、TLS 1.3 を推奨します。
- 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、API、または SDK AWS を使用して PCS 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。

保管中の暗号化

、AWS PCS AWS API AWS マネジメントコンソール AWS CLI、または SDK を使用して AWS Parallel Computing Service (PCS) クラスターを作成すると、保管中のデータの暗号化がデフォルトで有効になります。AWS SDKs AWS PCS は、AWS 所有の KMS キーを使用して保管中のデータを暗号化します。詳細については、「AWS KMS デベロッパーガイド」の[「カスタマーキーと AWS キー」](#)を参照してください。カスタマーマネージドキーを使用することもできます。詳細については、「[PCS の暗号化された EBS ボリュームで使用するために必要な KMS AWS キーポリシー](#)」を参照してください。

クラスターシークレットはに保存 AWS Secrets Manager され、Secrets Manager マネージド KMS キーで暗号化されます。詳細については、「[PCS AWS でのクラスターシークレットの使用](#)」を参照してください。

AWS PCS クラスターでは、次のデータは保管中です。

- スケジューラの状態 – クラスター内の実行中のジョブとプロビジョニングされたノードに関するデータが含まれます。これは、で StateSaveLocation 定義されたに Slurm が保持するデータです slurm.conf。詳細については、Slurm ドキュメントの [StateSaveLocation](#) の説明を参照してください。AWS PCS は、ジョブの完了後にジョブデータを削除します。
- スケジューラ認証シークレット – AWS PCS はそれを使用してクラスター内のすべてのスケジューラ通信を認証します。

スケジューラの状態情報の場合、AWS PCS はデータとメタデータをファイルシステムに書き込む前に自動的に暗号化します。暗号化されたファイルシステムは、保管中のデータに業界標準の AES-256 暗号化アルゴリズムを使用します。

転送中の暗号化

AWS PCS API への接続では、AWS Command Line Interface (AWS CLI) と AWS SDKs のどちらを使用するかにかかわらず、署名バージョン 4 の署名プロセスで TLS 暗号化が使用されます。詳細については、「[ユーザーガイド](#)」の AWS [「API リクエストの署名」](#)を参照してください。は、接続に使用するセキュリティ認証情報の IAM ポリシーを使用して API を介したアクセスコントロール AWS を管理します。AWS Identity and Access Management

AWS PCS は TLS を使用して他の AWS サービスに接続します。

Slurm クラスター内では、スケジューラはすべてのスケジューラ通信に auth/slurm 認証を提供する認証プラグインで設定されます。Slurm は通信のためにアプリケーションレベルで暗号化を提供しません。クラスターインスタンスをまたいで流れるすべてのデータは EC2 VPC のローカルに留まるため、それらのインスタンスが転送中の暗号化をサポートしている場合は VPC 暗号化の対象となります。詳細については、「Amazon Elastic Compute Cloud ユーザーガイド」の「[転送中の暗号化](#)」を参照してください。通信は、アカウントのクラスターノードのコントローラー (サービスアカウントでプロビジョニング) 間で暗号化されます。

キー管理

AWS PCS は、AWS 所有の KMS キーを使用してデータを暗号化します。詳細については、「AWS KMS デベロッパーガイド」の「[カスタマーキーと AWS キー](#)」を参照してください。カスタマーマネージドキーを使用することもできます。詳細については、「[PCS の暗号化された EBS ボリュームで使用するために必要な KMS AWS キーポリシー](#)」を参照してください。

クラスターシークレットは に保存 AWS Secrets Manager され、Secrets Manager マネージド KMS キーで暗号化されます。詳細については、「[PCS AWS でのクラスターシークレットの使用](#)」を参照してください。

ネットワーク間トラフィックのプライバシー

AWS クラスターの PCS コンピューティングリソースは、お客様のアカウントの 1 VPC 内に存在します。したがって、クラスター内のすべての内部 AWS PCS サービストラフィックは AWS ネットワーク内にとどまり、インターネットを経由しません。ユーザーノードと AWS PCS ノード間の通信はインターネットを通過する可能性があるため、SSH または Systems Manager を使用してノードに接続することをお勧めします。詳細については、「AWS Systems Manager ユーザーガイド」の「[What is AWS Systems Manager?](#)」を参照してください。

以下のサービスを使用して、オンプレミスネットワークを に接続することもできます AWS。

- AWS Site-to-Site VPN。詳細については、「AWS Site-to-Site VPN ユーザーガイド」の「[What is AWS Site-to-Site VPN?](#)」を参照してください。
- AWS Direct Connect。詳細については、「AWS Direct Connect ユーザーガイド」の「[What is AWS Direct Connect?](#)」を参照してください。

PCS API AWS にアクセスして、サービスの管理タスクを実行します。ユーザーとユーザーは Slurm エンドポイントポートにアクセスして、スケジューラと直接やり取りします。

API トラフィックの暗号化

AWS PCS API にアクセスするには、クライアントが Transport Layer Security (TLS) 1.2 以降をサポートしている必要があります。TLS 1.2 は必須ですが、TLS 1.3 を推奨します。クライアントは、Ephemeral Diffie-Hellman (DHE) や Elliptic Curve Diffie-Hellman Ephemeral (ECDHE) などの Perfect Forward Secrecy (PFS) を備えた暗号スイートもサポートする必要があります。モードは、Java 7 以降など、最近のほとんどのシステムでサポートされています。また、リクエストにはアクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。AWS Security Token Service (AWS STS) を使用して一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

データトラフィックの暗号化

転送中のデータの暗号化は、スケジューラエンドポイントにアクセスするサポートされている EC2 インスタンスから、および 内から ComputeNodeGroup インスタンス間で有効になります AWS クラウド。詳細については、「[転送中の暗号化](#)」を参照してください。

PCS の暗号化された EBS ボリュームで使用するために必要な KMS AWS キーポリシー

AWS PCS は、[サービスにリンクされたロール](#)を使用してアクセス許可を他の に委任します AWS のサービス。AWS PCS サービスにリンクされたロールは事前定義されており、PCS AWS が AWS のサービス ユーザーに代わって他の を呼び出すために必要なアクセス許可が含まれています。事前定義されたアクセス許可には、へのアクセスも含まれます AWS マネージドキー が、カスタマー マネージドキーへのアクセスは含まれません。

このトピックでは、Amazon EBS 暗号化のカスタマー マネージドキーを指定するときにインスタンスを起動するために必要なキーポリシーを設定する方法について説明します。

Note

AWS PCS では、アカウントの暗号化されたボリュームを保護するためにデフォルトを使用する AWS マネージドキー ための追加の認可は必要ありません。

内容

- [概要:](#)
- [キーポリシーを設定する](#)

- [例 1: カスタマー管理キーへのアクセスを許可するキーポリシーセクション](#)
- [例 2: カスタマー管理キーへのクロスアカウントアクセスを許可するキーポリシーセクション](#)
- [AWS KMS コンソールでキーポリシーを編集する](#)

概要:

PCS がインスタンスを起動するときに、Amazon EBS AWS 暗号化 AWS KMS keys に以下を使用できます。

- [AWS マネージドキー](#) – Amazon EBS によって作成、所有、および管理されるアカウント内の暗号化キー。これは、新しいアカウントのデフォルトの暗号化キーです。Amazon EBS は、カスタマーマネージドキーを指定しない限り、暗号化 AWS マネージドキー に を使用します。
- [カスタマーマネージド型キー](#) – お客様が作成、所有、および管理するカスタム暗号化キー。詳細については、[「デベロッパーガイド」の「KMS キーの作成」](#)を参照してください。AWS Key Management Service

Note

KMS キーは対称である必要があります。Amazon EBS は、非対称カスタマーマネージドキーをサポートしていません。

暗号化されたスナップショットまたは暗号化されたボリュームを指定する起動テンプレートを作成するとき、またはデフォルトで暗号化を有効にするときに、カスタマーマネージドキーを設定します。

キーポリシーを設定する

KMS キーには、PCS AWS がカスタマーマネージドキーで暗号化された Amazon EBS ボリュームを使用してインスタンスを起動できるようにするキーポリシーが必要です。

このページの例を使用して、PCS AWS にカスタマーマネージドキーへのアクセスを許可するキーポリシーを設定します。カスタマーマネージドキーのキーポリシーは、キーの作成時または後で変更できます。

キーポリシーには、次のステートメントが必要です。

- Principal 要素で指定された IAM ID がカスタマーマネージドキーを直接使用できるようにするステートメント。これには AWS KMS Encrypt、キーに対して

Decrypt、GenerateDataKey*、および ReEncrypt*DescribeKey オペレーションを実行するアクセス許可が含まれます。

- Principal 要素で指定された IAM ID が CreateGrant オペレーションを使用して、AWS KMS または別のプリンシパルと統合された 独自のアクセス許可のサブセットを委任 AWS のサービス する許可を生成できるようにするステートメント。これにより、それらのサービスはお客様に代わって、キーを使用して、暗号化されたリソースを作成できるようになります。

新しいポリシーステートメントをキーポリシーに追加するときは、ポリシー内の既存のステートメントを変更しないでください。

詳細については、以下を参照してください。

- AWS CLI コマンドリファレンスの [create-key](#)
- AWS CLI コマンドリファレンスの [put-key-policy](#)
- デAWS Key Management Service ベロッパーガイドの [キー ID とキー ARN](#) を検索する
- [PCS AWS のサービスにリンクされたロール](#)
- [Amazon EBS ユーザーガイド](#) の「Amazon EBS 暗号化」
- 「AWS Key Management Service デベロッパーガイド」の「[AWS Key Management Service](#)」

例 1: カスタマー管理キーへのアクセスを許可するキーポリシーセクション

カスターマネージドキーのキーポリシーに次のポリシーステートメントを追加します。サンプル ARN を、AWSServiceRoleForPCS サービスにリンクされたロールの ARN に置き換えます。このポリシー例では、AWS PCS サービスにリンクされたロール (AWSServiceRoleForPCS) に、カスターマネージドキーを使用するためのアクセス許可を付与します。

```
{
  "Sid": "Allow service-linked role use of the customer managed key",
  "Effect": "Allow",
  "Principal": {
    "AWS": [
      "arn:aws:iam::account-id:role/aws-service-role/pcs.amazonaws.com/
AWSServiceRoleForPCS"
    ]
  },
  "Action": [
    "kms:Encrypt",
```

```

    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
  ],
  "Resource": "*"
}

```

```

{
  "Sid": "Allow attachment of persistent resources",
  "Effect": "Allow",
  "Principal": {
    "AWS": [
      "arn:aws:iam::account-id:role/aws-service-role/pcs.amazonaws.com/AWSServiceRoleForPCS"
    ]
  },
  "Action": [
    "kms:CreateGrant"
  ],
  "Resource": "*",
  "Condition": {
    "Bool": {
      "kms:GrantIsForAWSResource": true
    }
  }
}

```

例 2: カスタマー管理キーへのクロスアカウントアクセスを許可するキーポリシーセクション

AWS PCS クラスターとは異なるアカウントにカスタマーマネージドキーを作成する場合は、キーポリシーと組み合わせて許可を使用して、キーへのクロスアカウントアクセスを許可する必要があります。

キーへのアクセスを許可するには

1. カスタマーマネージドキーのキーポリシーに次のポリシーステートメントを追加します。サンプル ARN を他のアカウントの ARN に置き換えます。**111122223333** を、PCS クラスター AWS アカウント を作成する の実際のアカウント ID AWS に置き換えます。そうすることで、次の CLI コマンドを使用して、指定されたアカウント内の IAM ユーザーまたはロールに、キーに対

するグラントを作成する許可を提供できます。デフォルトでは、ユーザーは キーにアクセスできません。

```
{.
  "Sid": "Allow external account 111122223333 use of the customer managed key",
  "Effect": "Allow",
  "Principal": {
    "AWS": [
      "arn:aws:iam::111122223333:root"
    ]
  },
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
  ],
  "Resource": "*"
}
```

```
{
  "Sid": "Allow attachment of persistent resources in external
account 111122223333",
  "Effect": "Allow",
  "Principal": {
    "AWS": [
      "arn:aws:iam::111122223333:root"
    ]
  },
  "Action": [
    "kms:CreateGrant"
  ],
  "Resource": "*"
}
```

2. PCS クラスターを作成するアカウントから、関連するアクセス許可を AWS PCS AWS サービスにリンクされたロールに委任する許可を作成します。の値は、サービスにリンクされたロールの ARN grantee-principal です。の値はキーの ARN key-id です。

次の [create-grant](#) CLI コマンドの例では、アカウント **111122223333** AWSServiceRoleForPCSで という名前のサービスにリンクされたロールに、アカウント **444455556666** でカスタマーマネージドキーを使用するためのアクセス許可を付与します。

```
aws kms create-grant \  
  --region us-west-2 \  
  --key-id arn:aws:kms:us-west-2:444455556666:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d \  
  --grantee-principal arn:aws:iam::111122223333:role/aws-service-role/  
pcs.amazonaws.com/AWSServiceRoleForPCS \  
  --operations "Encrypt" "Decrypt" "ReEncryptFrom" "ReEncryptTo" "GenerateDataKey"  
"GenerateDataKeyWithoutPlaintext" "DescribeKey" "CreateGrant"
```

Note

リクエストを行うユーザーには、`kms:CreateGrant` アクションを使用するためのアクセス許可が必要です。

次の IAM ポリシーの例では、アカウント **111122223333** 内の IAM アイデンティティ (ユーザーまたはロール) がアカウント **444455556666** 内でカスタマーマネージド型キーの付与を作成できるようにします。

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "AllowCreationOfGrantForTheKMSKeyinExternalAccount444455556666",  
      "Effect": "Allow",  
      "Action": "kms:CreateGrant",  
      "Resource": "arn:aws:kms:us-west-2:444455556666:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d"  
    }  
  ]  
}
```

異なる AWS アカウント内での KMS キーに対するグラントの作成に関する詳細については、「AWS Key Management Service デベロッパーガイド」の「[Grants in AWS KMS](#)」を参照してください。

 Important

被付与者のプリンシパルとして指定されたサービスにリンクされたロール名は、既存のロールの名前でなければなりません。権限を作成した後、その権限で PCS AWS が指定された KMS キーを使用できるように、サービスにリンクされたロールを削除して再作成しないでください。

AWS KMS コンソールでキーポリシーを編集する

以前のセクションの例では、キーポリシーにステートメントを追加する方法のみを示しています。これは、キーポリシーを変更する 1 つの方法にすぎません。キーポリシーを変更する最も簡単な方法は、キーポリシーに AWS KMS コンソールのデフォルトビューを使用し、IAM ID (ユーザーまたはロール) を適切なキーポリシーのキーユーザーの 1 つにすることです。詳細については、「AWS Key Management Service デベロッパーガイド」の [AWS マネジメントコンソール「デフォルトビュー」の使用](#) を参照してください。

 Warning

コンソールのデフォルトのビューポリシーステートメントには、カスターマネージドキーでオペレーションを実行する AWS KMS Revoke アクセス許可が含まれています。アカウントのカスターマネージドキー AWS アカウント へのアクセス権を付与する許可を取り消すと、 のユーザーは暗号化されたデータとキーにアクセス AWS アカウント できなくなります。

インターフェイスエンドポイント (AWS PrivateLink) AWS Parallel Computing Service を使用した へのアクセス

を使用して AWS PrivateLink、VPC と AWS Parallel Computing Service () の間にプライベート接続を作成できますAWS PCS。インターネットゲートウェイ、NAT デバイス、VPN 接続、または

Direct Connect 接続を使用せずに、VPC 内にある AWS PCS かのよう にアクセスできます。VPC 内のインスタンスは AWS PCSにアクセスするためにパブリック IP アドレスを必要としません。

このプライベート接続を確立するには、AWS PrivateLinkを利用したインターフェイスエンドポイントを作成します。インターフェイスエンドポイントに対して有効にする各サブネットにエンドポイントネットワークインターフェイスを作成します。これらは、AWS PCS宛てのトラフィックのエントリポイントとして機能するリクエスト管理型ネットワークインターフェイスです。

詳細については、「AWS PrivateLink ガイド」の [「Access AWS のサービス through AWS PrivateLink」](#) を参照してください。

に関する考慮事項 AWS PCS

のインターフェイスエンドポイントを設定する前に AWS PCS、「AWS PrivateLink ガイド」の [「インターフェイス VPC エンドポイントを使用して AWS サービスにアクセスする」](#) を参照してください。

AWS PCS は、インターフェイスエンドポイントを介したすべての API アクションの呼び出しをサポートしています。

VPC に直接インターネットアクセスがない場合は、コンピューティングノードグループインスタンスが [RegisterComputeNodeGroupInstance](#) API アクションを呼び出す AWS PCS ことができるように VPC エンドポイントを設定する必要があります。

のインターフェイスエンドポイントを作成する AWS PCS

Amazon VPC コンソールまたは AWS Command Line Interface () AWS PCS を使用して、のインターフェイスエンドポイントを作成できますAWS CLI。詳細については、「AWS PrivateLink ガイド」の [「インターフェイスエンドポイントを作成」](#) を参照してください。

次のサービス名 AWS PCS を使用して のインターフェイスエンドポイントを作成します。

```
com.amazonaws.region.pcs
```

region を、などのエンドポイント AWS リージョン を作成する の ID に置き換えますus-east-1。

インターフェイスエンドポイントのプライベート DNS を有効にすると、リージョンのデフォルト DNS 名を使用して、AWS PCS への API リクエストを実行できます。例えば、pcs.us-east-1.amazonaws.com。

インターフェイスエンドポイントのエンドポイントポリシーを作成する

エンドポイントポリシーは、インターフェイスエンドポイントにアタッチできる IAM リソースです。デフォルトのエンドポイントポリシーでは、インターフェイスエンドポイント AWS PCS を介した へのフルアクセスが許可されます。VPC AWS PCS から に許可されるアクセスを制御するには、カスタムエンドポイントポリシーをインターフェイスエンドポイントにアタッチします。

エンドポイントポリシーは以下の情報を指定します。

- アクションを実行できるプリンシパル (AWS アカウント、IAM ユーザー、IAM ロール)。
- 実行可能なアクション。
- このアクションを実行できるリソース。

詳細については、AWS PrivateLink ガイドの[Control access to services using endpoint policies \(エンドポイントポリシーを使用してサービスへのアクセスをコントロールする\)](#)を参照してください。

例: AWS PCS アクションの VPC エンドポイントポリシー

以下は、カスタムエンドポイントポリシーの例です。このポリシーをインターフェイスエンドポイントにアタッチすると、指定された *cluster-id* を持つクラスターのすべてのプリンシパルに対して、リスト AWS PCS されたアクションへのアクセスが許可されます。*region* を、などのクラスターの の AWS リージョン ID に置き換えます *us-east-1*。 *account-id* をクラスター AWS アカウント の数に置き換えます。

```
{
  "Statement": [
    {
      "Action": [
        "pcs:CreateCluster",
        "pcs:ListClusters",
        "pcs>DeleteCluster",
        "pcs:GetCluster",
      ],
      "Effect": "Allow",
      "Principal": "*",
      "Resource": [
        "arn:aws:pcs:region:account-id:cluster/cluster-id*"
      ]
    }
  ]
}
```

```
}
```

AWS 並列コンピューティングサービスの Identity and Access Management

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御 AWS のサービス するのに役立つです。IAM 管理者は、誰を認証 (サインイン) し、誰に PCS リソースの使用を許可する (アクセス許可を付与する) AWS かを制御します。IAM は、追加料金なしで利用できる AWS のサービス です。

トピック

- [オーディエンス](#)
- [アイデンティティを使用した認証](#)
- [ポリシーを使用したアクセスの管理](#)
- [AWS Parallel Computing Service と IAM の連携方法](#)
- [AWS Parallel Computing Service のアイデンティティベースのポリシーの例](#)
- [AWS AWS Parallel Computing Service の マネージドポリシー](#)
- [PCS AWS のサービスにリンクされたロール](#)
- [PCS AWS の Amazon EC2 スポットロール](#)
- [PCS AWS の最小アクセス許可](#)
- [AWS Parallel Computing Service の IAM インスタンスプロファイル](#)
- [AWS Parallel Computing Service のアイデンティティとアクセスのトラブルシューティング](#)

オーディエンス

AWS Identity and Access Management (IAM) の使用方法是、ロールによって異なります。

- サービスユーザー - 機能にアクセスできない場合は、管理者にアクセス許可をリクエストします (「[AWS Parallel Computing Service のアイデンティティとアクセスのトラブルシューティング](#)」を参照)。
- サービス管理者 - ユーザーアクセスを決定し、アクセス許可リクエストを送信します (「[AWS Parallel Computing Service と IAM の連携方法](#)」を参照)
- IAM 管理者 - アクセスを管理するためのポリシーを作成します (「[AWS Parallel Computing Service のアイデンティティベースのポリシーの例](#)」を参照)

アイデンティティを使用した認証

認証とは、ID 認証情報 AWS を使用して にサインインする方法です。、IAM ユーザー AWS アカウントのルートユーザー、または IAM ロールを引き受けることで認証される必要があります。

(AWS IAM アイデンティティセンター IAM Identity Center)、シングルサインオン認証、Google/Facebook 認証情報などの ID ソースからの認証情報を使用して、フェデレーテッド ID としてサインインできます。サインインの詳細については、「AWS サインイン ユーザーガイド」の「[AWS アカウントにサインインする方法](#)」を参照してください。

プログラムによるアクセスの場合、は SDK と CLI AWS を提供してリクエストを暗号化して署名します。詳細については、「IAM ユーザーガイド」の「[API リクエストに対するAWS 署名バージョン 4](#)」を参照してください。

AWS アカウント ルートユーザー

を作成するときは AWS アカウント、すべての AWS のサービス および リソースへの完全なアクセス権を持つ AWS アカウント ルートユーザーと呼ばれる 1 つのサインインアイデンティティから始めます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザー認証情報を必要とするタスクについては、「IAM ユーザーガイド」の「[ルートユーザー認証情報が必要なタスク](#)」を参照してください。

フェデレーテッドアイデンティティ

ベストプラクティスとして、人間のユーザーが一時的な認証情報 AWS のサービス を使用して にアクセスするには、ID プロバイダーとのフェデレーションを使用する必要があります。

フェデレーテッド ID は、エンタープライズディレクトリ、ウェブ ID プロバイダー、または ID ソースの認証情報 AWS のサービス を使用して Directory Service にアクセスするユーザーです。フェデレーテッドアイデンティティは、一時的な認証情報を提供するロールを引き受けます。

アクセスを一元管理する場合は、AWS IAM アイデンティティセンターをお勧めします。詳細については、「AWS IAM アイデンティティセンター ユーザーガイド」の「[IAM アイデンティティセンターとは](#)」を参照してください。

IAM ユーザーとグループ

[IAM ユーザー](#)は、特定の個人やアプリケーションに対する特定のアクセス許可を持つアイデンティティです。長期認証情報を持つ IAM ユーザーの代わりに一時的な認証情報を使用することをお勧め

します。詳細については、IAM ユーザーガイドの「[ID プロバイダーとのフェデレーションを使用してにアクセスする必要がある AWS](#)」を参照してください。

[IAM グループ](#)は、IAM ユーザーの集合を指定し、大量のユーザーに対するアクセス許可の管理を容易にします。詳細については、「IAM ユーザーガイド」の「[IAM ユーザーに関するユースケース](#)」を参照してください。

IAM ロール

[IAM ロール](#)は、特定のアクセス許可を持つアイデンティティであり、一時的な認証情報を提供します。[ユーザーから IAM ロール \(コンソール\) に切り替えるか、または API オペレーションを呼び出すことで、ロール](#)を引き受けることができます。AWS CLI AWS 詳細については、「IAM ユーザーガイド」の「[ロールを引き受けるための各種方法](#)」を参照してください。

IAM ロールは、フェデレーションユーザーアクセス、一時的な IAM ユーザーのアクセス許可、クロスアカウントアクセス、クロスサービスアクセス、および Amazon EC2 で実行するアプリケーションに役立ちます。詳細については、IAM ユーザーガイドの [IAM でのクロスアカウントリソースアクセス](#) を参照してください。

ポリシーを使用したアクセスの管理

でアクセスを制御する AWS には、ポリシーを作成し、ID AWS またはリソースにアタッチします。ポリシーは、ID またはリソースに関連付けられたときにアクセス許可を定義します。は、プリンシパルがリクエストを行うときにこれらのポリシー AWS を評価します。ほとんどのポリシーは JSON ドキュメント AWS としてに保存されます。JSON ポリシードキュメントの詳細については、「IAM ユーザーガイド」の「[JSON ポリシー概要](#)」を参照してください。

管理者は、ポリシーを使用して、どのプリンシパルがどのリソースに対して、どのような条件でアクションを実行できるかを定義することで、誰が何にアクセスできるかを指定します。

デフォルトでは、ユーザーやロールにアクセス許可はありません。IAM 管理者は IAM ポリシーを作成してロールに追加し、このロールをユーザーが引き受けられるようにします。IAM ポリシーは、オペレーションの実行方法を問わず、アクセス許可を定義します。

アイデンティティベースのポリシー

アイデンティティベースのポリシーは、アイデンティティ (ユーザー、グループ、またはロール) にアタッチできる JSON アクセス許可ポリシードキュメントです。これらのポリシーは、アイデンティティがどのリソースに対してどのような条件下でどのようなアクションを実行できるかを制御し

ます。アイデンティティベースポリシーの作成方法については、IAM ユーザーガイドの [カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する](#) を参照してください。

アイデンティティベースのポリシーは、インラインポリシー (単一の ID に直接埋め込む) または管理ポリシー (複数の ID にアタッチされたスタンドアロンポリシー) にすることができます。管理ポリシーとインラインポリシーのいずれかを選択する方法については、「IAM ユーザーガイド」の「[管理ポリシーとインラインポリシーのいずれかを選択する](#)」を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。例としては、IAM ロール信頼ポリシーや Amazon S3 バケットポリシーなどがあります。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。リソースベースのポリシーでは、[プリンシパルを指定する](#) 必要があります。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーでは、IAM の AWS マネージドポリシーを使用できません。

その他のポリシータイプ

AWS は、より一般的なポリシータイプによって付与されるアクセス許可の上限を設定できる追加のポリシータイプをサポートしています。

- アクセス許可の境界 – アイデンティティベースのポリシーで IAM エンティティに付与することのできるアクセス許可の数の上限を設定します。詳細については、「IAM ユーザーガイド」の「[IAM エンティティのアクセス許可境界](#)」を参照してください。
- サービスコントロールポリシー (SCP) - AWS Organizations内の組織または組織単位の最大のアクセス許可を指定します。詳細については、「AWS Organizations ユーザーガイド」の「[サービスコントロールポリシー](#)」を参照してください。
- リソースコントロールポリシー (RCP) – は、アカウント内のリソースで利用できる最大数のアクセス許可を定義します。詳細については、「AWS Organizations ユーザーガイド」の「[リソースコントロールポリシー \(RCP\)](#)」を参照してください。
- セッションポリシー – ロールまたはフェデレーションユーザーの一時セッションを作成する際にパラメータとして渡される高度なポリシーです。詳細については、「IAM ユーザーガイド」の「[セッションポリシー](#)」を参照してください。

複数のポリシータイプ

1つのリクエストに複数のタイプのポリシーが適用されると、結果として作成されるアクセス許可を理解するのがさらに難しくなります。が複数のポリシータイプが関与する場合にリクエストを許可するかどうかわかる方法については、「IAM ユーザーガイド」の[「ポリシー評価ロジック」](#)を参照してください。

AWS Parallel Computing Service と IAM の連携方法

IAM を使用して PCS AWS へのアクセスを管理する前に、PCS で使用できる IAM AWS 機能について説明します。

AWS Parallel Computing Service で使用できる IAM 機能

IAM 機能	AWS PCS サポート
アイデンティティベースのポリシー	あり
リソースベースのポリシー	なし
ポリシーアクション	あり
ポリシーリソース	はい
ポリシー条件キー (サービス固有)	はい
ACL	なし
ABAC (ポリシー内のタグ)	あり
一時的な認証情報	あり
プリンシパルアクセス権限	あり
サービスロール	いいえ
サービスリンクロール	はい

AWS PCS およびその他の AWS のサービスがほとんどの IAM 機能と連携する方法の概要については、「IAM ユーザーガイド」の[AWS 「IAM と連携する のサービス」](#)を参照してください。

PCS AWS のアイデンティティベースのポリシー

アイデンティティベースのポリシーのサポート: あり

アイデンティティベースポリシーは、IAM ユーザー、ユーザーグループ、ロールなど、アイデンティティにアタッチできる JSON 許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。アイデンティティベースポリシーの作成方法については、「IAM ユーザーガイド」の「[カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する](#)」を参照してください。

IAM アイデンティティベースのポリシーでは、許可または拒否するアクションとリソース、およびアクションを許可または拒否する条件を指定できます。JSON ポリシーで使用するすべての要素について学ぶには、「IAM ユーザーガイド」の「[IAM JSON ポリシーの要素のリファレンス](#)」を参照してください。

PCS AWS のアイデンティティベースのポリシーの例

AWS PCS アイデンティティベースのポリシーの例を表示するには、「」を参照してください[AWS Parallel Computing Service のアイデンティティベースのポリシーの例](#)。

PCS AWS 内のリソースベースのポリシー

リソースベースのポリシーのサポート: なし

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスをコントロールできます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーで、[プリンシパルを指定する](#)必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、またはを含めることができます AWS のサービス。

クロスアカウントアクセスを有効にするには、全体のアカウント、または別のアカウントの IAM エンティティを、リソースベースのポリシーのプリンシパルとして指定します。詳細については、IAM ユーザーガイドの[IAM でのクロスアカウントリソースアクセス](#)を参照してください。

PCS AWS のポリシーアクション

ポリシーアクションのサポート: あり

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

JSON ポリシーの Action 要素にはポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。このアクションは関連付けられたオペレーションを実行するためのアクセス許可を付与するポリシーで使用されます。

AWS PCS アクションのリストを確認するには、「サービス認可リファレンス」の[AWS 「Parallel Computing Service」で定義されるアクション](#)」を参照してください。

PCS AWS のポリシーアクションは、アクションの前に次のプレフィックスを使用します。

```
pcs
```

単一のステートメントで複数のアクションを指定するには、アクションをカンマで区切ります。

```
"Action": [
    "pcs:action1",
    "pcs:action2"
]
```

PCS AWS のポリシーリソース

ポリシーリソースのサポート: あり

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

Resource JSON ポリシー要素はアクションが適用されるオブジェクトを指定します。ベストプラクティスとして、[Amazon リソースネーム \(ARN\)](#) を使用してリソースを指定します。リソースレベルのアクセス許可をサポートしないアクションの場合は、ステートメントがすべてのリソースに適用されることを示すために、ワイルドカード (*) を使用します。

```
"Resource": "*" 
```

AWS PCS リソースタイプとその ARNs [AWS 「並列コンピューティングサービスで定義されるリソース」](#) を参照してください。各リソースの ARN を指定できるアクションについては、[AWS 「Parallel Computing Service」で定義されるアクション](#)」を参照してください。

AWS PCS アイデンティティベースのポリシーの例を表示するには、「」を参照してください[AWS Parallel Computing Service のアイデンティティベースのポリシーの例](#)。

PCS AWS のポリシー条件キー

サービス固有のポリシー条件キーのサポート: あり

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

Condition 要素は、定義された基準に基づいてステートメントが実行される時期を指定します。イコールや未満などの[条件演算子](#)を使用して条件式を作成して、ポリシーの条件とリクエスト内の値を一致させることができます。すべての AWS グローバル条件キーを確認するには、「IAM ユーザーガイド」の[AWS 「グローバル条件コンテキストキー」](#)を参照してください。

AWS PCS 条件キーのリストを確認するには、「サービス認可リファレンス」の[AWS 「並列コンピューティングサービスの条件キー」](#)を参照してください。条件キーを使用できるアクションとリソースについては、[AWS 「Parallel Computing Service で定義されるアクション」](#)を参照してください。

AWS PCS アイデンティティベースのポリシーの例を表示するには、「」を参照してください[AWS Parallel Computing Service のアイデンティティベースのポリシーの例](#)。

PCS AWS ACLs

ACL のサポート: なし

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするためのアクセス許可を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

PCS AWS での ABAC

ABAC (ポリシー内のタグ) のサポート: あり

属性ベースのアクセス制御 (ABAC) は、タグと呼ばれる属性に基づいてアクセス許可を定義する認可戦略です。IAM エンティティと AWS リソースにタグをアタッチし、プリンシパルのタグがリソースのタグと一致するときにオペレーションを許可するように ABAC ポリシーを設計できます。

タグに基づいてアクセスを管理するには、aws:ResourceTag/*key-name*、aws:RequestTag/*key-name*、または aws:TagKeys の条件キーを使用して、ポリシーの[条件要素](#)でタグ情報を提供します。

サービスがすべてのリソースタイプに対して 3 つの条件キーすべてをサポートする場合、そのサービスの値はありです。サービスが一部のリソースタイプに対してのみ 3 つの条件キーのすべてをサポートする場合、値は「部分的」になります。

ABAC の詳細については、「IAM ユーザーガイド」の「[ABAC 認可でアクセス許可を定義する](#)」を参照してください。ABAC をセットアップする手順を説明するチュートリアルについては、「IAM ユーザーガイド」の「[属性ベースのアクセスコントロール \(ABAC\) を使用する](#)」を参照してください。

PCS AWS での一時的な認証情報の使用

一時的な認証情報のサポート: あり

一時的な認証情報は、AWS リソースへの短期的なアクセスを提供し、フェデレーションまたは切り替えロールの使用時に自動的に作成されます。長期的なアクセスキーを使用する代わりに、一時的な認証情報を動的に生成 AWS することをお勧めします。詳細については、「IAM ユーザーガイド」の「[IAM の一時的な認証情報](#)」および「[AWS のサービス と IAM との連携](#)」を参照してください。

PCS AWS のクロスサービスプリンシパルのアクセス許可

転送アクセスセッション (FAS) のサポート: あり

転送アクセスセッション (FAS) は、 を呼び出すプリンシパルのアクセス許可と AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストをリクエストする を使用します。FAS リクエストを行う際のポリシーの詳細については、「[転送アクセスセッション](#)」を参照してください。

PCS AWS のサービスロール

サービスロールのサポート: なし

サービスロールとは、サービスがユーザーに代わってアクションを実行するために引き受ける [IAM ロール](#)です。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、IAM ユーザーガイドの [AWS のサービスに許可を委任するロールを作成する](#)を参照してください。

⚠ Warning

サービスロールのアクセス許可を変更すると、PCS AWS 機能が破損する可能性があります。PCS AWS が指示する場合にのみ、サービスロールを編集します。

PCS AWS のサービスにリンクされたロール

サービスリンクロールのサポート: あり

サービスにリンクされたロールは、にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント、サービスによって所有されます。IAM 管理者は、サービスリンクロールのアクセス許可を表示できますが、編集することはできません。

PCS サービスにリンクされたロールの作成または管理の詳細については、AWS 「」を参照してください [PCS AWS のサービスにリンクされたロール](#)。

AWS Parallel Computing Service のアイデンティティベースのポリシーの例

デフォルトでは、ユーザーとロールには PCS AWS リソースを作成または変更するアクセス許可はありません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。

これらのサンプルの JSON ポリシードキュメントを使用して IAM アイデンティティベースのポリシーを作成する方法については、「IAM ユーザーガイド」の「[IAM ポリシーを作成する \(コンソール\)](#)」を参照してください。

各リソースタイプの ARN の形式など、PCS AWS で定義されるアクションとリソースタイプの詳細については、「サービス認可リファレンス」の [AWS 「並列コンピューティングサービスのアクション、リソース、および条件キー」](#) を参照してください。ARNs

トピック

- [ポリシーに関するベストプラクティス](#)
- [PCS AWS コンソールの使用](#)
- [自分の権限の表示をユーザーに許可する](#)

ポリシーに関するベストプラクティス

アイデンティティベースのポリシーは、アカウント内で誰かが PCS AWS リソースを作成、アクセス、または削除できるかどうかを決定します。これらのアクションでは、AWS アカウントに費用が発生する場合があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

- AWS 管理ポリシーを開始し、最小特権のアクセス許可に移行する – ユーザーとワークロードにアクセス許可の付与を開始するには、多くの一般的なユースケースにアクセス許可を付与する AWS 管理ポリシーを使用します。これらはで利用できます AWS アカウント。ユースケースに固有の AWS カスタマー管理ポリシーを定義することで、アクセス許可をさらに減らすことをお勧めします。詳細については、IAM ユーザーガイドの [AWS マネージドポリシー](#) または [ジョブ機能のAWS マネージドポリシー](#) を参照してください。
- 最小特権を適用する – IAM ポリシーでアクセス許可を設定する場合は、タスクの実行に必要な許可のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権アクセス許可とも呼ばれています。IAM を使用して許可を適用する方法の詳細については、IAM ユーザーガイドの [IAM でのポリシーとアクセス許可](#) を参照してください。
- IAM ポリシーで条件を使用してアクセスをさらに制限する – ポリシーに条件を追加して、アクションやリソースへのアクセスを制限できます。たとえば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。条件を使用して、サービスアクションがなどの特定のを通じて使用されている場合に AWS のサービス、サービスアクションへのアクセスを許可することもできます CloudFormation。詳細については、IAM ユーザーガイドの [IAM JSON ポリシー要素:条件](#) を参照してください。
- IAM アクセスアナライザーを使用して IAM ポリシーを検証し、安全で機能的な権限を確保する – IAM アクセスアナライザーは、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語 (JSON) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、IAM ユーザーガイドの [IAM Access Analyzer でポリシーを検証する](#) を参照してください。
- 多要素認証 (MFA) を要求する – IAM ユーザーまたはルートユーザーを必要とするシナリオがある場合は AWS アカウント、MFA をオンにしてセキュリティを強化します。API オペレーションが呼び出されるときに MFA を必須にするには、ポリシーに MFA 条件を追加します。詳細については、IAM ユーザーガイドの [MFA を使用した安全な API アクセス](#) を参照してください。

IAM でのベストプラクティスの詳細については、IAM ユーザーガイドの [IAM でのセキュリティのベストプラクティス](#) を参照してください。

PCS AWS コンソールの使用

AWS Parallel Computing Service コンソールにアクセスするには、最小限のアクセス許可が必要です。これらのアクセス許可により、の PCS AWS リソースの詳細を一覧表示および表示できます AWS アカウント。最小限必要な許可よりも制限が厳しいアイデンティティベースのポリシーを作成すると、そのポリシーを持つエンティティ (ユーザーまたはロール) に対してコンソールが意図したとおりに機能しません。

AWS CLI または AWS API のみを呼び出すユーザーには、最小限のコンソールアクセス許可を付与する必要はありません。代わりに、実行しようとしている API オペレーションに一致するアクションのみへのアクセスが許可されます。

PCS コンソールを使用するために必要な最小限のアクセス許可の詳細については、AWS 「」を参照してください [PCS AWS の最小アクセス許可](#)。

自分の権限の表示をユーザーに許可する

この例では、ユーザーアイデンティティにアタッチされたインラインおよびマネージドポリシーの表示を IAM ユーザーに許可するポリシーの作成方法を示します。このポリシーには、コンソールで、または AWS CLI または AWS API を使用してプログラムでこのアクションを実行するアクセス許可が含まれています。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupsForUser",
        "iam:ListAttachedUserPolicies",
        "iam:ListUserPolicies",
        "iam:GetUser"
      ],
      "Resource": ["arn:aws:iam::*:user/${aws:username}"]
    }
  ],
  {
```

```
    "Sid": "NavigateInConsole",
    "Effect": "Allow",
    "Action": [
        "iam:GetGroupPolicy",
        "iam:GetPolicyVersion",
        "iam:GetPolicy",
        "iam:ListAttachedGroupPolicies",
        "iam:ListGroupPolicies",
        "iam:ListPolicyVersions",
        "iam:ListPolicies",
        "iam:ListUsers"
    ],
    "Resource": "*"
  }
]
```

AWS AWS Parallel Computing Service の マネージドポリシー

AWS 管理ポリシーは、によって作成および管理されるスタンドアロンポリシーです AWS。AWS 管理ポリシーは、ユーザー、グループ、ロールにアクセス許可の割り当てを開始できるように、多くの一般的なユースケースにアクセス許可を付与するように設計されています。

AWS 管理ポリシーは、すべての AWS お客様が使用できるため、特定のユースケースに対して最小特権のアクセス許可を付与しない場合があります。ユースケースに固有の[カスタマー管理ポリシー](#)を定義して、アクセス許可を絞り込むことをお勧めします。

AWS 管理ポリシーで定義されているアクセス許可は変更できません。が AWS マネージドポリシーで定義されたアクセス許可 AWS を更新すると、ポリシーがアタッチされているすべてのプリンシパル ID (ユーザー、グループ、ロール) に影響します。AWS は、新しい が起動されるか、新しい API オペレーション AWS のサービス が既存のサービスで使用できるようになったときに、AWS マネージドポリシーを更新する可能性が最も高くなります。

詳細については、「IAM ユーザーガイド」の「[AWS マネージドポリシー](#)」を参照してください。

AWS マネージドポリシー: AWSPCSCComputeNodePolicy

AWSPCSCComputeNodePolicy を IAM エンティティにアタッチできます。このポリシーは、指定した AWS PCS コンピューティングノード IAM ロールにアタッチして、そのロールを使用するノードが PCS AWS クラスターに接続できるようにします。

AWS コンソールを使用してコンピューティングノードグループを作成すると、PCS はこのポリシーをコンピューティングノードグループロールにアタッチします。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- pcs:RegisterComputeNodeGroupInstance – PCS コンピューティングノード (EC2 インスタンス) AWS が PCS AWS クラスターに登録できるようにします。

このポリシーのアクセス許可を確認するには、「AWS マネージドポリシーリファレンス」の[AWSPCSCComputeNodePolicy](#)を参照してください。

AWS マネージドポリシー: AWSPCSServiceRolePolicy

AWSPCSServiceRolePolicy を IAM エンティティにアタッチすることはできません。このポリシーは、PCS AWS がユーザーに代わってアクションを実行できるようにするサービスにリンクされた

ロールにアタッチされます。詳細については、「[PCS AWS のサービスにリンクされたロール](#)」を参照してください。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- ec2 – PCS AWS が Amazon EC2 リソースを作成および管理できるようにします。
- iam – PCS AWS が Amazon EC2 フリートのサービスにリンクされたロールを作成し、ロールを Amazon EC2 に渡すことを許可します。
- cloudwatch – PCS AWS が Amazon CloudWatch にサービスメトリクスを発行できるようにします。
- secretsmanager – PCS が AWS PCS AWS クラスターリソースのシークレットを管理できるようにします。

このポリシーのアクセス許可を確認するには、「AWS マネージドポリシーリファレンス」の[AWSPCSServiceRolePolicy](#)」を参照してください。

AWS AWS 管理ポリシーの PCS 更新

このサービスがこれらの変更の追跡を開始してからの PCS AWS の AWS 管理ポリシーの更新に関する詳細を表示します。このページの変更に関する自動アラートについては、PCS ドキュメント履歴ページの RSS AWS フィードにサブスクライブしてください。

変更	説明	日付
AWSPCSServiceRolePolicy – 既存のポリシーの更新	AWS PCS は、予測可能なコンピューティング容量のキャパシティブロックをサポートする新しいアクセス許可を追加しました。 PCS AWS がコンピューティングノードグループのキャパシティブロック予約を検出して使用できるようにするec2:DescribeCapaci	2025 年 9 月 11 日

変更	説明	日付
	tyReservations アクセス許可を追加しました。	
AWSPCSComputeNodePolicy – 新しいポリシー	AWS PCS は、PCS AWS クラスターに接続するためのアクセス許可を PCS AWS コンピューティングノードに付与する新しいポリシーを追加しました。 AWS PCS コンソールでコンピューティングノードグループを作成すると、PCS はこのポリシーを IAM AWS ロールにアタッチします。	2025 年 6 月 23 日
このドキュメントの JSON を更新しました	このドキュメントの JSON を修正して、を含めました"arn:aws:ec2:*:*:spot-instances-request/*" 。	2024 年 9 月 5 日
AWS PCS が変更の追跡を開始しました	AWS PCS は、AWS 管理ポリシーの変更の追跡を開始しました。	2024 年 8 月 28 日

PCS AWS のサービスにリンクされたロール

AWS Parallel Computing Service は AWS Identity and Access Management 、 (IAM) [サービスにリンクされたロール](#)を使用します。サービスにリンクされたロールは、PCS に直接リンクされた一意のタイプの IAM AWS ロールです。サービスにリンクされたロールは PCS AWS によって事前定義されており、サービスがユーザーに代わって他の AWS サービスを呼び出すために必要なすべてのアクセス許可が含まれています。

サービスにリンクされたロールを使用すると、必要なアクセス許可を手動で追加する必要がなくなるため、PCS AWS の設定が簡単になります。AWS PCS はサービスにリンクされたロールのアクセ

ス許可を定義し、特に定義されている場合を除き、PCS AWS のみがそのロールを引き受けることができます。定義されるアクセス許可には、信頼ポリシーと許可ポリシーが含まれており、その許可ポリシーを他の IAM エンティティにアタッチすることはできません。

サービスリンクロールを削除するには、まずその関連リソースを削除します。これにより AWS、リソースへのアクセス許可を誤って削除できないため、PCS リソースが保護されます。

サービスにリンクされたロールをサポートする他のサービスの詳細については、[AWS「IAM と連携するサービス」](#)を参照し、「サービスにリンクされたロール」列で「はい」があるサービスを探します。サービスリンクロールに関するドキュメントをサービスで表示するには、リンクで [はい] を選択します。

PCS AWS のサービスにリンクされたロールのアクセス許可

AWS PCS は、AWSServiceRoleForPCS という名前のサービスにリンクされたロールを使用します – Amazon EC2 リソースを管理するアクセス許可を AWS PCS に付与します。

AWSServiceRoleForPCS サービスにリンクされたロールは、次のサービスを信頼してロールを引き受けます。

- `pcs.amazonaws.com`

[AWSPCSServiceRolePolicy](#) という名前のロールアクセス許可ポリシーにより、PCS AWS は特定のリソースに対するアクションを完了できます。

ユーザー、グループ、またはロールにサービスリンクロールの作成、編集、または削除を許可するには、アクセス許可を設定する必要があります。詳細についてはIAM ユーザーガイドの「[サービスにリンクされた役割のアクセス許可](#)」を参照してください。

PCS AWS のサービスにリンクされたロールの作成

サービスにリンクされたロールを手動で作成する必要はありません。AWS PCS は、クラスターの作成時にサービスにリンクされたロールを作成します。

PCS AWS のサービスにリンクされたロールの編集

AWS PCS では、AWSServiceRoleForPCS サービスにリンクされたロールを編集することはできません。サービスリンクロールの作成後は、さまざまなエンティティがロールを参照する可能性があるため、ロール名を変更することはできません。ただし、IAM を使用してロールの説明を編集することはできます。詳細については、「IAM ユーザーガイド」の「[サービスリンクロールの編集](#)」を参照してください。

PCS AWS のサービスにリンクされたロールの削除

サービスリンクロールを必要とする機能やサービスが不要になった場合は、ロールを削除することをお勧めします。そうすることで、積極的にモニタリングまたは保守されていない未使用のエンティティを排除できます。ただし、手動で削除する前に、サービスリンクロールのリソースをクリーンアップする必要があります。

Note

リソースを削除しようとしたときに PCS AWS サービスがロールを使用している場合、削除が失敗する可能性があります。失敗した場合は数分待ってから操作を再試行してください。

AWSServiceRoleForPCS AWS で使用される PCS リソースを削除するには

AWSServiceRoleForPCS サービスにリンクされたロールを削除するには、すべてのクラスターを削除する必要があります。詳細については、[「クラスターの削除」](#)を参照してください。

サービスリンクロールを IAM で手動削除するには

IAM コンソール、AWS CLI、または AWS API を使用して、AWSServiceRoleForPCS サービスにリンクされたロールを削除します。詳細については、「IAM ユーザーガイド」の[「サービスにリンクされたロールの削除」](#)を参照してください。

PCS AWS サービスにリンクされたロールでサポートされているリージョン

AWS PCS は、サービスが利用可能なすべてのリージョンでサービスにリンクされたロールの使用をサポートしています。詳細については、[「AWS リージョンとエンドポイント」](#)を参照してください。

PCS AWS の Amazon EC2 スポットロール

スポットを購入オプションとして使用する AWS PCS コンピューティングノードグループを作成する場合は、に AWSServiceRoleForEC2Spot サービスにリンクされたロールも必要です AWS アカウント。次の AWS CLI コマンドを使用してロールを作成できます。詳細については、[「ユーザーガイド」の「サービスにリンクされたロールを作成する」](#)および[「AWS サービスにアクセス許可を委任するロールを作成する」](#)を参照してください。AWS Identity and Access Management

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

 Note

に AWS アカウント 既に IAM AWSServiceRoleForEC2Spot ロールがある場合、次のエラーが表示されます。

```
An error occurred (InvalidInput) when calling the CreateServiceLinkedRole operation: Service role name AWSServiceRoleForEC2Spot has been taken in this account, please try a different suffix.
```

PCS AWS の最小アクセス許可

このセクションでは、IAM ID (ユーザー、グループ、またはロール) がサービスを使用するために必要な最小限の IAM アクセス許可について説明します。

目次

- [API アクションを使用するための最小限のアクセス許可](#)
- [タグを使用するための最小限のアクセス許可](#)
- [ログをサポートする最小アクセス許可](#)
- [キャパシティブロックを使用するための最小限のアクセス許可](#)
- [サービス管理者の最小アクセス許可](#)

API アクションを使用するための最小限のアクセス許可

API アクション	最小アクセス許可	コンソールに対する追加のアクセス許可
CreateCluster	ec2:CreateNetworkInterface, ec2:DescribeVpcs, ec2:DescribeSubnets, ec2:DescribeSecurityGroups, ec2:GetSecurityGroupsForVpc,	

API アクション	最小アクセス許可	コンソールに対する追加のアクセス許可
	iam:CreateServiceLinkedRole, secretsmanager:CreateSecret, secretsmanager:TagResource, secretsmanager:RotateSecret, pcs:CreateCluster	
ListClusters	pcs:ListClusters	
GetCluster	pcs:GetCluster	ec2:DescribeSubnets
DeleteCluster	pcs>DeleteCluster	
CreateComputeNodeGroup	ec2:DescribeVpcs, ec2:DescribeSubnets, ec2:DescribeSecurityGroups, ec2:DescribeLaunchTemplates, ec2:DescribeLaunchTemplateVersions, ec2:DescribeInstanceTypes, ec2:DescribeInstanceTypeOfferings, ec2:RunInstances, ec2:CreateFleet, ec2:CreateTags, iam:PassRole, iam:GetInstanceProfile, pcs:CreateComputeNodeGroup	iam:ListInstanceProfiles, ec2:DescribeImages, pcs:GetCluster

API アクション	最小アクセス許可	コンソールに対する追加のアクセス許可
ListComputerNodeGroups	<code>pcs:ListComputeNodeGroups</code>	<code>pcs:GetCluster</code>
GetComputeNodeGroup	<code>pcs:GetComputeNodeGroup</code>	<code>ec2:DescribeSubnets</code>
UpdateComputeNodeGroup	<code>ec2:DescribeVpcs,</code> <code>ec2:DescribeSubnets,</code> <code>ec2:DescribeSecurityGroups,</code> <code>ec2:DescribeLaunchTemplates,</code> <code>ec2:DescribeLaunchTemplateVersions,</code> <code>ec2:DescribeInstanceTypes,</code> <code>ec2:DescribeInstanceTypeOfferings,</code> <code>ec2:RunInstances,</code> <code>ec2:CreateFleet,</code> <code>ec2:CreateTags,</code> <code>iam:PassRole,</code> <code>iam:GetInstanceProfile,</code> <code>pcs:UpdateComputeNodeGroup</code>	<code>pcs:GetComputeNodeGroup,</code> <code>iam:ListInstanceProfiles,</code> <code>ec2:DescribeImages,</code> <code>pcs:GetCluster</code>
DeleteComputeNodeGroup	<code>pcs>DeleteComputeNodeGroup</code>	
CreateQueue	<code>pcs>CreateQueue</code>	<code>pcs:ListComputeNodeGroups,</code> <code>pcs:GetCluster</code>

API アクション	最小アクセス許可	コンソールに対する追加のアクセス許可
ListQueues	<code>pcs:ListQueues</code>	<code>pcs:GetCluster</code>
GetQueue	<code>pcs:GetQueue</code>	
UpdateQueue	<code>pcs:UpdateQueue</code>	<code>pcs:ListComputeNodesGroups,</code> <code>pcs:GetQueue</code>
DeleteQueue	<code>pcs>DeleteQueue</code>	

タグを使用するための最小限のアクセス許可

AWS PCS のリソースでタグを使用するには、次のアクセス許可が必要です。

```
pcs:ListTagsForResource,
pcs:TagResource,
pcs:UntagResource
```

ログをサポートする最小アクセス許可

AWS PCS は Amazon CloudWatch Logs (CloudWatch Logs) にログデータを送信します。ID に CloudWatch Logs を使用するための最小限のアクセス許可があることを確認する必要があります。詳細については、「Amazon [CloudWatch Logs ユーザーガイド](#)」の「[CloudWatch Logs リソースへのアクセス許可の管理の概要](#)」を参照してください。Amazon CloudWatch

サービスが CloudWatch Logs にログを送信するために必要なアクセス許可については、「Amazon CloudWatch Logs ユーザーガイド」の[AWS 「サービスからのログ記録の有効化」](#)を参照してください。Amazon CloudWatch

キャパシティブロックを使用するための最小限のアクセス許可

Amazon EC2 Capacity Blocks for ML は、短期間のワークロードをサポートするために、特定の日時範囲内で GPU ベースの高速コンピューティングインスタンスを事前に予約するための Amazon

EC2 購入オプションです。詳細については、「[PCS での ML での Amazon EC2 AWS キャパシティブロックの使用](#)」を参照してください。

コンピューティングノードグループを作成または更新するときに、キャパシティブロックを使用することを選択します。コンピューティングノードグループを作成または更新するために使用する IAM ID には、次のアクセス許可が必要です。

```
ec2:DescribeCapacityReservations
```

サービス管理者の最小アクセス許可

次の IAM ポリシーは、IAM ID (ユーザー、グループ、またはロール) が PCS AWS サービスを設定および管理するために必要な最小限のアクセス許可を指定します。

Note

サービスを設定および管理しないユーザーには、これらのアクセス許可は必要ありません。ジョブのみを実行するユーザーは、セキュアシェル (SSH) を使用してクラスターに接続します。AWS Identity and Access Management (IAM) は SSH の認証または認可を処理しません。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PCSAccess",
      "Effect": "Allow",
      "Action": [
        "pcs:*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "EC2Access",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface",
        "ec2:DescribeImages",
        "ec2:GetSecurityGroupsForVpc",
        "ec2:DescribeSubnets",
```

```

        "ec2:DescribeSecurityGroups",
        "ec2:DescribeVpcs",
        "ec2:DescribeLaunchTemplates",
        "ec2:DescribeLaunchTemplateVersions",
        "ec2:DescribeInstanceTypes",
        "ec2:DescribeInstanceTypeOfferings",
        "ec2:RunInstances",
        "ec2:CreateFleet",
        "ec2:CreateTags",
        "ec2:DescribeCapacityReservations"
    ],
    "Resource": "*"
},
{
    "Sid": "IamInstanceProfile",
    "Effect": "Allow",
    "Action": [
        "iam:GetInstanceProfile"
    ],
    "Resource": "*"
},
{
    "Sid": "IamPassRole",
    "Effect": "Allow",
    "Action": [
        "iam:PassRole"
    ],
    "Resource": [
        "arn:aws:iam::*:role/*/AWSPCS*",
        "arn:aws:iam::*:role/AWSPCS*",
        "arn:aws:iam::*:role/aws-pcs/*",
        "arn:aws:iam::*:role/*/aws-pcs*"
    ],
    "Condition": {
        "StringEquals": {
            "iam:PassedToService": [
                "ec2.amazonaws.com"
            ]
        }
    }
},
{
    "Sid": "SLRAccess",
    "Effect": "Allow",

```

```

    "Action": [
      "iam:CreateServiceLinkedRole"
    ],
    "Resource": [
      "arn:aws:iam::*:role/aws-service-role/pcs.amazonaws.com/AWSServiceRoleFor*",
      "arn:aws:iam::*:role/aws-service-role/spot.amazonaws.com/AWSServiceRoleFor*"
    ],
    "Condition": {
      "StringLike": {
        "iam:AWSServiceName": [
          "pcs.amazonaws.com",
          "spot.amazonaws.com"
        ]
      }
    }
  },
  {
    "Sid": "AccessKMSKey",
    "Effect": "Allow",
    "Action": [
      "kms:Decrypt",
      "kms:Encrypt",
      "kms:GenerateDataKey",
      "kms:CreateGrant",
      "kms:DescribeKey"
    ],
    "Resource": "*"
  },
  {
    "Sid": "SecretManagementAccess",
    "Effect": "Allow",
    "Action": [
      "secretsmanager:CreateSecret",
      "secretsmanager:TagResource",
      "secretsmanager:UpdateSecret",
      "secretsmanager:RotateSecret"
    ],
    "Resource": "*"
  },
  {
    "Sid": "ServiceLogsDelivery",
    "Effect": "Allow",
    "Action": [
      "pcs:AllowVendedLogDeliveryForResource",

```

```
        "logs:PutDeliverySource",
        "logs:PutDeliveryDestination",
        "logs:CreateDelivery"
    ],
    "Resource": "*"
}
]
```

AWS Parallel Computing Service の IAM インスタンスプロファイル

EC2 インスタンスで実行されるアプリケーションは、作成する AWS API リクエストに AWS 認証情報を含める必要があります。EC2 インスタンスの一時的な認証情報を管理するには、IAM ロールを使用することをお勧めします。これを行うインスタンスプロファイルを定義し、インスタンスにアタッチできます。詳細については、[Amazon EC2 の IAM ロール](#) を参照してください。

Note

を使用して Amazon EC2 の IAM ロール AWS マネジメントコンソール を作成すると、コンソールによってインスタンスプロファイルが自動的に作成され、IAM ロールと同じ名前が付けられます。AWS CLI、AWS API アクション、または AWS SDK を使用して IAM ロールを作成する場合は、別のアクションとしてインスタンスプロファイルを作成します。詳細については、「Amazon Elastic Compute Cloud ユーザーガイド」の「[インスタンスプロファイル](#)」を参照してください。

コンピューティングノードグループを作成するときは、インスタンスプロファイルの Amazon リソースネーム (ARN) を指定する必要があります。コンピューティングノードグループの一部またはすべてに異なるインスタンスプロファイルを選択できます。

要件

インスタンスプロファイルの IAM ロール

インスタンスプロファイルに関連付けられた IAM ロールは、パス `/aws-pcs/` に含まれているか、名前が `g` で始まる必要があります AWSPCS。

IAM ロール ARNs の例

- `arn:aws:iam::*:role/AWSPCS-example-role-1`

- `arn:aws:iam::*:role/aws-pcs/example-role-2`

アクセス許可

PCS のインスタンスプロファイルに関連付けられた IAM AWS ロールには、次のポリシーを含める必要があります。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "pcs:RegisterComputeNodeGroupInstance"
      ],
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}
```

追加ポリシー

インスタンスプロファイルに 管理ポリシーを追加することを検討してください。例えば、次のようになります。

- [AmazonS3ReadOnlyAccess](#) は、すべての S3 バケットへの読み取り専用アクセスを提供します。
- [AmazonSSMManagedInstanceCore](#) は、Amazon マネジメントコンソールから直接のリモートアクセスなど、AWS Systems Manager のサービスコア機能を有効にします。
- [CloudWatchAgentServerPolicy](#) には、サーバーで AmazonCloudWatchAgent を使用するために必要なアクセス許可が含まれています。

特定のユースケースをサポートする独自の IAM ポリシーを含めることもできます。

PCS AWS のインスタンスプロファイルを作成する

AWS PCS console

コンピューティングノードグループを作成するときに基本プロファイルを作成するを選択し、必要最小限のポリシーで PCS AWS に作成させます。

Amazon EC2 console

インスタンスプロファイルは、Amazon EC2 コンソールから直接作成できます。詳細については、「AWS Identity and Access Management ユーザーガイド」の[「インスタンスプロファイルの使用」](#)を参照してください。

Important

IAM ロール名AWSPCSには、必ず必要なプレフィックスを使用してください。

AWS CLI

AWS CLI を使用した基本的なインスタンスプロファイルの設定

Note

次の例の *example-role* を IAM ロールの名前に置き換えます。

1. をパス属性として、または で始まる名前/aws-pcs/として IAM ロールを作成しますAWSPCS。
 - a. 次のコンテンツをコピーして、 という名前の新しいテキストファイルに貼り付けますtrust_policy.json。

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

```
        ],
      },
      "Action": [
        "sts:AssumeRole"
      ]
    }
  ]
}
```

- b. IAM ロールを作成するには、次のいずれかのコマンドを使用します。

```
aws iam create-role --path /aws-pcs/ --role-name example-role --assume-role-policy-document file://trust_policy.json
```

または

```
aws iam create-role --role-name AWSPCS-example-role --assume-role-policy-document file://trust_policy.json
```

2. アクセス許可をアタッチします。

- a. 次のコンテンツをコピーして、という名前の新しいテキストファイルに貼り付けます `policy_document.json`。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "pcs:RegisterComputeNodeGroupInstance"
      ],
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}
```

- b. ポリシードキュメントをロールにアタッチします。このコマンドは、ポリシーをインラインポリシーとしてアタッチします。

```
aws iam put-role-policy \
```

```
--role-name example-role \  
--policy-name pcsRegisterInstancePolicy \  
--policy-document file://policy_document.json
```

3. インスタンスプロファイルを作成します。*example-profile* をインスタンスプロファイルの名前に置き換えます。

```
aws iam create-instance-profile --instance-profile-name example-profile
```

4. IAM ロールをインスタンスプロファイルに関連付けます。

```
aws iam add-role-to-instance-profile \  
--instance-profile-name example-profile \  
--role-name example-role
```

PCS AWS で使用されるインスタンスプロファイルを検索する

1. PCS の IAM ロールの正確な名前がわからない場合は、次の AWS CLI コマンドを使用して、PCS AWS 名要件を満たす IAM AWS ロールを一覧表示します。

```
aws iam list-roles --query "Roles[?starts_with(RoleName, 'AWSPCS')] |  
contains(Path, '/aws-pcs/')].[RoleName]" --output text
```

2. 次の AWS CLI コマンドを使用して、特定の IAM ロールに関連付けられたインスタンスプロファイルを一覧表示します。*role-name* を、PCS 名要件を満たす IAM AWS ロールの名前に置き換えます。

```
aws iam list-instance-profiles-for-role --role-name role-name
```

AWS Parallel Computing Service のアイデンティティとアクセスのトラブルシューティング

以下の情報は、AWS PCS と IAM の使用時に発生する可能性がある一般的な問題の診断と修正に役立ちます。

トピック

- [PCS AWS でアクションを実行する権限がない](#)

- [iam:PassRole を実行する権限がありません](#)
- [自分の 以外のユーザーに PCS AWS リソース AWS アカウント へのアクセスを許可したい](#)

PCS AWS でアクションを実行する権限がない

アクションを実行する権限がないというエラーが表示された場合は、そのアクションを実行できるようにポリシーを更新する必要があります。

次のエラー例は、mateojackson IAM ユーザーがコンソールを使用して、ある *my-example-widget* リソースに関する詳細情報を表示しようとしたことを想定して、その際に必要な pcs:GetWidget アクセス許可を持っていない場合に発生するものです。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
pcs:GetWidget on resource: my-example-widget
```

この場合、pcs:GetWidget アクションを使用して *my-example-widget* リソースへのアクセスを許可するように、mateojackson ユーザーのポリシーを更新する必要があります。

サポートが必要な場合は、AWS 管理者にお問い合わせください。サインイン認証情報を提供した担当者が管理者です。

iam:PassRole を実行する権限がありません

iam:PassRole アクションを実行する権限がないというエラーが表示された場合は、ポリシーを更新して PCS AWS にロールを渡すことができるようにする必要があります。

一部の AWS のサービスでは、新しいサービスロールまたはサービスにリンクされたロールを作成する代わりに、既存のロールをそのサービスに渡すことができます。そのためには、サービスにロールを渡すアクセス許可が必要です。

次の例のエラーは、という IAM ユーザーがコンソールを使用して marymajor AWS PCS でアクションを実行しようとするると発生します。ただし、このアクションをサービスが実行するには、サービスロールから付与されたアクセス許可が必要です。Mary には、ロールをサービスに渡すアクセス許可がありません。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

この場合、Mary のポリシーを更新してメアリーに iam:PassRole アクションの実行を許可する必要があります。

サポートが必要な場合は、AWS 管理者にお問い合わせください。サインイン資格情報を提供した担当者が管理者です。

自分の 以外のユーザーに PCS AWS リソース AWS アカウント へのアクセスを許可したい

他のアカウントのユーザーや組織外の人、リソースにアクセスするために使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまたはアクセスコントロールリスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください:

- PCS がこれらの機能をサポートしているかどうかを確認するには、AWS 「」を参照してください [AWS Parallel Computing Service と IAM の連携方法](#)。
- 所有 AWS アカウント している のリソースへのアクセスを提供する方法については、「[IAM ユーザーガイド](#)」の「[所有 AWS アカウント している別の の IAM ユーザーへのアクセスを提供する](#)」を参照してください。
- リソースへのアクセスをサードパーティーに提供する方法については AWS アカウント、IAM ユーザーガイドの「[サードパーティーが所有する へのアクセスを提供する AWS アカウント](#)」を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、IAM ユーザーガイドの [外部で認証されたユーザー \(ID フェデレーション\) へのアクセスの許可](#) を参照してください。
- クロスアカウントアクセスにおけるロールとリソースベースのポリシーの使用方法の違いについては、IAM ユーザーガイドの [IAM でのクロスアカウントのリソースへのアクセス](#) を参照してください。

AWS Parallel Computing Service のコンプライアンス検証

AWS のサービス が特定のコンプライアンスプログラムの範囲内にあるかどうかを確認するには、「[コンプライアンスAWS のサービス プログラムによるスコープ](#)」の「コンプライアンス」を参照して、関心のあるコンプライアンスプログラムを選択します。一般的な情報については、[AWS 「コンプライアンスプログラム」](#)を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、「[Downloading Reports in AWS Artifact](#)」を参照してください。

を使用する際のお客様のコンプライアンス責任 AWS のサービスは、お客様のデータの機密性、貴社のコンプライアンス目的、適用可能な法律および規制によって決まります。を使用する際のコンプライアンス責任の詳細については AWS のサービス、[AWS 「セキュリティドキュメント」](#)を参照してください。

AWS Parallel Computing Service の耐障害性

AWS グローバルインフラストラクチャは、AWS リージョン およびアベイラビリティゾーンを中心に構築されています。は、低レイテンシー、高スループット、高度に冗長なネットワークで接続された、物理的に分離および分離された複数のアベイラビリティゾーン AWS リージョン を提供します。アベイラビリティゾーンでは、ゾーン間で中断することなく自動的にフェールオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性、フォールトトレランス、および拡張性が優れています。

AWS リージョン およびアベイラビリティゾーンの詳細については、[AWS 「グローバルインフラストラクチャ」](#)を参照してください。

AWS 並列コンピューティングサービスのインフラストラクチャセキュリティ

マネージドサービスである AWS パラレルコンピューティングサービスは、AWS グローバルネットワークセキュリティで保護されています。AWS セキュリティサービスと ガインフラストラクチャ AWS を保護する方法については、[AWS 「クラウドセキュリティ」](#)を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して環境を AWS 設計するには、「Security Pillar AWS Well-Architected Framework」の「[Infrastructure Protection](#)」を参照してください。

AWS 公開された API コールを使用して、ネットワーク経由で AWS PCS にアクセスします。クライアントは次をサポートする必要があります。

- Transport Layer Security (TLS)。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- DHE (楕円ディフィー・ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。これらのモードは Java 7 以降など、ほとんどの最新システムでサポートされています。

AWS PCS がクラスターを作成すると、サービスはアカウント内のコンピューティングノードとは別に、サービス所有のアカウントで Slurm コントローラーを起動します。コントローラーとコ

コンピューティングノード間の通信をブリッジするために、AWS PCS は VPC にクロスアカウント Elastic Network Interface (ENI) を作成します。Slurm コントローラーは ENI を使用して、さまざまなコンピューティングノードを管理および通信し AWS アカウント、リソースのセキュリティと分離を維持しながら、効率的な HPC および AI/ML オペレーションを促進します。

AWS Parallel Computing Service での脆弱性分析と管理

設定と IT コントロールは、AWS とお客様の間で責任を共有します。詳細については、[AWS 「責任共有モデル」](#) を参照してください。は、コントローラーインスタンスのオペレーティングシステムへのパッチ適用、ファイアウォール設定、インフラストラクチャディザスタリカバリなど、サービスアカウントの基盤となる AWS インフラストラクチャの基本的なセキュリティタスク AWS を処理します。これらの手順は適切なサードパーティーによって確認され、認証されています。詳細については、「[セキュリティ、アイデンティティ、コンプライアンスのベストプラクティス](#)」を参照してください。

Note

Slurm コントローラーは、更新中は利用できません。実行中のジョブは影響を受けません。クラスターのコントローラーが使用できないときに送信されたジョブは、コントローラーが使用可能になるまで保持されます。

お客様は、お客様の基盤となるインフラストラクチャのセキュリティに責任を負います AWS アカウント。

- 更新やセキュリティパッチなど、コードを維持します。
- コンピューティングノードグループの Amazon マシンイメージ (AMI) のオペレーティングシステムにパッチを適用して更新し、更新された AMI を使用するようにコンピューティングノードグループを更新します。
- スケジューラを更新して、サポートされているバージョン内に保持します。コンピューティングノードグループの AMI を更新し、更新された AMI を使用するようにコンピューティングノードグループを更新します。
- ユーザークライアントと接続先のノード間の通信を認証および暗号化します。

コンピューティングノードグループの AMI の更新の詳細については、「」を参照してください [PCS 用の Amazon AWS マシンイメージ \(AMIs\)](#)。

サービス間の混乱した代理の防止

「混乱した代理」問題は、アクションを実行するためのアクセス許可を持たないエンティティが、より特権のあるエンティティにアクションの実行を強制できてしまう場合に生じる、セキュリティ上の問題です。では AWS、サービス間のなりすましにより、混乱した代理問題が発生する可能性があります。サービス間でのなりすましは、1つのサービス (呼び出し元サービス) が、別のサービス (呼び出し対象サービス) を呼び出すときに発生する可能性があります。呼び出し元サービスは、本来ならアクセスすることが許可されるべきではない方法でその許可を使用して、別のお客様のリソースに対する処理を実行するように操作される場合があります。これを防ぐため、AWS では、アカウントのリソースへのアクセス権が付与されたサービスプリンシパルで、すべてのサービスのデータを保護するために役立つツールを提供しています。

リソースポリシーで [aws:SourceArn](#) および [aws:SourceAccount](#) グローバル条件コンテキストキーを使用して、AWS Parallel Computing Service (AWS PCS) がリソースに別のサービスに付与するアクセス許可を制限することをお勧めします。クロスサービスアクセスにリソースを 1 つだけ関連付けたい場合は、`aws:SourceArn` を使用します。そのアカウント内のリソースをクロスサービスの使用に関連付けることを許可する場合は、`aws:SourceAccount` を使用します。

混乱した代理問題から保護するための最も効果的な方法は、リソースの完全な ARN を指定して、`aws:SourceArn` グローバル条件コンテキストキーを使用することです。リソースの完全な ARN が不明な場合や、複数のリソースを指定する場合には、グローバルコンテキスト条件キー `aws:SourceArn` で、ARN の未知部分を示すためにワイルドカード文字 (*) を使用します。例えば、`arn:aws:servicename:*:123456789012:*`。

`aws:SourceArn` の値に Amazon S3 バケット ARN などのアカウント ID が含まれていない場合は、両方のグローバル条件コンテキストキーを使用して、アクセス許可を制限する必要があります。

の値はクラスター ARN `aws:SourceArn` である必要があります。

次の例は、PCS で AWS `aws:SourceArn` および `aws:SourceAccount` グローバル条件コンテキストキーを使用して、混乱した代理問題を防ぐ方法を示しています。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "pcs.amazonaws.com"
    }
  },
}
```



```

    "Action": "sts:AssumeRole",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": [
          "arn:aws:pcs:us-east-1:123456789012:cluster/*"
        ]
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}

```

コンピューティングノードグループの一部としてプロビジョニングされた Amazon EC2 インスタンスの IAM ロール

AWS PCS は、クラスター内の設定された各コンピューティングノードグループの Amazon EC2 容量を自動的にオーケストレーションします。コンピューティングノードグループを作成する場合、ユーザーは `iamInstanceProfileArn` フィールドを使用して IAM インスタンスプロファイルを指定する必要があります。インスタンスプロファイルは、プロビジョニングされた EC2 インスタンスに関連付けられたアクセス許可を指定します。AWS PCS は、`AWSPCS` のプレフィックス `AWSPCS` として、または `role/aws-pcs/` の一部として持つすべてのロールを受け入れます。アクセス `iam:PassRole` 許可は、コンピューティングノードグループを作成または更新する IAM ID (ユーザーまたはロール) が必要です。ユーザーが `CreateComputeNodeGroup` または `UpdateComputeNodeGroup` API アクションを呼び出すと、AWS PCS はユーザーが `iam:PassRole` アクションを実行できるかどうかを確認します。

次のポリシーの例では、名前が `AWSPCS` で始まる IAM ロールのみを渡す許可を付与します。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::123456789012:role/AWSPCS*",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": [
            "ec2.amazonaws.com"
          ]
        }
      }
    }
  ]
}

```

```
    }  
  }  
}  
]  
}
```

AWS Parallel Computing Service のセキュリティのベストプラクティス

このセクションでは、AWS Parallel Computing Service (AWS PCS) に固有のセキュリティのベストプラクティスについて説明します。のセキュリティのベストプラクティスの詳細については AWS、[「セキュリティ、アイデンティティ、コンプライアンスのベストプラクティス」](#)を参照してください。

AMI 関連のセキュリティ

- 本番ワークロードには AWS PCS サンプル AMIs を使用しないでください。サンプル AMIs はサポートされておらず、テストのみを目的としています。
- コンピューティングノードグループの AMI のオペレーティングシステムとソフトウェアを定期的に更新して、脆弱性を軽減します。
- 公式 AWS ソースからダウンロードした認証済みの公式 AWS PCS パッケージのみを使用してください。
- コンピューティングノードグループの AMI の AWS PCS パッケージを定期的に更新し、更新された AMI を使用するようにコンピューティングノードを更新します。脆弱性を最小限に抑えるために、このプロセスの自動化を検討してください。

詳細については、[「PCS のカスタム Amazon AWS マシンイメージ \(AMIs\)」](#)を参照してください。

Slurm Workload Manager のセキュリティ

- アクセスコントロールとネットワーク制限を実装して、Slurm コントロールノードとコンピューティングノードを保護します。信頼できるユーザーとシステムのみがジョブを送信し、Slurm 管理コマンドにアクセスできるようにします。
- Slurm 認証などの Slurm の組み込みセキュリティ機能を使用して、ジョブの送信と通信が認証されるようにします。

- Slurm バージョンを更新して、スムーズなオペレーションとクラスターサポートを維持します。

Important

サポート終了 (EOSL) に達したバージョンの Slurm を使用するクラスターは、直ちに停止します。ユーザーガイドページの上にあるリンクを使用して PCS AWS ドキュメントの RSS フィードをサブスクライブし、Slurm バージョンが EOSL に近づいたときに通知を受け取ります。

詳細については、「[PCS の Slurm AWS バージョン](#)」を参照してください。

- クラスターシークレットを定期的に更新してセキュリティコンプライアンスを維持し、潜在的なセキュリティ侵害を修正します。これは HIPAA と FedRAMP のコンプライアンスに必要です。

詳細については、「[PCS AWS でのクラスターシークレットの更新](#)」を参照してください。

モニタリングとログ記録

- Amazon CloudWatch Logs と AWS CloudTrail を使用して、クラスターと のアクションをモニタリングおよび記録します AWS アカウント。データを使用してトラブルシューティングと監査を行います。

ネットワークセキュリティ

- AWS PCS クラスターを別の VPC にデプロイして、HPC 環境を他のネットワークトラフィックから分離します。
- セキュリティグループとネットワークアクセスコントロールリスト (ACLs) を使用して、PCS AWS インスタンスとサブネットへのインバウンドトラフィックとアウトバウンドトラフィックを制御します。
- AWS PrivateLink または VPC エンドポイントを使用して、クラスターとネットワーク内の他の AWS サービス間の AWS ネットワークトラフィックを維持します。詳細については、「[インターフェイスエンドポイント \(AWS PrivateLink\) AWS Parallel Computing Service を使用した へのアクセス](#)」を参照してください。

PCS AWS のログ記録とモニタリング

モニタリングは、PCS やその他の AWS AWS リソースの信頼性、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、PCS AWS を監視し、問題が発生したときに報告し、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 で実行され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 アカウントによって、またはアカウントに代わって行われた API コールおよび関連イベントをキャプチャし、指定した Amazon S3 バケットにログファイルを配信します。呼び出し元のユーザーとアカウント AWS、呼び出し元の送信元 IP アドレス、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。

PCS AWS のジョブ完了ログ

ジョブ完了ログは、完了時に AWS Parallel Computing Service (AWS PCS) ジョブに関する重要な詳細を追加料金なしで提供します。Amazon CloudWatch Logs、Amazon Simple Storage Service (Amazon S3)、Amazon Data Firehose などの他の AWS サービスを使用して、ログデータにアクセスして処理できます。AWS PCS は、次のようなジョブに関するメタデータを記録します。

- ジョブ ID と名前
- ユーザーとグループの情報
- ジョブの状態 (COMPLETED、FAILED、 などCANCELLED)
- 使用されているパーティション
- 時間制限

- 開始、終了、送信、および対象時間
- ノードのリストと数
- プロセッサ数
- 作業ディレクトリパス
- リソース使用量 (CPU、メモリ)
- 終了コード
- ノードの詳細 (名前、インスタンス IDs、インスタンスタイプ)

目次

- [前提条件](#)
- [ジョブ完了ログの設定](#)
- [ジョブ完了ログの検索方法](#)
 - [CloudWatch Logs](#)
 - [Amazon S3](#)
- [ジョブ完了ログフィールド](#)
- [ジョブ完了ログの例](#)

前提条件

PCS クラスターを管理する IAM AWS プリンシパルは、`pcs:AllowVendedLogDeliveryForResource` アクションを許可する必要があります。

次の IAM ポリシーの例では、必要なアクセス許可を付与します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PcsAllowVendedLogsDelivery",
      "Effect": "Allow",
      "Action": ["pcs:AllowVendedLogDeliveryForResource"],
      "Resource": [
        "arn:aws:pcs:*::cluster/*"
      ]
    }
  ]
}
```

```
    ]  
  }  
]  
}
```

ジョブ完了ログの設定

AWS マネジメントコンソール または を使用して、PCS AWS クラスターのジョブ完了ログを設定できます AWS CLI。

AWS マネジメントコンソール

コンソールでジョブ完了ログを設定するには

1. [AWS PCS コンソール](#)を開きます。
2. ナビゲーションペインで [Clusters] (クラスター) を選択してください。
3. ジョブ完了ログを追加するクラスターを選択します。
4. クラスターの詳細ページで、ログタブを選択します。
5. ジョブ完了ログ で、追加 を選択して、CloudWatch Logs、Amazon S3、Firehose の中から最大 3 つのログ配信先を追加します。
6. ログ配信の更新 を選択します。

AWS CLI

を使用してジョブ完了ログを設定するには AWS CLI

1. ログ配信先を作成します。

```
aws logs put-delivery-destination --region region \  
  --name pcs-logs-destination \  
  --delivery-destination-configuration \  
    destinationResourceArn=resource-arn
```

置き換え前:

- *region* — など、送信先を作成する AWS リージョン *# us-east-1*
- *pcs-logs-destination* — 送信先の名前

- *resource-arn* — CloudWatch Logs ロググループ、S3 バケット、または Firehose 配信ストリームの Amazon リソースネーム (ARN)。

詳細については、「Amazon CloudWatch Logs API Reference」の「[PutDeliveryDestination](#)」を参照してください。

2. PCS クラスターをログ配信ソースとして設定します。

```
aws logs put-delivery-source --region region \  
  --name cluster-logs-source-name \  
  --resource-arn cluster-arn \  
  --log-type PCS_JOBCOMP_LOGS
```

置き換え前:

- *region* — クラスター AWS リージョン の など。 us-east-1
- *cluster-logs-source-name* — ソースの名前
- *cluster-arn* — AWS PCS クラスターの ARN

詳細については、Amazon CloudWatch Logs API リファレンスの「[PutDeliverySource](#)」を参照してください。

3. 配信ソースを配信先に接続します。

```
aws logs create-delivery --region region \  
  --delivery-source-name cluster-logs-source \  
  --delivery-destination-arn destination-arn
```

置き換え前:

- *region* — AWS リージョンなど us-east-1
- *cluster-logs-source* — 配信ソースの名前
- *destination-arn* — 配信先の ARN

詳細については、Amazon CloudWatch Logs API リファレンスの[CreateDelivery](#)」を参照してください。

ジョブ完了ログの検索方法

CloudWatch Logs と Amazon S3 でログの送信先を設定できます。AWS PCS は次の構造化パス名とファイル名を使用します。

CloudWatch Logs

AWS PCS は CloudWatch Logs ストリームに次の名前形式を使用します。

```
AWSLogs/PCS/cluster-id/jobcomp.log
```

例: AWSLogs/PCS/pcs_abc123de45/jobcomp.log

Amazon S3

AWS PCS は S3 パスに次の名前形式を使用します。

```
AWSLogs/account-id/PCS/region/cluster-id/jobcomp/year/month/day/hour/
```

例: AWSLogs/111122223333/PCS/us-east-1/pcs_abc123de45/jobcomp/2025/06/19/11/

AWS PCS はログファイルに次の名前形式を使用します。

```
PCS_jobcomp_year-month-day-hour_cluster-id_random-id.log.gz
```

例: PCS_jobcomp_2025-06-19-11_pcs_abc123de45_04be080b.log.gz

ジョブ完了ログフィールド

AWS PCS はジョブ完了ログデータを JSON オブジェクトとして書き込みます。JSON コンテナはジョブの詳細jobcompを保持します。次の表に、jobcompコンテナ内のフィールドを示します。一部のフィールドは、配列ジョブや異種ジョブなど、特定の状況でのみ存在します。

ジョブ完了ログフィールド

名前	値の例	必須	コメント
job_id	11	はい	常に値とともに存在する
user	"root"	はい	常に値とともに存在する

名前	値の例	必須	コメント
user_id	0	はい	常に値とともに存在する
group	"root"	はい	常に値とともに存在する
group_id	0	はい	常に値とともに存在する
name	"wrap"	はい	常に値とともに存在する
job_state	"COMPLETED"	はい	常に値とともに存在する
partition	"Hydra-Mp iQueue-ab cdef01-7"	はい	常に値とともに存在する
time_limit	"UNLIMITED"	はい	常に存在しますが、"UNLIMITED"
start_time	"2025-06- 19T10:58: 57"	はい	常に存在しますが、"Unknown"
end_time	"2025-06- 19T10:58: 57"	はい	常に存在しますが、"Unknown"
node_list	"Hydra-Mp iNG-abcde f01-2345- 1"	はい	常に値とともに存在する
node_cnt	1	はい	常に値とともに存在する
proc_cnt	1	はい	常に値とともに存在する
work_dir	"/root"	はい	常に存在しますが、"Unknown"
reservati on_name	"weekly_m aintenanc e"	はい	常に存在しますが、空の文字列である可能性があります ""

名前	値の例	必須	コメント
tres.cpu	1	はい	常に値とともに存在する
tres.mem.val	600	はい	常に値とともに存在する
tres.mem.unit	"M"	はい	または "M" にすることができます "bb"
tres.node	1	はい	常に値とともに存在する
tres.billing	1	はい	常に値とともに存在する
account	"finance"	はい	常に存在しますが、空の文字列である可能性があります ""
qos	"normal"	はい	常に存在しますが、空の文字列である可能性があります ""
wc_key	"project_1"	はい	常に存在しますが、空の文字列である可能性があります ""
cluster	"unknown"	はい	常に存在しますが、 "unknown"
submit_time	"2025-06-19T10:55:46"	はい	常に存在しますが、 "Unknown"
eligible_time	"2025-06-19T10:55:46"	はい	常に存在しますが、 "Unknown"
array_job_id	12	なし	ジョブが配列ジョブの場合にのみ存在します
array_task_id	1	なし	ジョブが配列ジョブの場合にのみ存在します

名前	値の例	必須	コメント
het_job_id	10	なし	ジョブが異種ジョブの場合にのみ存在します
het_job_offset	0	なし	ジョブが異種ジョブの場合にのみ存在します
derived_exit_code_status	0	はい	常に値とともに存在する
derived_exit_code_signal	0	はい	常に値とともに存在する
exit_code_status	0	はい	常に値とともに存在する
exit_code_signal	0	はい	常に値とともに存在する
node_details[0].name	"Hydra-Mp iNG-abcde f01-2345- 1"	なし	常に存在しますが、node_details "[]"
node_details[0].instance_id	"i-0abcde f01234567 a"	なし	常に存在しますが、node_details "[]"
node_details[0].instance_type	"t4g.micro"	なし	常に存在しますが、node_details "[]"

ジョブ完了ログの例

次の例は、さまざまなジョブタイプと状態のジョブ完了ログを示しています。

```
{ "jobcomp": { "job_id": 1, "user": "root", "user_id": 0, "group": "root", "group_id":
  0, "name": "wrap", "job_state": "COMPLETED", "partition": "Hydra-MpiQueue-
  abcdef01-7", "time_limit": "UNLIMITED", "start_time": "2025-06-19T16:32:57",
  "end_time": "2025-06-19T16:33:03", "node_list": "Hydra-MpiNG-abcdef01-2345-
  [1-2]", "node_cnt": 2, "proc_cnt": 2, "work_dir": "/usr/bin", "reservation_name":
  "", "tres": { "cpu": 2, "mem": { "val": 1944, "unit": "M" }, "node": 2,
  "billing": 2 }, "account": "", "qos": "", "wc_key": "", "cluster": "unknown",
  "submit_time": "2025-06-19T16:29:40", "eligible_time": "2025-06-19T16:29:41",
  "derived_exit_code_status": 0, "derived_exit_code_signal": 0, "exit_code_status":
  0, "exit_code_signal": 0, "node_details": [ { "name": "Hydra-MpiNG-abcdef01-2345-1",
  "instance_id": "i-0abc123def45678", "instance_type": "t4g.micro" }, { "name":
  "Hydra-MpiNG-abcdef01-2345-2", "instance_id": "i-0def456abc78901", "instance_type":
  "t4g.micro" } ] } }
{ "jobcomp": { "job_id": 2, "user": "root", "user_id": 0, "group": "root", "group_id":
  0, "name": "wrap", "job_state": "COMPLETED", "partition": "Hydra-MpiQueue-
  abcdef01-7", "time_limit": "UNLIMITED", "start_time": "2025-06-19T16:33:13",
  "end_time": "2025-06-19T16:33:14", "node_list": "Hydra-MpiNG-abcdef01-2345-
  [1-2]", "node_cnt": 2, "proc_cnt": 2, "work_dir": "/usr/bin", "reservation_name":
  "", "tres": { "cpu": 2, "mem": { "val": 1944, "unit": "M" }, "node": 2,
  "billing": 2 }, "account": "", "qos": "", "wc_key": "", "cluster": "unknown",
  "submit_time": "2025-06-19T16:33:13", "eligible_time": "2025-06-19T16:33:13",
  "derived_exit_code_status": 0, "derived_exit_code_signal": 0, "exit_code_status":
  0, "exit_code_signal": 0, "node_details": [ { "name": "Hydra-MpiNG-abcdef01-2345-1",
  "instance_id": "i-0abc123def45678", "instance_type": "t4g.micro" }, { "name":
  "Hydra-MpiNG-abcdef01-2345-2", "instance_id": "i-0def456abc78901", "instance_type":
  "t4g.micro" } ] } }
{ "jobcomp": { "job_id": 3, "user": "root", "user_id": 0, "group": "root", "group_id":
  0, "name": "wrap", "job_state": "COMPLETED", "partition": "Hydra-MpiQueue-abcdef01-7",
  "time_limit": "UNLIMITED", "start_time": "2025-06-19T22:58:57", "end_time":
  "2025-06-19T22:58:57", "node_list": "Hydra-MpiNG-abcdef01-2345-1", "node_cnt":
  1, "proc_cnt": 1, "work_dir": "/root", "reservation_name": "", "tres": { "cpu":
  1, "mem": { "val": 972, "unit": "M" }, "node": 1, "billing": 1 }, "account": "",
  "qos": "", "wc_key": "", "cluster": "unknown", "submit_time": "2025-06-19T22:55:46",
  "eligible_time": "2025-06-19T22:55:46", "derived_exit_code_status": 0,
  "derived_exit_code_signal": 0, "exit_code_status": 0, "exit_code_signal":
  0, "node_details": [ { "name": "Hydra-MpiNG-abcdef01-2345-1", "instance_id":
  "i-0abc234def56789", "instance_type": "t4g.micro" } ] } }
{ "jobcomp": { "job_id": 4, "user": "root", "user_id": 0, "group": "root",
  "group_id": 0, "name": "wrap", "job_state": "COMPLETED", "partition": "Hydra-
```

```

MpiQueue-abcdef01-7", "time_limit": "525600", "start_time": "2025-06-19T23:04:27",
  "end_time": "2025-06-19T23:04:27", "node_list": "Hydra-MpiNG-abcdef01-2345-
[1-2]", "node_cnt": 2, "proc_cnt": 2, "work_dir": "/root", "reservation_name":
  "", "tres": { "cpu": 2, "mem": { "val": 1944, "unit": "M" }, "node": 2,
  "billing": 2 }, "account": "", "qos": "", "wc_key": "", "cluster": "unknown",
  "submit_time": "2025-06-19T23:01:38", "eligible_time": "2025-06-19T23:01:38",
  "derived_exit_code_status": 0, "derived_exit_code_signal": 0, "exit_code_status":
  0, "exit_code_signal": 0, "node_details": [ { "name": "Hydra-MpiNG-abcdef01-2345-1",
  "instance_id": "i-0abc234def56789", "instance_type": "t4g.micro" }, { "name":
  "Hydra-MpiNG-abcdef01-2345-2", "instance_id": "i-0def345abc67890", "instance_type":
  "t4g.micro" } ] } }
{ "jobcomp": { "job_id": 5, "user": "root", "user_id": 0, "group": "root", "group_id":
  0, "name": "wrap", "job_state": "FAILED", "partition": "Hydra-MpiQueue-abcdef01-7",
  "time_limit": "UNLIMITED", "start_time": "2025-06-19T23:09:00", "end_time":
  "2025-06-19T23:09:00", "node_list": "(null)", "node_cnt": 0, "proc_cnt": 0,
  "work_dir": "/root", "reservation_name": "", "tres": { "cpu": 1, "mem": { "val":
  1, "unit": "G" }, "node": 1, "billing": 1 }, "account": "", "qos": "", "wc_key":
  "", "cluster": "unknown", "submit_time": "2025-06-19T23:09:00", "eligible_time":
  "2025-06-19T23:09:00", "derived_exit_code_status": 0, "derived_exit_code_signal": 0,
  "exit_code_status": 0, "exit_code_signal": 1, "node_details": [] } }
{ "jobcomp": { "job_id": 6, "user": "root", "user_id": 0, "group": "root", "group_id":
  0, "name": "wrap", "job_state": "CANCELLED", "partition": "Hydra-MpiQueue-
abcdef01-7", "time_limit": "UNLIMITED", "start_time": "2025-06-19T23:09:36",
  "end_time": "2025-06-19T23:09:36", "node_list": "(null)", "node_cnt": 0, "proc_cnt":
  0, "work_dir": "/root", "reservation_name": "", "tres": { "cpu": 1, "mem":
  { "val": 400, "unit": "M" }, "node": 1, "billing": 1 }, "account": "", "qos":
  "", "wc_key": "", "cluster": "unknown", "submit_time": "2025-06-19T23:09:35",
  "eligible_time": "2025-06-19T23:09:36", "het_job_id": 6, "het_job_offset": 0,
  "derived_exit_code_status": 0, "derived_exit_code_signal": 0, "exit_code_status": 0,
  "exit_code_signal": 1, "node_details": [] } }
{ "jobcomp": { "job_id": 7, "user": "root", "user_id": 0, "group": "root", "group_id":
  0, "name": "wrap", "job_state": "CANCELLED", "partition": "Hydra-MpiQueue-
abcdef01-7", "time_limit": "UNLIMITED", "start_time": "2025-06-19T23:10:03",
  "end_time": "2025-06-19T23:10:03", "node_list": "(null)", "node_cnt": 0, "proc_cnt":
  0, "work_dir": "/root", "reservation_name": "", "tres": { "cpu": 1, "mem":
  { "val": 400, "unit": "M" }, "node": 1, "billing": 1 }, "account": "", "qos":
  "", "wc_key": "", "cluster": "unknown", "submit_time": "2025-06-19T23:10:03",
  "eligible_time": "2025-06-19T23:10:03", "het_job_id": 7, "het_job_offset": 0,
  "derived_exit_code_status": 0, "derived_exit_code_signal": 0, "exit_code_status": 0,
  "exit_code_signal": 1, "node_details": [] } }
{ "jobcomp": { "job_id": 8, "user": "root", "user_id": 0, "group": "root", "group_id":
  0, "name": "wrap", "job_state": "COMPLETED", "partition": "Hydra-MpiQueue-abcdef01-7",
  "time_limit": "UNLIMITED", "start_time": "2025-06-19T23:11:24", "end_time":
  "2025-06-19T23:11:24", "node_list": "Hydra-MpiNG-abcdef01-2345-1", "node_cnt":

```

```

1, "proc_cnt": 1, "work_dir": "/root", "reservation_name": "", "tres": { "cpu":
1, "mem": { "val": 400, "unit": "M" }, "node": 1, "billing": 1 }, "account": "",
"qos": "", "wc_key": "", "cluster": "unknown", "submit_time": "2025-06-19T23:11:23",
"eligible_time": "2025-06-19T23:11:23", "het_job_id": 8, "het_job_offset": 0,
"derived_exit_code_status": 0, "derived_exit_code_signal": 0, "exit_code_status":
0, "exit_code_signal": 0, "node_details": [ { "name": "Hydra-MpiNG-abcdef01-2345-1",
"instance_id": "i-0abc234def56789", "instance_type": "t4g.micro" } ] } }
{ "jobcomp": { "job_id": 9, "user": "root", "user_id": 0, "group": "root", "group_id":
0, "name": "wrap", "job_state": "COMPLETED", "partition": "Hydra-MpiQueue-abcdef01-7",
"time_limit": "UNLIMITED", "start_time": "2025-06-19T23:11:24", "end_time":
"2025-06-19T23:11:24", "node_list": "Hydra-MpiNG-abcdef01-2345-2", "node_cnt":
1, "proc_cnt": 1, "work_dir": "/root", "reservation_name": "", "tres": { "cpu":
1, "mem": { "val": 400, "unit": "M" }, "node": 1, "billing": 1 }, "account": "",
"qos": "", "wc_key": "", "cluster": "unknown", "submit_time": "2025-06-19T23:11:23",
"eligible_time": "2025-06-19T23:11:23", "het_job_id": 8, "het_job_offset": 1,
"derived_exit_code_status": 0, "derived_exit_code_signal": 0, "exit_code_status":
0, "exit_code_signal": 0, "node_details": [ { "name": "Hydra-MpiNG-abcdef01-2345-2",
"instance_id": "i-0def345abc67890", "instance_type": "t4g.micro" } ] } }
{ "jobcomp": { "job_id": 10, "user": "root", "user_id": 0, "group": "root", "group_id":
0, "name": "wrap", "job_state": "COMPLETED", "partition": "Hydra-MpiQueue-abcdef01-7",
"time_limit": "UNLIMITED", "start_time": "2025-06-19T23:12:24", "end_time":
"2025-06-19T23:12:24", "node_list": "Hydra-MpiNG-abcdef01-2345-1", "node_cnt":
1, "proc_cnt": 1, "work_dir": "/root", "reservation_name": "", "tres": { "cpu":
1, "mem": { "val": 400, "unit": "M" }, "node": 1, "billing": 1 }, "account": "",
"qos": "", "wc_key": "", "cluster": "unknown", "submit_time": "2025-06-19T23:12:14",
"eligible_time": "2025-06-19T23:12:14", "het_job_id": 10, "het_job_offset": 0,
"derived_exit_code_status": 0, "derived_exit_code_signal": 0, "exit_code_status":
0, "exit_code_signal": 0, "node_details": [ { "name": "Hydra-MpiNG-abcdef01-2345-1",
"instance_id": "i-0abc234def56789", "instance_type": "t4g.micro" } ] } }
{ "jobcomp": { "job_id": 11, "user": "root", "user_id": 0, "group": "root", "group_id":
0, "name": "wrap", "job_state": "COMPLETED", "partition": "Hydra-MpiQueue-abcdef01-7",
"time_limit": "UNLIMITED", "start_time": "2025-06-19T23:12:24", "end_time":
"2025-06-19T23:12:24", "node_list": "Hydra-MpiNG-abcdef01-2345-2", "node_cnt":
1, "proc_cnt": 1, "work_dir": "/root", "reservation_name": "", "tres": { "cpu":
1, "mem": { "val": 600, "unit": "M" }, "node": 1, "billing": 1 }, "account": "",
"qos": "", "wc_key": "", "cluster": "unknown", "submit_time": "2025-06-19T23:12:14",
"eligible_time": "2025-06-19T23:12:14", "het_job_id": 10, "het_job_offset": 1,
"derived_exit_code_status": 0, "derived_exit_code_signal": 0, "exit_code_status":
0, "exit_code_signal": 0, "node_details": [ { "name": "Hydra-MpiNG-abcdef01-2345-2",
"instance_id": "i-0def345abc67890", "instance_type": "t4g.micro" } ] } }
{ "jobcomp": { "job_id": 13, "user": "root", "user_id": 0, "group": "root", "group_id":
0, "name": "wrap", "job_state": "COMPLETED", "partition": "Hydra-MpiQueue-abcdef01-7",
"time_limit": "UNLIMITED", "start_time": "2025-06-19T23:47:57", "end_time":
"2025-06-19T23:47:58", "node_list": "Hydra-MpiNG-abcdef01-2345-1", "node_cnt":

```

```
1, "proc_cnt": 1, "work_dir": "/root", "reservation_name": "", "tres": { "cpu":
1, "mem": { "val": 972, "unit": "M" }, "node": 1, "billing": 1 }, "account": "",
"qos": "", "wc_key": "", "cluster": "unknown", "submit_time": "2025-06-19T23:43:56",
"eligible_time": "2025-06-19T23:43:56" , "array_job_id": 12, "array_task_id": 1,
"derived_exit_code_status": 0, "derived_exit_code_signal": 0, "exit_code_status":
0, "exit_code_signal": 0, "node_details": [ { "name": "Hydra-MpiNG-abcdef01-2345-1",
"instance_id": "i-0abc345def67890", "instance_type": "t4g.micro" } ] } }
{ "jobcomp": { "job_id": 12, "user": "root", "user_id": 0, "group": "root", "group_id":
0, "name": "wrap", "job_state": "COMPLETED", "partition": "Hydra-MpiQueue-abcdef01-7",
"time_limit": "UNLIMITED", "start_time": "2025-06-19T23:47:58", "end_time":
"2025-06-19T23:47:58", "node_list": "Hydra-MpiNG-abcdef01-2345-1", "node_cnt":
1, "proc_cnt": 1, "work_dir": "/root", "reservation_name": "", "tres": { "cpu":
1, "mem": { "val": 972, "unit": "M" }, "node": 1, "billing": 1 }, "account": "",
"qos": "", "wc_key": "", "cluster": "unknown", "submit_time": "2025-06-19T23:43:56",
"eligible_time": "2025-06-19T23:43:56" , "array_job_id": 12, "array_task_id": 2,
"derived_exit_code_status": 0, "derived_exit_code_signal": 0, "exit_code_status":
0, "exit_code_signal": 0, "node_details": [ { "name": "Hydra-MpiNG-abcdef01-2345-1",
"instance_id": "i-0abc345def67890", "instance_type": "t4g.micro" } ] } } }
```

PCS AWS のスケジューラログ

クラスタースケジューラから Amazon CloudWatch Logs、Amazon Simple Storage Service (Amazon S3)、Amazon Data Firehose に詳細なログデータを送信するように AWS PCS を設定できます。これは、モニタリングとトラブルシューティングに役立ちます。

目次

- [前提条件](#)
- [スケジューラログの設定](#)
- [スケジューラログストリームのパスと名前](#)
- [スケジューラログレコードの例](#)

前提条件

PCS クラスターを管理する IAM AWS プリンシパルは、`pcs:AllowVendedLogDeliveryForResource` アクションを許可する必要があります。

次の IAM ポリシーの例では、必要なアクセス許可を付与します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PcsAllowVendedLogsDelivery",
      "Effect": "Allow",
      "Action": ["pcs:AllowVendedLogDeliveryForResource"],
      "Resource": [
        "arn:aws:pcs:*::cluster/*"
      ]
    }
  ]
}
```

スケジューラログの設定

AWS マネジメントコンソール または を使用して、PCS AWS クラスターのスケジューラログを設定できます AWS CLI。

AWS マネジメントコンソール

コンソールでスケジューラログを設定するには

1. [AWS PCS コンソール](#)を開きます。
2. ナビゲーションペインで [Clusters] (クラスター) を選択してください。
3. スケジューラログを追加するクラスターを選択します。
4. クラスターの詳細ページで、ログタブを選択します。
5. スケジューラログで、追加 を選択して、CloudWatch Logs、Amazon S3、Firehose の中から最大 3 つのログ配信先を追加します。
6. ログ配信の更新 を選択します。

AWS CLI

を使用してスケジューラログを設定するには AWS CLI

1. ログ配信先を作成します。


```
aws logs put-delivery-destination --region region \  
  --name pcs-logs-destination \  
  --delivery-destination-configuration \  
    destinationResourceArn=resource-arn
```

置き換え前:

- *region* — など、送信先を作成する AWS リージョン # *us-east-1*
- *pcs-logs-destination* — 送信先の名前
- *resource-arn* — CloudWatch Logs ロググループ、S3 バケット、または Firehose 配信ストリームの Amazon リソースネーム (ARN)。

詳細については、「Amazon CloudWatch Logs API Reference」の「[PutDeliveryDestination](#)」を参照してください。

2. PCS クラスターをログ配信ソースとして設定します。

```
aws logs put-delivery-source --region region \  
  --name cluster-logs-source-name \  
  --resource-arn cluster-arn \  
  --log-type PCS_SCHEDULER_LOGS
```

置き換え前:

- *region* — クラスター AWS リージョン の など。 *us-east-1*
- *cluster-logs-source-name* — ソースの名前
- *cluster-arn* — AWS PCS クラスターの ARN

詳細については、Amazon CloudWatch Logs API リファレンス」の「[PutDeliverySource](#)」を参照してください。

3. 配信ソースを配信先に接続します。

```
aws logs create-delivery --region region \  
  --delivery-source-name cluster-logs-source \  
  --delivery-destination-arn destination-arn
```

置き換え前:

- *region* — AWS リージョンなど us-east-1
- *cluster-logs-source* — 配信ソースの名前
- *destination-arn* — 配信先の ARN

詳細については、Amazon CloudWatch Logs API リファレンスの[CreateDelivery](#)」を参照してください。

スケジューラログストリームのパスと名前

AWS PCS スケジューラログのパスと名前は、送信先タイプによって異なります。

- CloudWatch Logs
 - CloudWatch Logs ストリームは、この命名規則に従います。

```
AWSLogs/PCS/${cluster_id}/${log_name}_${scheduler_major_version}.log
```

Example

```
AWSLogs/PCS/abcdef0123/slurmctld_24.05.log
```

- S3 バケット
 - S3 バケットの出力パスは、次の命名規則に従います。

```
AWSLogs/${account-id}/PCS/${region}/${cluster_id}/${log_name}/  
${scheduler_major_version}/yyyy/MM/dd/HH/
```

Example

```
AWSLogs/111111111111/PCS/us-east-2/abcdef0123/slurmctld/24.05/2024/09/01/00.
```

- S3 オブジェクト名は次の規則に従います。

```
PCS_${log_name}_${scheduler_major_version}_#{expr date 'event_timestamp', format:  
"yyyy-MM-dd-HH"}_${cluster_id}_${hash}.log
```

Example

```
PCS_slurmctld_24.05_2024-09-01-00_abcdef0123_0123abcdef.log
```

スケジューラログレコードの例

AWS PCS スケジューラログは構造化されています。これには、Slurm コントローラープロセスから出力されるログメッセージに加えて、クラスター識別子、スケジューラタイプ、メジャーバージョン、パッチバージョンなどのフィールドが含まれます。以下はその例です。

```
{
  "resource_id": "s3431v9rx2",
  "resource_type": "PCS_CLUSTER",
  "event_timestamp": 1721230979,
  "log_level": "info",
  "log_name": "slurmctld",
  "scheduler_type": "slurm",
  "scheduler_major_version": "25.05",
  "scheduler_patch_version": "3",
  "node_type": "controller_primary",
  "message": "[2024-07-17T15:42:58.614+00:00] Running as primary controller\n"
}
```

Amazon CloudWatch による AWS 並列コンピューティングサービスのモニタリング

Amazon CloudWatch は、クラスターからメトリクスを一定の間隔で収集することで、AWS Parallel Computing Service (AWS PCS) クラスターのヘルスとパフォーマンスをモニタリングします。これらのメトリクスは保持されるため、履歴データにアクセスし、時間の経過とともにクラスターのパフォーマンスに関するインサイトを得ることができます。

CloudWatch では、スケーリング要件を満たすために PCS AWS によって起動された EC2 インスタンスをモニタリングすることもできます。実行中のインスタンスのログを検査できますが、CloudWatch メトリクスとログ記録データは通常、インスタンスが終了すると削除されます。ただし、EC2 起動テンプレートを使用してインスタンスの CloudWatch エージェントを設定することで、インスタンスの終了後もメトリクスとログを保持し、長期的なモニタリングと分析を行うことができます。

CloudWatch を使用した PCS AWS のモニタリングの詳細については、このセクションのトピックを参照してください。

トピック

- [CloudWatch AWS を使用した PCS メトリクスのモニタリング](#)
- [Amazon CloudWatch AWS を使用した PCS インスタンスのモニタリング](#)

CloudWatch AWS を使用した PCS メトリクスのモニタリング

Amazon CloudWatch を使用して AWS Amazon CloudWatch はクラスターからデータを収集し、ほぼリアルタイムのメトリクスに変換します。これらの統計は 15 か月間保持されるため、履歴情報にアクセスしてクラスターのパフォーマンスをより的確に把握できます。クラスターメトリクスは 1 分間隔で CloudWatch に送信されます。CloudWatch の詳細については、「Amazon CloudWatch ユーザーガイド」の「[Amazon CloudWatch とは?](#)」を参照してください。

AWS PCS は、CloudWatch の AWS/PCS 名前空間に次のメトリクスを発行します。1 つのディメンション `ClusterId` があります。

名前	説明	単位
ActualCapacity	IdleCapacity + UtilizedCapacity	カウント
CapacityUtilization	UtilizedCapacity/ActualCapacity	カウント
DesiredCapacity	ActualCapacity + PendingCapacity	カウント
IdleCapacity	実行中のがジョブに割り当てられていないインスタンスの数	カウント
UtilizedCapacity	実行中でジョブに割り当てられたインスタンスの数	Count (カウント)

Amazon CloudWatch AWS を使用した PCS インスタンスのモニタリング

AWS PCS は、PCS コンピューティングノードグループで定義されているスケーリング要件を満たすために、必要に応じて Amazon EC2 インスタンスを起動します。これらのインスタンスは、Amazon CloudWatch を使用して実行中にモニタリングできます。実行中のインスタンスのログを確認するには、インスタンスにログインし、インタラクティブなコマンドラインツールを使用します。ただし、デフォルトでは、CloudWatch メトリクスデータはインスタンスが終了すると一定期間のみ保持され、インスタンスログは通常、インスタンスをバックアップする EBS ボリュームとともに削除されます。終了後に PCS によって起動されたインスタンスからメトリクスまたはログデータを保持するには、EC2 起動テンプレートを使用してインスタンスで CloudWatch エージェントを設定できます。このトピックでは、実行中のインスタンスのモニタリングの概要と、永続インスタンスのメトリクスとログを設定する方法の例を示します。

実行中のインスタンスのモニタリング

AWS PCS インスタンスの検索

PCS によって起動されたインスタンスをモニタリングするには、クラスターまたはコンピューティングノードグループに関連付けられた実行中のインスタンスを見つけます。次に、特定のインスタンスの EC2 コンソールで、ステータスとアラーム、モニタリングセクションを検査します。これらのインスタンスのログインアクセスが設定されている場合は、インスタンスに接続して、インスタンスのさまざまなログファイルを検査できます。PCS によって管理されるインスタンスの特定の詳細については、「」を参照してください[PCS AWS でのコンピューティングノードグループインスタンスの検索](#)。

詳細なメトリクスの有効化

デフォルトでは、インスタンスメトリクスは 5 分間隔で収集されます。1 分間隔でメトリクスを収集するには、コンピューティングノードグループの起動テンプレートで詳細な CloudWatch モニタリングを有効にします。詳細については、「[詳細な CloudWatch モニタリングを有効にする](#)」を参照してください。

永続インスタンスのメトリクスとログの設定

インスタンスに Amazon CloudWatch エージェントをインストールして設定することで、インスタンスのメトリクスとログを保持できます。これは、主に 3 つのステップで構成されます。

1. CloudWatch エージェント設定を作成します。
2. PCS インスタンスで取得できる設定を保存します。

3. CloudWatch エージェントソフトウェアをインストールし、設定を取得し、設定を使用して CloudWatch エージェントを開始する EC2 起動テンプレートを作成します。

詳細については、「Amazon [CloudWatch ユーザーガイド](#)」の「[CloudWatch エージェントを使用してメトリクス、ログ、トレースを収集する](#)」および「[」を参照してください](#)PCS での Amazon EC2 AWS 起動テンプレートの使用。 Amazon CloudWatch

CloudWatch エージェント設定を作成する

CloudWatch エージェントをインスタンスにデプロイする前に、収集するメトリクス、ログ、トレースを指定する JSON 設定ファイルを作成する必要があります。設定ファイルは、ウィザードを使用して作成することも、テキストエディタを使用して手動で作成することもできます。設定ファイルは、このデモ用に手動で作成されます。

AWS CLI がインストールされているコンピュータで、config.json という名前の CloudWatch 設定ファイルを以下の内容で作成します。次の URL を使用して、ファイルのコピーをダウンロードすることもできます。

```
https://aws-hpc-recipes.s3.amazonaws.com/main/recipes/pcs/cloudwatch/assets/config.json
```

注意事項

- サンプルファイルのログパスは Amazon Linux 2 用です。インスタンスが別の基本オペレーティングシステムを使用する場合は、必要に応じてパスを変更します。
- 他のログをキャプチャするには、の下にエントリを追加しますcollect_list。
- の値はテンプレート変数{brackets}です。サポートされている変数の完全なリストについては、「Amazon [CloudWatch ユーザーガイド](#)」の「[CloudWatch エージェント設定ファイルを手動で作成または編集する](#)」を参照してください。 Amazon CloudWatch
- これらの情報タイプを収集しない場合は、logsを省略するかmetrics、を省略するかを選択できます。

```
{
  "agent": {
    "metrics_collection_interval": 60
  },
  "logs": {
    "logs_collected": {
      "files": {
```

```
"collect_list": [  
  {  
    "file_path": "/var/log/cloud-init.log",  
    "log_group_class": "STANDARD",  
    "log_group_name": "/PCSLogs/instances",  
    "log_stream_name": "{instance_id}.cloud-init.log",  
    "retention_in_days": 30  
  },  
  {  
    "file_path": "/var/log/cloud-init-output.log",  
    "log_group_class": "STANDARD",  
    "log_stream_name": "{instance_id}.cloud-init-output.log",  
    "log_group_name": "/PCSLogs/instances",  
    "retention_in_days": 30  
  },  
  {  
    "file_path": "/var/log/amazon/pcs/bootstrap.log",  
    "log_group_class": "STANDARD",  
    "log_stream_name": "{instance_id}.bootstrap.log",  
    "log_group_name": "/PCSLogs/instances",  
    "retention_in_days": 30  
  },  
  {  
    "file_path": "/var/log/slurmd.log",  
    "log_group_class": "STANDARD",  
    "log_stream_name": "{instance_id}.slurmd.log",  
    "log_group_name": "/PCSLogs/instances",  
    "retention_in_days": 30  
  },  
  {  
    "file_path": "/var/log/messages",  
    "log_group_class": "STANDARD",  
    "log_stream_name": "{instance_id}.messages",  
    "log_group_name": "/PCSLogs/instances",  
    "retention_in_days": 30  
  },  
  {  
    "file_path": "/var/log/secure",  
    "log_group_class": "STANDARD",  
    "log_stream_name": "{instance_id}.secure",  
    "log_group_name": "/PCSLogs/instances",  
    "retention_in_days": 30  
  }  
]
```

```

    }
  }
},
"metrics": {
  "aggregation_dimensions": [
    [
      "InstanceId"
    ]
  ],
  "append_dimensions": {
    "AutoScalingGroupName": "${aws:AutoScalingGroupName}",
    "ImageId": "${aws:ImageId}",
    "InstanceId": "${aws:InstanceId}",
    "InstanceType": "${aws:InstanceType}"
  },
  "metrics_collected": {
    "cpu": {
      "measurement": [
        "cpu_usage_idle",
        "cpu_usage_iowait",
        "cpu_usage_user",
        "cpu_usage_system"
      ],
      "metrics_collection_interval": 60,
      "resources": [
        "*"
      ],
      "totalcpu": false
    },
    "disk": {
      "measurement": [
        "used_percent",
        "inodes_free"
      ],
      "metrics_collection_interval": 60,
      "resources": [
        "*"
      ]
    },
    "diskio": {
      "measurement": [
        "io_time"
      ],
      "metrics_collection_interval": 60,

```



```

        "resources": [
            "*"
        ],
    },
    "mem": {
        "measurement": [
            "mem_used_percent"
        ],
        "metrics_collection_interval": 60
    },
    "swap": {
        "measurement": [
            "swap_used_percent"
        ],
        "metrics_collection_interval": 60
    }
}
}
}

```

このファイルは、インスタンスのブートストラップ、認証とログイン、およびその他のトラブルシューティングドメインのエラーの診断に役立つ複数のファイルをモニタリングするように CloudWatch エージェントに指示します。具体的には次のとおりです。

- `/var/log/cloud-init.log` – インスタンス設定の初期ステージからの出力
- `/var/log/cloud-init-output.log` – インスタンス設定中に実行されるコマンドからの出力
- `/var/log/amazon/pcs/bootstrap.log` – インスタンス設定中に実行される PCS 固有のオペレーションからの出力
- `/var/log/slurmd.log` – Slurm ワークロードマネージャーのデーモンスラムからの出力
- `/var/log/messages` – カーネル、システムサービス、アプリケーションからのシステムメッセージ
- `/var/log/secure` – SSH、sudo、その他のセキュリティイベントなどの認証試行に関連するログ

ログファイルは、という名前の CloudWatch ロググループに送信されます `/PCSLogs/instances`。ログストリームは、インスタンス ID とログファイルのベース名の組み合わせです。ロググループの保持期間は 30 日です。

さらに、ファイルは CloudWatch エージェントにいくつかの一般的なメトリクスを収集し、インスタンス ID 別に集約するように指示します。

設定を保存する

CloudWatch エージェント設定ファイルは、PCS コンピューティングノードインスタンスがアクセスできる場所に保存する必要があります。これを行うには、2 つの一般的な方法があります。コンピューティングノードグループインスタンスがインスタンスプロファイルを通じてアクセスできる Amazon S3 バケットにアップロードするか、Amazon Systems Manager パラメータストアに SSM パラメータとして保存できます。

S3 バケットへのアップロード

ファイルを S3 に保存するには、次の AWS CLI コマンドを使用します。コマンドを実行する前に、次の置き換えを行います。

- `amzn-s3-demo-bucket` を独自の S3 バケット名に置き換える

まず、(既存のバケットがある場合はオプション) 設定ファイルを保持するバケットを作成します (複数可)。

```
aws s3 mb s3://amzn-s3-demo-bucket
```

次に、ファイルを バケットにアップロードします。

```
aws s3 cp ./config.json s3://amzn-s3-demo-bucket/
```

SSM パラメータとして保存する

ファイルを SSM パラメータとして保存するには、次のコマンドを使用します。コマンドを実行する前に、次の置き換えを行います。

- `region-code` を、AWS PCS を使用している AWS リージョンに置き換えます。
- (オプション) `AmazonCloudWatch-PCS` を パラメータの独自の名前に置き換えます。名前のプレフィックスを から変更する場合 AmazonCloudWatch-、ノードグループインスタンスプロファイルの SSM パラメータに読み取りアクセスを具体的に追加する必要があります。

```
aws ssm put-parameter \  
  --region region-code \  
  --name AmazonCloudWatch-PCS \  
  --type String --value ./config.json
```

```
--name "AmazonCloudWatch-PCS" \  
--type String \  
--value file://config.json
```

EC2 起動テンプレートを記述する

起動テンプレートの具体的な詳細は、設定ファイルが S3 または SSM に保存されているかどうかによって異なります。

S3 に保存されている設定を使用する

このスクリプトは、CloudWatch エージェントをインストールし、S3 バケットから設定ファイルをインポートし、それを使用して CloudWatch エージェントを起動します。このスクリプトの次の値を独自の詳細に置き換えます。

- *amzn-s3-demo-bucket* – アカウントが読み取ることができる S3 バケットの名前
- */config.json* – 設定が保存されている S3 バケットルートへの相対パス

```
MIME-Version: 1.0  
Content-Type: multipart/mixed; boundary=="MYBOUNDARY=="  
  
--==MYBOUNDARY==  
Content-Type: text/cloud-config; charset="us-ascii"  
  
packages:  
- amazon-cloudwatch-agent  
  
runcmd:  
- aws s3 cp s3://amzn-s3-demo-bucket/config.json /etc/s3-cw-config.json  
- /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m  
  ec2 -s -c file://etc/s3-cw-config.json  
  
--==MYBOUNDARY==--
```

ノードグループの IAM インスタンスプロファイルには、バケットへのアクセス権が必要です。上記のユーザーデータスクリプトのバケットの IAM ポリシーの例を次に示します。

JSON

```
{
```

```

    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": [
          "s3:GetObject",
          "s3:ListBucket"
        ],
        "Resource": [
          "arn:aws:s3:::amzn-s3-demo-bucket",
          "arn:aws:s3:::amzn-s3-demo-bucket/*"
        ]
      }
    ]
  }
}

```

また、インスタンスは S3 エンドポイントと CloudWatch エンドポイントへのアウトバウンドトラフィックを許可する必要があります。これは、クラスターアーキテクチャに応じて、セキュリティグループまたは VPC エンドポイントを使用して実現できます。

SSM に保存されている設定を使用する

このスクリプトは、CloudWatch エージェントをインストールし、SSM パラメータから設定ファイルをインポートし、それを使用して CloudWatch エージェントを起動します。このスクリプトの次の値を独自の詳細に置き換えます。

- (オプション) *AmazonCloudWatch-PCS* を パラメータの独自の名前に置き換えます。

```

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

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

packages:
- amazon-cloudwatch-agent

runcmd:
- /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -s -c ssm:AmazonCloudWatch-PCS

```

```
---MYBOUNDARY---
```

ノードグループの IAM インスタンスポリシーには、CloudWatchAgentServerPolicy がアタッチされている必要があります。

パラメータ名が で始まらないAmazonCloudWatch-場合は、ノードグループインスタンスプロファイルの SSM パラメータに読み取りアクセスを具体的に追加する必要があります。以下は、プレフィックス *DOC-EXAMPLE-PREFIX* の IAM ポリシーの例です。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CustomCwSsmMParamReadOnly",
      "Effect": "Allow",
      "Action": [
        "ssm:GetParameter"
      ],
      "Resource": "arn:aws:ssm:*:*:parameter/DOC-EXAMPLE-PREFIX*"
    }
  ]
}
```

また、インスタンスは SSM エンドポイントと CloudWatch エンドポイントへのアウトバウンドトラフィックを許可する必要があります。これは、クラスターアーキテクチャに応じて、セキュリティグループまたは VPC エンドポイントを使用して実現できます。

を使用した AWS 並列コンピューティングサービス API コールのログ記録 AWS CloudTrail

AWS PCS は、PCS のユーザー AWS CloudTrail、ロール、または サービスによって実行されたアクションを記録する AWS サービスである AWS と統合されています。CloudTrail は、AWS PCS のすべての API コールをイベントとしてキャプチャします。キャプチャされた呼び出しには、PCS コンソールからの呼び出しと AWS PCS API AWS オペレーションへのコード呼び出しが含まれます。証跡を作成する場合は、PCS のイベントなど、Amazon S3 バケットへの CloudTrail AWS イベントの継続的な配信を有効にすることができます。証跡を設定しない場合でも、CloudTrail コンソール

の [イベント履歴] で最新のイベントを表示できます。CloudTrail によって収集された情報を使用して、PCS AWS に対して行われたリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

CloudTrail の詳細については、「[AWS CloudTrail ユーザーガイド](#)」を参照してください。

AWS CloudTrail の PCS 情報

CloudTrail は、アカウントの作成 AWS アカウント 時に 有効になります。AWS PCS でアクティビティが発生すると、そのアクティビティはイベント履歴の他の AWS サービスイベントとともに CloudTrail イベントに記録されます。で最近のイベントを表示、検索、ダウンロードできます AWS アカウント。詳細については、[CloudTrail イベント履歴でのイベントの表示](#)を参照してください。

PCS のイベントなど AWS アカウント、AWS のイベントの継続的な記録については、証跡を作成します。追跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。デフォルトでは、コンソールで証跡を作成するときに、証跡がすべての AWS リージョンに適用されます。証跡は、AWS パーティション内のすべてのリージョンからのイベントをログに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、CloudTrail ログで収集されたイベントデータをより詳細に分析し、それに基づいて行動するように、他の AWS サービスを設定できます。詳細については、次を参照してください:

- [追跡を作成するための概要](#)
- 「[CloudTrail がサポートされているサービスと統合](#)」
- 「[CloudTrail の Amazon SNS 通知の設定](#)」
- 「[複数のリージョンから CloudTrail ログファイルを受け取る](#)」および「[複数のアカウントから CloudTrail ログファイルを受け取る](#)」

すべての AWS PCS アクションは CloudTrail によってログに記録され、[AWS](#) [「Parallel Computing Service API リファレンス」](#)に記載されています。例え

ば、CreateComputeNodeGroup、UpdateQueue、DeleteCluster の各アクションを呼び出すと、CloudTrail ログファイルにエントリが生成されます。

各イベントまたはログエントリには、リクエストの生成者に関する情報が含まれます。アイデンティティ情報は、以下を判別するのに役立ちます。

- リクエストがルートまたは AWS Identity and Access Management (IAM) ユーザー認証情報を使用して行われたかどうか。

- リクエストがロールまたはフェデレーションユーザーのテンポラリなセキュリティ認証情報を使用して行われたかどうか。
- リクエストが別の AWS サービスによって行われたかどうか。

詳細については、「[CloudTrail userIdentity エlement](#)」を参照してください。

PCS からの CloudTrail AWS ログファイルエントリについて

証跡は、指定した S3 バケットにイベントをログファイルとして配信するように設定できます。CloudTrail ログファイルには、1 つ以上のログエントリがあります。イベントは任意ソースからの単一リクエストを表し、リクエストされたアクション、アクションの日時、リクエストパラメータなどの情報を含みます。CloudTrail ログファイルは、パブリック API 呼び出しの順序付けられたスタックトレースではないため、特定の順序では表示されません。

次の例は、CreateQueueアクションの CloudTrail ログエントリを示しています。

```
{
  "eventVersion": "1.09",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE:admin",
    "arn": "arn:aws:sts::012345678910:assumed-role/Admin/admin",
    "accountId": "012345678910",
    "accessKeyId": "ASIAY36PTPIEXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAY36PTPIEEXAMPLE",
        "arn": "arn:aws:iam::012345678910:role/Admin",
        "accountId": "012345678910",
        "userName": "Admin"
      },
      "attributes": {
        "creationDate": "2024-07-16T17:05:51Z",
        "mfaAuthenticated": "false"
      }
    }
  },
  "eventTime": "2024-07-16T17:13:09Z",
  "eventSource": "pcs.amazonaws.com",
  "eventName": "CreateQueue",
```

```

    "awsRegion": "us-east-1",
    "sourceIPAddress": "127.0.0.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/126.0.0.0 Safari/537.36",
    "requestParameters": {
      "clientToken": "c13b7baf-2894-42e8-acec-example",
      "clusterIdentifier": "abcdef0123",
      "computeNodeGroupConfigurations": [
        {
          "computeNodeId": "abcdef0123"
        }
      ],
      "queueName": "all"
    },
    "responseElements": {
      "queue": {
        "arn": "arn:aws:pcs:us-east-1:609783872011:cluster/abcdef0123/queue/
abcdef0123",
        "clusterId": "abcdef0123",
        "computeNodeGroupConfigurations": [
          {
            "computeNodeId": "abcdef0123"
          }
        ],
        "createdAt": "2024-07-16T17:13:09.276069393Z",
        "id": "abcdef0123",
        "modifiedAt": "2024-07-16T17:13:09.276069393Z",
        "name": "all",
        "status": "CREATING"
      }
    },
    "requestID": "a9df46d7-3f6d-43a0-9e3f-example",
    "eventID": "7ab18f88-0040-47f5-8388-example",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "012345678910",
    "eventCategory": "Management",
    "tlsDetails": {
      "tlsVersion": "TLSv1.3",
      "cipherSuite": "TLS_AES_128_GCM_SHA256",
      "clientProvidedHostHeader": "pcs.us-east-1.amazonaws.com"
    },
    "sessionCredentialFromConsole": "true"

```



```
}
```

AWS PCS のエンドポイントとサービスクォータ

以下のセクションでは、AWS Parallel Computing Service (AWS PCS) のエンドポイントとサービスクォータについて説明します。以前は制限と呼ばれていたサービスクォータは、のサービスリソースまたはオペレーションの最大数です AWS アカウント。

AWS アカウント には、AWS サービスごとにデフォルトのクォータがあります。特に明記されていない限り、クォータは地域固有です。一部のクォータについては引き上げをリクエストできますが、その他のクォータについては引き上げることはできません。

詳細については、「AWS 全般のリファレンス」の「[AWS Service Quotas](#)」を参照してください。

目次

- [サービスエンドポイント](#)
- [Service Quotas](#)
 - [内部クォータ](#)
 - [他の AWS サービスの関連するクォータ](#)

サービスエンドポイント

リージョン名	リージョン	エンドポイント	プロトコル
米国東部 (オハイオ)	us-east-2	pcs.us-east-2.amazonaws.com	HTTPS
		pcs-fips.us-east-2.amazonaws.com	
		pcs-fips.us-east-2.api.aws	
		pcs.us-east-2.api.aws	
米国東部 (バージニア北部)	us-east-1	pcs.us-east-1.amazonaws.com	HTTPS

リージョン名	リージョン	エンドポイント	プロトコル
		pcs-fips.us-east-1 .amazonaws.com	
		pcs-fips.us-east-1 .api.aws	
		pcs.us-east-1.api.aws	
米国西部 (オレゴン)	us-west-2	pcs.us-west-2.amaz onaws.com	HTTPS
		pcs-fips.us-west-2 .amazonaws.com	
		pcs-fips.us-west-2 .api.aws	
		pcs.us-west-2.api.aws	
アジアパシフィック (シンガポール)	ap-southeast-1	pcs.ap-southeast-1 .amazonaws.com	HTTPS
		pcs.ap-southeast-1 .api.aws	
アジアパシフィック (シドニー)	ap-southeast-2	pcs.ap-southeast-2 .amazonaws.com	HTTPS
		pcs.ap-southeast-2 .api.aws	
アジアパシフィック (東京)	ap-northeast-1	pcs.ap-northeast-1 .amazonaws.com	HTTPS
		pcs.ap-northeast-1 .api.aws	

リージョン名	リージョン	エンドポイント	プロトコル
欧州 (フランクフルト)	eu-central-1	pcs.eu-central-1.amazonaws.com pcs.eu-central-1.api.aws	HTTPS
欧州 (アイルランド)	eu-west-1	pcs.eu-west-1.amazonaws.com pcs.eu-west-1.api.aws	HTTPS
欧州 (ロンドン)	eu-west-2	pcs.eu-west-2.amazonaws.com pcs.eu-west-2.api.aws	HTTPS
欧州 (ストックホルム)	eu-north-1	pcs.eu-north-1.amazonaws.com pcs.eu-north-1.api.aws	HTTPS
AWS GovCloud (米国東部)	us-gov-east-1	pcs.us-gov-east-1.amazonaws.com pcs-fips.us-gov-east-1.amazonaws.com pcs-fips.us-gov-east-1.api.aws pcs.us-gov-east-1.api.aws	HTTPS

リージョン名	リージョン	エンドポイント	プロトコル
AWS GovCloud (米国西部)	us-gov-west-1	pcs.us-gov-west-1. amazonaws.com pcs-fips.us-gov-west-1. amazonaws.com pcs-fips.us-gov-west-1. api.aws pcs.us-gov-west-1. api.aws	HTTPS

Service Quotas

名前	デフォルト	調整可能	説明
クラスター	5	はい	あたりのクラスターの最大数 AWS リージョン。

Note

デフォルト値は、によって設定された初期クォータです AWS。これらのデフォルト値は、実際に適用されたクォータ値および適用可能な最大 Service Quotas とは異なります。詳細については、「Service Quotas ユーザーガイド」の「[Service Quotas の用語](#)」を参照してください。

これらのサービスクォータは、の AWS Parallel Computing Service (PCS) に記載されています [AWS マネジメントコンソール](#)。調整可能として表示される値のクォータ引き上げをリクエストするには、「[Service Quotas ユーザーガイド](#)」の「[クォータ引き上げのリクエスト](#)」を参照してください。 Service Quotas

⚠ Important

で現在の AWS リージョン 設定を必ず確認してください AWS マネジメントコンソール。

内部クォータ

次のクォータは内部クォータであり、調整不可能です。

名前	デフォルト	調整可能	説明
クラスターの同時作成	1	いいえ	AWS リージョンあたりの Creating 状態のクラスターの最大数。
クラスターあたりのコンピューティングノードグループ	10	いいえ	クラスターあたりのコンピューティングノードグループの最大数。
クラスターあたりのキュー数	10	いいえ	クラスターあたりのキューの最大数。

他の AWS サービスの関連するクォータ

AWS PCS は他の AWS サービスを使用します。これらのサービスのサービスクォータは、PCS AWS の使用に影響します。

PCS に影響する Amazon EC2 AWS サービスクォータ

- スポットインスタンスリクエスト
- オンデマンドインスタンスの実行
- 起動テンプレート
- 起動テンプレートのバージョン
- Amazon EC2 API リクエスト

詳細については、[Amazon EC2 サービスクォータ](#)」を参照してください。

AWS Parallel Computing Service の問題のトラブルシューティング

以下のトピックでは、PCS AWS で発生する可能性のあるいくつかの問題をトラブルシューティングするためのガイダンスを提供します。

- [クラスターの更新](#)
- [コンピューティングノードのブートストラップの問題](#)
- [カスタム Slurm 設定](#)
- [再起動後に終了した EC2 インスタンス](#)
- [ID とアクセス](#)
- [Slurm の再起動の問題](#)

再起動後に PCS AWS の EC2 インスタンスが終了して置き換えられる

問題の概要

コンピューティングノードグループの EC2 インスタンスを再起動すると、AWS PCS は自動的に終了してインスタンスを置き換えます。

これが発生する理由

AWS PCS はインスタンスの再起動をサポートしていません。EC2 インスタンスを再起動すると、AWS PCS はインスタンスを異常と見なし、置き換えます。AWS PCS がインスタンスを継続的に終了して置き換える場合は、起動後にインスタンスが再起動されることが原因である可能性があります。例としては、EC2 インスタンスでのオートメーションによる再起動 (パッチ適用後の自動再起動など)、EC2 インスタンス外部のオートメーション (ネットワーク管理アプリケーションなど)、別の AWS サービス (など AWS Systems Manager)、人による手動再起動などがあります。

対応方法

slurmctld または slurmd ログをチェックして、インスタンスが再起動されたかどうかを確認できます。詳細については、「[PCS AWS のスケジューラログ](#)」および「[Amazon CloudWatch AWS を使用した PCS インスタンスのモニタリング](#)」を参照してください。次の slurmctld ログエントリの例は、インスタンスが再起動したことを示しています。

Example

```
[2024-09-12T06:42:50.393+00:00] validate_node_specs: Node Login-1 unexpectedly rebooted  
boot_time=1726123354 last_response=1726123285
```

パッチ適用による再起動

パッチを適用した後は、再起動が必要になることがよくあります。AWS PCS コンピューティングノードグループの一部である EC2 インスタンスに直接パッチを適用しないでください。EC2 インスタンスにパッチを適用する必要がある場合は、更新された Amazon マシンイメージ (AMI) にパッチを適用し、更新された AMI を使用するようにコンピューティングノードグループを更新する必要があります。これらのコンピューティングノードグループに対して AWS PCS が起動する新しい EC2 インスタンスは、更新された (パッチが適用された) AMI を使用します。詳細については、「[PCS のカスタム Amazon AWS マシンイメージ \(AMIs\)](#)」を参照してください。

PCS AWS のコンピューティングノードのブートストラップと登録に関する問題のトラブルシューティング

コンピューティングノードのブートストラップまたは PCS AWS クラスターへの適切な登録に失敗すると、次の症状が発生する可能性があります。

- ジョブが開始されない
- のインスタンスに接続できません AWS Systems Manager
- インスタンスが予期せずシャットダウンする
- インスタンスは継続的に置き換えられます

これらの障害は、EC2 インスタンスの起動中または PCS AWS コンピューティングノードのブートストラッププロセス中に発生する可能性があります。このトピックでは、PCS AWS ノードのブートストラッププロセス中の問題のトラブルシューティングに役立つ手順について説明します。EC2 インスタンスの起動のトラブルシューティングの詳細については、[Amazon EC2 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](#)」を参照してください。

ブートストラップエラーは、EC2 インスタンスが正常に起動されたが、PCS AWS クラスターの結合プロセス中に失敗した場合に発生します。ブートストラッププロセスには、主に 2 つのフェーズがあります。

- ノード登録 – EC2 インスタンスは [RegisterComputeNodeGroupInstance](#) AWS PCS API アクションを呼び出して PCS AWS サービスに登録します。以下の問題が原因で障害が発生する可能性があります。
 - アクセス許可
 - [インスタンスプロファイルが正しくありません](#)
 - ネットワーク
 - [AWS PCS エンドポイントに接続できない](#)
 - [PCS AWS エンドポイントの設定ミス](#)
 - [パブリック IP のないパブリックサブネット内のインスタンス](#)
 - [パブリックサブネットのマルチ NIC インスタンス](#)
 - クラスターシークレット
 - [クラスターシークレットが削除されたか、削除対象としてマークされている](#)
- Slurm 統合 – インスタンスは Slurm クラスターを実行してslurmd結合します。以下の問題が原因で障害が発生する可能性があります。
 - アクセス許可
 - [セキュリティグループの構成](#)
 - [Slurmctld がコンピューティングノードに ping を実行できない](#)
 - カスタム AMI セットアップ
 - [NVIDIA ドライバーが見つからない](#)
 - [ResumeTimeout に到達しました](#)

Slurm が PCS AWS で機能する仕組み

これは、Slurm の標準的な動作と Slurm が AWS PCS で動作する方法を比較するのに役立ちます。

標準 Slurm ジョブ処理

次の手順は、標準の Slurm ジョブ処理で発生します。

1. ジョブを送信すると、 はジョブslurmctldを検証してキューに入れます。
2. リソースが利用可能になると、 は既存のノードをslurmctld割り当てます。
3. slurmd デーモンは、割り当てられたノードでジョブを実行します。

PCS での Slurm AWS ジョブ処理

PCS AWS ジョブ処理では、次の手順を実行します。

1. ジョブを送信すると、 はジョブ `slurmctld` を検証してキューに入れます。
2. 追加の容量が必要な場合、AWS PCS はコンピューティングノードグループの起動テンプレートを使用して新しい EC2 インスタンスを起動します。
3. 新しいインスタンスはクラスターにブートストラップします。
 - a. インスタンスは PCS AWS に登録されます。
 - b. インスタンスは Slurm クラスターに参加します。
4. リソースの準備ができたら、 はノード (新しくブートストラップされたノードを含む) を `slurmctld` 割り当てます。
5. `slurmd` デーモンは、割り当てられたノードでジョブを実行します。

インスタンスログを取得する

コンピューティングノードのブートストラップ問題のトラブルシューティングの最初のステップは、インスタンスログの取得です。次のいずれかの方法を使用します。

AWS CLI

次のコマンドを使用して、コンピューティングノードからコンソール出力を取得します。

```
aws ec2 get-console-output --region us-east-1 --instance-id i-1234567890abcdef0 --output text
```

us-east-1 を AWS リージョンに、*i-1234567890abcdef0* をインスタンス ID に置き換えます。

AWS Systems Manager

Systems Manager を使用してインスタンスに接続できる場合は、ブートストラップログファイルを直接表示できます。

1. Systems Manager を使用してインスタンスに接続します。詳細については、「[Systems Manager ユーザーガイド](#)」の「セッションの開始」を参照してください。
2. ブートストラップログファイルを表示します。

```
sudo cat /var/log/amazon/pcs/bootstrap.log
```

Note

初期化フェーズ中に問題が発生した場合は、インスタンスに接続できるようになるまでに約 20 分待つ必要がある場合があります。Systems Manager と SSH サービスは、初期化が完了した後、または障害発生時にブートストラップ実行がタイムアウトに達したときのみ開始されます。

インスタンス ID から VPC/Subnet/Securityグループを取得する

コンピューティングノードの問題をトラブルシューティングするには、インスタンスに関連付けられた VPC、サブネット、およびセキュリティグループに関する情報を取得する必要がある場合があります。インスタンス IDs「」を参照してください[PCS AWS でのコンピューティングノードグループインスタンスの検索](#)。

AWS マネジメントコンソール

VPC、サブネット、セキュリティグループを取得するには

1. [Amazon EC2 コンソール](#)を開きます。
2. [Instances] を選択します。
3. インスタンステーブルで、インスタンス ID を選択します。
4. 表示されたインスタンスの概要で、VPC ID とサブネット ID を見つけます。
5. インスタンスの概要で、セキュリティタブを選択します。
6. セキュリティタブでセキュリティグループを見つけてます。

AWS CLI

インスタンスの VPC、サブネット、およびセキュリティグループ情報を取得するには、次のコマンドを使用します。

```
aws ec2 describe-instances --instance-ids i-1234567890abcdef0 --query  
'Reservations[*].Instances[*].  
{InstanceId:InstanceId,VpcId:VpcId,SubnetId:SubnetId,SecurityGroups:SecurityGroups[*].GroupI  
--output table
```

ノード登録の問題

ノード登録は、ブートストラップ中にコンピューティングノードによって実行される最初のアクションです。ノードは PCS API AWS エンドポイントを呼び出して、自身を PCS AWS に登録します。登録の失敗は通常、次のようなエラーメッセージを表示します。

```
<13>Nov  5 08:10:27 user-data: Recipe: aws-pcs-environment::node_registration
<13>Nov  5 08:10:27 user-data:  * ruby_block[Register NodeGroup Instance] action
  run[2024-11-05T08:10:27+00:00] INFO: Processing ruby_block[Register NodeGroup
  Instance] action run (aws-pcs-environment::node_registration line 19)
<13>Nov  5 08:15:46 user-data:
<13>Nov  5 08:15:46 user-data:
<13>Nov  5 08:15:46 user-data:
=====
<13>Nov  5 08:15:46 user-data:      Error executing action `run` on resource
  'ruby_block[Register NodeGroup Instance]'
<13>Nov  5 08:15:46 user-data:
=====
<13>Nov  5 08:15:46 user-data:
<13>Nov  5 08:15:46 user-data:      EOFError
```

インスタンスプロファイルが正しくありません

インスタンスを登録できない場合は、コンピューティングノードに関連付けられたインスタンスプロファイルに `pcs:RegisterComputeNodeGroupInstance` 許可があることを確認します。

有効なインスタンスプロファイルを作成する方法の詳細については、「」を参照してください [PCS AWS のインスタンスプロファイルを作成する](#)。

AWS PCS エンドポイントに接続できない

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場合は、PCS の VPC AWS エンドポイントが設定されているか、サブネットにインターネットアクセス用の NAT ゲートウェイへのルートがあることを確認してください。詳細については次を参照してください:

- Amazon Virtual Private Cloud AWS PrivateLink ガイドの [インターフェイス VPC エンドポイントを使用して AWS サービスにアクセスします](#)。
- [AWS PCS のエンドポイントとサービスクォータ](#)。
- 「[Amazon Virtual Private Cloud ユーザーガイド](#)」の「[VPC を他のネットワークに接続する Amazon Virtual Private Cloud](#)」

- [AWS PCS ネットワーク](#)

PCS AWS エンドポイントの設定ミス

次のようなエラーメッセージが表示された場合は、PCS VPC AWS エンドポイントに関連付けられているポリシーを確認します。

```
com.amazon.coral.security.AccessDeniedException: User: arn:aws:sts::xxx:assumed-role/rolename/i-instanceid is not authorized to perform: pcs:RegisterComputeNodeGroupInstance on resource: arn:aws:pcs:us-west-2:xxx:cluster/cluster-id as either the resource does not exist, some policy explicitly denies access, or no policy grants access
```

PCS の VPC AWS インターフェイスエンドポイントを設定する方法の詳細については、「」を参照してください[インターフェイスエンドポイント \(AWS PrivateLink\) AWS Parallel Computing Service を使用した へのアクセス](#)。

パブリック IP のないパブリックサブネット内のインスタンス

サブネットでパブリック IP の自動割り当てが有効になっていず、ルート設定でインターネットゲートウェイを使用している場合、インスタンスは AWS PCS API と通信できません。

インターネットゲートウェイを持つサブネット内のインスタンスには、パブリック IP アドレスが必要です。この問題を解決するには、次のいずれかのオプションを選択します。

- PCS AWS の VPC エンドポイントをクラスター VPC に追加します。これにより、パブリック IP アドレスがインターネットゲートウェイを通過することなく、インスタンスが AWS PCS と通信できるようになります。
- NAT ゲートウェイでプライベートサブネットを使用して、パブリック IP アドレスが不要になるようにします。
- サブネットまたは起動テンプレートを介して自動パブリック IP アドレス割り当てを有効にして、インスタンスがインターネットゲートウェイを介して API に連絡できるようにします。このオプションは、マルチネットワークインターフェイスインスタンスでは有効ではないことに注意してください。

パブリックサブネットのマルチ NIC インスタンス

複数のネットワークインターフェイス (NICs) を持つインスタンスタイプを使用する場合は、プライベートサブネットを使用する必要があります。

AWS パブリック IP アドレスは、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「[インスタンスの起動時にパブリック IPv4 アドレスを割り当てる](#)」を参照してください。Amazon EC2

マルチ NIC インスタンスタイプでは、PCS エンドポイントにアクセスするには、サブネット内の NAT AWS ゲートウェイまたは内部プロキシが必要です。または、PCS AWS の VPC エンドポイントをクラスター VPC に追加することもできます。

クラスターシークレットが削除されたか、削除対象としてマークされている

AWS Secrets Manager の Slurm 共有シークレットが削除された場合、または削除対象としてマークされている場合、コンピューティングノードは登録に失敗し、クラスターに障害が発生します。

AWS PCS は、クラスターの作成時に AWS Secrets Manager (名前形式: pcs!slurm-secret-`<cluster-id>`) で Slurm 共有シークレットを自動的に作成します。このシークレットは、クラスター内の安全な通信に必要です。詳細については、「[PCS AWS でのクラスターシークレットの使用](#)」を参照してください。

このシークレットが削除または削除対象としてマークされている場合、新しいノードはクラスターに参加できず、再起動するとコントローラーやその他のクラスターデーモン (slurmdや などslurmdbd) はクラスターに再参加できない可能性があります。

この問題を解決するには、削除されたシークレットがまだ復旧ウィンドウ内にある場合に復元できます。詳細な手順については、「[AWS Secrets Manager シークレットの復元](#)」を参照してください。

復旧ウィンドウの有効期限が切れると、シークレットは復元できず、影響を受ける AWS PCS クラスターは復元できません。同じ設定で新しいクラスターを作成する必要があります。AWS PCS は新しいスケジューラシークレットを自動的に作成します。

Slurm クラスター結合の問題

ノード登録が成功すると、コンピューティングノードは Slurm クラスターに参加しようとします。ノードのslurmdデーモンは Slurm コントローラーに連絡して、クラスターに登録します。通常、Slurm 結合の失敗は次のようなエラーメッセージを表示します。

```
<13>Nov  5 17:20:29 user-data: [2024-11-05T17:20:28+00:00] FATAL:
Mixlib::ShellOut::ShellCommandFailed: service[slurmd] (aws-pcs-slurm::finalize_slurm
line 18) had an error: Mixlib::ShellOut::ShellCommandFailed: Expected process to exit
with [0], but received '1'
```

```
<13>Nov  5 17:20:29 user-data: ---- Begin output of ["/usr/bin/systemctl", "--system",
"start", "slurmd"] ----
<13>Nov  5 17:20:29 user-data: STDOUT:
<13>Nov  5 17:20:29 user-data: STDERR: Job for slurmd.service failed because the
control process exited with error code. See "systemctl status slurmd.service" and
"journalctl -xe" for details.
<13>Nov  5 17:20:29 user-data: ---- End output of ["/usr/bin/systemctl", "--system",
"start", "slurmd"] ----
```

セキュリティグループの構成

コンピューティングノードと Slurm コントローラー間の通信を許可するようにセキュリティグループが正しく設定されていることを確認します。セキュリティグループは、次のトラフィックを許可する必要があります。

- と通信slurmdするための のポート 6817 slurmd
- から ping slurmdへのポート 6818 slurmd

セキュリティグループの要件の詳細については、以下のトピックを参照してください。

- [PCS AWS のセキュリティグループを作成する](#)
- [PCS AWS の起動テンプレートを作成する](#)
- [セキュリティグループの要件と考慮事項](#)

Important

クラスターの作成時にクラスターに関連付けたクラスターセキュリティグループは、コンピューティングノードがコントローラーと通信できるように、コンピューティングノードグループセキュリティグループでも設定する必要があります。

NVIDIA ドライバーが見つからない

インスタンスが正しくブートストラップされてもジョブが起動せず、インスタンスログに次のようなエラーメッセージが表示される場合は、NVIDIA ドライバーがない可能性があります。

```
<13>Dec  2 13:52:00 user-data: [2024-12-02T13:52:00.094+00:00] - /opt/aws/pcs/bin/
pcs_bootstrap_config_always.sh: INFO: nvidia-smi not found!
...
```



```
<13>Dec  2 13:54:10 user-data: Job for slurmd.service failed because the control
process exited with error code. See "systemctl status slurmd.service" and "journalctl
-xe" for details.
<13>Dec  2 13:54:12 user-data: [2024-12-02T13:54:12.718+00:00] - /opt/aws/pcs/bin/
pcs_bootstrap_finalize.sh: INFO: systemctl could not start slurmd!
```

インスタンスに接続してslurmdデーモンのステータスを確認すると、次のようなエラーが表示されることがあります。

```
$ systemctl status slurmd
...
fatal: can't stat gres.conf file /dev/nvidia0: No such file or directory
```

この問題を解決するには、カスタム AMI に NVIDIA ドライバーをインストールします。詳細については、「[ステップ 4 – \(オプション\) 追加のドライバー、ライブラリ、アプリケーションソフトウェアをインストールする](#)」を参照してください。

ResumeTimeout に到達しました

ノードが異常であるためにコンピューティングノードとその EC2 インスタンスが終了した場合、AWS PCS は AMI をサポートしていないか、ネットワークに問題がある可能性があります。EC2 インスタンスは、Slurm の ResumeTimeout に達するまで約 30 分間実行され、ノードをとしてマークしますDOWN。

インスタンスが正しくブートストラップされず、PCS に登録されていない場合 (EC2 AWS インスタンスのRegisterComputeNodeGroupInstance呼び出しがない場合)、インスタンスログに次のようなエラーメッセージがないか確認してください。

```
/opt/aws/pcs/bin/pcs_bootstrap_init.sh: No such file or directory
```

このエラーは、PCS AWS ブートストラップソフトウェアが AMI の一部ではないことを示します。この問題を解決するには、カスタム AMI に PCS AWS ブートストラップソフトウェアが含まれていることを確認します。詳細については、「[PCS のカスタム Amazon AWS マシンイメージ \(AMIs\)](#)」を参照してください。

Slurmctld がコンピューティングノードに ping を実行できない

インスタンスがブートストラッププロセスを正しく実行し、PCS AWS に登録されているが、それを表示してジョブを送信slurmctldできない場合、インスタンスはしばらくDOWNしてからに設定され、終了します。

これは、セキュリティグループの設定ミスが原因である可能性があります。たとえば、ポート 6817 で `slurmd` 通信が許可されているが `slurmctld`、ポート 6818 で `slurmctld ping` が許可されていない場合です `slurmd`。

「」で説明されているように、セキュリティグループにすべての必要なルールが含まれていることを確認します [セキュリティグループの要件と考慮事項](#)。

PCS AWS ユーザーガイドのドキュメント履歴

次の表に、PCS AWS のドキュメントの重要な変更点を示します。

日付	変更	ドキュメントの更新	API バージョンの更新
2026 年 3 月 10 日	更新された PCS エージェント	PCS エージェント 1.3.2-1 AWS の AMI トピックを更新しました。RHEL 8.10 と Rocky Linux 8.10 コンピューティングノードのブートストラップに影響する問題を修正しました。詳細については、「 PCS 用のカスタム AMIs AWS を構築するためのソフトウェアインストーラ 」および「 AWS PCS エージェントバージョン 」を参照してください。	該当なし
2026 年 2 月 11 日	AWS アジアパシフィック (ムンバイ) および欧州 (パリ) でリリースされた PCS	AWS PCS がアジアパシフィック (ムンバイ) (ap-south-1) および欧州 (パリ) (eu-west-3) で利用可能になりました。 CloudFormation テンプレートは、アジアパシフィック (ムンバイ) AWS リージョン および欧州 (パリ) で開始できます AWS リージョン。詳細については、「 CloudFormation を使用してサンプル PCS AWS クラス 」	該当なし

日付	変更	ドキュメントの更新	API バージョンの更新
		ターを作成する 」および「 CloudFormation サンプル PCS AWS クラスターを作成するための テンプレート 」を参照してください。	
2025 年 11 月 18 日	新機能: Slurm REST API	Slurm REST API が Slurm 25.05 以降でサポートされるようになりました。詳細については、「 PCS の Slurm AWS REST API 」を参照してください。	AWS SDK: 2025-11-18

日付	変更	ドキュメントの更新	API バージョンの更新
2025 年 11 月 10 日	新機能: Slurm CLI フィルタープラグインのサポート	AWS PCS は、Slurm CLI フィルタープラグインをサポートし、Slurm コントローラーに到達する前にジョブ送信パラメータを検証および変更するカスタム Lua スクリプトを実行するようになりました。CLI フィルターを使用して、カスタムポリシーを適用し、デフォルトのパラメータを設定し、ジョブの送信中にユーザーガイダンスを提供します。この機能には Slurm バージョン 25.05 以降が必要です。詳細については、「 Slurm CLI フィルタープラグインを使用して PCS AWS でのジョブ送信をカスタマイズする 」を参照してください。	該当なし
2025 年 11 月 7 日	更新された PCS エージェント	PCS エージェント 1.3.1-1 AWS の AMI トピックを更新しました。詳細については、「 PCS 用のカスタム AMIs AWS を構築するためのソフトウェアインストーラ 」および「 AWS PCS エージェントバージョン 」を参照してください。	該当なし

日付	変更	ドキュメントの更新	API バージョンの更新
2025 年 11 月 3 日	PCS エージェントと Slurm インストーラの更新	PCS エージェント 1.3.0-1 AWS および Slurm インストーラ 24.11.6-2、24.05.8-2、および 23.11.10-4 の AMI トピックを更新しました。サポートされているオペレーティングシステムのリストを更新しました。詳細については、「 PCS 用のカスタム AMIs AWS を構築するためのソフトウェアインストーラ 」および「 AWS PCS エージェントバージョン 」を参照してください。	該当なし
2025 年 10 月 23 日	更新されたコンテンツ: pcs-multi-cluster-login-configure.sh	マルチクラスターログインノード設定スクリプトのエラーを修正しました。詳細については、「 AWS PCS マルチクラスターログインノード設定スクリプトコード 」を参照してください。	該当なし

日付	変更	ドキュメントの更新	API バージョンの更新
2025 年 10 月 21 日	新機能: クラスターシークレットのローテーション	AWS PCS は、セキュリティを強化するためにクラスターシークレットローテーションをサポートするようになりました。詳細については、「PCS AWS でのクラスターシークレットの更新」を参照してください。 クラスターシークレットのローテーションをサポートするために、最小管理者権限を更新しました。詳細については、「PCS AWS の最小アクセス許可」を参照してください。	該当なし

日付	変更	ドキュメントの更新	API バージョンの更新
2025 年 10 月 17 日	新しいトピック: マルチクラスターログインノード設定スクリプト	複数の AWS PCS クラスターに接続するスタンドアロンログインノードを設定するスクリプトを提供する新しいトピックを追加しました。このスクリプトは、複数の Slurm sackd デーモンの設定を自動化し、クラスターインタラクション用のアクティベーションスクリプトを作成します。 詳細については、「スタンドアロンログインノードを PCS AWS の複数のクラスターに接続する」を参照してください。	該当なし
2025 年 10 月 16 日	Slurm 25.05 用に更新	Slurm 25.05 サポートのユーザーガイドを更新しました。Slurm 25.05 がデフォルトバージョンになりました。詳細については次を参照してください: • PCS の Slurm AWS バージョン • PCS 用のカスタム AMIs AWS を構築するためのソフトウェアインストーラ • PCS サンプル AWS AMIs リリースノート	該当なし

日付	変更	ドキュメントの更新	API バージョンの更新
2025 年 10 月 16 日	更新された PCS エージェント	PCS エージェント 1.2.2-1 AWS の AMI トピックを更新しました。 詳細については、「 PCS 用のカスタム AMIs AWS を構築するためのソフトウェアインストーラ 」および「 AWS PCS エージェントバージョン 」を参照してください。	該当なし

日付	変更	ドキュメントの更新	API バージョンの更新
2025 年 10 月 2 日	新機能: Slurm ノードの再起動、クラスターの更新、カスタム Slurm 設定	AWS PCS は、複数の新機能のサポートを追加します。 • Slurm ノードの再起動 – Slurm のネイティブ <code>scontrol reboot</code> コマンドを使用して、インスタンスを置き換えることなくコンピューティングノードを再起動します。詳細については、「PCS での Slurm を使用したコンピューティングノード AWS の再起動」を参照してください。 • クラスターの更新 – 再ビルドせずに作成後にクラスター設定を変更します。詳細については、「PCS AWS でのクラスターの更新」を参照してください。 • Slurm カスタム設定 – クラスター、キュー、コンピューティングノードグループのリソース全体で高度な Slurm パラメータを設定します。詳細については、「PCS でのカスタム Slurm AWS 設定」	2025-10-01

日付	変更	ドキュメントの更新	API バージョンの更新
		の設定 」を参照してください。	
2025 年 9 月 23 日	新しいトラブルシューティングトピック: コンピューティングノードのブートストラップの問題	コンピューティングノードのブートストラップの問題の診断と解決に関するトラブルシューティングガイダンスを追加しました。詳細については、「 PCS AWS のコンピューティングノードのブートストラップと登録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。	該当なし

日付	変更	ドキュメントの更新	API バージョンの更新
2025 年 9 月 17 日	新機能: ML のキャパシティブロック	AWS PCS で ML 用の Amazon EC2 キャパシティブロックがサポートされるようになりました。これにより、クラスターの GPU ベースの高速コンピューティングインスタンスを予約できます。詳細については、「PCS での ML での Amazon EC2 AWS キャパシティブロックの使用」を参照してください。 キャパシティブロックをサポートする最小アクセス許可は、サービス管理者の最小アクセス許可の一部になりました。詳細については、「PCS AWS の最小アクセス許可」を参照してください。	2025-09-17
2025 年 9 月 11 日	AWS 管理ポリシーの更新	AWS PCS は、キャパシティブロックをサポートするように AWSPCSServiceRolePolicy を更新しました。詳細については、「AWS AWS Parallel Computing Service の マネージドポリシー」を参照してください。	該当なし

日付	変更	ドキュメントの更新	API バージョンの更新
2025 年 8 月 14 日	インスタンスプロファイルドキュメントの更新	IAM ロールとインスタンスプロファイルを作成するための包括的な CLI 手順を使用して、インスタンスプロファイルドキュメントを強化しました。 を使用してインスタンスプロファイルを設定する step-by-step の手順を追加し、AWS CLI PCS AWS で使用されるインスタンスプロファイルを検索するためのガイダンスを改善しました。 詳細については、「AWS Parallel Computing Service の IAM インスタンスプロファイル」を参照してください。	2025-08-14

日付	変更	ドキュメントの更新	API バージョンの更新
2025 年 8 月 1 日	新しいトピック: SPANK プラグイン	PCS クラスターでのジョブの起動と実行中に Slurm の動作を拡張および変更するために使用できる SPANK (Slurm Plugin Architecture for Node and job Kontrol) AWS プラグインのドキュメントを追加しました。 詳細については、「SPANK プラグインを使用して AWS PCS で Slurm 機能を拡張する」を参照してください。	該当なし
2025 年 8 月 1 日	IPv6 ネットワークのサポート	PCS クラスターの作成時に IPv6 AWS ネットワーキングのサポートが追加されました。クラスターのネットワークタイプとして IPv6 を選択できるようになりました。それに応じて、VPC 要件、サブネット設定、セキュリティグループ設定、クラスター作成手順が更新されます。 詳細については、「AWS PCS VPC とサブネットの要件と考慮事項」および「PCS AWS でのクラスターの作成」を参照してください。	2025-08-01

日付	変更	ドキュメントの更新	API バージョンの更新
2025 年 7 月 3 日	AWS 欧州 (ロンドン) で PCS をリリース	AWS PCS が欧州 (ロンドン) (eu-west-2) で利用可能になりました。 CloudFormation テンプレートは、欧州 (ロンドン) で開始できます AWS リージョン。詳細については、「CloudFormation を使用してサンプル PCS AWS クラスターを作成する」および「CloudFormation サンプル PCS AWS クラスターを作成するための テンプレート」を参照してください。	該当なし

日付	変更	ドキュメントの更新	API バージョンの更新
2025 年 7 月 1 日	コンソールの手順を更新しました	コンソールでクラスターとコンピューティングノードグループを作成するときに、基本的なインスタンスプロファイルとセキュリティグループを AWS PCS で作成できるようになりました。詳細については、以下を参照してください。 • PCS AWS でのクラスターの作成 • PCS AWS でのコンピューティングノードグループの作成 • AWS Parallel Computing Service の IAM インスタンスプロファイル	該当なし
2025 年 6 月 23 日	新しい管理ポリシー: AWSPCSComputeNodePolicy	PCS AWS クラスターに接続するためのアクセス許可を PCS AWS コンピューティングノードに付与する新しいマネージドポリシーを追加しました。詳細については、「AWS マネージドポリシー: AWSPCSComputeNodePolicy」を参照してください。	該当なし

日付	変更	ドキュメントの更新	API バージョンの更新
2025 年 6 月 19 日	新しいトピック: ジョブ完了ログ	ジョブ完了ログを使用して、ジョブの完了時に追加コストなしでジョブの詳細を記録します。詳細については、「 PCS AWS のジョブ完了ログ 」を参照してください。	該当なし

日付	変更	ドキュメントの更新	API バージョンの更新
2025 年 6 月 18 日	AWS での PCS リリース AWS GovCloud (US)	AWS PCS が AWS GovCloud (米国東部) (us-gov-east-1) および AWS GovCloud (米国西部) (us-gov-west-1) で利用可能になりました。 CloudFormation テンプレートは、で開始できます AWS GovCloud (US) Regions。詳細については、「CloudFormation を使用してサンプル PCS AWS クラスターを作成する」および「CloudFormation サンプル PCS AWS クラスターを作成するための テンプレート」を参照してください。 の PCS AWS サービスエンドポイントの詳細については AWS GovCloud (US) Regions、「」を参照してくださいAWS PCS のエンドポイントとサービスクォータ。 の違いの詳細については AWS GovCloud (US) Regions、「AWS GovCloud (US) ユーザーガイド」の「AWS の PCS AWS GovCloud	該当なし

日付	変更	ドキュメントの更新	API バージョンの更新
		(US) 」を参照してください。	
2025 年 6 月 18 日	更新された PCS エージェント	PCS エージェント 1.2.1-1 AWS の AMI トピックを更新しました。詳細については、「 PCS 用のカスタム AMIs AWS を構築するためのソフトウェアインストーラ 」を参照してください。	該当なし
2025 年 5 月 15 日	新機能: アカウンティング	Slurm アカウンティングが Slurm 24.11 以降でサポートされるようになりました。詳細については、「 PCS での Slurm AWS アカウンティング 」を参照してください。	AWS SDK: 2025-05-15
2025 年 5 月 15 日	Slurm 24.11 用に更新	Slurm 24.11.5 サポートのユーザーガイドを更新しました。詳細については次を参照してください: • PCS の Slurm AWS バージョン • PCS 用のカスタム AMIs AWS を構築するためのソフトウェアインストーラ • PCS サンプル AWS AMIs リリースノート	該当なし

日付	変更	ドキュメントの更新	API バージョンの更新
2025 年 5 月 5 日	更新された Slurm バージョンに関するよくある質問	Slurm バージョンに関するよくある質問 (FAQ) を、サポート終了 (EOL) 間近またはそれ以降に更新しました。詳細については、「 PCS の Slurm AWS バージョンに関するよくある質問 」を参照してください。	該当なし
2025 年 4 月 17 日	新しいトピック: コンピューティングノードグループの詳細を取得する方法	ID、ARN、AMI ID など、PCS AWS コンピューティングノードグループの詳細を取得する方法について説明します。詳細については、「 PCS AWS でコンピューティングノードグループの詳細を取得する 」を参照してください。	該当なし
2025 年 4 月 2 日	更新された Slurm インストーラ	Slurm インストーラ 24.05.7-1 の AMI トピックを更新しました。詳細については、「 PCS 用のカスタム AMIs AWS を構築するためのソフトウェアインストーラ 」を参照してください。	該当なし

日付	変更	ドキュメントの更新	API バージョンの更新
2025 年 3 月 28 日	コンピューティングノードグループとキューの最大数のクォータを追加	クラスターあたりのコンピューティングノードグループの最大数とクラスターあたりのキューの最大数の内部非調整クォータを追加しました。 詳細については、「 内部クォータ 」を参照してください。	該当なし
2025 年 3 月 14 日	CloudFormation テンプレートのプロパティキーを変更しました	Id は、CloudFormation テンプレートの CustomLaunchTemplate プロパティTemplateId のになりました。詳細については、「 PCS AWS 用の CloudFormation テンプレートの一部 」の リソース を参照してください。	該当なし

日付	変更	ドキュメントの更新	API バージョンの更新
2025 年 3 月 13 日	PCS エージェントと AWS Slurm のバージョン情報を追加しました	AWS PCS エージェントの各バージョンの変更を説明する新しいトピックを追加しました。詳細については、「AWS PCS エージェントバージョン」を参照してください。 Slurm バージョントピックに、Slurm の AWS PCS サポートの重要なサポート日と詳細なリリースノートについて説明する詳細情報を追加しました。詳細については、「PCS の Slurm AWS バージョン」を参照してください。	該当なし
2025 年 3 月 7 日	更新された PCS エージェント	PCS エージェント 1.2.0-1 AWS の AMI トピックを更新しました。詳細については、「 PCS 用のカスタム AMIs AWS を構築するためのソフトウェアインストーラ 」を参照してください。	該当なし

日付	変更	ドキュメントの更新	API バージョンの更新
2025 年 2 月 3 日	PCS AWS CloudFormation での AWS の使用に関するトピックを追加しました	ユーザーガイドに、PCS AWS CloudFormation でを使用する方法の例を示すトピックを追加しました。このトピックでは、サンプルの CloudFormation テンプレートを使用してサンプル PCS AWS クラスターを作成する手順と、そのテンプレートのセクションを簡単に説明します。詳細については、「 CloudFormation と PCS AWS の使用を開始する 」を参照してください。	該当なし
2024 年 12 月 18 日	Slurm 24.05 用に更新	Slurm 24.05 サポートのユーザーガイドを更新しました。詳細については、「 PCS 用のカスタム AMIs AWS を構築するためのソフトウェアインストール 」および「 PCS サンプル AWS AMIs リリースノート 」を参照してください。	該当なし
2024 年 12 月 18 日	Slurm 23.11 サンプル AMIs の NVIDIA バージョンを更新	Slurm 23.11 サンプル AMIs。詳細については、「 PCS サンプル AWS AMIs リリースノート 」を参照してください。	該当なし

日付	変更	ドキュメントの更新	API バージョンの更新
2024 年 12 月 17 日	更新された Slurm インストーラ	Slurm インストーラ 23.11.10-3 の AMI トピックを更新しました。詳細については、「 PCS 用のカスタム AMIs AWS を構築するためのソフトウェアインストーラ 」を参照してください。	該当なし
2024 年 12 月 13 日	更新された PCS エージェント	PCS エージェント 1.1.1-1 AWS の AMI トピックを更新しました。詳細については、「 PCS 用のカスタム AMIs AWS を構築するためのソフトウェアインストーラ 」を参照してください。	該当なし
2024 年 12 月 6 日	PCS エージェントと Slurm インストーラの更新	PCS エージェント 1.1.0-1 AWS と Slurm インストーラ 23.11.10-2 の AMI トピックを更新しました。詳細については、「 PCS 用のカスタム AMIs AWS を構築するためのソフトウェアインストーラ 」を参照してください。	該当なし
2024 年 12 月 6 日	OS サポートに関するトピックを追加しました	詳細については、「 PCS AWS でサポートされているオペレーティングシステム 」を参照してください。	該当なし

日付	変更	ドキュメントの更新	API バージョンの更新
2024 年 11 月 8 日	再編成されたユーザーガイド	ユーザーガイドを再編成して、トピックを最上位レベルにし、いくつかのトピックを独自のページに移動し、同様のトピックをグループ化しました。	該当なし
2024 年 11 月 7 日	更新された AMI トピック	Slurm 23.11.10 および libjwt 17.0 の AMI トピックを更新しました。詳細については、「 PCS 用のカスタム AMIs AWS を構築するためのソフトウェアインストーラ 」および「 ステップ 3 – Slurm をインストールする 」を参照してください。 AMIs。詳細については、「 PCS サンプル AWS AMIs リリースノート 」を参照してください。	該当なし
2024 年 11 月 7 日	PCS での暗号化された EBS AWS ボリュームの使用に関する新しいトピックを追加しました	PCS の暗号化された EBS ボリュームに必要な KMS AWS キーポリシーを説明するトピックを追加しました。詳細については、「 PCS の暗号化された EBS ボリュームで使用するために必要な KMS AWS キーポリシー 」を参照してください。	該当なし

日付	変更	ドキュメントの更新	API バージョンの更新
2024 年 10 月 18 日	AWS PCS エージェント 1.0.1-1 をリリース	PCS エージェントバージョン 1.0.1-1 AWS を参照するように AMI 関連のドキュメントを更新しました。詳細については、「 PCS 用のカスタム AMIs AWS を構築するためのソフトウェアインストーラ 」および「 ステップ 2 – AWS PCS エージェントをインストールする 」を参照してください。	該当なし
2024 年 10 月 10 日	トラブルシューティングの章を追加しました	再起動後に自動的に置き換えられる EC2 インスタンスに関するトピックを含むトラブルシューティングの章を追加しました。詳細については、「 AWS Parallel Computing Service の問題のトラブルシューティング 」を参照してください。	該当なし

日付	変更	ドキュメントの更新	API バージョンの更新
2024 年 9 月 23 日	API アクションとサービス管理者の最小アクセス許可を更新しました	アクセス ec2:DescribeInstanceTypeOfferings 許可が CreateComputeNodeGroup および UpdateComputeNodeGroup API アクションに必要になりました。詳細については、「 PCS AWS の最小アクセス許可 」を参照してください。	該当なし
2024 年 9 月 5 日	サービス管理者の最小アクセス許可の IAM ポリシーの例を更新しました	詳細については、「 サービス管理者の最小アクセス許可 」を参照してください。	該当なし
2024 年 9 月 5 日	管理ポリシーページで JSON にアクセス許可が欠落している	これはドキュメントの単なる修正です。実際の管理ポリシーは変更されませんでした。詳細については、「 AWS AWS Parallel Computing Service の マネージドポリシー 」を参照してください。	該当なし
2024 年 8 月 28 日	管理ポリシーページが追加されました	詳細については、「 AWS AWS Parallel Computing Service の マネージドポリシー 」を参照してください。	該当なし

日付	変更	ドキュメントの更新	API バージョンの更新
2024 年 8 月 28 日	AWS PCS リリース	PCS AWS ユーザーガイドの初回リリース。	AWS SDK: 2024-08-28

AWS 用語集

最新の AWS 用語については、「AWS の用語集 リファレンス」の[AWS 「用語集」](#)を参照してください。

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。