AWS ParallelCluster ユーザーガイド (v2)

AWS ParallelCluster

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

AWS ParallelCluster: AWS ParallelCluster ユーザーガイド (v2)

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

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

Table of Contents

とは AWS ParallelCluster 1
料金 1
セットアップ AWS ParallelCluster 2
のインストール AWS ParallelCluster 2
AWS ParallelCluster 仮想環境に をインストールする (推奨) 2
pip を使用して AWS ParallelCluster 非仮想環境に をインストールする 3
インストール後に実行する手順 3
各環境の詳細な手順 4
仮想環境 4
リナックス 6
macOS 11
Windows 13
の設定 AWS ParallelCluster 16
ベストプラクティス 24
ベストプラクティス:マスターインスタンスタイプの選択 24
ベストプラクティス: ネットワークパフォーマンス 25
ベストプラクティス: 予算アラート 25
ベストプラクティス: AWS ParallelCluster クラスターを新しいマイナーバージョンまたはパッチバージョンに移動する 26
CfnCluster から への移行 AWS ParallelCluster 27
サポート対象の リージョン 28
の使用 AWS ParallelCluster 31
ネットワークの設定 31
AWS ParallelCluster 1 つのパブリックサブネット内の 32
AWS ParallelCluster 2 つのサブネットの使用 33
AWS ParallelCluster を使用して接続された単一のプライベートサブネット内の AWS Direct Connect 34
AWS ParallelCluster スawsbatchケジューラを使用する 35
カスタムブートストラップアクション 36
設定 38
引数 38
例 38
Amazon S3 の操作 40
例 40

スポットインスタンスの操作 ... 41
シナリオ 1: 実行中のジョブがないスポットインスタンスが中断される ... 41
シナリオ 2: 単一ノードジョブを実行しているスポットインスタンスが中断される ... 42
シナリオ 3: マルチノードジョブを実行しているスポットインスタンスが中断される ... 43
AWS Identity and Access Management での ロール AWS ParallelCluster ... 45
クラスター作成のデフォルト設定 ... 45
Amazon EC2 ロールに既存の IAM ロールを使用する ... 45
AWS ParallelCluster インスタンスポリシーとユーザーポリシーの例 ... 46
でサポートされているスケジューラ AWS ParallelCluster ... 87
Son of Grid Engine ... 88
Slurm Workload Manager ... 88
Torque Resource Manager ... 101
AWS Batch ... 101
Tagging ... 109
Amazon CloudWatch ダッシュボード ... 112
Amazon CloudWatch Logs との統合 ... 114
Elastic Fabric Adapter ... 117
インテル Select ソリューション ... 118
Intel MPI を有効にする ... 119
インテル HPC プラットフォームの仕様 ... 121
Arm パフォーマンスライブラリ ... 121
Amazon DCV を介してヘッドノードに接続する ... 123
Amazon DCV HTTPS 証明書 ... 124
Amazon DCV のライセンス ... 125
`pcluster update` を使用する ... 125
AMI のパッチ適用と EC2 インスタンスの交換 ... 128
ヘッドノードインスタンスの更新または交換 ... 129
インスタンスストアの制限事項 ... 129
インスタンスストアの制限回避策 ... 130
クラスターのヘッドノードを停止して起動します。 ... 131
AWS ParallelCluster CLI コマンド ... 133
`pcluster` ... 133
引数 ... 133
サブコマンド: ... 133
`pcluster configure` ... 134
`pcluster create` ... 135

pcluster createami 137
pcluster dcv 141
pcluster delete 143
pcluster instances 145
pcluster list 146
pcluster ssh 147
pcluster start 148
pcluster status 149
pcluster stop 150
pcluster update 151
pcluster version 153
pcluster-config 154
名前付き引数 154
設定 156
[レイアウト] 157
[global] セクション 157
cluster_template 157
update_check 158
sanity_check 158
[aws] セクション 158
[aliases] セクション 159
[cluster] セクション 160
additional_cfn_template 162
additional_iam_policies 162
base_os 163
cluster_resource_bucket 165
cluster_type 166
compute_instance_type 167
compute_root_volume_size 167
custom_ami 168
cw_log_settings 169
dashboard_settings 169
dcv_settings 170
desired_vcpus 170
disable_cluster_dns 171
disable_hyperthreading 171

ebs_settings 172
ec2_iam_role 173
efs_settings 173
enable_efa 173
enable_efa_gdr 174
enable_intel_hpc_platform 175
encrypted_ephemeral 176
ephemeral_dir 176
extra_json 176
fsx_settings 177
iam_lambda_role 177
initial_queue_size 178
key_name 179
maintain_initial_size 179
master_instance_type 180
master_root_volume_size 180
max_queue_size 181
max_vcpus 181
min_vcpus 182
placement 182
placement_group 183
post_install 184
post_install_args 184
pre_install 184
pre_install_args 185
proxy_server 185
queue_settings 185
raid_settings 186
s3_read_resource 187
s3_read_write_resource 187
scaling_settings 187
scheduler 188
shared_dir 189
spot_bid_percentage 189
spot_price 190
tags 190

template_url 191
vpc_settings 192
[compute_resource] セクション 192
initial_count 193
instance_type 193
max_count 194
min_count 194
spot_price 195
[cw_log] セクション 195
enable 195
retention_days 196
[dashboard] セクション 196
enable 196
[dcv] セクション 197
access_from 198
enable 198
port 198
[ebs] セクション 199
shared_dir 200
ebs_kms_key_id 200
ebs_snapshot_id 201
ebs_volume_id 201
encrypted 201
volume_iops 201
volume_size 202
volume_throughput 203
volume_type 204
[efs] セクション 205
efs_fs_id 206
efs_kms_key_id 207
encrypted 207
performance_mode 207
provisioned_throughput 208
shared_dir 209
throughput_mode 209
[fsx] セクション 209

auto_import_policy 212
automatic_backup_retention_days 213
copy_tags_to_backups 213
daily_automatic_backup_start_time 214
data_compression_type 214
deployment_type 215
drive_cache_type 216
export_path 216
fsx_backup_id 217
fsx_fs_id 217
fsx_kms_key_id 218
import_path 218
imported_file_chunk_size 219
per_unit_storage_throughput 219
shared_dir 220
storage_capacity 220
storage_type 221
weekly_maintenance_start_time 223
[queue] セクション 223
compute_resource_settings 224
compute_type 224
disable_hyperthreading 225
enable_efa 226
enable_efa_gdr 226
placement_group 227
[raid] セクション 228
shared_dir 228
ebs_kms_key_id 229
encrypted 229
num_of_raid_volumes 229
raid_type 230
volume_iops 230
volume_size 231
volume_throughput 232
volume_type 232
[scaling] セクション 233

scaledown_idletime 234
[vpc] セクション 234
additional_sg 235
compute_subnet_cidr 235
compute_subnet_id 235
master_subnet_id 235
ssh_from 236
use_public_ips 236
vpc_id 237
vpc_security_group_id 237
例 40
Slurm の例 238
SGE および Torque の例 239
AWS Batch 例 240
の AWS ParallelCluster 仕組み 242
AWS ParallelCluster プロセス 242
SGE and Torque integration processes 243
Slurm integration processes 249
AWS が使用する サービス AWS ParallelCluster 249
AWS Auto Scaling 250
AWS Batch 251
AWS CloudFormation 251
Amazon CloudWatch 251
Amazon CloudWatch Logs 252
AWS CodeBuild 252
Amazon DynamoDB 252
Amazon Elastic Block Store 253
Amazon Elastic Compute Cloud 253
Amazon Elastic Container Registry 253
Amazon EFS 253
Amazon FSx for Lustre 254
AWS Identity and Access Management 254
AWS Lambda 254
Amazon DCV 255
Amazon Route 53 255
Amazon Simple Notification Service 255

Amazon Simple Queue Service 256
Amazon Simple Storage Service 256
Amazon VPC 256
AWS ParallelCluster Auto Scaling 257
スケールアップ 258
スケールダウン 259
静的クラスター 259
チュートリアル 260
で最初のジョブを実行する AWS ParallelCluster 260
インストールを確認する 260
初めてクラスターを作成する 261
ヘッドノードにログインする 261
SGE を使用して最初のジョブを実行する 262
カスタム AMI AWS ParallelCluster の構築 263
AMI AWS ParallelCluster をカスタマイズする方法 264
AMI を変更する 264
カスタム AMI AWS ParallelCluster の構築 267
実行時にカスタム AMI を使用する 268
AWS ParallelCluster とスawsbatchケジューラを使用した MPI ジョブの実行 269
クラスターの作成 269
ヘッドノードにログインする 261
を使用して最初のジョブを実行する AWS Batch 271
マルチノード並列環境で MPI ジョブを実行する 273
カスタム KMS キーを使用したディスク暗号化 277
ロールの作成 278
キーのアクセス許可を付与する 278
クラスターの作成 269
マルチキューモードのチュートリアル 279
マルチキューモードで AWS ParallelCluster でジョブを実行する 279
開発 292
カスタム AWS ParallelCluster クックブックのセットアップ 292
ステップ 292
カスタム AWS ParallelCluster ノードパッケージのセットアップ 294
ステップ 294
トラブルシューティング 296
ログの取得と保存 296

スタックデプロイ時のトラブルシューティング 297
マルチキューモードクラスターでの問題のトラブルシューティング 297
キーログ 298
ノードの初期化に関する問題のトラブルシューティング 299
予期しないノードの置換や終了のトラブルシューティング 301
問題のあるインスタンスやノードの置換、終了、電源オフ 302
その他の既知のノードやジョブの問題のトラブルシューティング 303
シングルキューモードのクラスターにおける問題のトラブルシューティング 303
キーログ 303
起動および参加オペレーションの失敗のトラブルシューティング 305
スケーリング問題のトラブルシューティング 305
他のクラスター関連の問題のトラブルシューティング 306
プレイスメントグループとインスタンスの起動に関する問題 306
置き換えられないディレクトリ 307
Amazon DCV の問題のトラブルシューティング 308
Amazon DCV のログ 308
Amazon DCV インスタンスタイプメモリ 308
Ubuntu の Amazon DCV の問題 308
AWS Batch 統合されたクラスターにおける問題のトラブルシューティング 309
ヘッドノードの問題 309
AWS Batch マルチノード並列ジョブの送信に関する問題 309
コンピューティングの問題 309
ジョブの失敗 309
リソースの作成に失敗したときのトラブルシューティング 309
IAM ポリシーのサイズに関する問題のトラブルシューティング 311
追加のサポート 311
AWS ParallelCluster サポートポリシー 312
セキュリティ 313
が使用する サービスのセキュリティ情報 AWS ParallelCluster 313
データ保護 314
データの暗号化 315
関連情報 316
Identity and Access Management 317
コンプライアンス検証 317
TLS 1.2 の適用 318
現在サポートされているプロトコルの確認 319

OpenSSL と Python のコンパイル 320
リリースノートとドキュメント履歴 322
..... ccclxvi

とは AWS ParallelCluster

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

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

料金

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

セットアップ AWS ParallelCluster

トピック

- のインストール AWS ParallelCluster
- の設定 AWS ParallelCluster
- ベストプラクティス
- CfnCluster から への移行 AWS ParallelCluster
- サポート対象の リージョン

のインストール AWS ParallelCluster

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

インストール方法 AWS ParallelCluster：

- 仮想環境を使用する (推奨)
- pip を使用する

最新の CLI のバージョン番号は、GitHub のリリースページで確認できます。

このガイドのコマンド例では、Python v 3 がインストールされていることを前提としています。pip コマンド例は pip3 バージョンを使用します。

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

AWS ParallelCluster 仮想環境に をインストールすることをお勧めします。 AWS ParallelCluster でをインストールしようとしたときに問題が発生した場合はpip3、AWS ParallelCluster 仮想環境にをインストールして、ツールとその依存関係を分離できます。または、通常使用しているものと異なるバージョンの Python を使用することもできます。

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

Linux、Windows、macOS AWS ParallelCluster での のプライマリディストリビューション方法は`pip`、Python のパッケージマネージャーである です。Python パッケージとその依存関係をインストール、アップグレード、削除する方法です。

現在の AWS ParallelCluster バージョン

AWS ParallelCluster は定期的に更新されます。最新バージョンがあるかどうかを確認するには、「GitHub のリリースページ」を参照してください。

`pip` とサポートされているバージョンの Python がすでにある場合は、次のコマンド AWS ParallelCluster を使用して をインストールできます。Python version 3+ がインストールされている場合は、**`pip3`** コマンドの使用が推奨されます。

```
$ pip3 install "aws-parallelcluster<3.0" --upgrade --user
```

インストール後に実行する手順

インストール後 AWS ParallelCluster、 `PATH`変数に実行可能ファイルパスを追加する必要がある場合があります。プラットフォーム固有の手順については、以下のトピックを参照してください。

- Linux – コマンドラインパスに AWS ParallelCluster 実行可能ファイルを追加する
- macOS – コマンドラインパスに AWS ParallelCluster 実行可能ファイルを追加する
- Windows – コマンドラインパスに AWS ParallelCluster 実行可能ファイルを追加する

を実行して、 が正しく AWS ParallelCluster インストールされていることを確認できます`pcluster version`。

```
$ pcluster version
2.11.9
```

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

```
$ pip3 install "aws-parallelcluster<3.0" --upgrade --user
```

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

```
$ pip3 uninstall "aws-parallelcluster<3.0"
```

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

各環境の詳細な手順

- 仮想環境に AWS ParallelCluster をインストールする (推奨)
- Linux AWS ParallelCluster に をインストールする
- macOS AWS ParallelCluster に をインストールする
- Windows AWS ParallelCluster に をインストールする

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

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

前提条件

- pip と Python がインストールされていることを確認します。pip3 と、Python 3 バージョン 3.8 の使用をお勧めします。Python 2 を使用している場合は、pip3 の代わりに pip を、venv の代わりに virtualenv を使用します。

AWS ParallelCluster 仮想環境に をインストールするには

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

   Linux, macOS, or Unix

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

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

Windows

```
C:\>pip3 install --user --upgrade virtualenv
```

`exit` を実行して現在のコマンドプロンプトを終了し、新しいコマンドプロンプトを開いて環境への変更を取得します。

2. 仮想環境を作成して名前を付けます。

   Linux, macOS, or Unix

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

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

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

   Windows

   ```
   C:\>virtualenv %USERPROFILE%\apc-ve
   ```

3. 新しい仮想環境をアクティブ化します。

   Linux, macOS, or Unix

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

   Windows

   ```
   C:\>%USERPROFILE%\apc-ve\Scripts\activate
   ```

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

   Linux, macOS, or Unix

   ```
   (apc-ve)~$ python3 -m pip install --upgrade "aws-parallelcluster<3.0"
   ```

Windows

```
(apc-ve) C:\>pip3 install --upgrade "aws-parallelcluster<3.0"
```

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

   Linux, macOS, or Unix

   ```
   $ pcluster version
   2.11.9
   ```

   Windows

   ```
   (apc-ve) C:\>pcluster version
   2.11.9
   ```

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

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

Linux, macOS, or Unix

```
(apc-ve)~$ python3 -m pip install --upgrade "aws-parallelcluster<3.0"
```

Windows

```
(apc-ve) C:\>pip3 install --upgrade "aws-parallelcluster<3.0"
```

Linux AWS ParallelCluster に をインストールする

Python のパッケージマネージャーである を使用して`pip`、 AWS ParallelCluster とその依存関係をほとんどの Linux ディストリビューションにインストールできます。まず、Python と `pip` がインストールされているかどうかを判断します。

1. お使いの Linux に Python と `pip` が付属していることを確認するには、`pip --version` を実行します。

```
$ pip --version
```

pip をインストールした場合は、「pip AWS ParallelCluster でインストール」トピックに進みます。それ以外の場合は、ステップ 2 に進みます。

2. Python がインストールされているかどうかを確認するには、python --version を実行します。

```
$ python --version
```

Python 3 バージョン 3.6 以降または Python 2 バージョン 2.7 がインストールされている場合は、「pip AWS ParallelCluster でのインストール」トピックに進みます。それ以外の場合は、Python をインストールし、この手順に戻って pip をインストールします。

3. pip をインストールするには、Python Packaging Authority より提供されているスクリプトを使用します。

4. curl コマンドを使用してインストールスクリプトをダウンロードします。

```
$ curl -O https://bootstrap.pypa.io/get-pip.py
```

5. Python でスクリプトを実行して、pip の最新バージョンとその他の必要なサポートパッケージをダウンロードしてインストールします。

```
$ python get-pip.py --user
```

or

```
$ python3 get-pip.py --user
```

--user スイッチを含めると、スクリプトは pip をパス ~/.local/bin にインストールします。

6. pip を含むフォルダが PATH 変数の一部であることを確認するには、以下の操作を行います。

   a. ユーザーフォルダーでシェルのプロファイルスクリプトを見つけます。現在使用しているシェルが不明な場合は、basename $SHELL を実行します。

```
$ ls -a ~
```

```
.  ..  .bash_logout  .bash_profile  .bashrc  Desktop  Documents  Downloads
```

- Bash - `.bash_profile`、`.profile`、または `.bash_login`
- Zsh - `.zshrc`
- Tcsh - `.tcshrc`、`.cshrc`、または `.login`

b. 次の例のように、プロファイルスクリプトの末尾にエクスポートコマンドを追加します。

```
export PATH=~/.local/bin:$PATH
```

エクスポートコマンドでは、パス (この例では `~/.local/bin`) が、既存の PATH 変数の前に挿入されます。

c. 変更を適用するには、プロファイルを現在のセッションに再ロードします。

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

7. `pip` が正しくインストールされたことを確認します。

```
$ pip3 --version
pip 21.3.1 from ~/.local/lib/python3.6/site-packages (python 3.6)
```

セクション

- AWS ParallelCluster で をインストールする pip
- コマンドラインパスに AWS ParallelCluster 実行可能ファイルを追加する
- Linux での Python のインストール

AWS ParallelCluster で をインストールする **pip**

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

```
$ python3 -m pip install "aws-parallelcluster<3.0" --upgrade --user
```

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

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

```
$ pcluster version
2.11.9
```

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

```
$ python3 -m pip install "aws-parallelcluster<3.0" --upgrade --user
```

コマンドラインパスに AWS ParallelCluster 実行可能ファイルを追加する

`pip` を使用してインストールした後は、オペレーティングシステムの `PATH` 環境変数への `pcluster` 実行ファイルの追加が必要になる場合があります。

`pip` がインストールされているフォルダを確認するには AWS ParallelCluster、次のコマンドを実行します。

```
$ which pcluster
/home/username/.local/bin/pcluster
```

インストール時に`--user`スイッチを省略した場合 AWS ParallelCluster、実行可能ファイルは Python インストールの `bin`フォルダにある可能性があります。Python がインストールされた場所が不明な場合は、次のコマンドを実行します。

```
$ which python
/usr/local/bin/python
```

出力は、実際の実行可能ファイルではなく symlink へのパスになる場合があります。symlink の示す場所を確認するには、`ls -al` を実行します。

```
$ ls -al $(which python)
/usr/local/bin/python -> ~/.local/Python/3.6/bin/python3.6
```

これが のインストール AWS ParallelCluster のステップ 3 でパスに追加したのと同じフォルダである場合、インストール作業は完了です。それ以外の場合は、ステップ 3a ~ 3c を再び実行します。これにより、この追加フォルダがパスに追加されます。

Linux での Python のインストール

ディストリビューションに Python が付属していない場合、または以前のバージョンが付属している場合は、 `pip`と をインストールする前に Python をインストールします AWS ParallelCluster。

Linux に Python 3 をインストールするには

1. Python がインストール済みかどうかを確認します。

```
$ python3 --version
```

or

```
$ python --version
```

> **Note**
> ご使用の Linux ディストリビューションに Python が付属している場合は、Python 開発者パッケージをインストールする必要があります。開発者パッケージには、拡張機能をコンパイルして AWS ParallelClusterをインストールするのに必要なヘッダーとライブラリが含まれます。パッケージマネージャーを使用して、開発者パッケージをインストールします。通常、このファイル名は `python-dev` または `python-devel` です。

2. Python 2.7 以降がインストールされていない場合は、ご使用のディストリビューションのパッケージマネージャーを使用して Python をインストールします。コマンドとパッケージ名は、場合によって異なります。

   - Debian から派生した OS (Ubuntu など) では、`apt` を使用します。

     ```
     $ sudo apt-get install python3
     ```

   - Red Hat およびそれから派生した OS では、`yum` を使用します。

     ```
     $ sudo yum install python3
     ```

   - SUSE およびそれから派生した OS では、`zypper` を使用します。

     ```
     $ sudo zypper install python3
     ```

3. Python が正しくインストールされたことを確認するには、コマンドプロンプトまたはシェルを開き、次のコマンドを実行します。

```
$ python3 --version
Python 3.8.11
```

macOS AWS ParallelCluster に をインストールする

セクション

- 前提条件
- pip を使用して macOS AWS ParallelCluster に をインストールする
- コマンドラインパスに AWS ParallelCluster 実行可能ファイルを追加する

前提条件

- Python 3 バージョン 3.7+ または Python 2 バージョン 2.7

Python のインストールを確認します。

```
$ python --version
```

ご使用のコンピュータに Python がインストールされていない場合、または別のバージョンの Python をインストールする場合は、「Linux AWS ParallelCluster に をインストールする」の手順に従います。

pip を使用して macOS AWS ParallelCluster に をインストールする

`pip` を直接使用してインストールすることもできます AWS ParallelCluster。`pip` がない場合は、メインの「インストールに関するトピック」の手順に従います。`pip3 --version` を実行して、使用する macOS のバージョンにすでに Python と `pip3` が含まれているかどうかを確認します。

```
$ pip3 --version
```

macOS AWS ParallelCluster に をインストールするには

1. Python.org のダウンロードページから最新の Python をダウンロードしてインストールします。
2. Python Packaging Authority が提供する `pip3` インストールスクリプトをダウンロードして実行します。

   ```
   $ curl -O https://bootstrap.pypa.io/get-pip.py
   $ python3 get-pip.py --user
   ```

3. 新しくインストールした pip3を使用して をインストールします AWS ParallelCluster。Python バージョン 3+ を使用する場合、pip3 コマンドの使用が推奨されます。

```
$ python3 -m pip install "aws-parallelcluster<3.0" --upgrade --user
```

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

```
$ pcluster version
2.11.9
```

プログラムが見つからない場合は、そのプログラムをコマンドラインパスに追加します。

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

```
$ pip3 install "aws-parallelcluster<3.0" --upgrade --user
```

コマンドラインパスに AWS ParallelCluster 実行可能ファイルを追加する

pip を使用してインストールした後は、オペレーティングシステムの PATH 環境変数への pcluster プログラムの追加が必要になる場合があります。プログラムの場所は、Python のインストール先によって異なります。

Example AWS ParallelCluster インストール場所 - Python 3.6 および **pip** (ユーザーモード) を使用した macOS

```
~/Library/Python/3.6/bin
```

前の例のバージョン用に使用する Python のバージョンで置き換えます。

Python のインストール先がわからない場合は、which python を実行します。

```
$ which python3
/usr/local/bin/python3
```

出力は、実際のプログラムへのパスではなくシンボリックリンクへのパスになる場合があります。ls -al を実行して、その参照先を確認します。

```
$ ls -al /usr/local/bin/python3
```

```
lrwxr-xr-x  1 username  admin  36 Mar 12 12:47 /usr/local/bin/python3 -> ../Cellar/
python/3.6.8/bin/python3
```

pip は、Python プログラムが含まれている同じフォルダに、アプリケーションをインストールします。このフォルダを PATH 変数に追加します。

PATH 変数を変更するには (Linux、Unix、macOS)

1. ユーザーフォルダーでシェルのプロファイルスクリプトを見つけます。現在使用しているシェルが不明な場合は、echo $SHELL を実行します。

   ```
   $ ls -a ~
   .  ..  .bash_logout  .bash_profile  .bashrc  Desktop  Documents  Downloads
   ```

   - Bash - .bash_profile、.profile、または .bash_login
   - Zsh - .zshrc
   - Tcsh - .tcshrc、.cshrc、または .login
2. プロファイルスクリプトにエクスポートコマンドを追加します。

   ```
   export PATH=~/.local/bin:$PATH
   ```

   このコマンドは、現在の ~/.local/bin 変数にパス (この例では PATH) を追加します。
3. 現在のセッションにプロファイルをロードします。

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

Windows AWS ParallelCluster に をインストールする

Python のパッケージマネージャーである を使用して pipWindows AWS ParallelCluster に をインストールできます。pip がすでに存在する場合は、メインのインストールに関するトピックの手順に従います。

セクション

- Python AWS ParallelCluster を使用して Windows pipに をインストールする
- コマンドラインパスに AWS ParallelCluster 実行可能ファイルを追加する

Python AWS ParallelCluster を使用して Windows **pip**に をインストールする

Python Software Foundation は、`pip` を含む Windows 用インストーラを提供しています。

Python と **`pip`** をインストールするには (Windows)

1. Python.org のダウンロードページから Python Windows x86-64 のインストーラーをダウンロードします。
2. インストーラーを実行します。
3. [Add Python 3 to PATH] (Python 3 を PATH に追加する) を選択します。
4. [Install Now] (今すぐインストールする) を選択します。

インストーラはユーザーフォルダに Python をインストールし、プログラムフォルダをユーザーパスに追加します。

AWS ParallelCluster で をインストールするには **`pip3`** (Windows)

Python version 3+ を使用する場合は、`pip3` コマンドの使用が推奨されます。

1. [Start] (スタート) メニューから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、Python と `pip` のいずれも正しくインストールされたことを確認します。

```
C:\>py --version
Python 3.8.11
C:\>pip3 --version
pip 21.3.1 from c:\python38\lib\site-packages\pip (python 3.8)
```

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

```
C:\>pip3 install "aws-parallelcluster<3.0"
```

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

```
C:\>pcluster version
2.11.9
```

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

```
C:\>pip3 install --user --upgrade "aws-parallelcluster<3.0"
```

コマンドラインパスに AWS ParallelCluster 実行可能ファイルを追加する

AWS ParallelCluster でインストールしたらpip、オペレーティングシステムのPATH環境変数にpclusterプログラムを追加します。

次のコマンドを実行すると、pcluster プログラムがインストールされた場所を確認できます。

```
C:\>where pcluster
C:\Python38\Scripts\pcluster.exe
```

このコマンドで結果が返らない場合は、手動でパスを追加する必要があります。コマンドラインまたは Windows Explorer を使用して、コンピュータのインストールされている場所を見つけます。一般的なパスは、次のとおりです。

- Python 3 と**pip3** - `C:\Python38\Scripts\`
- Python 3 と **pip3** --user オプション - `%APPDATA%\Python\Python38\Scripts`

> **Note**
>
> バージョン番号が含まれるフォルダ名は、異なる場合があります。前述の例は Python38 を示しています。必要に応じて使用しているバージョン番号に置き換えます。

PATH 変数を変更するには (Windows)

1. Windows キーを押し、「**environment variables**」と入力します。
2. [Edit environment variables for your account] (アカウントの環境変数を編集する) を選択します。
3. PATH を選択して、編集 を選択します。
4. このパスを [Variable value] (変数値) フィールドに追加します。例: ***C:\new\path***。
5. [OK] を 2 回選択して、新しい設定を適用します。
6. 実行中のコマンドプロンプトを閉じ、コマンドプロンプトウィンドウを再度開きます。

の設定 AWS ParallelCluster

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

AWS アカウントに CLI `pcluster` の実行に必要なアクセス許可を含むロールがあることを確認します。詳細については、「AWS ParallelCluster インスタンスポリシーとユーザーポリシーの例」を参照してください。

AWS 認証情報を設定します。詳細については、「AWS CLI ユーザーガイド」の「AWS CLIを設定する」を参照してください。

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

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

```
$ pcluster configure
```

設定ウィザードでは、クラスターを作成するために必要な情報をすべて入力するよう求められます。スケジューラ AWS Batch として を使用する場合と を使用する場合では、シーケンスの詳細が異なりますSlurm。クラスターの設定に関する詳細は、「設定」を参照してください。

> **Note**
>
> バージョン 2.11.5 AWS ParallelCluster 以降、 は SGEまたは スTorqueケジューラの使用をサポートしていません。2.11.4 以前のバージョンで引き続き使用できますが、 AWS サービスおよび AWS サポートチームによる今後の更新やトラブルシューティングのサポートを受けることはできません。

スラム

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

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

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

クラスターで使用するスケジューラを選択します。

```
Allowed values for Scheduler:
```

```
1. slurm
2. awsbatch
Scheduler [slurm]:
```

オペレーティングシステムを選択します。

```
Allowed values for Operating System:
1. alinux2
2. centos7
3. ubuntu1804
4. ubuntu2004
Operating System [alinux2]:
```

> **Note**
>
> AWS ParallelCluster バージョン 2.6.0 で のサポートが追加されalinux2ました。

コンピューティングノードのクラスターの最小サイズと最大サイズを入力します。これは、インスタンスの数で測定されます。

```
Minimum cluster size (instances) [0]:
Maximum cluster size (instances) [10]:
```

ヘッドノードとコンピューティングノードのインスタンスタイプが入力されます。インスタンスタイプの場合、アカウントインスタンスの制限は、要件を満たすのに十分な大きさです。詳細については、「Amazon EC2 ユーザーガイド」の「オンデマンドインスタンス制限」を参照してください。

```
Master instance type [t2.micro]:
Compute instance type [t2.micro]:
```

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

```
Allowed values for EC2 Key Pair Name:
1. prod-uswest1-key
2. test-uswest1-key
```

```
EC2 Key Pair Name [prod-uswest1-key]:
```

前のステップが完了したら、既存の VPC を使用するか、 が VPC AWS ParallelCluster を作成するかを決定します。適切に設定された VPC がない場合は、新しい VPC AWS ParallelCluster を作成できます。同じパブリックサブネット内のヘッダーノードとコンピューティングノードの両方を使用するか、プライベートサブネット内のすべてのノードを持つパブリックサブネット内のヘッダーノードのみを使用します。内の VPCs 数の制限に達する可能性があります AWS リージョン。デフォルトの制限は、それぞれ 5 つの VPCsです AWS リージョン。この制限と引き上げをリクエストする方法の詳細については、「Amazon VPC User Guide」(Amazon VPC ユーザーガイド) の「VPC and subnets」(VPC とサブネット) を参照してください。

で VPC AWS ParallelCluster を作成する場合は、すべてのノードをパブリックサブネットに配置するかどうかを決定する必要があります。

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

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

```
Automate VPC creation? (y/n) [n]: y
Allowed values for Network Configuration:
1. Master in a public subnet and compute fleet in a private subnet
2. Master and compute fleet in the same public subnet
Network Configuration [Master in a public subnet and compute fleet in a private
 subnet]: 1
Beginning VPC creation. Please do not leave the terminal until the creation is
 finalized
```

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

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

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

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

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

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

AWS Batch

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

```
Allowed values for AWS ##### ID:
1. ap-northeast-1
2. ap-northeast-2
3. ap-south-1
4. ap-southeast-1
5. ap-southeast-2
6. ca-central-1
7. eu-central-1
8. eu-north-1
9. eu-west-1
10. eu-west-2
11. eu-west-3
12. sa-east-1
13. us-east-1
14. us-east-2
```

```
15. us-west-1
16. us-west-2
AWS ##### ID [ap-northeast-1]:
```

クラスターで使用するスケジューラを選択します。

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

awsbatch がスケジューラとして選択されている場合、alinux2 がオペレーティングシステムとして使用されます。

コンピューティングノードのクラスターの最小サイズと最大サイズを入力します。これは vCPU 単位で測定されます。

```
Minimum cluster size (vcpus) [0]:
Maximum cluster size (vcpus) [10]:
```

ヘッドノードのインスタンスタイプが入力されます。awsbatch スケジューラを使用する場合、コンピューティングノードは optimal のインスタンスタイプを使用します。

```
Master instance type [t2.micro]:
```

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

```
Allowed values for EC2 Key Pair Name:
1. prod-uswest1-key
2. test-uswest1-key
EC2 Key Pair Name [prod-uswest1-key]:
```

既存の VPCs を使用するか、 で VPCs AWS ParallelCluster を作成するかを決定します。適切に設定された VPC がない場合は、新しい VPC AWS ParallelCluster を作成できます。同じパブリックサブネット内のヘッダーノードとコンピューティングノードの両方を使用するか、プライベートサブネット内のすべてのノードを持つパブリックサブネット内のヘッダーノードのみを使用します。内の VPCs 数の制限に達する可能性があります AWS リージョン。VPC のデフォルト

数は 5 です。この制限と引き上げをリクエストする方法の詳細については、「Amazon VPC User Guide」(Amazon VPC ユーザーガイド) の「VPC and subnets」(VPC とサブネット) を参照してください。

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

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

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

```
Automate VPC creation? (y/n) [n]: y
Allowed values for Network Configuration:
1. Master in a public subnet and compute fleet in a private subnet
2. Master and compute fleet in the same public subnet
Network Configuration [Master in a public subnet and compute fleet in a private
 subnet]: 1
Beginning VPC creation. Please do not leave the terminal until the creation is
 finalized
```

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

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

```
Automate VPC creation? (y/n) [n]: n
Allowed values for VPC ID:
```

```
  #  id                     name                                number_of_subnets
---  ---------------------  ----------------------------------  -------------------
  1  vpc-0b4ad9c4678d3c7ad  ParallelClusterVPC-20200118031893                    2
  2  vpc-0e87c753286f37eef  ParallelClusterVPC-20191118233938                    5
VPC ID [vpc-0b4ad9c4678d3c7ad]: 1
```

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

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

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

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

- VPC には `DNS Resolution = yes` と `DNS Hostnames = yes` が必要です。
- VPC は AWS リージョンに対して正しい `domain-name` を設定している DHCP オプションも必要です。デフォルトの DHCP オプションセットでは、必要な AmazonProvidedDNS がすでに指定されています。複数のドメインネームサーバーを指定する場合は、「Amazon VPC User Guide」(Amazon VPC ユーザーガイド) の「DHCP options sets」(DHCP オプションセット) を参照してください。プライベートサブネットを使用する場合は、NAT ゲートウェイまたは内部プロキシを使用して、コンピューティングノードへのウェブアクセスを有効にします。詳細については、「ネットワークの設定」を参照してください。

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

```
$ pcluster create mycluster
```

クラスターが「CREATE_COMPLETE」ステータスになったら、通常の SSH クライアント/設定を使用して接続できます。Amazon EC2 インスタンスへの接続の詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EC2 の使用を開始する」を参照してください。

次のコマンドを実行してクラスターを削除します。

```
$ pcluster delete --region us-east-1 mycluster
```

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

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

スタックは、delete-stack コマンドで削除することができます。

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

`pcluster configure` が作成する VPC は CloudFormation のネットワークスタックには作成されません。コンソールまたは AWS CLIを使用して手動で VPC を削除できます。

```
$ aws --region us-east-1 ec2 delete-vpc --vpc-id vpc-0b4ad9c4678d3c7ad
```

ベストプラクティス

ベストプラクティス:マスターインスタンスタイプの選択

マスターノードはジョブを実行しませんが、その機能とサイジングは、クラスターの全体的なパフォーマンスにとって重要です。

マスターノードに使用するインスタンスタイプを選択するときは、次の項目を評価します。

- クラスターサイズ: マスターノードはいかなるジョブも実行しませんが、その機能とサイズはクラスターの全体的なパフォーマンスにとって重要です。かなりの数のノードのクラスターをスケールアップまたはスケールダウンする必要がある場合、マスターノードの処理能力を向上させることを検討してください。
- 共有ファイルシステム: 共有ファイルシステムを使用してコンピューティングノードとマスターノード間でアーティファクトを共有する場合、マスターが NFS サーバーを公開しているノードで

あることを考慮してください。このため、ワークフローを処理するのに十分なネットワーク帯域幅と Amazon EBS の専用帯域幅を持つインスタンスタイプを選択する必要があります。

ベストプラクティス: ネットワークパフォーマンス

ネットワーク通信を改善するための可能性を網羅した 3 つのヒントがあります。

- プレイスメントグループ: クラスタープレイスメントグループは、単一のアベイラビリティーゾーン内のインスタンスを論理的にグループ化したものです。プレイスメントグループの詳細については、「Amazon EC2 ユーザーガイド」の「プレイスメントグループ」を参照してください。`placement_group = your-placement-group-name` で独自の配置グループを使用するようにクラスターを設定したり、`placement_group = DYNAMIC` で `"compute"` 戦略による配置グループを AWS ParallelCluster に作成させることができます。詳細については、「`placement_group` for multiple queue mode」、「`placement_group` for single queue mode」を参照してください。
- 拡張ネットワーキング: 拡張ネットワークをサポートするインスタンスタイプの選択を検討してください。詳細については、「Amazon EC2 ユーザーガイド」の「Linux での拡張ネットワーキング」を参照してください。
- Elastic Fabric Adapter: スケーラブルで高いレベルのインスタンス間通信をサポートするには、お使いのネットワークに EFA ネットワークインターフェイスを選択することを検討してください。EFA のカスタムビルドオペレーティングシステム (OS) バイパスハードウェアは、 AWS クラウドのオンデマンドの伸縮性と柔軟性により、インスタンス間の通信を強化します。EFA を使用するように単一の Slurm クラスターキューを設定するには、`enable_efa = true` を設定します。で EFA を使用する方法の詳細については AWS ParallelCluster、Elastic Fabric Adapter「」および「」を参照してください`enable_efa`。EFA の詳細については、「Linux インスタンス用 Amazon EC2 ユーザーガイド」の「Elastic Fabric Adapter」を参照してください。
- インスタンス帯域幅: 帯域幅はインスタンスサイズに応じてスケールするため、ニーズに合ったインスタンスタイプの選択を検討する必要があります。詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EBS 最適化インスタンス」と「Amazon EBS ボリュームの種類」を参照してください。

ベストプラクティス: 予算アラート

AWS ParallelCluster リソースコストを管理するには、 AWS Budgets アクションを使用して、選択した AWS リソースの予算と定義済みの予算しきい値アラートを作成することをお勧めします。詳細については、「AWS Budgets ユーザーガイド」の「予算アクションを設定する」を参照してくださ

い。Amazon CloudWatch を使用して、請求アラームを作成することもできます。詳細については、「推定請求額をモニタリングするための請求アラームの作成 AWS」を参照してください。

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

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

クラスターを新しいマイナーバージョンに移行するプロセスを最適化するため、またはその他の理由で共有ストレージデータを保存したりするには、次のベストプラクティスを使用することをお勧めします。

- Amazon EFS や FSx for Lustre などの外部ボリュームに個人データを保存します。これにより、あるクラスターから別のクラスターにデータを簡単に移動できます。
- AWS CLI または を使用して、以下に示すタイプの共有ストレージシステムを作成します AWS Management Console。
  - `[ebs]` セクション
  - `[efs]` セクション
  - `[fsx]` セクション

  それらを既存のファイルシステムとして新しいクラスター構成に追加します。こうすることで、クラスターを削除しても保存され、新しいクラスターにアタッチすることができます。共有ストレージシステムは、通常、クラスターに接続されていても、クラスターから切り離されていても料金が発生します。

  Amazon EFS または Amazon FSx for Lustre ファイルシステムを使用することをお勧めします。これは、複数のクラスターに同時にアタッチでき、古いクラスターを削除する前に新しいクラスターにアタッチできるためです。詳細については、「Amazon EFS ユーザーガイド」の「Mounting Amazon EFS file systems」と、「Amazon FSx for Lustre ユーザーガイド」の「FSx for Lustre ファイルシステムへのアクセス」を参照してください。
- カスタム AMI ではなく、カスタムブートストラップアクションを使用してインスタンスをカスタマイズします。これにより、新しいバージョンごとに新しいカスタム AMI を作成する必要がなくなるため、作成プロセスが最適化されます。
- 推奨シーケンスです。

  1. 既存のファイルシステム定義を使用するようにクラスター構成を更新します。

2. `pcluster` バージョンを確認し、必要に応じて更新してください。
3. 新しいクラスターを作成してテストします。
   - 新しいクラスターでデータが利用可能であることを確認します。
   - 新しいクラスターでアプリケーションが動作することを確認します。
4. 新しいクラスターが完全にテストされ、動作しており、古いクラスターを使用しないことが確実な場合は、そのクラスターを削除します。

CfnCluster から への移行 AWS ParallelCluster

AWS ParallelCluster は CfnCluster の拡張バージョンです。

現在 CfnCluster を使用している場合は、 AWS ParallelCluster 代わりに を使用して新しいクラスターを作成することをお勧めします。今後も CfnCluster を使用することができますが、CfnCluster の開発は終了しており、新しい機能や特徴は追加されません。

CfnCluster と の主な違い AWS ParallelCluster については、以下のセクションで説明します。

AWS ParallelCluster CLI は異なるクラスターセットを管理します

`cfncluster` CLI で作成されたクラスターを `pcluster` CLI で管理することはできません。以下のコマンドは、CfnCluster によって作成されたクラスターでは動作しません。

```
pcluster list
pcluster update cluster_name
pcluster start cluster_name
pcluster status cluster_name
```

CfnCluster で作成したクラスターを管理するには、`cfncluster` CLI を使用する必要があります。

古いクラスターを管理するために CfnCluster パッケージが必要な場合は、Python 仮想環境からインストールして使用することをお勧めします。

AWS ParallelCluster と CfnCluster が異なる IAM カスタムポリシーを使用する

以前に CfnCluster クラスター作成で使用した、カスタム IAM ポリシーを AWS ParallelClusterで使用することはできません。のカスタムポリシーが必要な場合は AWS ParallelCluster、新しいポリシーを作成する必要があります。 AWS ParallelCluster ガイドを参照してください。

AWS ParallelCluster と CfnCluster は異なる設定ファイルを使用します

AWS ParallelCluster 設定ファイルは `~/.parallelcluster`フォルダにあります。CfnCluster 設定ファイルは、`~/.cfncluster` フォルダにあります。

で既存の CfnCluster 設定ファイルを使用する場合は AWS ParallelCluster、次のアクションを実行する必要があります。

1. 設定ファイルを `~/.cfncluster/config` から `~/.parallelcluster/config` に移動します。
2. extra_json 設定パラメータを使用する場合は、以下のように変更します。

CfnCluster 設定:

```
extra_json = { "cfncluster" : { } }
```

AWS ParallelCluster 設定：

```
extra_json = { "cluster" : { } }
```

では AWS ParallelCluster、ganglia はデフォルトで無効になっています

では AWS ParallelCluster、ganglia はデフォルトで無効になっています。Ganglia を有効にするには、次の手順を実行します。

1. 以下に示すように extra_json を設定します。

   ```
   extra_json = { "cluster" : { "ganglia_enabled" : "yes" } }
   ```

2. また、ポート 80 への接続を許可するようにヘッドセキュリティグループを変更します。

   パブリック IP からポート 80 へのインバウンド接続を許可するには、新しいセキュリティグループルールを追加して、`parallelcluster-<CLUSTER_NAME>-MasterSecurityGroup-<xxx>` セキュリティグループを変更する必要があります。詳細については、「Amazon EC2 ユーザーガイド」の「セキュリティグループへのルールの追加」を参照してください。

サポート対象の リージョン

AWS ParallelCluster バージョン 2.x は、以下から入手できます AWS リージョン。

リージョン名	リージョン
米国東部 (オハイオ)	us-east-2
米国東部 (バージニア北部)	us-east-1
米国西部 (北カリフォルニア)	us-west-1
米国西部 (オレゴン)	us-west-2
アフリカ (ケープタウン)	af-south-1
アジアパシフィック (香港)	ap-east-1
アジアパシフィック (ムンバイ)	ap-south-1
アジアパシフィック (ソウル)	ap-northeast-2
アジアパシフィック (シンガポール)	ap-southeast-1
アジアパシフィック (シドニー)	ap-southeast-2
アジアパシフィック (東京)	ap-northeast-1
カナダ (中部)	ca-central-1
中国 (北京)	cn-north-1
中国 (寧夏)	cn-northwest-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
ヨーロッパ (ミラノ)	eu-south-1
欧州 (パリ)	eu-west-3
欧州 (ストックホルム)	eu-north-1

リージョン名	リージョン
中東 (バーレーン)	me-south-1
南米 (サンパウロ)	sa-east-1
AWS GovCloud (米国東部)	us-gov-east-1
AWS GovCloud (米国西部)	us-gov-west-1

の使用 AWS ParallelCluster

トピック

- ネットワークの設定
- カスタムブートストラップアクション
- Amazon S3 の操作
- スポットインスタンスの操作
- AWS Identity and Access Management での ロール AWS ParallelCluster
- でサポートされているスケジューラ AWS ParallelCluster
- AWS ParallelCluster リソースとタグ付け
- Amazon CloudWatch ダッシュボード
- Amazon CloudWatch Logs との統合
- Elastic Fabric Adapter
- インテル Select ソリューション
- Intel MPI を有効にする
- インテル HPC プラットフォームの仕様
- Arm パフォーマンスライブラリ
- Amazon DCV を介してヘッドノードに接続する
- pcluster update を使用する
- AMI のパッチ適用と EC2 インスタンスの交換

ネットワークの設定

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

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

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

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

これらの設定はすべて、パブリック IP アドレス指定の有無にかかわらず動作します。 は、すべての AWS リクエストに HTTP プロキシを使用するようにデプロイ AWS ParallelCluster することもできます。これらの設定を組み合わせることで、さまざまなデプロイシナリオにつながります。例えば、インターネットですべてのアクセスが可能な単一のパブリックサブネットを設定することができます。または、すべてのトラフィックに対して AWS Direct Connect と HTTP プロキシを使用して、完全プライベートネットワークを設定できます。

これらのシナリオのいくつかについては、以下のアーキテクチャ図を参照してください。

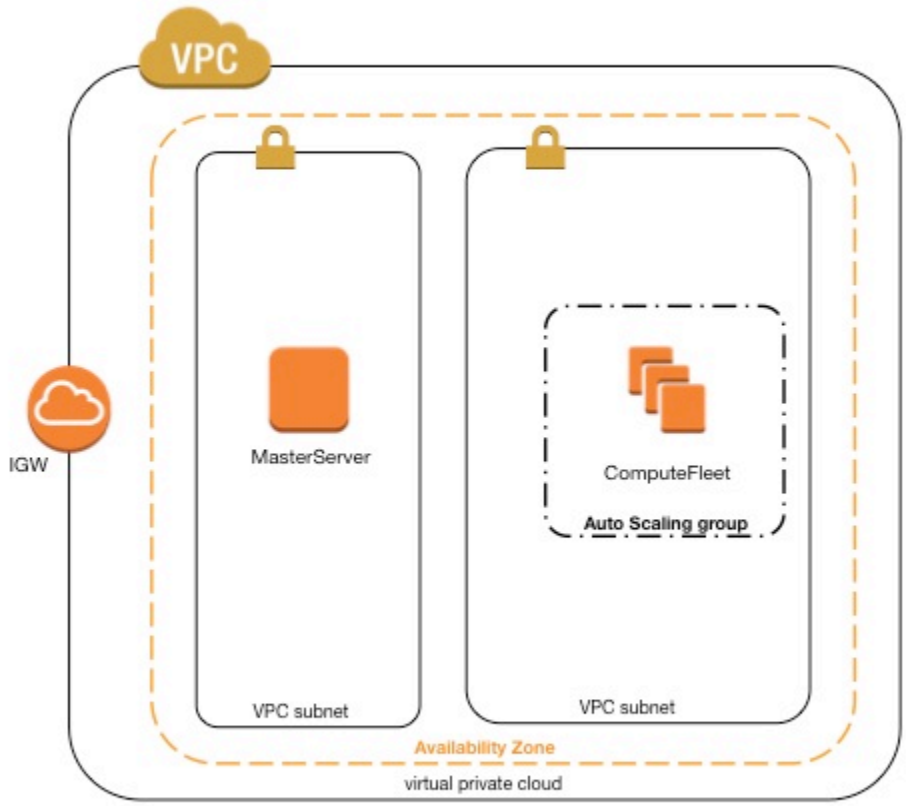
AWS ParallelCluster 1 つのパブリックサブネット内の

このアーキテクチャの設定には、次の設定が必要です。

```
[vpc public]
vpc_id = vpc-xxxxxx
master_subnet_id = subnet-<public>
use_public_ips = true
```

use_public_ips 設定を false に指定できません。インターネットゲートウェイでは、すべてのインスタンスにグローバルに一意の IP アドレスが必要であるためです。詳細については、「Amazon VPC User Guide」(Amazon VPC ユーザーガイド) の「Enabling internet access」(インターネットアクセスを有効にする)を参照してください。

AWS ParallelCluster 2 つのサブネットの使用



コンピューティングインスタンス用に新しいプライベートサブネットを作成するための設定には、次の設定が必要です。

値はすべてサンプルです。

```
[vpc public-private-new]
vpc_id = vpc-xxxxxx
master_subnet_id = subnet-<public>
compute_subnet_cidr = 10.0.1.0/24
```

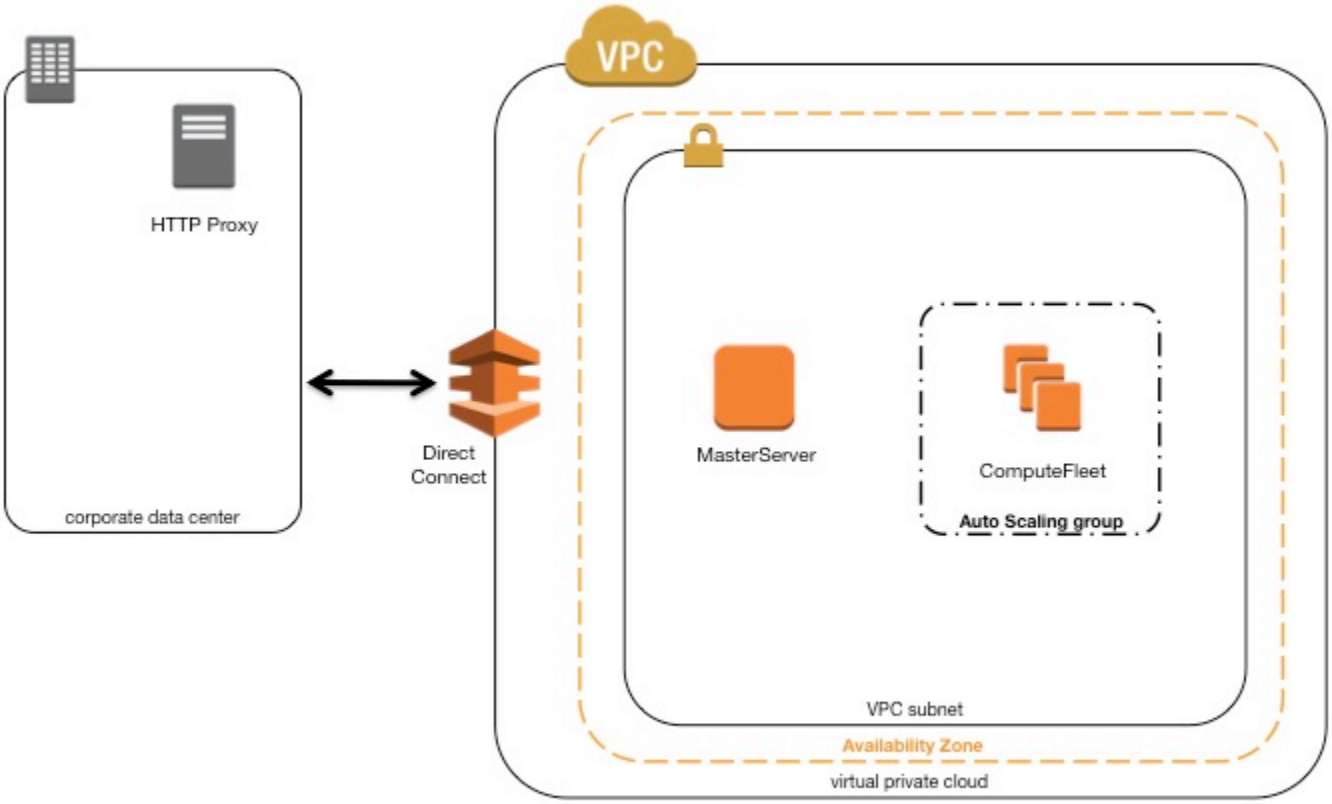
既存のプライベートネットワークを使用するための設定には、次の設定が必要です。

```
[vpc public-private-existing]
```

```
vpc_id = vpc-xxxxxx
master_subnet_id = subnet-<public>
compute_subnet_id = subnet-<private>
```

どちらの設定でも、コンピューティングインスタンスへのウェブアクセスを有効にするには、NAT ゲートウェイまたは内部プロキシが必要です。

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



このアーキテクチャの設定には、次の設定が必要です。

```
[cluster private-proxy]
proxy_server = http://proxy.corp.net:8080
```

```
[vpc private-proxy]
vpc_id = vpc-xxxxxx
master_subnet_id = subnet-<private>
use_public_ips = false
```

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

AWS ParallelCluster スawsbatchケジューラを使用する

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

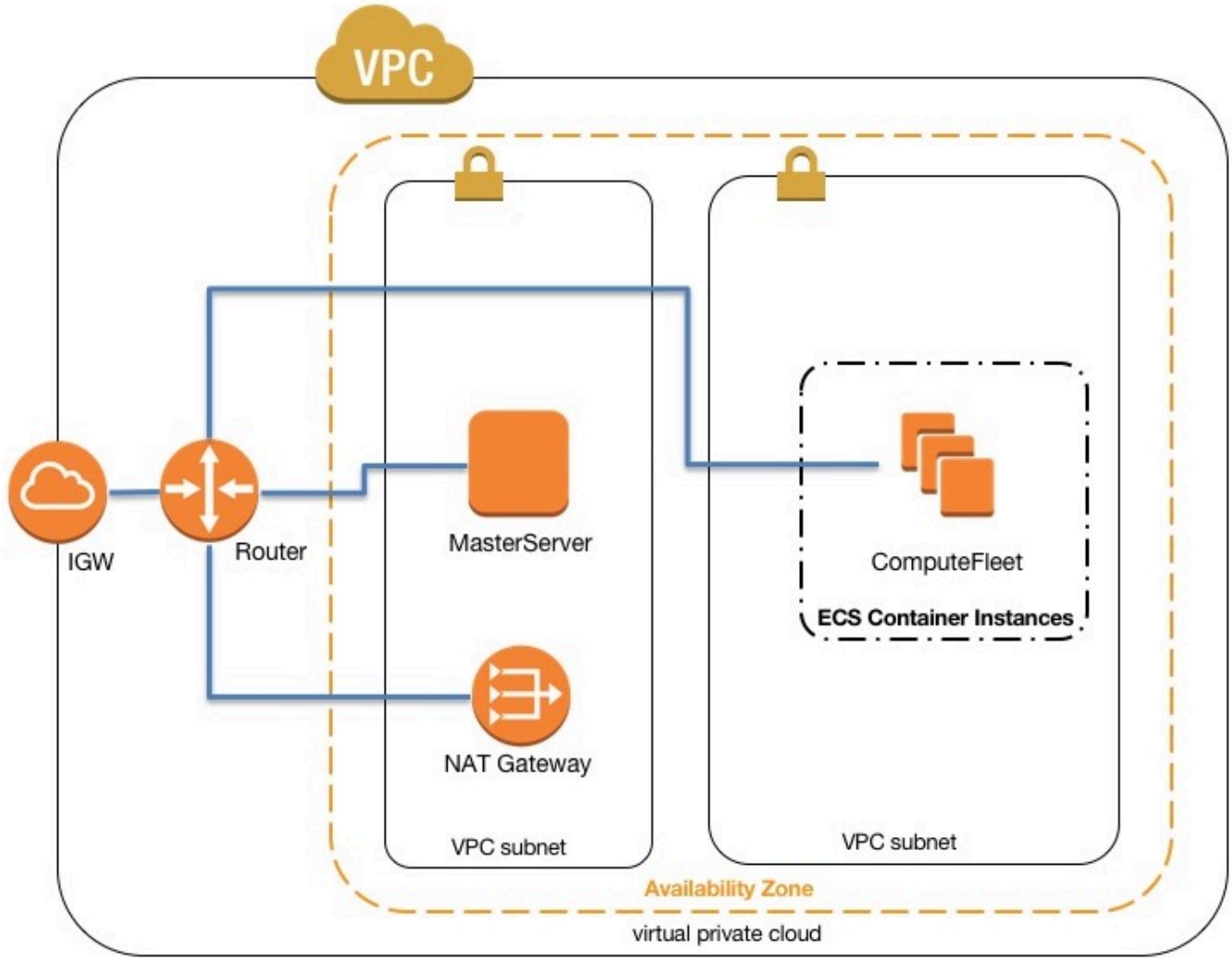
- compute_subnet は NAT ゲートウェイを使用してインターネットにアクセスします。(この手法をお勧めします)
- compute_subnet で起動されたインスタンスはパブリック IP アドレスを持ち、インターネットゲートウェイを介してインターネットにアクセスできます。

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

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

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

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



詳細については、以下の各トピックを参照してください。

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

カスタムブートストラップアクション

AWS ParallelCluster は、クラスターの作成時にメインブートストラップアクションの前 (インストール前) または後 (インストール後) に任意のコードを実行できます。通常、このコードは Amazon

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

インストール前のブートストラップアクションは、NAT、Amazon Elastic Block Store (Amazon EBS)、スケジューラの設定など、クラスターのデプロイの前に呼び出されます。一般的なインストール前のアクションには、ストレージの変更、追加のユーザーやパッケージの追加などがあります。

ポストインストールのアクションは、クラスターのブートストラッププロセスが完了した後に呼び出されます。インストール後のアクションは、インスタンスが完全に設定されて完了したと見なされる前に実行される最後のアクションです。一般的なポストインストールのアクションには、スケジューラ設定の変更、ストレージやパッケージの変更などがあります。

設定時に引数を指定することで、スクリプトに引数を渡すことができます。これを実行するには、プレインストールおよびポストインストールのアクションに引数を二重引用符で囲んで渡します。

プレインストールまたはポストインストールのアクションに失敗すると、インスタンスのブートストラップも失敗します。成功すると、終了コード 0 で通知されます。それ以外の終了コードは、インスタンスのブートストラップが失敗したことを示します。

ランニングヘッドとコンピューティングノードを区別することができます。`/etc/parallelcluster/cfnconfig` ファイルをソースとし、ヘッドノードとコンピューティングノードにそれぞれ「`MasterServer`」と「`ComputeFleet`」という値を持つ `cfn_node_type` 環境変数を評価します。

```
#!/bin/bash

. "/etc/parallelcluster/cfnconfig"

case "${cfn_node_type}" in
    MasterServer)
        echo "I am the head node" >> /tmp/head.txt
    ;;
    ComputeFleet)
        echo "I am a compute node" >> /tmp/compute.txt
    ;;
    *)
    ;;
esac
```

設定

次の設定は、プレインストールおよびポストインストールのアクションと引数を定義するために使用されます。

```
# URL to a preinstall script. This is run before any of the boot_as_* scripts are run
# (no default)
pre_install = https://<bucket-name>.s3.amazonaws.com/my-pre-install-script.sh
# Arguments to be passed to preinstall script
# (no default)
pre_install_args = argument-1 argument-2
# URL to a postinstall script. This is run after any of the boot_as_* scripts are run
# (no default)
post_install = https://<bucket-name>.s3.amazonaws.com/my-post-install-script.sh
# Arguments to be passed to postinstall script
# (no default)
post_install_args = argument-3 argument-4
```

引数

最初の 2 つの引数 `$0` と `$1` はスクリプト名と URL 用に予約されています。

```
$0 => the script name
$1 => s3 url
$n => args set by pre/post_install_args
```

例

以下のステップでは、R パッケージをクラスターにインストールする簡単なポストインストールスクリプトを作成します。

1. [Create a script].(スクリプトを作成します)。

   ```
   #!/bin/bash

   echo "post-install script has $# arguments"
   for arg in "$@"
   do
       echo "arg: ${arg}"
   done
   ```

```
yum -y install "${@:2}"
```

2. Amazon S3 に正しいアクセス許可でスクリプトをアップロードしてください。パブリック読み取り権限が適切でない場合は、s3_read_resource または s3_read_write_resource パラメータのいずれかを使用してアクセスを許可してください。詳細については、「Amazon S3 の操作」を参照してください。

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

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

3. 新しいインストール後アクションを含めるように AWS ParallelCluster 設定を更新します。

```
[cluster default]
...
post_install = https://bucket-name.s3.amazonaws.com/myscript.sh
post_install_args = 'R curl wget'
```

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

```
[cluster default]
...
post_install = s3://bucket-name/myscript.sh
post_install_args = 'R curl wget'
```

4. クラスターを起動します。

```
$ pcluster create mycluster
```

5. 出力の検証

```
$ less /var/log/cfn-init.log
2019-04-11 10:43:54,588 [DEBUG] Command runpostinstall output: post-install script
 has 4 arguments
arg: s3://bucket-name/test.sh
arg: R
```

```
arg: curl
arg: wget
Loaded plugins: dkms-build-requires, priorities, update-motd, upgrade-helper
Package R-3.4.1-1.52.amzn1.x86_64 already installed and latest version
Package curl-7.61.1-7.91.amzn1.x86_64 already installed and latest version
Package wget-1.18-4.29.amzn1.x86_64 already installed and latest version
Nothing to do
```

Amazon S3 の操作

Amazon S3 バケットへのアクセス許可をクラスターリソースに付与するには、 AWS ParallelCluster 設定の `s3_read_resource`および `s3_read_write_resource`パラメータでバケット ARNs を指定します。によるアクセスの制御の詳細については AWS ParallelCluster、「」を参照してください[AWS Identity and Access Management での ロール AWS ParallelCluster](#)。

```
# Specify Amazon S3 resource which AWS ParallelCluster nodes will be granted read-only
 access
# (no default)
s3_read_resource = arn:aws:s3:::my_corporate_bucket*
# Specify Amazon S3 resource which AWS ParallelCluster nodes will be granted read-write
 access
# (no default)
s3_read_write_resource = arn:aws:s3:::my_corporate_bucket/*
```

どちらのパラメータも * または有効な Amazon S3 ARN を受け入れます。Amazon S3 ARN の指定については、「AWS 全般のリファレンス」の「[Amazon S3 ARN format](#)」を参照してください。

例

次の例では、Amazon S3 バケット my_corporate_bucket 内の任意のオブジェクトへの読み取りアクセス権を付与しています。

```
s3_read_resource = arn:aws:s3:::my_corporate_bucket/*
```

以下の例では、バケットへの読み取りアクセスを付与していますが、バケットからアイテムを読み取ることはできません。

```
s3_read_resource = arn:aws:s3:::my_corporate_bucket
```

この最後の例では、バケットおよびバケットに保存されたアイテムへの読み取りアクセスが付与されます。

```
s3_read_resource = arn:aws:s3:::my_corporate_bucket*
```

スポットインスタンスの操作

AWS ParallelCluster クラスター設定で `cluster_type` = spot が設定されている場合、 はスポットインスタンスを使用します。スポットインスタンスはオンデマンドインスタンスよりも費用対効果が高いが、中断される可能性があります。中断の効果は、使用する特定のスケジューラによって異なります。これは、Amazon EC2 がスポットインスタンスを停止または終了する前に 2 分間の警告を提供するスポットインスタンスの中断通知を活用するのに役立ちます。詳細については、「Amazon EC2 ユーザーガイド」の「スポットインスタンスの中断」を参照してください。以下のセクションでは、スポットインスタンスを中断できる 3 つのシナリオについて説明します。

> **Note**
>
> スポットインスタンスを使用するには、お客様のアカウントに `AWSServiceRoleForEC2Spot` サービスにリンクしたロールが必要です。を使用してアカウントにこのロールを作成するには AWS CLI、次のコマンドを実行します。
>
> ```
> aws iam create-service-linked-role --aws-service-name spot.amazonaws.com
> ```
>
> 詳細については、「Amazon EC2 ユーザーガイド」の「スポットインスタンスリクエスト向けのサービスにリンクされたロール」を参照してください。

シナリオ 1: 実行中のジョブがないスポットインスタンスが中断される

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

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

この中断の動作は、使用されているスケジューラによって異なります。

Slurm

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

> **Note**
>
> この動作は AWS ParallelCluster バージョン 2.9.0 で変更されました。以前のバージョンでは、ステートコード NODE_FAIL でジョブが終了し、そのノードがスケジューラのキューから削除されました。

SGE

> **Note**
>
> これは、 AWS ParallelCluster バージョン 2.11.4 以前のバージョンにのみ適用されます。バージョン 2.11.5 以降、 AWS ParallelCluster は SGE または Torque スケジューラの使用をサポートしていません。

ジョブが終了します。ジョブが再実行フラグを有効にしている場合 (qsub -r yes または qalter -r yes を使用)、またはキューで rerun 設定が TRUE に設定されている場合、ジョブは再スケジュールされます。コンピューティングインスタンスがスケジューラキューから削除されます。この動作は、次の SGE 設定パラメータから起こります。

- reschedule_unknown 00:00:30
- ENABLE_FORCED_QDEL_IF_UNKNOWN
- ENABLE_RESCHEDULE_KILL=1

Torque

> **Note**
>
> これは、 AWS ParallelCluster バージョン 2.11.4 以前のバージョンにのみ適用されます。バージョン 2.11.5 以降、 AWS ParallelCluster は SGE または Torque スケジューラの使用をサポートしていません。

ジョブがシステムから削除され、ノードがスケジューラから削除されます。ジョブは再実行されません。中断時にインスタンス上で複数のジョブが実行されている場合、ノードの削除中にトルクがタイムアウトすることがあります。`sqswatcher` ログファイルにエラーが表示されることがあります。これはスケーリングロジックには影響せず、その後の再試行によって適切なクリーンアップが実行されます。

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

この中断の動作は、使用されているスケジューラによって異なります。

Slurm

ステートコード `NODE_FAIL` で失敗し、ジョブは再キューされます (ジョブ投入時に `--no-requeue` が指定されていない場合)。静的ノードの場合は、置換されます。動的ノードの場合は、ノードは終了してリセットされます。終了したジョブを実行していた他のノードは、他の保留中のジョブに割り当てられたり、設定された `scaledown_idletime` 時間が経過した後にスケールダウンされたりする場合があります。

> **Note**
>
> この動作は AWS ParallelCluster バージョン 2.9.0 で変更されました。以前のバージョンでは、ステートコード `NODE_FAIL` でジョブが終了し、そのノードがスケジューラのキューから削除されました。終了したジョブを実行していた他のノードは、設定された `scaledown_idletime` 時間が経過した後に縮小される可能性があります。

SGE

> **Note**
>
> これは、 AWS ParallelCluster バージョン 2.11.4 以前のバージョンにのみ適用されます。バージョン 2.11.5 以降、 AWS ParallelCluster は SGE または Torque スケジューラの使用をサポートしていません。

ジョブは終了せず、残りのノードで引き続き実行されます。コンピューティングノードはスケジューラキューから削除されますが、ホストリストには孤立した使用できないノードとして表示されます。

この場合、ユーザーはジョブを削除する必要があります (`qdel <jobid>`)。ノードは引き続きホストリスト (`qhost`) に表示されますが、 AWS ParallelClusterに影響はありません。リストからホストを削除するには、インスタンスを置き換えた後に次のコマンドを実行します。

```
sudo -- bash -c 'source /etc/profile.d/sge.sh; qconf -dattr hostgroup
 hostlist <hostname> @allhosts; qconf -de <hostname>'
```

Torque

> **Note**
>
> これは、 AWS ParallelCluster バージョン 2.11.4 以前のバージョンにのみ適用されます。バージョン 2.11.5 以降、 AWS ParallelCluster は SGE または Torque スケジューラの使用をサポートしていません。

ジョブがシステムから削除され、ノードがスケジューラから削除されます。ジョブは再実行されません。中断時にインスタンス上で複数のジョブが実行されている場合、ノードの削除中にトルクがタイムアウトすることがあります。`sqswatcher` ログファイルにエラーが表示されることがあります。これはスケーリングロジックには影響せず、その後の再試行によって適切なクリーンアップが実行されます。

スポットインスタンスの詳細については、「Amazon EC2 ユーザーガイド」の「スポットインスタンス」を参照してください。

AWS Identity and Access Management での ロール AWS ParallelCluster

AWS ParallelCluster は Amazon EC2 の AWS Identity and Access Management (IAM) ロールを使用して、インスタンスがクラスターのデプロイとオペレーション AWS のサービスにアクセスできるようにします。デフォルトでは、Amazon EC2 の IAM ロールは、クラスターの作成時に作成されます。そのため、次のセクションで説明するように、クラスターを作成するユーザーに適切なレベルのアクセス許可が必要です。

AWS ParallelCluster は、複数の AWS サービスを使用してクラスターをデプロイおよび運用します。詳細なリストは、「AWS Services used in AWS ParallelCluster」セクションを参照してください。

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

トピック

- クラスター作成のデフォルト設定
- Amazon EC2 ロールに既存の IAM ロールを使用する
- AWS ParallelCluster インスタンスポリシーとユーザーポリシーの例

クラスター作成のデフォルト設定

クラスター作成にデフォルト設定を使用すると、デフォルト Amazon EC2 の IAM ロールがクラスターによって作成されます。通常、ユーザーには、クラスターの起動に必要なすべてのリソースを作成するための適切なレベルのアクセス許可が必要です。これには、Amazon EC2 の IAM ロールの作成が含まれます。通常、デフォルト設定を使用するとき、ユーザーには AdministratorAccess マネージドポリシーのアクセス許可が必要です。マネージドポリシーの詳細については、「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。

Amazon EC2 ロールに既存の IAM ロールを使用する

クラスターを作成する際、デフォルト設定の代わりに既存の `ec2_iam_role` を使用できますが、まず IAM ポリシーとロールを定義してからクラスターを起動する必要があります。通常、クラスターを起動する際ユーザーに付与されているアクセス許可を最小限に抑えるためには、Amazon EC2 の既存の IAM ロールを選択します。には、 AWS ParallelCluster とその機能に必要な最小限のアクセス許可AWS ParallelCluster インスタンスポリシーとユーザーポリシーの例が含まれています。両方を

IAM で個別のポリシーとして作成してから、適切なリソースにアタッチする必要があります。ロールポリシーの中には、サイズが大きくなり、クォータエラーの原因となるものもあります。詳細については、「IAM ポリシーのサイズ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。ポリシーで、*<REGION>*、*#AWS ACCOUNT ID>*、および同様の文字列を適切な値に置き換えます。

クラスターノードのデフォルト設定にポリシーを追加する場合は、`ec2_iam_role` 設定を使用する代わりに、`additional_iam_policies` 設定で追加のカスタム IAM ポリシーを渡すことをお勧めします。

AWS ParallelCluster インスタンスポリシーとユーザーポリシーの例

次のポリシーの例では、リソースの Amazon リソースネーム (ARN) が含まれています。 AWS GovCloud (US) または AWS 中国パーティションで作業している場合は、ARNs を変更する必要があります。具体的には、 AWS GovCloud (US) パーティションの場合は「arn:aws」から「arn:aws-us-gov」、 AWS 中国パーティションの場合は「arn:aws-cn」に変更する必要があります。詳細については、「 ユーザーガイド」の「 リージョンの AWS GovCloud (US) Amazon リソースネーム (ARNs)」および「中国での AWS サービスの開始方法」のARNs」を参照してください。 AWS GovCloud (US)　AWS

これらのポリシーには、 が現在必要とする最小限のアクセス許可 AWS ParallelCluster、その機能、リソースが含まれます。ロールポリシーの中には、サイズが大きくなり、クォータエラーの原因となるものもあります。詳細については、「IAM ポリシーのサイズ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。

トピック

- SGE、Slurm または Torque を使用する ParallelClusterInstancePolicy
- awsbatch を使用する ParallelClusterInstancePolicy
- Slurm を使用する ParallelClusterUserPolicy
- SGE または Torque を使用する ParallelClusterUserPolicy
- awsbatch を使用する ParallelClusterUserPolicy
- SGE、Slurm または Torque を使用する ParallelClusterLambdaPolicy
- awsbatch を使用する ParallelClusterLambdaPolicy
- ユーザー用の ParallelClusterUserPolicy

SGE、Slurm または Torque を使用する **ParallelClusterInstancePolicy**

> **Note**
>
> バージョン 2.11.5 以降、 AWS ParallelCluster は SGEまたは スTorqueケジューラの使用をサポートしていません。2.11.4 までのバージョンで引き続き使用できますが、 AWS サービスおよび AWS サポートチームによる今後の更新やトラブルシューティングのサポートを受けることはできません。

トピック

- Slurm を使用する ParallelClusterInstancePolicy
- SGE または Torque を使用する ParallelClusterInstancePolicy

Slurm を使用する **ParallelClusterInstancePolicy**

次の例では、スケジューラとして Slurm を使用して `ParallelClusterInstancePolicy` を設定します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ec2:DescribeVolumes",
                "ec2:AttachVolume",
                "ec2:DescribeInstanceAttribute",
                "ec2:DescribeInstanceStatus",
                "ec2:DescribeInstanceTypes",
                "ec2:DescribeInstances",
                "ec2:DescribeRegions",
                "ec2:TerminateInstances",
                "ec2:DescribeLaunchTemplates",
                "ec2:CreateTags"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow",
            "Sid": "EC2"
```

```
        },
        {
            "Action": "ec2:RunInstances",
            "Resource": [
                "arn:aws:ec2:<REGION>:<AWS ACCOUNT ID>:subnet/<COMPUTE SUBNET ID>",
                "arn:aws:ec2:<REGION>:<AWS ACCOUNT ID>:network-interface/*",
                "arn:aws:ec2:<REGION>:<AWS ACCOUNT ID>:instance/*",
                "arn:aws:ec2:<REGION>:<AWS ACCOUNT ID>:volume/*",
                "arn:aws:ec2:<REGION>::image/<IMAGE ID>",
                "arn:aws:ec2:<REGION>:<AWS ACCOUNT ID>:key-pair/<KEY NAME>",
                "arn:aws:ec2:<REGION>:<AWS ACCOUNT ID>:security-group/*",
                "arn:aws:ec2:<REGION>:<AWS ACCOUNT ID>:launch-template/*",
                "arn:aws:ec2:<REGION>:<AWS ACCOUNT ID>:placement-group/*"
            ],
            "Effect": "Allow",
            "Sid": "EC2RunInstances"
        },
        {
            "Action": [
                "dynamodb:ListTables"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow",
            "Sid": "DynamoDBList"
        },
        {
            "Action": [
                "cloudformation:DescribeStacks",
                "cloudformation:DescribeStackResource",
                "cloudformation:SignalResource"
            ],
            "Resource": [
                "arn:aws:cloudformation:<REGION>:<AWS ACCOUNT ID>:stack/
parallelcluster-*/*"
            ],
            "Effect": "Allow",
            "Sid": "CloudFormation"
        },
        {
            "Action": [
                "dynamodb:PutItem",
                "dynamodb:Query",
```

```
                "dynamodb:GetItem",
                "dynamodb:BatchWriteItem",
                "dynamodb:DeleteItem",
                "dynamodb:DescribeTable"
            ],
            "Resource": [
                "arn:aws:dynamodb:<REGION>:<AWS ACCOUNT ID>:table/parallelcluster-*"
            ],
            "Effect": "Allow",
            "Sid": "DynamoDBTable"
        },
        {
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::<REGION>-aws-parallelcluster/*"
            ],
            "Effect": "Allow",
            "Sid": "S3GetObj"
        },
        {
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow",
            "Sid": "IAMPassRole",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "ec2.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::dcv-license.<REGION>/*"
```

```
            ],
            "Effect": "Allow",
            "Sid": "DcvLicense"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::parallelcluster-*/*"
            ],
            "Effect": "Allow",
            "Sid": "GetClusterConfig"
        },
        {
            "Action": [
                "fsx:DescribeFileSystems"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow",
            "Sid": "FSx"
        },
        {
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow",
            "Sid": "CWLogs"
        },
        {
            "Action": [
                "route53:ChangeResourceRecordSets"
            ],
            "Resource": [
                "arn:aws:route53:::hostedzone/*"
            ],
            "Effect": "Allow",
```

```
            "Sid": "Route53"
        }
    ]
}
```

SGE または Torque を使用する **ParallelClusterInstancePolicy**

次の例では、スケジューラとして SGE または Torque を使用して `ParallelClusterInstancePolicy` を設定します。

> **Note**
>
> このポリシーは、 AWS ParallelCluster バージョン 2.11.4 以前のバージョンにのみ適用されます。バージョン 2.11.5 以降、 AWS ParallelCluster は SGE または Torque スケジューラの使用をサポートしていません。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ec2:DescribeVolumes",
                "ec2:AttachVolume",
                "ec2:DescribeInstanceAttribute",
                "ec2:DescribeInstanceStatus",
                "ec2:DescribeInstanceTypes",
                "ec2:DescribeInstances",
                "ec2:DescribeRegions",
                "ec2:TerminateInstances",
                "ec2:DescribeLaunchTemplates",
                "ec2:CreateTags"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow",
            "Sid": "EC2"
        },
        {
            "Action": "ec2:RunInstances",
            "Resource": [
```

```
                "arn:aws:ec2:<REGION>:<AWS ACCOUNT ID>:subnet/<COMPUTE SUBNET ID>",
                "arn:aws:ec2:<REGION>:<AWS ACCOUNT ID>:network-interface/*",
                "arn:aws:ec2:<REGION>:<AWS ACCOUNT ID>:instance/*",
                "arn:aws:ec2:<REGION>:<AWS ACCOUNT ID>:volume/*",
                "arn:aws:ec2:<REGION>::image/<IMAGE ID>",
                "arn:aws:ec2:<REGION>:<AWS ACCOUNT ID>:key-pair/<KEY NAME>",
                "arn:aws:ec2:<REGION>:<AWS ACCOUNT ID>:security-group/*",
                "arn:aws:ec2:<REGION>:<AWS ACCOUNT ID>:launch-template/*",
                "arn:aws:ec2:<REGION>:<AWS ACCOUNT ID>:placement-group/*"
            ],
            "Effect": "Allow",
            "Sid": "EC2RunInstances"
        },
        {
            "Action": [
                "dynamodb:ListTables"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow",
            "Sid": "DynamoDBList"
        },
        {
            "Action": [
                "sqs:SendMessage",
                "sqs:ReceiveMessage",
                "sqs:ChangeMessageVisibility",
                "sqs:DeleteMessage",
                "sqs:GetQueueUrl"
            ],
            "Resource": [
                "arn:aws:sqs:<REGION>:<AWS ACCOUNT ID>:parallelcluster-*"
            ],
            "Effect": "Allow",
            "Sid": "SQSQueue"
        },
        {
            "Action": [
                "autoscaling:DescribeAutoScalingGroups",
                "autoscaling:TerminateInstanceInAutoScalingGroup",
                "autoscaling:SetDesiredCapacity",
                "autoscaling:UpdateAutoScalingGroup",
                "autoscaling:DescribeTags",
```

```
                "autoscaling:SetInstanceHealth"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow",
            "Sid": "Autoscaling"
        },
        {
            "Action": [
                "cloudformation:DescribeStacks",
                "cloudformation:DescribeStackResource",
                "cloudformation:SignalResource"
            ],
            "Resource": [
                "arn:aws:cloudformation:<REGION>:<AWS ACCOUNT ID>:stack/
parallelcluster-*/*"
            ],
            "Effect": "Allow",
            "Sid": "CloudFormation"
        },
        {
            "Action": [
                "dynamodb:PutItem",
                "dynamodb:Query",
                "dynamodb:GetItem",
                "dynamodb:BatchWriteItem",
                "dynamodb:DeleteItem",
                "dynamodb:DescribeTable"
            ],
            "Resource": [
                "arn:aws:dynamodb:<REGION>:<AWS ACCOUNT ID>:table/parallelcluster-*"
            ],
            "Effect": "Allow",
            "Sid": "DynamoDBTable"
        },
        {
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::<REGION>-aws-parallelcluster/*"
            ],
            "Effect": "Allow",
```

```
            "Sid": "S3GetObj"
        },
        {
            "Action": [
                "sqs:ListQueues"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow",
            "Sid": "SQSList"
        },
        {
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow",
            "Sid": "IAMPassRole",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "ec2.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::dcv-license.<REGION>/*"
            ],
            "Effect": "Allow",
            "Sid": "DcvLicense"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
```

```
            "Resource": [
                "arn:aws:s3:::parallelcluster-*/*"
            ],
            "Effect": "Allow",
            "Sid": "GetClusterConfig"
        },
        {
            "Action": [
                "fsx:DescribeFileSystems"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow",
            "Sid": "FSx"
        },
        {
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow",
            "Sid": "CWLogs"
        },
        {
            "Action": [
                "route53:ChangeResourceRecordSets"
            ],
            "Resource": [
                "arn:aws:route53:::hostedzone/*"
            ],
            "Effect": "Allow",
            "Sid": "Route53"
        }
    ]
}
```

awsbatch を使用する ParallelClusterInstancePolicy

次の例では、スケジューラとして awsbatch を使用して ParallelClusterInstancePolicy を設定します。 AWS Batch AWS CloudFormation ネストされたスタックで定義されている BatchUserRole に割り当てられたのと同じポリシーを含める必要があります。BatchUserRole ARN はスタック出力として提供されます。この例では、*#<RESOURCES S3 BUCKET>#*が cluster_resource_bucket の設定値です。cluster_resource_bucket を指定しない場合は、*#<RESOURCES S3 BUCKET>#*が「parallelcluster-*」となります。次の例は、必要なパーミッションの概要です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "batch:RegisterJobDefinition",
                "logs:GetLogEvents"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "batch:SubmitJob",
                "cloudformation:DescribeStacks",
                "ecs:ListContainerInstances",
                "ecs:DescribeContainerInstances",
                "logs:FilterLogEvents",
                "s3:PutObject",
                "s3:Get*",
                "s3:DeleteObject",
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:batch:<REGION>:<AWS ACCOUNT ID>:job-
definition/<AWS_BATCH_STACK - JOB_DEFINITION_SERIAL_NAME>:1",
                "arn:aws:batch:<REGION>:<AWS ACCOUNT ID>:job-
definition/<AWS_BATCH_STACK - JOB_DEFINITION_MNP_NAME>*",
                "arn:aws:batch:<REGION>:<AWS ACCOUNT ID>:job-queue/<AWS_BATCH_STACK -
 JOB_QUEUE_NAME>",
```

```
                "arn:aws:cloudformation:<REGION>:<AWS ACCOUNT ID>:stack/<STACK NAME>/
*",
                "arn:aws:s3:::<RESOURCES S3 BUCKET>/batch/*",
                "arn:aws:iam::<AWS ACCOUNT ID>:role/<AWS_BATCH_STACK - JOB_ROLE>",
                "arn:aws:ecs:<REGION>:<AWS ACCOUNT ID>:cluster/<ECS COMPUTE
 ENVIRONMENT>",
                "arn:aws:ecs:<REGION>:<AWS ACCOUNT ID>:container-instance/*",
                "arn:aws:logs:<REGION>:<AWS ACCOUNT ID>:log-group:/aws/batch/job:log-
stream:*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::<RESOURCES S3 BUCKET>"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "batch:DescribeJobQueues",
                "batch:TerminateJob",
                "batch:DescribeJobs",
                "batch:CancelJob",
                "batch:DescribeJobDefinitions",
                "batch:ListJobs",
                "batch:DescribeComputeEnvironments"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "ec2:DescribeInstances",
                "ec2:AttachVolume",
                "ec2:DescribeVolumes",
                "ec2:DescribeInstanceAttribute"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "EC2"
```

```
        },
        {
            "Action": [
                "cloudformation:DescribeStackResource",
                "cloudformation:SignalResource"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "CloudFormation"
        },
        {
            "Action": [
                "fsx:DescribeFileSystems"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow",
            "Sid": "FSx"
        },
        {
            "Action": [
                "logs:CreateLogGroup",
                "logs:TagResource",
                "logs:UntagResource",
                "logs:CreateLogStream"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow",
            "Sid": "CWLogs"
        }
    ]
}
```

Slurm を使用する **ParallelClusterUserPolicy**

次の例では、スケジューラとして Slurm を使用して ParallelClusterUserPolicy を設定します。この例では、*#<RESOURCES S3 BUCKET>#*が cluster_resource_bucket の設定値です。cluster_resource_bucket を指定しない場合は、*#<RESOURCES S3 BUCKET>#*が「parallelcluster-*」となります。

> **Note**
>
> ec2_iam_role = *<role_name>* というカスタムロールを使用する場合は、IAM リソースからそのロール名を含めるように変更する必要があります。
>
> `"Resource": "arn:aws:iam::<AWS ACCOUNT ID>:role/parallelcluster-*"`
>
> 変更後:
>
> `"Resource": "arn:aws:iam::<AWS ACCOUNT ID>:role/<role_name>"`

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ec2:DescribeKeyPairs",
                "ec2:DescribeRegions",
                "ec2:DescribeVpcs",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribePlacementGroups",
                "ec2:DescribeImages",
                "ec2:DescribeInstances",
                "ec2:DescribeInstanceStatus",
                "ec2:DescribeInstanceTypes",
                "ec2:DescribeInstanceTypeOfferings",
                "ec2:DescribeSnapshots",
                "ec2:DescribeVolumes",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeAddresses",
                "ec2:CreateTags",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeAvailabilityZones"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "EC2Describe"
        },
        {
            "Action": [
                "ec2:CreateVpc",
                "ec2:ModifyVpcAttribute",
                "ec2:DescribeNatGateways",
```

```
                "ec2:CreateNatGateway",
                "ec2:DescribeInternetGateways",
                "ec2:CreateInternetGateway",
                "ec2:AttachInternetGateway",
                "ec2:DescribeRouteTables",
                "ec2:CreateRoute",
                "ec2:CreateRouteTable",
                "ec2:AssociateRouteTable",
                "ec2:CreateSubnet",
                "ec2:ModifySubnetAttribute"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "NetworkingEasyConfig"
        },
        {
            "Action": [
                "ec2:CreateVolume",
                "ec2:RunInstances",
                "ec2:AllocateAddress",
                "ec2:AssociateAddress",
                "ec2:AttachNetworkInterface",
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:CreateNetworkInterface",
                "ec2:CreateSecurityGroup",
                "ec2:ModifyVolumeAttribute",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteVolume",
                "ec2:TerminateInstances",
                "ec2:DeleteSecurityGroup",
                "ec2:DisassociateAddress",
                "ec2:RevokeSecurityGroupIngress",
                "ec2:RevokeSecurityGroupEgress",
                "ec2:ReleaseAddress",
                "ec2:CreatePlacementGroup",
                "ec2:DeletePlacementGroup"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "EC2Modify"
        },
        {
```

```
            "Action": [
                "autoscaling:CreateAutoScalingGroup",
                "ec2:CreateLaunchTemplate",
                "ec2:CreateLaunchTemplateVersion",
                "ec2:ModifyLaunchTemplate",
                "ec2:DeleteLaunchTemplate",
                "ec2:DescribeLaunchTemplates",
                "ec2:DescribeLaunchTemplateVersions"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "ScalingModify"
        },
        {
            "Action": [
                "dynamodb:DescribeTable",
                "dynamodb:ListTagsOfResource"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "DynamoDBDescribe"
        },
        {
            "Action": [
                "dynamodb:CreateTable",
                "dynamodb:DeleteTable",
                "dynamodb:GetItem",
                "dynamodb:PutItem",
                "dynamodb:Query",
                "dynamodb:TagResource"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "DynamoDBModify"
        },
        {
            "Action": [
                "route53:ChangeResourceRecordSets",
                "route53:ChangeTagsForResource",
                "route53:CreateHostedZone",
                "route53:DeleteHostedZone",
                "route53:GetChange",
                "route53:GetHostedZone",
                "route53:ListResourceRecordSets",
```

```
                "route53:ListQueryLoggingConfigs"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "Route53HostedZones"
        },
        {
            "Action": [
                "cloudformation:DescribeStackEvents",
                "cloudformation:DescribeStackResource",
                "cloudformation:DescribeStackResources",
                "cloudformation:DescribeStacks",
                "cloudformation:ListStacks",
                "cloudformation:GetTemplate"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "CloudFormationDescribe"
        },
        {
            "Action": [
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:UpdateStack"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Sid": "CloudFormationModify"
        },
        {
            "Action": [
                "s3:*"
            ],
            "Resource": [
                "arn:aws:s3:::<RESOURCES S3 BUCKET>"
            ],
            "Effect": "Allow",
            "Sid": "S3ResourcesBucket"
        },
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
```

```
            "Resource": [
                "arn:aws:s3:::<REGION>-aws-parallelcluster*"
            ],
            "Effect": "Allow",
            "Sid": "S3ParallelClusterReadOnly"
        },
        {
            "Action": [
                "s3:DeleteBucket",
                "s3:DeleteObject",
                "s3:DeleteObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::<RESOURCES S3 BUCKET>"
            ],
            "Effect": "Allow",
            "Sid": "S3Delete"
        },
        {
            "Action": [
                "iam:PassRole",
                "iam:CreateRole",
                "iam:DeleteRole",
                "iam:GetRole",
                "iam:TagRole",
                "iam:SimulatePrincipalPolicy"
            ],
            "Resource": [
                "arn:aws:iam::<AWS ACCOUNT ID>:role/<PARALLELCLUSTER EC2 ROLE NAME>",
                "arn:aws:iam::<AWS ACCOUNT ID>:role/parallelcluster-*"
            ],
            "Effect": "Allow",
            "Sid": "IAMModify"
        },
        {
          "Condition": {
              "StringEquals": {
                  "iam:AWSServiceName": [
                      "fsx.amazonaws.com",
                      "s3.data-source.lustre.fsx.amazonaws.com"
                  ]
              }
          },
          "Action": [
```

```
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "arn:aws:iam::<AWS ACCOUNT ID>:role/aws-service-role/*",
            "Effect": "Allow",
            "Sid": "IAMServiceLinkedRole"
        },
        {
            "Action": [
                "iam:CreateInstanceProfile",
                "iam:DeleteInstanceProfile"
            ],
            "Resource": "arn:aws:iam::<AWS ACCOUNT ID>:instance-profile/*",
            "Effect": "Allow",
            "Sid": "IAMCreateInstanceProfile"
        },
        {
            "Action": [
                "iam:AddRoleToInstanceProfile",
                "iam:RemoveRoleFromInstanceProfile",
                "iam:GetRolePolicy",
                "iam:GetPolicy",
                "iam:AttachRolePolicy",
                "iam:DetachRolePolicy",
                "iam:PutRolePolicy",
                "iam:DeleteRolePolicy"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "IAMInstanceProfile"
        },
        {
            "Action": [
                "elasticfilesystem:DescribeMountTargets",
                "elasticfilesystem:DescribeMountTargetSecurityGroups",
                "ec2:DescribeNetworkInterfaceAttribute"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "EFSDescribe"
        },
        {
            "Action": [
                "ssm:GetParametersByPath"
            ],
```

```
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "SSMDescribe"
        },
        {
            "Action": [
                "fsx:*"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "FSx"
        },
        {
            "Action": [
                "elasticfilesystem:*"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "EFS"
        },
        {
            "Action": [
                "logs:DeleteLogGroup",
                "logs:PutRetentionPolicy",
                "logs:DescribeLogGroups",
                "logs:CreateLogGroup",
                "logs:TagResource",
                "logs:UntagResource"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "CloudWatchLogs"
        },
        {
            "Action": [
                "lambda:CreateFunction",
                "lambda:DeleteFunction",
                "lambda:GetFunctionConfiguration",
                "lambda:GetFunction",
                "lambda:InvokeFunction",
                "lambda:AddPermission",
                "lambda:RemovePermission",
                "lambda:TagResource",
                "lambda:ListTags",
```

```
                "lambda:UntagResource"
            ],
            "Resource": [
                "arn:aws:lambda:<REGION>:<AWS ACCOUNT ID>:function:parallelcluster-*",
                "arn:aws:lambda:<REGION>:<AWS ACCOUNT ID>:function:pcluster-*"
            ],
            "Effect": "Allow",
            "Sid": "Lambda"
        },
        {
            "Sid": "CloudWatch",
            "Effect": "Allow",
            "Action": [
              "cloudwatch:PutDashboard",
              "cloudwatch:ListDashboards",
              "cloudwatch:DeleteDashboards",
              "cloudwatch:GetDashboard"
            ],
            "Resource": "*"
        }
    ]
}
```

SGE または Torque を使用する **ParallelClusterUserPolicy**

> **Note**
>
> このセクションは、 AWS ParallelCluster バージョン 2.11.4 以前のバージョンにのみ適用されます。バージョン 2.11.5 以降、 AWS ParallelCluster は SGE または Torque スケジューラの使用をサポートしていません。

次の例では、スケジューラとして、 SGE または Torque を使用して `ParallelClusterUserPolicy` を設定します。この例では、*#<RESOURCES S3 BUCKET>#*が `cluster_resource_bucket` の設定値です。`cluster_resource_bucket` を指定しない場合は、*#<RESOURCES S3 BUCKET>#*が「parallelcluster-*」となります。

> **Note**
>
> `ec2_iam_role = <role_name>` というカスタムロールを使用する場合は、IAM リソースからそのロール名を含めるように変更する必要があります。

```
    "Resource": "arn:aws:iam::<AWS ACCOUNT ID>:role/parallelcluster-*"
    変更後:
    "Resource": "arn:aws:iam::<AWS ACCOUNT ID>:role/<role_name>"
```

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ec2:DescribeKeyPairs",
                "ec2:DescribeRegions",
                "ec2:DescribeVpcs",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribePlacementGroups",
                "ec2:DescribeImages",
                "ec2:DescribeInstances",
                "ec2:DescribeInstanceStatus",
                "ec2:DescribeInstanceTypes",
                "ec2:DescribeInstanceTypeOfferings",
                "ec2:DescribeSnapshots",
                "ec2:DescribeVolumes",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeAddresses",
                "ec2:CreateTags",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeAvailabilityZones"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "EC2Describe"
        },
        {
            "Action": [
                "ec2:CreateVpc",
                "ec2:ModifyVpcAttribute",
                "ec2:DescribeNatGateways",
                "ec2:CreateNatGateway",
                "ec2:DescribeInternetGateways",
                "ec2:CreateInternetGateway",
                "ec2:AttachInternetGateway",
```

```
                "ec2:DescribeRouteTables",
                "ec2:CreateRoute",
                "ec2:CreateRouteTable",
                "ec2:AssociateRouteTable",
                "ec2:CreateSubnet",
                "ec2:ModifySubnetAttribute"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "NetworkingEasyConfig"
        },
        {
            "Action": [
                "ec2:CreateVolume",
                "ec2:RunInstances",
                "ec2:AllocateAddress",
                "ec2:AssociateAddress",
                "ec2:AttachNetworkInterface",
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:CreateNetworkInterface",
                "ec2:CreateSecurityGroup",
                "ec2:ModifyVolumeAttribute",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteVolume",
                "ec2:TerminateInstances",
                "ec2:DeleteSecurityGroup",
                "ec2:DisassociateAddress",
                "ec2:RevokeSecurityGroupIngress",
                "ec2:RevokeSecurityGroupEgress",
                "ec2:ReleaseAddress",
                "ec2:CreatePlacementGroup",
                "ec2:DeletePlacementGroup"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "EC2Modify"
        },
        {
            "Action": [
                "autoscaling:DescribeAutoScalingGroups",
                "autoscaling:DescribeAutoScalingInstances"
            ],
```

```
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "AutoScalingDescribe"
        },
        {
            "Action": [
                "autoscaling:CreateAutoScalingGroup",
                "ec2:CreateLaunchTemplate",
                "ec2:CreateLaunchTemplateVersion",
                "ec2:ModifyLaunchTemplate",
                "ec2:DeleteLaunchTemplate",
                "ec2:DescribeLaunchTemplates",
                "ec2:DescribeLaunchTemplateVersions",
                "autoscaling:PutNotificationConfiguration",
                "autoscaling:UpdateAutoScalingGroup",
                "autoscaling:PutScalingPolicy",
                "autoscaling:DescribeScalingActivities",
                "autoscaling:DeleteAutoScalingGroup",
                "autoscaling:DeletePolicy",
                "autoscaling:DisableMetricsCollection",
                "autoscaling:EnableMetricsCollection"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "AutoScalingModify"
        },
        {
            "Action": [
                "dynamodb:DescribeTable",
                "dynamodb:ListTagsOfResource"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "DynamoDBDescribe"
        },
        {
            "Action": [
                "dynamodb:CreateTable",
                "dynamodb:DeleteTable",
                "dynamodb:GetItem",
                "dynamodb:PutItem",
                "dynamodb:Query",
                "dynamodb:TagResource"
            ],
```

```
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "DynamoDBModify"
        },
        {
            "Action": [
                "sqs:GetQueueAttributes"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "SQSDescribe"
        },
        {
            "Action": [
                "sqs:CreateQueue",
                "sqs:SetQueueAttributes",
                "sqs:DeleteQueue",
                "sqs:TagQueue"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "SQSModify"
        },
        {
            "Action": [
                "sns:ListTopics",
                "sns:GetTopicAttributes"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "SNSDescribe"
        },
        {
            "Action": [
                "sns:CreateTopic",
                "sns:Subscribe",
                "sns:Unsubscribe",
                "sns:DeleteTopic"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "SNSModify"
        },
        {
```

```
            "Action": [
                "cloudformation:DescribeStackEvents",
                "cloudformation:DescribeStackResource",
                "cloudformation:DescribeStackResources",
                "cloudformation:DescribeStacks",
                "cloudformation:ListStacks",
                "cloudformation:GetTemplate"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "CloudFormationDescribe"
        },
        {
            "Action": [
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:UpdateStack"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Sid": "CloudFormationModify"
        },
        {
            "Action": [
                "s3:*"
            ],
            "Resource": [
                "arn:aws:s3:::<RESOURCES S3 BUCKET>"
            ],
            "Effect": "Allow",
            "Sid": "S3ResourcesBucket"
        },
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::<REGION>-aws-parallelcluster*"
            ],
            "Effect": "Allow",
            "Sid": "S3ParallelClusterReadOnly"
        },
        {
```

```
            "Action": [
                "s3:DeleteBucket",
                "s3:DeleteObject",
                "s3:DeleteObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::<RESOURCES S3 BUCKET>"
            ],
            "Effect": "Allow",
            "Sid": "S3Delete"
        },
        {
            "Action": [
                "iam:PassRole",
                "iam:CreateRole",
                "iam:DeleteRole",
                "iam:GetRole",
                "iam:TagRole",
                "iam:SimulatePrincipalPolicy"
            ],
            "Resource": [
                "arn:aws:iam::<AWS ACCOUNT ID>:role/<PARALLELCLUSTER EC2 ROLE NAME>",
                "arn:aws:iam::<AWS ACCOUNT ID>:role/parallelcluster-*"
            ],
            "Effect": "Allow",
            "Sid": "IAMModify"
        },
        {
          "Condition": {
              "StringEquals": {
                  "iam:AWSServiceName": [
                      "fsx.amazonaws.com",
                      "s3.data-source.lustre.fsx.amazonaws.com"
                  ]
              }
          },
          "Action": [
              "iam:CreateServiceLinkedRole"
          ],
          "Resource": "arn:aws:iam::<AWS ACCOUNT ID>:role/aws-service-role/*",
          "Effect": "Allow",
          "Sid": "IAMServiceLinkedRole"
        },
        {
```

```
            "Action": [
                "iam:CreateInstanceProfile",
                "iam:DeleteInstanceProfile"
            ],
            "Resource": "arn:aws:iam::<AWS ACCOUNT ID>:instance-profile/*",
            "Effect": "Allow",
            "Sid": "IAMCreateInstanceProfile"
        },
        {
            "Action": [
                "iam:AddRoleToInstanceProfile",
                "iam:RemoveRoleFromInstanceProfile",
                "iam:GetRolePolicy",
                "iam:GetPolicy",
                "iam:AttachRolePolicy",
                "iam:DetachRolePolicy",
                "iam:PutRolePolicy",
                "iam:DeleteRolePolicy"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "IAMInstanceProfile"
        },
        {
            "Action": [
                "elasticfilesystem:DescribeMountTargets",
                "elasticfilesystem:DescribeMountTargetSecurityGroups",
                "ec2:DescribeNetworkInterfaceAttribute"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "EFSDescribe"
        },
        {
            "Action": [
                "ssm:GetParametersByPath"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "SSMDescribe"
        },
        {
            "Action": [
                "fsx:*"
```

```
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "FSx"
        },
        {
            "Action": [
                "elasticfilesystem:*"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "EFS"
        },
        {
            "Action": [
                "logs:DeleteLogGroup",
                "logs:PutRetentionPolicy",
                "logs:DescribeLogGroups",
                "logs:CreateLogGroup",
                "logs:TagResource",
                "logs:UntagResource"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "CloudWatchLogs"
        },
        {
            "Action": [
                "lambda:CreateFunction",
                "lambda:DeleteFunction",
                "lambda:GetFunctionConfiguration",
                "lambda:GetFunction",
                "lambda:InvokeFunction",
                "lambda:AddPermission",
                "lambda:RemovePermission",
                "lambda:TagResource",
                "lambda:ListTags",
                "lambda:UntagResource"
            ],
            "Resource": [
                "arn:aws:lambda:<REGION>:<AWS ACCOUNT ID>:function:parallelcluster-*",
                "arn:aws:lambda:<REGION>:<AWS ACCOUNT ID>:function:pcluster-*"
            ],
            "Effect": "Allow",
```

```
            "Sid": "Lambda"
        },
        {
            "Sid": "CloudWatch",
            "Effect": "Allow",
            "Action": [
              "cloudwatch:PutDashboard",
              "cloudwatch:ListDashboards",
              "cloudwatch:DeleteDashboards",
              "cloudwatch:GetDashboard"
            ],
            "Resource": "*"
        }
    ]
}
```

awsbatch を使用する ParallelClusterUserPolicy

次の例では、スケジューラとして awsbatch を使用して ParallelClusterUserPolicy を設定します。この例では、*#<RESOURCES S3 BUCKET>#*が cluster_resource_bucket の設定値です。cluster_resource_bucket を指定しない場合は、*#<RESOURCES S3 BUCKET>#*が「parallelcluster-*」となります。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ec2:DescribeKeyPairs",
                "ec2:DescribeRegions",
                "ec2:DescribeVpcs",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribePlacementGroups",
                "ec2:DescribeImages",
                "ec2:DescribeInstances",
                "ec2:DescribeInstanceStatus",
                "ec2:DescribeInstanceTypes",
                "ec2:DescribeInstanceTypeOfferings",
                "ec2:DescribeSnapshots",
                "ec2:DescribeVolumes",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeAddresses",
```

```
                "ec2:CreateTags",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeAvailabilityZones"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "EC2Describe"
        },
        {
            "Action": [
                "ec2:CreateLaunchTemplate",
                "ec2:CreateLaunchTemplateVersion",
                "ec2:ModifyLaunchTemplate",
                "ec2:DeleteLaunchTemplate",
                "ec2:DescribeLaunchTemplates",
                "ec2:DescribeLaunchTemplateVersions"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "EC2LaunchTemplate"
        },
        {
            "Action": [
                "ec2:CreateVpc",
                "ec2:ModifyVpcAttribute",
                "ec2:DescribeNatGateways",
                "ec2:CreateNatGateway",
                "ec2:DescribeInternetGateways",
                "ec2:CreateInternetGateway",
                "ec2:AttachInternetGateway",
                "ec2:DescribeRouteTables",
                "ec2:CreateRoute",
                "ec2:CreateRouteTable",
                "ec2:AssociateRouteTable",
                "ec2:CreateSubnet",
                "ec2:ModifySubnetAttribute"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "NetworkingEasyConfig"
        },
        {
            "Action": [
                "ec2:CreateVolume",
```

```
                "ec2:RunInstances",
                "ec2:AllocateAddress",
                "ec2:AssociateAddress",
                "ec2:AttachNetworkInterface",
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:CreateNetworkInterface",
                "ec2:CreateSecurityGroup",
                "ec2:ModifyVolumeAttribute",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteVolume",
                "ec2:TerminateInstances",
                "ec2:DeleteSecurityGroup",
                "ec2:DisassociateAddress",
                "ec2:RevokeSecurityGroupIngress",
                "ec2:RevokeSecurityGroupEgress",
                "ec2:ReleaseAddress",
                "ec2:CreatePlacementGroup",
                "ec2:DeletePlacementGroup"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "EC2Modify"
        },
        {
            "Action": [
                "dynamodb:DescribeTable",
                "dynamodb:CreateTable",
                "dynamodb:DeleteTable",
                "dynamodb:GetItem",
                "dynamodb:PutItem",
                "dynamodb:Query",
                "dynamodb:TagResource"
            ],
            "Resource": "arn:aws:dynamodb:<REGION>:<AWS ACCOUNT ID>:table/
parallelcluster-*",
            "Effect": "Allow",
            "Sid": "DynamoDB"
        },
        {
            "Action": [
                "cloudformation:DescribeStackEvents",
                "cloudformation:DescribeStackResource",
```

```
                "cloudformation:DescribeStackResources",
                "cloudformation:DescribeStacks",
                "cloudformation:ListStacks",
                "cloudformation:GetTemplate",
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:UpdateStack"
            ],
            "Resource": "arn:aws:cloudformation:<REGION>:<AWS ACCOUNT ID>:stack/
parallelcluster-*",
            "Effect": "Allow",
            "Sid": "CloudFormation"
        },
        {
            "Action": [
                "route53:ChangeResourceRecordSets",
                "route53:ChangeTagsForResource",
                "route53:CreateHostedZone",
                "route53:DeleteHostedZone",
                "route53:GetChange",
                "route53:GetHostedZone",
                "route53:ListResourceRecordSets"
            ],
            "Resource": "arn:aws:route53:::hostedzone/*",
            "Effect": "Allow",
            "Sid": "Route53HostedZones"
        },
        {
            "Action": [
                "sqs:GetQueueAttributes",
                "sqs:CreateQueue",
                "sqs:SetQueueAttributes",
                "sqs:DeleteQueue",
                "sqs:TagQueue"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "SQS"
        },
        {
            "Action": [
                "sqs:SendMessage",
                "sqs:ReceiveMessage",
                "sqs:ChangeMessageVisibility",
```

```
                "sqs:DeleteMessage",
                "sqs:GetQueueUrl"
            ],
            "Resource": "arn:aws:sqs:<REGION>:<AWS ACCOUNT ID>:parallelcluster-*",
            "Effect": "Allow",
            "Sid": "SQSQueue"
        },
        {
            "Action": [
                "sns:ListTopics",
                "sns:GetTopicAttributes",
                "sns:CreateTopic",
                "sns:Subscribe",
                "sns:Unsubscribe",
                "sns:DeleteTopic"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "SNS"
        },
        {
            "Action": [
                "iam:PassRole",
                "iam:CreateRole",
                "iam:DeleteRole",
                "iam:GetRole",
                "iam:TagRole",
                "iam:SimulatePrincipalPolicy"
            ],
            "Resource": [
                "arn:aws:iam::<AWS ACCOUNT ID>:role/parallelcluster-*",
                "arn:aws:iam::<AWS ACCOUNT ID>:role/<PARALLELCLUSTER EC2 ROLE NAME>"
            ],
            "Effect": "Allow",
            "Sid": "IAMRole"
        },
        {
            "Action": [
                "iam:CreateInstanceProfile",
                "iam:DeleteInstanceProfile",
                "iam:GetInstanceProfile",
                "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::<AWS ACCOUNT ID>:instance-profile/*",
```

```
            "Effect": "Allow",
            "Sid": "IAMInstanceProfile"
        },
        {
            "Action": [
                "iam:AddRoleToInstanceProfile",
                "iam:RemoveRoleFromInstanceProfile",
                "iam:GetRolePolicy",
                "iam:PutRolePolicy",
                "iam:DeleteRolePolicy",
                "iam:GetPolicy",
                "iam:AttachRolePolicy",
                "iam:DetachRolePolicy"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "IAM"
        },
        {
            "Action": [
                "s3:*"
            ],
            "Resource": [
                "arn:aws:s3:::<RESOURCES S3 BUCKET>"
            ],
            "Effect": "Allow",
            "Sid": "S3ResourcesBucket"
        },
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::<REGION>-aws-parallelcluster/*"
            ],
            "Effect": "Allow",
            "Sid": "S3ParallelClusterReadOnly"
        },
        {
            "Action": [
                "s3:DeleteBucket",
                "s3:DeleteObject",
                "s3:DeleteObjectVersion"
```

```
            ],
            "Resource": [
                "arn:aws:s3:::<RESOURCES S3 BUCKET>"
            ],
            "Effect": "Allow",
            "Sid": "S3Delete"
        },
        {
            "Action": [
                "lambda:CreateFunction",
                "lambda:DeleteFunction",
                "lambda:GetFunction",
                "lambda:GetFunctionConfiguration",
                "lambda:InvokeFunction",
                "lambda:AddPermission",
                "lambda:RemovePermission",
                "lambda:TagResource",
                "lambda:ListTags",
                "lambda:UntagResource"
            ],
            "Resource": [
                "arn:aws:lambda:<REGION>:<AWS ACCOUNT ID>:function:parallelcluster-*",
                "arn:aws:lambda:<REGION>:<AWS ACCOUNT ID>:function:pcluster-*"
            ],
            "Effect": "Allow",
            "Sid": "Lambda"
        },
        {
            "Action": [
                "logs:*"
            ],
            "Resource": "arn:aws:logs:<REGION>:<AWS ACCOUNT ID>:*",
            "Effect": "Allow",
            "Sid": "Logs"
        },
        {
            "Action": [
                "codebuild:*"
            ],
            "Resource": "arn:aws:codebuild:<REGION>:<AWS ACCOUNT ID>:project/
parallelcluster-*",
            "Effect": "Allow",
            "Sid": "CodeBuild"
        },
```

```
        {
            "Action": [
                "ecr:*"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "ECR"
        },
        {
            "Action": [
                "batch:*"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "Batch"
        },
        {
            "Action": [
                "events:*"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Sid": "AmazonCloudWatchEvents"
        },
        {
            "Action": [
                "ecs:DescribeContainerInstances",
                "ecs:ListContainerInstances"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "ECS"
        },
        {
            "Action": [
                "elasticfilesystem:CreateFileSystem",
                "elasticfilesystem:CreateMountTarget",
                "elasticfilesystem:DeleteFileSystem",
                "elasticfilesystem:DeleteMountTarget",
                "elasticfilesystem:DescribeFileSystems",
                "elasticfilesystem:DescribeMountTargets"
            ],
            "Resource": "*",
            "Effect": "Allow",
```

```
            "Sid": "EFS"
        },
        {
            "Action": [
                "fsx:*"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "FSx"
        },
        {
            "Sid": "CloudWatch",
            "Effect": "Allow",
            "Action": [
              "cloudwatch:PutDashboard",
              "cloudwatch:ListDashboards",
              "cloudwatch:DeleteDashboards",
              "cloudwatch:GetDashboard"
            ],
            "Resource": "*"
        }
    ]
}
```

SGE、Slurm または Torque を使用する **ParallelClusterLambdaPolicy**

次の例では、スケジューラとして SGE、Slurm、または Torque を使用して `ParallelClusterLambdaPolicy` を設定します。

> **Note**
>
> バージョン 2.11.5 以降、 AWS ParallelCluster は SGEまたは スTorqueケジューラの使用をサポートしていません。

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

```
      ],
      "Resource": "arn:aws:logs:*:*:*",
      "Effect": "Allow",
      "Sid": "CloudWatchLogsPolicy"
    },
    {
      "Action": [
        "s3:DeleteBucket",
        "s3:DeleteObject",
        "s3:DeleteObjectVersion",
        "s3:ListBucket",
        "s3:ListBucketVersions"
      ],
      "Resource": [
        "arn:aws:s3:::*"
      ],
      "Effect": "Allow",
      "Sid": "S3BucketPolicy"
    },
    {
      "Action": [
        "ec2:DescribeInstances"
      ],
      "Resource": "*",
      "Effect": "Allow",
      "Sid": "DescribeInstances"
    },
    {
      "Action": [
        "ec2:TerminateInstances"
      ],
      "Resource": "*",
      "Effect": "Allow",
      "Sid": "FleetTerminatePolicy"
    },
    {
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:PutItem"
      ],
      "Resource": "arn:aws:dynamodb:<REGION>:<AWS ACCOUNT ID>:table/parallelcluster-*",
      "Effect": "Allow",
      "Sid": "DynamoDBTable"
    },
```

```
    {
      "Action": [
        "route53:ListResourceRecordSets",
        "route53:ChangeResourceRecordSets"
      ],
      "Resource": [
        "arn:aws:route53:::hostedzone/*"
      ],
      "Effect": "Allow",
      "Sid": "Route53DeletePolicy"
    }
  ]
}
```

awsbatch を使用する ParallelClusterLambdaPolicy

次の例では、スケジューラとして awsbatch を使用して ParallelClusterLambdaPolicy を設定します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:logs:*:*:*",
      "Sid": "CloudWatchLogsPolicy"
    },
    {
      "Action": [
        "ecr:BatchDeleteImage",
        "ecr:ListImages"
      ],
      "Effect": "Allow",
      "Resource": "*",
      "Sid": "ECRPolicy"
    },
    {
      "Action": [
        "codebuild:BatchGetBuilds",
```

```
        "codebuild:StartBuild"
      ],
      "Effect": "Allow",
      "Resource": "*",
      "Sid": "CodeBuildPolicy"
    },
    {
      "Action": [
        "s3:DeleteBucket",
        "s3:DeleteObject",
        "s3:DeleteObjectVersion",
        "s3:ListBucket",
        "s3:ListBucketVersions"
      ],
      "Effect": "Allow",
      "Resource": "*",
      "Sid": "S3BucketPolicy"
    }
  ]
}
```

ユーザー用の ParallelClusterUserPolicy

次の例では、クラスターを作成または更新する必要がないユーザーの ParallelClusterUserPolicy を設定します。以下のコマンドがサポートされています。

- pcluster dcv
- pcluster instances
- pcluster list
- pcluster ssh
- pcluster start
- pcluster status
- pcluster stop
- pcluster version

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

```
              "Sid": "MinimumModify",
              "Action": [
                 "autoscaling:UpdateAutoScalingGroup",
                 "batch:UpdateComputeEnvironment",
                 "cloudformation:DescribeStackEvents",
                 "cloudformation:DescribeStackResources",
                 "cloudformation:GetTemplate",
                 "dynamodb:GetItem",
                 "dynamodb:PutItem"
              ],
              "Effect": "Allow",
              "Resource": [
                  "arn:aws:autoscaling:<REGION>:<AWS ACCOUNT
 ID>:autoScalingGroup:*:autoScalingGroupName/parallelcluster-*",
                  "arn:aws:batch:<REGION>:<AWS ACCOUNT ID>:compute-environment/*",
                  "arn:aws:cloudformation:<REGION>:<AWS ACCOUNT ID>:stack/<CLUSTERNAME>/
*",
                  "arn:aws:dynamodb:<REGION>:<AWS ACCOUNT ID>:table/<CLUSTERNAME>"
              ]
        },
        {
           "Sid": "Describe",
           "Action": [
                "cloudformation:DescribeStacks",
                "ec2:DescribeInstances",
                "ec2:DescribeInstanceStatus"
           ],
           "Effect": "Allow",
           "Resource": "*"
        }
    ]
}
```

でサポートされているスケジューラ AWS ParallelCluster

AWS ParallelCluster は、 `scheduler`設定を使用して設定された複数のスケジューラをサポートします。

> **Note**
>
> バージョン 2.11.5 以降、 AWS ParallelCluster は SGEまたは スTorqueケジューラの使用をサポートしていません。2.11.4 までのバージョンで引き続き使用できますが、 AWS サービ

スおよび AWS サポートチームによる今後の更新やトラブルシューティングのサポートを受けることはできません。

トピック

- Son of Grid Engine (sge)
- Slurm Workload Manager (slurm)
- Torque Resource Manager (torque)
- AWS Batch (awsbatch)

Son of Grid Engine (`sge`)

> **Note**
>
> バージョン 2.11.5 以降、 AWS ParallelCluster は SGEまたは スTorqueケジューラの使用をサポートしていません。2.11.4 までのバージョンで引き続き使用できますが、 AWS サービスおよび AWS サポートチームによる今後の更新やトラブルシューティングのサポートを受けることはできません。

AWS ParallelCluster バージョン 2.11.4 以前では Son of Grid Engine 8.1.9 を使用しています。

Slurm Workload Manager (`slurm`)

AWS ParallelCluster バージョン 2.11.9 では 20.11.9 Slurm が使用されます。Slurm の詳細については「https://slurm.schedmd.com/」を参照してください。ダウンロードについては、「https://github.com/SchedMD/slurm/tags」を参照してください。出典コードについては、「https://github.com/SchedMD/slurm」を参照してください。

> **Important**
>
> AWS ParallelCluster は、デフォルトで提供されるSlurm設定パラメータでテストされます。これらの Slurm 設定パラメータを変更する場合は、お客様ご自身の責任で行ってください。ベストエフォート型のみサポートされます。

AWS ParallelCluster バージョン (複数可)	サポートされる Slurm のバージョン
2.11.7、2.11.8、2.11.9	20.11.9
2.11.4 から 2.11.6	20.11.8
2.11.0 から 2.11.3	20.11.7
2.10.4	20.02.7
2.9.0 から 2.10.3	20.02.4
2.6 から 2.8.1	19.05.5
2.5.0、2.5.1	19.05.3-2
2.3.1 から 2.4.1	18.08.6-2
2.3.1 より前	16.05.3-1

マルチキューモード

AWS ParallelCluster バージョン 2.9.0 では、複数のキューモードが導入されました。`scheduler` を `slurm` に設定し、`queue_settings` を定義した場合、マルチキューモードがサポートされます。このモードでは、異なるタイプのインスタンスをコンピューティングノードに共存させることができます。異なるインスタンスタイプを含むコンピューティングリソースは、必要に応じてスケールアップまたはスケールダウンすることができます。キューモードでは、最大 5 つのキューをサポートし、各 `[queue]` セクションは、最大 3 つの `[compute_resource]` セクションを参照することができます。この `[queue]` セクションは、それぞれ Slurm Workload Manager のパーティションにあたります。詳細については、「マルチキューモードの Slurm ガイド」および「マルチキューモードのチュートリアル」を参照してください。

キュー内の `[compute_resource]` セクションはそれぞれ異なるインスタンスタイプを持つ必要があり、この `[compute_resource]` のそれぞれはさらに静的ノードと動的ノードに分けられる。各 `[compute_resource]` の静的ノードには、1 から `min_count` の値まで番号が振られます。各 `[compute_resource]` の動的ノードには、(1) ~ (`max_count` ~ `min_count`) の番号が振られています。例えば、`min_count` が 2、`max_count` が 10 の場合、その `[compute_resource]` の動的ノードには (1) ~ (8) の番号が振られます。`[compute_resource]` には、常に 0 から最大数までの動的ノードが存在することができます。

コンピューティングフリートに起動されるインスタンスは、動的に割り当てられます。これを管理するために、各ノードにはホスト名が生成されます。hostname の形式は次のとおりです。

```
$HOSTNAME=$QUEUE-$STATDYN-$INSTANCE_TYPE-$NODENUM
```

- $QUEUE はキューの名前です。たとえば、 セクションが開始すると[queue *queue-name*]、$QUEUE「」は*#queue-name*」になります。
- $STATDYN は、静的ノードの場合は st、動的ノードの場合は dy です。
- $INSTANCE_TYPE は instance_type 設定による [compute_resource] 用のインスタンスタイプです。
- $NODENUM はノードの番号です。$NODENUM は、静的ノードの場合は 1 以上 min_count 以下の値、動的ノードの場合は 1 以上 max_count〜min_count 以下の値です。

ホスト名と完全修飾ドメイン名 (FQDN) の両方は、Amazon Route 53 のホストゾーンを使用して作成されます。FQDN は $HOSTNAME.$CLUSTERNAME.pcluster で、$CLUSTERNAME はクラスターに使用する [cluster] セクションの名前です。

設定をキューモードに変換するには、pcluster-config convert コマンドを使用します。[queue compute] という 1 つの [queue] セクションを持つ更新されたコンフィギュレーションを書き込みます。そのキューには、[compute_resource default] という名前の [compute_resource]セクションが 1 つ含まれています。[queue compute] と [compute_resource default] は、指定された [cluster] セクションから設定が移行されています。

マルチキューモードの Slurm ガイド

AWS ParallelCluster バージョン 2.9.0 では、複数のキューモードと Slurm Workload Manager () の新しいスケーリングアーキテクチャが導入されましたSlurm。

次のセクションでは、新しく導入されたスケーリングアーキテクチャを備えた Slurm クラスターを使用する際の一般的な概要を説明します。

概要

新しいスケーリングアーキテクチャは、Slurm の「Cloud Scheduling Guide」(クラウドスケジュールガイド) と省電力プラグインに基づいています。省電力プラグインの詳細については、「Slurm Power Saving Guide」を参照してください。新しいアーキテクチャでは、クラスターで利用できる可能性のあるリソースは、通常、クラウドノードとして Slurm 設定にあらかじめ定義されています。

クラウドノードのライフサイクル

クラウドノードはそのライフサイクルを通じて、すべてではないが以下のような状態になります。POWER_SAVING、POWER_UP (pow_up)、ALLOCATED (alloc)、および POWER_DOWN (pow_dn)。場合によっては、クラウドノードが OFFLINE 状態になることがあります。次のリストは、クラウドノードのライフサイクルにおけるこれらの状態のいくつかの側面を詳細に示しています。

- POWER_SAVING 状態のノードは、sinfo では ~ サフィックス (例: idle~) で表示されます。この状態では、ノードをバックアップする EC2 インスタンスは存在しません。ただし、Slurm は引き続きノードにジョブを割り当てることができます。
- POWER_UP 状態に遷移したノードは、sinfo では # サフィックス (例: idle#) で表示されます。
- Slurm が POWER_SAVING 状態のノードにジョブを割り当てると、そのノードは自動的に POWER_UP 状態に移行します。それ以外の場合は、scontrol update nodename=*nodename* state=power_up コマンドを使用して手動でノードを POWER_UP 状態にすることができます。この段階では、ResumeProgram が起動され、EC2 インスタンスが起動し、POWER_UP ノードをバックアップするように構成されます。
- 現在使用可能なノードは、sinfo にサフィックス (例: idle) なしで表示されます。ノードのセットアップが完了し、クラスターに参加した後、ジョブの実行が可能になります。この段階で、ノードは適切に設定され、使用可能な状態になります。原則として、EC2 のインスタンス数は利用可能なノード数と同じにすることを推奨します。通常、静的ノードはクラスター作成後、常に利用可能です。
- POWER_DOWN 状態に遷移しているノードは、sinfo に % サフィックス (例 : idle%) で表示されます。動的ノードは scaledown_idletime の後、自動的に POWER_DOWN 状態になります。対照的に、ほとんどの場合、静的ノードの電源はオフになりません。ただし、scontrol update nodename=*nodename* state=powering_down コマンドを使用して手動でノードを POWER_DOWN 状態にすることができます。この状態では、ノードに関連するインスタンスは終了し、ノードは scaledown_idletime 後に将来使用するために POWER_SAVING 状態に戻されます。scaledown-idletime の設定は、SuspendTimeout の設定として Slurm のコンフィギュレーションに保存されます。
- オフラインのノードは、sinfo に * サフィックス (例: down*) で表示されます。Slurm コントローラーがノードにコンタクトできない場合、または静的ノードが無効化され、バッキングインスタンスが終了した場合、ノードはオフラインになります。

ここで、次の sinfo 例で示されるノードの状態を考えてみます。

```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
efa          up   infinite      4  idle~ efa-dy-c5n18xlarge-[1-4]
efa          up   infinite      1   idle efa-st-c5n18xlarge-1
gpu          up   infinite      1  idle% gpu-dy-g38xlarge-1
gpu          up   infinite      9  idle~ gpu-dy-g38xlarge-[2-10]
ondemand     up   infinite      2   mix# ondemand-dy-c52xlarge-[1-2]
ondemand     up   infinite     18  idle~ ondemand-dy-c52xlarge-[3-10],ondemand-dy-
t2xlarge-[1-10]
spot*        up   infinite     13  idle~ spot-dy-c5xlarge-[1-10],spot-dy-t2large-[1-3]
spot*        up   infinite      2   idle spot-st-t2large-[1-2]
```

spot-st-t2large-[1-2] ノードと efa-st-c5n18xlarge-1 ノードには、すでにバッキングインスタンスが設定されており、使用可能です。ondemand-dy-c52xlarge-[1-2] ノードは POWER_UP 状態であり、数分以内に利用可能になるはずです。gpu-dy-g38xlarge-1 ノードは POWER_DOWN 状態であり、scaledown_idletime (デフォルトは 120 秒) 後に POWER_SAVING 状態へ遷移します。

他のノードはすべて POWER_SAVING 状態で、EC2 インスタンスのバックアップはありません。

使用可能なノードの操作

使用可能なノードは EC2 インスタンスによってバックアップされます。デフォルトでは、ノード名を使用してインスタンスに直接 SSH 接続することができます (例: ssh efa-st-c5n18xlarge-1)。インスタンスのプライベート IP アドレスは、scontrol show nodes *nodename* コマンドを使用して NodeAddr フィールドを確認することで取得することができます。利用できないノードについては、NodeAddr フィールドは稼働中の EC2 インスタンスにポイントしてはいけません。ノード名と同じにする必要があります。

ジョブの状態および送信

通常、送信されたジョブはすぐにシステム内のノードに割り当てられ、すべてのノードが割り当てられている場合は保留状態になります。

ジョブに割り当てられたノードに POWER_SAVING 状態のノードが含まれる場合、ジョブは、CF または CONFIGURING 状態で開始されます。このとき、ジョブは POWER_SAVING 状態のノードが POWER_UP 状態に遷移し、利用可能になるのを待ちます。

ジョブに割り当てられたすべてのノードが利用可能になると、RUNNING (R) 状態になります。

デフォルトでは、すべてのジョブはデフォルトのキュー (Slurm ではパーティションと呼ばれます) に送信されます。これは、キュー名の後に * サフィックスが付くことで示されます。-p ジョブ送信オプションを使用してキューを選択することができます。

すべてのノードには、ジョブ送信コマンドで使用可能な次の機能が設定されています。

- インスタンスタイプ (例: c5.xlarge)
- ノードタイプ (dynamic または static のいずれか)。

scontrol show nodes *nodename* コマンドを使用して AvailableFeatures リストを確認することで、特定のノードで利用可能なすべての機能を確認することができます。

もう一つ考慮しなければならないのは、ジョブです。まず、クラスターの初期状態を考えてみましょう。sinfo コマンドを実行することで確認できます。

```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
efa          up   infinite      4  idle~ efa-dy-c5n18xlarge-[1-4]
efa          up   infinite      1   idle efa-st-c5n18xlarge-1
gpu          up   infinite     10  idle~ gpu-dy-g38xlarge-[1-10]
ondemand     up   infinite     20  idle~ ondemand-dy-c52xlarge-[1-10],ondemand-dy-
t2xlarge-[1-10]
spot*        up   infinite     13  idle~ spot-dy-c5xlarge-[1-10],spot-dy-t2large-[1-3]
spot*        up   infinite      2   idle spot-st-t2large-[1-2]
```

なお、spot はデフォルトのキューです。* というサフィックスで表示されます。

1 つの静的ノードへのジョブをデフォルトキュー (spot) に送信します。

```
$ sbatch --wrap "sleep 300" -N 1 -C static
```

ある動的ノードへのジョブを EFA キューに送信します。

```
$ sbatch --wrap "sleep 300" -p efa -C dynamic
```

c5.2xlarge ノード 8 台、t2.xlarge ノード 2 台のジョブを ondemand キューに送信します。

```
$ sbatch --wrap "sleep 300" -p ondemand -N 10 -C "[c5.2xlarge*8&t2.xlarge*2]"
```

ある GPU ノードへのジョブを gpu キューに送信します。

```
$ sbatch --wrap "sleep 300" -p gpu -G 1
```

ここで、squeue コマンドを使ったジョブの状態を考えてみます。

```
$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                12  ondemand     wrap   ubuntu CF       0:36     10 ondemand-dy-
c52xlarge-[1-8],ondemand-dy-t2xlarge-[1-2]
                13       gpu     wrap   ubuntu CF       0:05      1 gpu-dy-g38xlarge-1
                 7      spot     wrap   ubuntu  R       2:48      1 spot-st-t2large-1
                 8       efa     wrap   ubuntu  R       0:39      1 efa-dy-
c5n18xlarge-1
```

ジョブ 7 と 8 (spot と efa のキュー) はすでに実行中です (R)。ジョブ 12 と 13 はまだ設定中 (CF) で、おそらくインスタンスが利用できるようになるのを待っています。

```
# Nodes states corresponds to state of running jobs
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
efa          up   infinite      3  idle~ efa-dy-c5n18xlarge-[2-4]
efa          up   infinite      1    mix efa-dy-c5n18xlarge-1
efa          up   infinite      1   idle efa-st-c5n18xlarge-1
gpu          up   infinite      1   mix~ gpu-dy-g38xlarge-1
gpu          up   infinite      9  idle~ gpu-dy-g38xlarge-[2-10]
ondemand     up   infinite     10   mix# ondemand-dy-c52xlarge-[1-8],ondemand-dy-
t2xlarge-[1-2]
ondemand     up   infinite     10  idle~ ondemand-dy-c52xlarge-[9-10],ondemand-dy-
t2xlarge-[3-10]
spot*        up   infinite     13  idle~ spot-dy-c5xlarge-[1-10],spot-dy-t2large-[1-3]
spot*        up   infinite      1    mix spot-st-t2large-1
spot*        up   infinite      1   idle spot-st-t2large-2
```

ノードの状態および機能

ほとんどの場合、ノードの状態は、このトピックで前述したクラウドノードライフサイクルの特定のプロセス AWS ParallelCluster に従って によって完全に管理されます。

ただし、 は、 DOWNおよび DRAINEDの状態の異常なノードと、異常なバッキングインスタンスを持つノード AWS ParallelCluster も置き換えまたは終了します。詳細については、「clustermgtd」を参照してください。

パーティションの状態

AWS ParallelCluster は、次のパーティション状態をサポートしています。Slurm パーティションは AWS ParallelClusterのキューです。

- UP: パーティションがアクティブ状態であることを示します。これは、パーティションのデフォルトの状態です。この状態では、パーティション内のすべてのノードがアクティブで、使用可能な状態です。
- INACTIVE: パーティションが非アクティブ状態であることを示す。この状態では、非アクティブなパーティションのノードをバックアップしているインスタンスはすべて終了します。非アクティブなパーティションのノードには、新しいインスタンスは起動しません。

pcluster の起動と停止

pcluster stop を実行すると、すべてのパーティションが INACTIVE状態になり、 AWS ParallelCluster プロセスはパーティションを INACTIVE状態のままにします。

pcluster start を実行すると、最初はすべてのパーティションが UP の状態になります。ただし、 AWS ParallelCluster プロセスはパーティションを UP状態に保持しません。パーティションの状態を手動で変更する必要があります。数分後、すべての静的ノードが使用可能になります。パーティションを UP に設定しても、動的容量は向上しません。initial_count が max_count より大きい場合、パーティションの状態が UP 状態に変化したときに initial_count が満たされない可能性があります。

pcluster start と pcluster stop が動作している場合、pcluster status コマンドを実行して ComputeFleetStatus を確認することで、クラスターの状態を確認することができます。考えられる状態を以下に示します。

- STOP_REQUESTED: pcluster stop リクエストがクラスターに送信されます。
- STOPPING: pcluster プロセスは現在、クラスターを停止しています。
- STOPPED: pcluster プロセスは停止処理を終了し、すべてのパーティションは INACTIVE 状態になり、すべてのコンピューティングインスタンスは終了しました。
- START_REQUESTED: pcluster start リクエストがクラスターに送信されます。
- STARTING: pcluster プロセスは現在、クラスターを起動中です
- RUNNING: pcluster プロセスは起動処理を終了し、すべてのパーティションが UP 状態になり、数分後に静的ノードが利用可能になります。

キューの手動制御

場合によっては、クラスター内のノードやキュー (Slurm ではパーティションと呼ばれます) を手動で制御したいことがあります。次の共通の手順でクラスター内のノードを管理することができます。

- POWER_SAVING 状態の動的ノードをパワーアップさせます。`scontrol update nodename=nodename state=power_up` コマンドを実行するか、特定のノード数を要求するプレースホルダ `sleep 1` ジョブを送信し、Slurm に依存して必要な数のノードをパワーアップさせます。
- より前の動的ノードの電源を切る[scaledown_idletime](): `scontrol update nodename=nodename state=down` コマンドDOWNを使用して動的ノードを に設定します。は、ダウンした動的ノード AWS ParallelCluster を自動的に終了およびリセットします。一般に、`scontrol update nodename=nodename state=power_down` コマンドを使用してノードを直接 POWER_DOWN に設定することは推奨しません。これは、 AWS ParallelCluster が自動的にパワーダウン処理を行うためです。手動で行う必要がありません。そのため、可能な限りノードを DOWN に設定することをお勧めします。
- キュー (パーティション) を無効にする、または特定のパーティションの静止ノードをすべて停止する: `scontrol update partition=queue name state=inactive` コマンドで特定のキューを INACTIVE に設定します。この操作により、パーティション内のすべてのインスタンスバッキングノードが終了します。
- キュー (パーティション) を有効にする: `scontrol update partition=queue name state=up` コマンドで INACTIVE に特定のキューを設定します。

スケーリング動作および調整

ここでは、通常のスケーリングワークフローの例を示します。

- スケジューラは、2 つのノードを必要とするジョブを受け取ります。
- スケジューラは 2 つのノードを POWER_UP 状態に遷移させ、ノード名 (例: `queue1-dy-c5xlarge-[1-2]`) で ResumeProgram を呼び出す。
- ResumeProgram は EC2 インスタンスを 2 台起動し、`queue1-dy-c5xlarge-[1-2]` のプライベート IP アドレスとホストネームを割り当て、ResumeTimeout (デフォルトの期間は 60 分 (1 時間)) を待ってからノードのリセットを行います。
- インスタンスが設定され、クラスターに参加します。インスタンス上でジョブの実行を開始します。
- ジョブが完了しました。

- 設定された SuspendTime が経過した後 (scaledown_idletime に設定されています)、インスタンスはスケジューラによって POWER_SAVING 状態になります。スケジューラは queue1-dy-c5xlarge-[1-2] を POWER_DOWN 状態にし、ノード名で SuspendProgram を呼び出します。
- SuspendProgram は 2 つのノードに対して呼び出されます。ノードは、例えば SuspendTimeout のために idle% を維持することで、POWER_DOWN 状態を維持します (デフォルトの期間は 120 秒 (2 分))。clustermgtd は、ノードがパワーダウンしていることを検出した後、バッキングインスタンスを終了します。その後、queue1-dy-c5xlarge-[1-2] をアイドル状態に設定し、プライベート IP アドレスとホスト名をリセットして、将来のジョブのために再びパワーアップできるようにします。

ここで、処理がうまくいかず、何らかの理由で特定のノードのインスタンスが起動できない場合、次のようなことが起こります。

- スケジューラは、2 つのノードを必要とするジョブを受け取ります。
- スケジューラは 2 つのクラウドでのバーストノードを POWER_UP 状態にし、ノード名で ResumeProgram を呼び出します (例: queue1-dy-c5xlarge-[1-2])。
- ResumeProgram は EC2 インスタンスを 1 台だけ起動し、queue1-dy-c5xlarge-1 の設定を行いましたが、queue1-dy-c5xlarge-2 用のインスタンスの起動に失敗しました。
- queue1-dy-c5xlarge-1 は影響を受けず、POWER_UP の状態になった後にオンラインになります。
- queue1-dy-c5xlarge-2 は POWER_DOWN 状態になり、Slurm がノード障害を検出したため、自動的にジョブが再クエリされます。
- queue1-dy-c5xlarge-2 は SuspendTimeout の後に利用可能になります (デフォルトは 120 秒 (2 分))。その間、ジョブは再キューされ、別のノードで実行を開始することができます。
- 上記のプロセスは、使用可能なノードで障害が発生せずにジョブを実行できるようになるまで繰り返されます。

必要に応じて調整可能な 2 つのタイミングパラメータがあります。

- ResumeTimeout (デフォルトは 60 分 (1 時間)): ResumeTimeout は、Slurm がノードをダウン状態にするまでの待ち時間を制御します。
  - インストール前後の処理に時間がかかる場合は、これを拡張するのも有効です。
  - これは、問題が発生した場合に がノードを交換またはリセットするまでに AWS ParallelCluster 待機する最大時間でもあります。コンピューティングノードは、起動時やセットアップ時に何ら

かのエラーが発生した場合、自己終了します。次に、 AWS ParallelCluster プロセスは、インスタンスが終了したことを確認すると、ノードの交換を行います。

- `SuspendTimeout` (デフォルトは 120 秒 (2 分))。`SuspendTimeout` は、ノードがシステムに戻され、再び使用できるようになるまでの時間を制御します。
  - `SuspendTimeout` が短いと、ノードのリセットが早くなり、Slurm はより頻繁にインスタンスの起動を試みることができます。
  - `SuspendTimeout` が長いと、故障したノードのリセットが遅くなります。その間、Slurm は他のノードの利用を試みます。`SuspendTimeout` が数分以上であれば、Slurm はシステム内の全ノードの循環を試みます。1,000 ノード以上の大規模システムでは、失敗したジョブを頻繁に再キューイングすることによって Slurm のストレスを軽減するために、より長い `SuspendTimeout` が有効であると考えられます。
  - `SuspendTimeout` は、ノードのバッキングインスタンスの終了を AWS ParallelCluster 待機した時間を参照しないことに注意してください。`power down` ノードのバックアップインスタンスは即座に終了します。通常、終了プロセスは数分で完了します。ただし、この間、ノードはパワーダウン状態にあり、スケジューラで利用することはできません。

新しいアーキテクチャのログ

次のリストは、マルチキューアーキテクチャのキーログを含んでいます。Amazon CloudWatch Logs で使用されるログストリーム名は、*`{hostname}`*`.`*`{instance_id}`*`.`*`{logIdentifier}`* という形式になっており、*`logIdentifier`* が ログ名の後に続きます。詳細については、「Amazon CloudWatch Logs との統合」を参照してください。

- `ResumeProgram`:

  `/var/log/parallelcluster/slurm_resume.log` (`slurm_resume`)
- `SuspendProgram`:

  `/var/log/parallelcluster/slurm_suspend.log` (`slurm_suspend`)
- `clustermgtd`:

  `/var/log/parallelcluster/clustermgtd.log` (`clustermgtd`)
- `computemgtd`:

  `/var/log/parallelcluster/computemgtd.log` (`computemgtd`)
- `slurmctld`:

`/var/log/slurmctld.log` (`slurmctld`)

- `slurmd`:

  `/var/log/slurmd.log` (`slurmd`)

よくある問題とデバッグの方法:

起動、パワーアップ、またはクラスターへの参加に失敗したノード:

- 動的ノード:
  - `ResumeProgram` ログを確認し、ノードで `ResumeProgram` が呼び出されたことがあるかどうかを確認します。そうでない場合は、`slurmctld` ログを確認し、Slurm がノードで `ResumeProgram` に電話をかけようとしたことがあるかどうかを判断します。`ResumeProgram` のパーミッションが正しくない場合、サイレントで失敗することがあります。
  - `ResumeProgram` が呼び出された場合、そのノードに対してインスタンスが起動されたかどうかを確認します。インスタンスを起動できない場合は、インスタンスの起動に失敗した理由を示す明確なエラーメッセージが表示されます。
  - インスタンスが起動した場合、ブートストラッププロセスの実行時に何らかの問題が発生した可能性があります。`ResumeProgram` ログから対応するプライベート IP アドレスとインスタンス ID を探し、CloudWatch Logs で特定のインスタンスの対応するブートストラップのログを見ます。
- 静的ノード:
  - `clustermgtd` ログをチェックして、ノードに対してインスタンスが起動されたかどうかを確認します。そうでない場合は、インスタンスの起動に失敗した原因について、明確なエラーが表示されるはずです。
  - インスタンスを起動していた場合、ブートストラップ処理中に何らかの問題が発生しています。`clustermgtd` ログから対応するプライベート IP とインスタンス ID を探し、CloudWatch Logs で特定のインスタンスの対応するブートストラップのログを見ます。

ノードが予期せず置換または終了したノードの障害

- ノードが予期せず置換または終了したノード
  - 通常、`clustermgtd` はすべてのノードのメンテナンスアクションを処理します。`clustermgtd` がノードを置換または終了させたかどうかを確認するには、`clustermgtd` のログを確認します。

  - clustermgtd がノードを置換または終了させた場合、その理由を示すメッセージが表示されます。スケジューラ関連 (ノードが DOWN だった場合など) の場合は、slurmctld ログで詳細を確認します。EC2 が原因の場合は、ツールを使用し、そのインスタンスのステータスやログを確認します。例えば、インスタンスにスケジュールされたイベントがあったかどうか、EC2 ヘルスステータスのチェックに失敗したかどうかを確認できます。
  - clustermgtd がノードを終了させなかった場合、computemgtd がノードを終了させたか、EC2 がスポットインスタンスを再要求するためにインスタンスを終了させたかどうかを確認します。
- ノードの障害
  - 通常、ノードに障害が発生した場合、ジョブは自動的に再キューされます。slurmctld ログを見て、ジョブやノードがなぜ失敗したかを確認し、そこから状況を分析します。

インスタンスの置換やターミネーション時の障害、ノードのパワーダウン時の障害

- 一般に、clustermgtd は期待されるすべてのインスタンス終了アクションを処理します。clustermgtd ログを見て、ノードの置換または終了に失敗した理由を確認する。
- scaledown_idletime に失敗した動的ノードの場合、SuspendProgram のログから、特定のノードを引数にした slurmctld によるプログラムがあるかどうかを確認します。SuspendProgram は実際に特定のアクションを実行するわけではありません。呼び出されたときだけログに記録されます。すべてのインスタンスの終了と NodeAddr リセットは clustermgtd により完了します。Slurm は IDLE の後、SuspendTimeout にノードを入れます。

その他の問題

- AWS ParallelCluster はジョブの割り当てやスケーリングの決定を行いません。Slurm の指示に従い、リソースを起動、終了、維持を試みるだけです。

  ジョブの割り当て、ノードの割り当て、スケーリングの決定に関する問題については、slurmctld ログを見てエラーを確認します。

Torque Resource Manager (`torque`)

> **Note**
>
> バージョン 2.11.5 以降、 AWS ParallelCluster は SGEまたは スTorqueケジューラの使用をサポートしていません。2.11.4 までのバージョンで引き続き使用できますが、 AWS サービスおよび AWS サポートチームによる今後の更新やトラブルシューティングのサポートを受けることはできません。

AWS ParallelCluster バージョン 2.11.4 以前では Torque Resource Manager 6.1.2 を使用しています。Torque Resource Manager 6.1.2 の詳細については、「http://docs.adaptivecomputing.com/torque/6-1-2/releaseNotes/torquerelnote.htm」を参照してください。ドキュメントについては、「http://docs.adaptivecomputing.com/torque/6-1-2/adminGuide/torque.htm」を参照してください。出典コードについては、「https://github.com/adaptivecomputing/torque/tree/6.1.2」を参照してください。

AWS ParallelCluster バージョン 2.4.0 以前では Torque Resource Manager 6.0.2 を使用しています。リリースノートについては、「http://docs.adaptivecomputing.com/torque/6-0-2/releaseNotes/torqueReleaseNotes6.0.2.pdf」を参照してください。ドキュメントについては、「http://docs.adaptivecomputing.com/torque/6-0-2/adminGuide/help.htm」を参照してください。出典コードについては、「https://github.com/adaptivecomputing/torque/tree/6.0.2」を参照してください。

AWS Batch (`awsbatch`)

詳細については AWS Batch、「」を参照してくださいAWS Batch。ドキュメントについては、「AWS Batch IAM ユーザーガイド」を参照してください。

AWS ParallelCluster の CLI コマンド AWS Batch

スケ`awsbatch`ジューラを使用すると、 の AWS ParallelCluster CLI コマンド AWS Batch が AWS ParallelCluster ヘッドノードに自動的にインストールされます。CLI は AWS Batch API オペレーションを使用し、次のオペレーションを許可します。

- ジョブの送信と管理
- ジョブ、キュー、ホストのモニタリング
- 従来のスケジューラコマンドのミラーリング

> **Important**
>
> AWS ParallelCluster は の GPU ジョブをサポートしていません AWS Batch。詳細については、「GPU jobs」を参照してください。

トピック

- awsbsub
- awsbstat
- awsbout
- awsbkill
- awsbqueues
- awsbhosts

awsbsub

ジョブをクラスターのジョブキューに送信します。

```
awsbsub [-h] [-jn JOB_NAME] [-c CLUSTER] [-cf] [-w WORKING_DIR]
        [-pw PARENT_WORKING_DIR] [-if INPUT_FILE] [-p VCPUS] [-m MEMORY]
        [-e ENV] [-eb ENV_DENYLIST] [-r RETRY_ATTEMPTS] [-t TIMEOUT]
        [-n NODES] [-a ARRAY_SIZE] [-d DEPENDS_ON]
        [command] [arguments [arguments ...]]
```

> **Important**
>
> AWS ParallelCluster は の GPU ジョブをサポートしていません AWS Batch。詳細については、「GPU jobs」を参照してください。

位置引数

command

ジョブを送信するか (指定したコマンドがコンピューティングインスタンスで使用可能である必要があります)、転送するファイル名を指定します。「`--command-file`」も参照してください。

arguments

(オプション) コマンドまたはコマンドファイルの引数を指定します。

名前付き引数

-jn *JOB_NAME*, --job-name *JOB_NAME*

ジョブの名前を指定します。最初の文字はアルファベットまたは数字でなければなりません。ジョブ名には、アルファベット (大文字、小文字)、数字、ハイフン、アンダースコアを含めることができ、最大 128 文字まで使用可能です。

-c *CLUSTER*, --cluster *CLUSTER*

使用するクラスターを指定します。

-cf, --command-file

コマンドがコンピューティングインスタンスに転送されるファイルであることを示します。

デフォルト: False

-w *WORKING_DIR*, --working-dir *WORKING_DIR*

ジョブの作業ディレクトリとして使用するフォルダを指定します。作業ディレクトリが指定されない場合、ジョブは、ユーザーのホームディレクトリの `job-<AWS_BATCH_JOB_ID>` サブフォルダで実行されます。このパラメータ、または `--parent-working-dir` パラメータを使用できます。

-pw *PARENT_WORKING_DIR*, --parent-working-dir *PARENT_WORKING_DIR*

ジョブの作業ディレクトリの親フォルダを指定します。親の作業ディレクトリが指定されない場合、デフォルトはユーザーのホームディレクトリに設定されます。`job-<AWS_BATCH_JOB_ID>` という名前のサブフォルダが親の作業ディレクトリに作成されます。このパラメータ、または `--working-dir` パラメータを使用できます。

-if *INPUT_FILE*, --input-file *INPUT_FILE*

コンピューティングインスタンスに転送するファイル (ジョブの作業ディレクトリ内) を指定します。複数の入力ファイルのパラメータを指定できます。

-p *VCPUS*, --vcpus *VCPUS*

コンテナ用に予約する vCPU の数を指定します。–nodes を指定すると、ノードごとの vCPU の数が識別されます。

デフォルト: 1

-m *MEMORY*, --memory *MEMORY*

ジョブに送信するメモリのハード制限 (MiB 単位) を指定します。ここで指定したメモリ制限を超えようとすると、ジョブは強制終了されます。

デフォルト: 128

-e *ENV*, --env *ENV*

ジョブ環境にエクスポートする環境変数名のカンマ区切りリストを指定します。すべての環境変数をエクスポートするには、「all」を指定します。「all」の環境変数のリストには、`-env-blacklist` パラメータに一覧表示されている変数や、プレフィックスが `PCLUSTER_*` または `AWS_*` の変数は含まれません。

-eb *ENV_DENYLIST*, --env-blacklist *ENV_DENYLIST*

ジョブ環境にエクスポートしない環境変数名のカンマ区切りリストを指定します。`HOME`、`PWD`、`USER`、`PATH`、`LD_LIBRARY_PATH`、`TERM`、および `TERMCAP` はデフォルトでエクスポートされません。

-r *RETRY_ATTEMPTS*, --retry-attempts *RETRY_ATTEMPTS*

ジョブを `RUNNABLE` ステータスに移行する回数を指定します。1〜10 回の試行を指定できます。試行回数の設定値が 1 より大きい場合にジョブが失敗すると、`RUNNABLE` ステータスに変わるまで、指定された回数分、再試行します。

デフォルト: 1

-t *TIMEOUT*, --timeout *TIMEOUT*

ジョブが終了していない場合に がジョブ AWS Batch を終了するまでの時間 (ジョブ試行の`startedAt`タイムスタンプから測定) を秒単位で指定します。タイムアウト値は 60 秒以上に指定する必要があります。

-n *NODES*, --nodes *NODES*

ジョブ用に予約するノード数を指定します。マルチノード並列送信が有効になるように、このパラメータに値を指定します。

> **Note**
>
> `cluster_type` パラメータが `spot` に設定されている場合、マルチノード並列ジョブはサポートされません。

-a *ARRAY_SIZE*, --array-size *ARRAY_SIZE*

配列のサイズを示します。2 から 10,000 までの値を指定できます。ジョブの配列プロパティを指定した場合は、配列ジョブになります。

-d *DEPENDS_ON*, --depends-on *DEPENDS_ON*

ジョブの依存関係のセミコロン区切りリストを指定します。ジョブは最大 20 個のジョブに依存します。配列ジョブのジョブ ID を指定せずに、SEQUENTIAL タイプの依存関係を指定できます。シーケンシャルな依存関係では、各子配列ジョブがインデックス 0 から開始して順番に完了します。また、配列ジョブのジョブ ID を使用して N_TO_N タイプの依存関係を指定することもできます。N_TO_N の依存関係では、このジョブの各インデックスの子は各依存関係の対応するインデックスの子が完了するまで待機してから開始されます。このパラメータの構文は、「jobId=*<string>*,type=*<string>*;...」です。

awsbstat

クラスターのジョブキューに送信されたジョブを表示します。

```
awsbstat [-h] [-c CLUSTER] [-s STATUS] [-e] [-d] [job_ids [job_ids ...]]
```

位置引数

job_ids

出力に表示するジョブ ID のスペース区切りリストを指定します。ジョブがジョブ配列の場合は、すべての子ジョブが表示されます。単一のジョブがリクエストされた場合は、詳細バージョンで表示されます。

名前付き引数

-c *CLUSTER*, --cluster *CLUSTER*

使用するクラスターを示します。

-s *STATUS*, --status *STATUS*

含めるジョブステータスのカンマ区切りリストを指定します。
デフォルトのジョブのステータスは「active」です。有効な値は、SUBMITTED、PENDING、RUNNABLE、STARTING、RUNNING、SUCCEEDED、FAILED、ALL です。

デフォルト値は、SUBMITTED、PENDING、RUNNABLE、STARTING、RUNNING です。

-e, --expand-children

子を含むジョブを拡張します (配列とマルチノードの並列ジョブのいずれも)。

デフォルト: False

-d, --details

ジョブの詳細を表示します。

デフォルト: False

awsbout

指定されたジョブの出力を表示します。

```
awsbout [ - h ] [ - c CLUSTER ] [ - hd HEAD ] [ - t TAIL ] [ - s ] [ - sp STREAM_PERIOD
 ] job_id
```

位置引数

job_id

ジョブ ID を指定します。

名前付き引数

-c *CLUSTER*, --cluster *CLUSTER*

使用するクラスターを示します。

-hd *HEAD*, --head *HEAD*

ジョブ出力の最初の *HEAD* 行を取得します。

-t *TAIL*, --tail *TAIL*

ジョブ出力の最後の <tail> 行を取得します。

-s, --stream

ジョブ出力を取得してから、追加の出力が生成されるのを待ちます。ジョブ出力の最新の <tail> 行から開始するには、この引数に –tail を指定します。

デフォルト: False

-sp *STREAM_PERIOD*, --stream-period *STREAM_PERIOD*

ストリーミング期間を設定します。

デフォルト: 5

awsbkill

クラスターに送信されたジョブをキャンセルし、終了します。

```
awsbkill [ - h ] [ - c CLUSTER ] [ - r REASON ] job_ids [ job_ids ... ]
```

位置引数

job_ids

キャンセルまたは終了するジョブ ID のスペース区切りリストを指定します。

名前付き引数

-c *CLUSTER*, --cluster *CLUSTER*

使用するクラスターの名前を示します。

-r *REASON*, --reason *REASON*

キャンセル理由と合わせて、ジョブにアタッチするメッセージを示します。

デフォルト:「Terminated by the user」

awsbqueues

クラスターに関連付けられているジョブキューを表示します。

```
awsbqueues [ - h ] [ - c CLUSTER ] [ - d ] [ job_queues [ job_queues ... ]]
```

位置引数

job_queues

表示するキュー名のスペース区切りリストを指定します。単一のキューがリクエストされた場合は、詳細バージョンで表示されます。

名前付き引数

-c *CLUSTER*, --cluster *CLUSTER*

使用するクラスターの名前を指定します。

-d, --details

キューの詳細を表示するかどうかを示します。

デフォルト: False

awsbhosts

クラスターのコンピューティング環境に属するホストを表示します。

```
awsbhosts [ - h ] [ - c CLUSTER ] [ - d ] [ instance_ids [ instance_ids ... ]]
```

位置引数

instance_ids

インスタンス ID のスペース区切りリストを指定します。単一のインスタンスがリクエストされた場合は、詳細バージョンで表示されます。

名前付き引数

-c *CLUSTER*, --cluster *CLUSTER*

使用するクラスターの名前を指定します。

-d, --details

ホストの詳細を表示するかどうかを示します。

デフォルト: False

AWS ParallelCluster リソースとタグ付け

AWS ParallelCluster を使用すると、 AWS ParallelCluster タグを作成してリソースを追跡および管理できます。クラスター設定ファイルの `tags`セクションで、 AWS CloudFormation 作成してすべてのクラスターリソースに伝達するタグを定義します。また、 AWS ParallelCluster が自動的に生成するタグを使用して、 リソースを追跡および管理することもできます。

クラスターを作成すると、クラスターとそのリソースには、このセクションで定義されている AWS ParallelCluster および AWS システムタグがタグ付けされます。

AWS ParallelCluster は、クラスターインスタンス、ボリューム、リソースにタグを適用します。クラスタースタックを識別するために、 はクラスターインスタンスに AWS システムタグ AWS CloudFormation を適用します。クラスターの EC2 起動テンプレートを識別するために、EC2 はインスタンスにシステムタグを適用します。これらのタグを使用して、 AWS ParallelCluster リソースを表示および管理できます。

AWS システムタグを変更することはできません。 AWS ParallelCluster 機能への影響を避けるため、 AWS ParallelCluster タグを変更しないでください。

以下は、 AWS ParallelCluster リソースの AWS システムタグの例です。これらは変更できません。

```
"aws:cloudformation:stack-name"="parallelcluster-clustername-
MasterServerSubstack-ABCD1234EFGH"
```

以下は、リソースに適用される AWS ParallelCluster タグの例です。これらは修正しないでください。

```
"aws-parallelcluster-node-type"="Master"
```

```
"Name"="Master"
```

```
"Version"="2.11.9"
```

これらのタグは、 AWS Management Consoleの EC2 セクションで表示できます。

タグの表示

1. https://console.aws.amazon.com/ec2/ で EC2 コンソールをナビゲートします。
2. クラスタータグをすべて表示するには、ナビゲーションペインで [タグ] を選択します。

3. クラスタータグをインスタンス別に表示するには、ナビゲーションペインで [インスタンス] を選択します。
4. クラスターインスタンスを選択します。
5. インスタンス詳細の [タグを管理] タブを選択し、タグを表示します。
6. インスタンス詳細の [ストレージ] タブを選択します。
7. [ボリューム ID] を選択します。
8. [ボリューム] で、ボリュームを選択します。
9. ボリューム詳細の [タグ] タブを選択し、タグを表示します。

AWS ParallelCluster ヘッドノードインスタンスタグ

キー	タグ値
ClusterName	*clustername*
Name	Master
Application	parallelcluster- *clustername*
aws:ec2launchtemplate:id	*lt-1234567890abcdef0*
aws:ec2launchtemplate:version	*1*
aws-parallelcluster-node-type	Master
aws:cloudformation:stack-name	parallelcluster- *clustername* -MasterServerSubstack- *ABCD1234EFGH*
aws:cloudformation:logical-id	MasterServer
aws:cloudformation:stack-id	arn:aws:cloudformation: *region-id* :*ACCOUNTID* :stack/parallelcluster- *clustername* -MasterServerSubstack- *ABCD1234EFGH* /*1234abcd-12ab-12ab-12ab-1234567890abcdef0*

キー	タグ値
Version	*2.11.9*

AWS ParallelCluster ヘッドノードルートボリュームタグ

タグキー	タグ値
ClusterName	*clustername*
Application	parallelcluster- *clustername*
aws-parallelcluster-node-type	Master

AWS ParallelCluster コンピューティングノードインスタンスタグ

キー	タグ値
ClusterName	*clustername*
aws-parallelcluster-node-type	Compute
aws:ec2launchtemplate:id	*lt-1234567890abcdef0*
aws:ec2launchtemplate:version	*1*
QueueName	*queue-name*
Version	*2.11.9*

AWS ParallelCluster コンピューティングノードのルートボリュームタグ

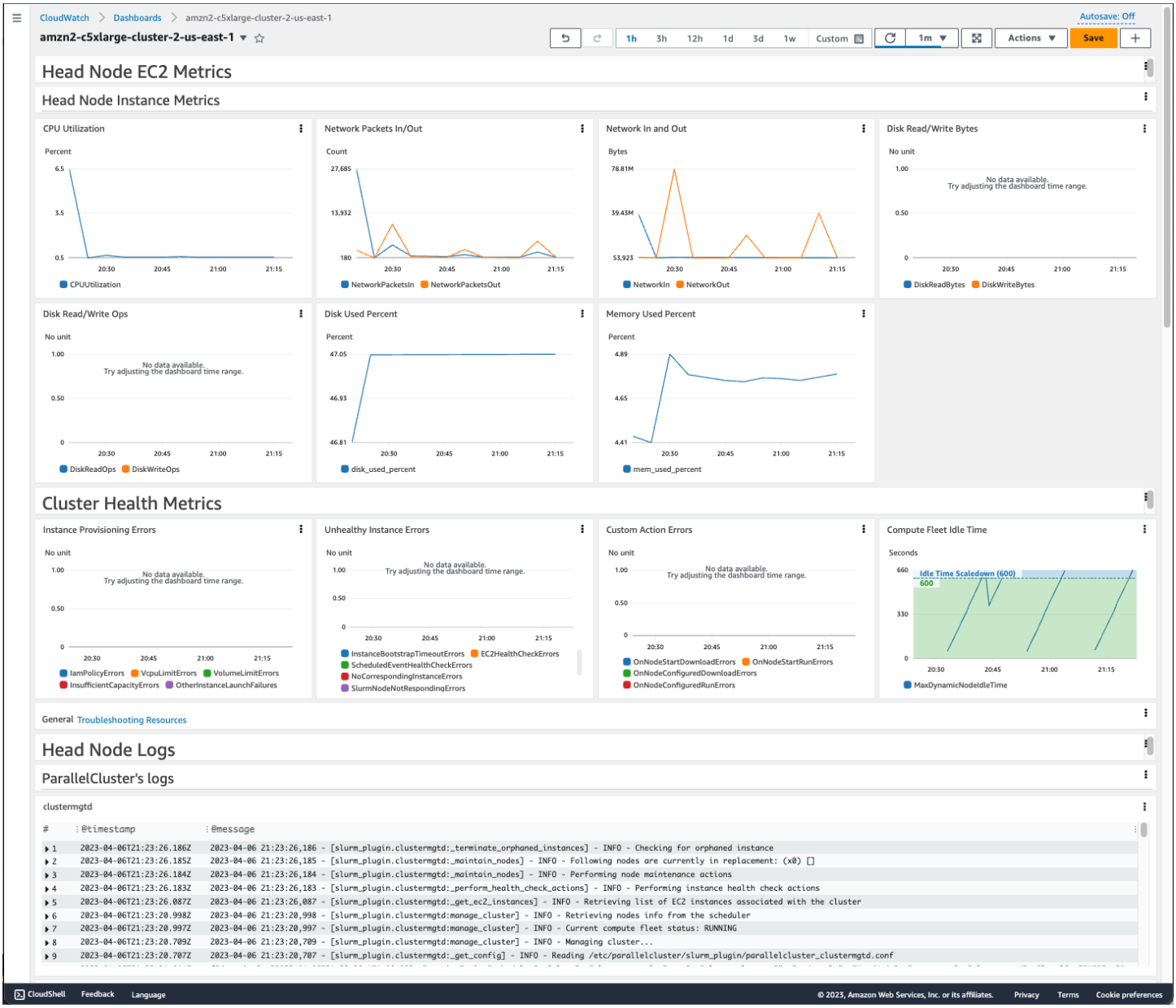
タグキー	タグ値
ClusterName	*clustername*
Application	parallelcluster- *clustername*
aws-parallelcluster-node-type	Compute

タグキー	タグ値
QueueName	*queue-name*
Version	*2.11.9*

Amazon CloudWatch ダッシュボード

AWS ParallelCluster バージョン 2.10.0 以降では、クラスターの作成時に Amazon CloudWatch ダッシュボードが作成されます。これにより、クラスター内のノードのモニタリングや、Amazon CloudWatch Logs に保存されたログの表示が容易になります。ダッシュボードの名前は `parallelcluster-`*`ClusterName`*`-`*`Region` です。*`ClusterName` はクラスターの名前、*`#####`*はクラスターの AWS リージョン です。ダッシュボードはコンソールから、または `https://console.aws.amazon.com/cloudwatch/home?region=`*`Region`*`#dashboards:name=parallelcluster-`*`ClusterName` を開くことでアクセスできます。

次の図は、クラスター用の CloudWatch ダッシュボード例を示しています。



ダッシュボードの最初のセクションには、ヘッドノード EC2 メトリックスのグラフが表示されます。クラスターに共有ストレージがある場合、次のセクションには共有ストレージメトリックスが表示されます。最後のセクションには、ParallelCluster のログ、スケジューラのログ、NICE DCV 統合ログ、およびシステムのログごとにグループ化されたヘッドノードログが一覧表示されます。

CloudWatch メトリクスの操作方法の詳細については、「Amazon CloudWatch User Guide」(Amazon CloudWatch ユーザーガイド) の「Using Amazon CloudWatch dashboards」(Amazon CloudWatch ダッシュボードの使用) を参照してください。

Amazon CloudWatch ダッシュボードを作成しない場合は、以下のステップを完了する必要があります。まず、設定ファイルに [dashboard] セクションを追加し、そのセクションの名前を

[cluster] セクションの dashboard_settings 設定の値として追加します。[dashboard] セクションで、enable = false を設定します。

例えば、[dashboard] セクションの名前をmyDashboard、[cluster] セクションの名前を myCluster とすると、以下のような変更になります。

```
[cluster MyCluster]
dashboard_settings = MyDashboard
...

[dashboard MyDashboard]
enable = false
```

Amazon CloudWatch Logs との統合

AWS ParallelCluster バージョン 2.6.0 以降、一般的なログはデフォルトで CloudWatch Logs に保存されます。CloudWatch Logs の詳細については、「Amazon CloudWatch Logs User Guide」(Amazon CloudWatch Logs ユーザーガイド) を参照してください。CloudWatch Logs の統合を設定するには、[cw_log] セクションと cw_log_settings 設定を参照してください。

ロググループは、クラスターごとに /aws/parallelcluster/*cluster-name* という名前 (例: /aws/parallelcluster/testCluster) で作成されます。ノード別の各ログ (またはパスに * が含まれている場合はログのセット) には、*{hostname}*.*{instance_id}*.*{logIdentifier}* という名前のログストリームが存在します。(例: ip-172-31-10-46.i-02587cf29cc3048f3.nodewatcher) ログデータは、すべてのクラスターインスタンス上で root として実行される CloudWatch エージェントによって CloudWatch に送信されます。

AWS ParallelCluster バージョン 2.10.0 以降では、クラスターの作成時に Amazon CloudWatch ダッシュボードが作成されます。このダッシュボードでは、CloudWatch Logs に保存されているログを簡単に確認することができます。詳細については、「Amazon CloudWatch ダッシュボード」を参照してください。

このリストには、プラットフォーム、スケジューラー、ノードで使用できるログストリームの *logIdentifier* とパスが含まれています。

プラットフォーム、スケジューラー、ノードで使用できるログストリーム

[Platforms] (プラットフォーム)	スケジューラ	ノード	ログストリーム
Amazon CentOS Ubuntu	awsbatch slurm	HeadNode	dcv-authenticator: `/var/log/parallelcluster/pcluster_dcv_authenticator.log` dcv-ext-authenticator: `/var/log/parallelcluster/pcluster_dcv_connect.log` dcv-agent: `/var/log/dcv/agent.*.log` dcv-xsession: `/var/log/dcv/dcv-xsession.*.log` dcv-server: `/var/log/dcv/server.log` dcv-session-launcher: `/var/log/dcv/sessionlauncher.log` Xdcv: `/var/log/dcv/Xdcv.*.log` cfn-init: `/var/log/cfn-init.log` chef-client: `/var/log/chef-client.log`
Amazon CentOS Ubuntu	awsbatch slurm	ComputeFleet HeadNode	cloud-init: `/var/log/cloud-init.log` supervisord: `/var/log/supervisord.log`
Amazon CentOS Ubuntu	slurm	ComputeFleet	cloud-init-output: `/var/log/cloud-init-output.log` computemgtd: `/var/log/parallelcluster/computemgtd` slurmd: `/var/log/slurmd.log`
Amazon	slurm	HeadNode	clustermgtd: `/var/log/parallelcluster/clustermgtd`

[Platforms] (プラットフォーム)	スケジューラ	ノード	ログストリーム
CentOS Ubuntu			slurm_resume: `/var/log/parallelcluster/slurm_resume.log` slurm_suspend: `/var/log/parallelcluster/slurm_suspend.log` slurmctld: `/var/log/slurmctld.log`
Amazon CentOS	awsbatc slurm	Computeet HeadNc	system-messages: `/var/log/messages`
Ubuntu	awsbatc slurm	Computeet HeadNc	syslog: `/var/log/syslog`

を使用するクラスター内のジョブは、`RUNNING`、、`SUCCEEDED`または `FAILED`状態に達したジョブの出力を CloudWatch Logs に AWS Batch 保存します。ロググループは `/aws/batch/job`、ログストリーム名形式は *`jobDefinitionName`*`/default/`*`ecs_task_id`* です。デフォルトでは、このログは永久に失効しませんが、保持期間を変更することもできます。詳細については、「Amazon CloudWatch Logs User Guide」(Amazon CloudWatch Logs ユーザーガイド) の「Change log data retention in CloudWatch Logs」(CloudWatch ログでのログデータ保管期間の変更) を参照してください。

> **Note**
>
> `chef-client`、、`cloud-init-outputclustermgtd`、`computemgtd`、`slurm_resume`、 が AWS ParallelCluster バージョン 2.9.0 で追加`slurm_suspend`されました。 AWS ParallelCluster バージョン 2.6.0

では、`/var/log/cfn-init-cmd.log` (`cfn-init-cmd`) と`/var/log/cfn-wire.log` (`cfn-wire`) も CloudWatch Logs に保存されました。

Elastic Fabric Adapter

Elastic Fabric Adapter (EFA) は、同じサブネット上の他のインスタンスとの低レイテンシーのネットワーク通信用の OS バイパス機能を備えたネットワークデバイスです。EFA は Libfabric を使用して公開され、メッセージングパッシングインターフェイス (MPI) を使用するアプリケーションで使用できます。

で EFA を使用するには AWS ParallelCluster、 [queue]セクション`enable_efa = true`に 行を追加します。

EFA をサポートする EC2 インスタンスのリストを表示するには、「Linux インスタンス用 Amazon EC2 ユーザーガイド」の「サポートされるインスタンスタイプ」を参照してください。

`enable_efa` 設定の詳細については、「[queue] section」の「`enable_efa`」を参照してください。

クラスタープレイスメントグループは、インスタンス間のレイテンシーを最小限に抑えるために使用する必要があります。詳細については、「`placement`」および「`placement_group`」を参照してください。

詳細については、「Amazon EC2 ユーザーガイド」の「Elastic Fabric Adapter」と、「AWS Open Source Blog」の「Scale HPC workloads with elastic fabric adapter and AWS ParallelCluster」を参照してください。

> **Note**
>
> デフォルトでは、Ubuntu ディストリビューションにより ptrace (プロセストレース) 保護が有効になります。 AWS ParallelCluster 2,6.0 以降、Libfabric が適切に機能するように、ptrace 保護が無効になります。詳細については、「Amazon EC2 ユーザーガイド」の「ptrace 保護を無効にする」を参照してください。

> **Note**
>
> Arm ベースの Graviton2 インスタンスでの EFA のサポートが AWS ParallelCluster バージョン 2.10.1 で追加されました。

インテル Select ソリューション

AWS ParallelCluster は、シミュレーションとモデリング用のインテル Select ソリューションとして利用できます。設定は、インテル HPC プラットフォーム仕様で設定された標準を満たすように検証され、特定のインテルインスタンスタイプを使用し、 Elastic Fabric Adapter (EFA) ネットワークインターフェイスを使用するように設定されています。 AWS ParallelCluster は、インテル Select Solutions プログラムの要件を満たす最初のクラウドソリューションです。サポートしているインスタンスタイプには `c5n.18xlarge`、`m5n.24xlarge`、`r5n.24xlarge` などがあります。インテル Select ソリューションズ規格に対応した構成例を以下に示します。

Example インテル Select ソリューションズの構成

```
[global]
update_check = true
sanity_check = true
cluster_template = intel-select-solutions

[aws]
aws_region_name = <Your AWS #####>

[scaling demo]
scaledown_idletime = 5

[cluster intel-select-solutions]
key_name = <Your SSH key name>
base_os = centos7
scheduler = slurm
enable_intel_hpc_platform = true
master_instance_type = c5.xlarge
vpc_settings = <Your VPC section>
scaling_settings = demo
queue_settings = c5n,m5n,r5n
master_root_volume_size = 200
compute_root_volume_size = 80
```

```
[queue c5n]
compute_resource_settings = c5n_i1
enable_efa = true
placement_group = DYNAMIC

[compute_resource c5n_i1]
instance_type = c5n.18xlarge
max_count = 5

[queue m5n]
compute_resource_settings = m5n_i1
enable_efa = true
placement_group = DYNAMIC

[compute_resource m5n_i1]
instance_type = m5n.24xlarge
max_count = 5

[queue r5n]
compute_resource_settings = r5n_i1
enable_efa = true
placement_group = DYNAMIC

[compute_resource r5n_i1]
instance_type = r5n.24xlarge
max_count = 5
```

AWS ParallelCluster および Intel HPC プラットフォーム仕様の詳細については、「」を参照してください[インテル HPC プラットフォームの仕様](#)。

Intel MPI を有効にする

Intel MPI は AWS ParallelCluster AMIsで使用できます。インテル MPI を使用するには、「Intel simplified software license」(インテル簡易ソフトウェアライセンス) の条項を確認し、同意する必要があります。デフォルトでは、Open MPI はパス上に配置されます。Open MPI の代わりにインテル MPI を有効にするには、まずインテル MPI モジュールをロードする必要があります。その後、`module load intelmpi` を使用して最新版をインストールする必要があります。モジュールの正確な名前は、更新ごとに変更されます。使用可能なモジュールを確認するには、`module avail` を実行します。出力は次のとおりです。

```
$ module avail
```

```
------------------------------------------ /usr/share/Modules/modulefiles
 -----------------------------------------
dot                        libfabric-aws/1.8.1amzn1.3 module-info                null
                     use.own
module-git                 modules                    openmpi/4.0.2

------------------------------------------------- /etc/modulefiles
 -------------------------------------------------

--------------------------------- /opt/intel/impi/2019.7.217/intel64/modulefiles
 ----------------------------------
intelmpi
```

```
$ module load intelmpi
```

ロードされているモジュールを確認するには、`module list` を実行します。

```
$ module list
Currently Loaded Modulefiles:
  1) intelmpi
```

Intel MPI が有効になっていることを確認するには、`mpirun --version` を実行します。

```
$ mpirun --version
Intel(R) MPI Library for Linux* OS, Version 2019 Update 7 Build 20200312 (id:
 5dc2dd3e9)
Copyright 2003-2020, Intel Corporation.
```

Intel MPI モジュールがロードされると、Intel MPI ツールを使用するように複数のパスが変更されます。Intel MPI ツールでコンパイルされたコードを実行するには、まず Intel MPI モジュールをロードします。

> **Note**
>
> Intel MPI は Graviton AWS ベースのインスタンスと互換性がありません。

> **Note**
>
> AWS ParallelCluster バージョン 2.5.0 以前は、Intel MPI は中国 (北京) および中国 (寧夏) リージョンの AWS ParallelCluster AMIs では使用できませんでした。

インテル HPC プラットフォームの仕様

AWS ParallelCluster は Intel HPC プラットフォーム仕様に準拠しています。インテル HPC プラットフォームの仕様は、コンピューティング、ファブリック、メモリ、ストレージ、ソフトウェア要件のセットを提供し、高水準の品質と HPC ワークロードとの互換性を確保します。詳細については、「Intel HPC Platform Specification」(インテル HPC プラットフォームの仕様) および「Applications Verified Compatible with the Intel HPC Platform Specification」(アプリケーションのインテル HPC プラットフォームの仕様との互換性の確認) を参照してください。

Intel HPC Platform Specification に準拠するには、以下の要件を満たす必要があります。

- オペレーティングシステムは CentOS 7 (`base_os = centos7`) でなければなりません。
- コンピューティングノードのインスタンスタイプには、Intel CPU と 64 GB 以上のメモリが必要です。インスタンスタイプの `c5` ファミリーの場合、インスタンスタイプは少なくとも `c5.9xlarge` (`compute_instance_type = c5.9xlarge`) であることが必要です。
- ヘッドノードには 200 GB 以上のストレージが必要です。
- Intel Parallel Studio のエンドユーザー使用許諾契約書に同意する必要があります (`enable_intel_hpc_platform = true`)。
- 各コンピューティングノードには、少なくとも 80 GB のストレージが必要です (`compute_root_volume_size = 80`)。

ストレージは、ローカルまたはネットワーク (ヘッドノードの NFS 共有、Amazon EBS または FSx for Lustre) に存在でき、共有することもできます。

Arm パフォーマンスライブラリ

AWS ParallelCluster バージョン 2.10.1 以降、ARM パフォーマンスライブラリは、 `base_os` 設定の `alinux2`、、`ubuntu1804`、および `ubuntu2004`値の AWS ParallelCluster AMIs `centos8`で使用できます。Arm パフォーマンスライブラリは、Arm プロセッサ上のハイパフォーマンスコンピューティングアプリケーション向けに最適化された標準コア算術ライブラリを提供します。Arm

パフォーマンスライブラリを使用するには、Arm パフォーマンスライブラリ (無償版) - エンドユーザーライセンスの条項を確認し、同意する必要があります。Arm パフォーマンスライブラリの詳細については、「Free Arm Performance Libraries」(Free Arm パフォーマンスライブラリ) を参照してください。

Arm パフォーマンスライブラリを有効にするには、まず Arm パフォーマンスライブラリモジュールをロードする必要があります。`Armpl-21.0.0` は要件として GCC-9.3 が必要で、`armpl/21.0.0` モジュールをロードすると、`gcc/9.3` モジュールもロードされます。モジュールの正確な名前は、更新ごとに変更されます。使用可能なモジュールを確認するには、`module avail` を実行します。その後、`module load armpl` を使用して最新版をインストールする必要があります。出力は次の通りです。

```
$ module avail

------------------------------ /usr/share/Modules/modulefiles
 --------------------------------
armpl/21.0.0         dot                  libfabric-aws/1.11.1amzn1.0
 module-git
module-info          modules              null                 openmpi/4.1.0
 use.own
```

モジュールをロードするには、`module load` *`modulename`* を実行します。`mpirun` を実行するために使用するスクリプトにこれを追加できます。

```
$ module load armpl

Use of the free of charge version of Arm Performance Libraries is subject to the terms
 and
conditions of the Arm Performance Libraries (free version) -  End User License
 Agreement
(EULA). A copy of the EULA can be found in the
'/opt/arm/armpl/21.0.0/arm-performance-libraries_21.0_gcc-9.3/license_terms' folder
```

ロードされているモジュールを確認するには、`module list` を実行します。

```
$ module list
Currently Loaded Modulefiles:
1) /opt/arm/armpl/21.0.0/modulefiles/armpl/gcc-9.3
2) /opt/arm/armpl/21.0.0/modulefiles/armpl/21.0.0_gcc-9.3
3) armpl/21.0.0
```

Arm パフォーマンスライブラリが有効であることを確認するために、サンプルテストを実行します。

```
$ sudo chmod 777 /opt/arm/armpl/21.0.0/armpl_21.0_gcc-9.3/examples
$ cd /opt/arm/armpl/21.0.0/armpl_21.0_gcc-9.3/examples
$ make
...
Testing: no example difference files were generated.
Test passed OK
```

Arm パフォーマンスライブラリモジュールがロードされた後、Arm パフォーマンスライブラリツールを使用するために複数のパスが変更されます。Arm パフォーマンスライブラリツールでコンパイルされたコードを実行するには、まず Arm パフォーマンスライブラリをロードします。

> **Note**
>
> AWS ParallelCluster 2.10.1 から 2.10.4 までのバージョンでは、 を使用しますarmpl/20.2.1。

Amazon DCV を介してヘッドノードに接続する

Amazon DCV はリモート視覚化テクノロジーで、リモートの高性能サーバーでホストされている、グラフィックを多用する 3D アプリケーションに安全に接続することを可能にします。詳細については、「Amazon DCV」を参照してください。

base_os = alinux2、base_os = centos7、base_os = ubuntu1804、または base_os = ubuntu2004 を使用すると、Amazon DCV ソフトウェアが自動的にヘッドノードにインストールされます。

ヘッドノードが ARM インスタンスである場合は、base_os = alinux2、base_os = centos7、または base_os = ubuntu1804 を使用すると、Amazon DCV ソフトウェアが自動的にヘッドノードにインストールされます。

ヘッドノードで Amazon DCV を有効にするには、enable = master を持つ [dcv] セクションの名前を dcv_settings 内に含め、base_os を alinux2、centos7、ubuntu1804、または ubuntu2004 に設定する必要があります。ヘッドノードが ARM インスタンスの場合、base_os を alinux2、centos7、または ubuntu1804 に設定する必要があります。これにより、 はクラス

ター設定パラメータshared_dirを DCV サーバーストレージフォルダ AWS ParallelCluster に設定します。

```
[cluster custom-cluster]
...
dcv_settings = custom-dcv
...
[dcv custom-dcv]
enable = master
```

Amazon DCV の設定パラメータの詳細については、「dcv_settings」を参照してください。Amazon DCV セッションに接続するには、pcluster dcv コマンドを使用します。

> **Note**
>
> バージョン 2.10.4 では、 での Amazon DCV AWS ParallelCluster のサポートcentos8が削除されました。バージョン 2.10.0 で での Amazon DCV AWS ParallelCluster のサポートが追加されcentos8ました。Graviton AWS ベースのインスタンスでの Amazon DCV のサポートが AWS ParallelCluster バージョン 2.9.0 で追加されました。alinux2 および での Amazon DCV のサポートubuntu1804が AWS ParallelCluster バージョン 2.6.0 で追加されました。 AWS ParallelCluster バージョン 2.5.0 で での Amazon DCV のサポートが追加されcentos7ました。

> **Note**
>
> Amazon DCV は、 AWS ParallelCluster バージョン AWS 2.8.0 および 2.8.1 の Graviton ベースのインスタンスではサポートされていません。

Amazon DCV HTTPS 証明書

Amazon DCV は、Amazon DCV クライアント と Amazon DCV サーバー間のトラフィックを保護するための自己署名証明書を自動的に生成します。

デフォルトの自己署名 Amazon DCV 証明書を別の証明書に置き換えるには、まずヘッドノードに接続します。次に、pcluster dcv コマンドを実行する前に、証明書とキーの両方を /etc/dcv フォルダにコピーします。

詳細については、「Amazon DCV 管理者ガイド」の「TLS 証明書の変更」を参照してください。

Amazon DCV のライセンス

Amazon DCV は、Amazon EC2 インスタンスで実行する場合、ライセンスサーバーを必要としません。ただし、Amazon DCV サーバーは定期的に Amazon S3 バケットに接続して、有効なライセンスが利用可能かどうかを判断する必要があります。

AWS ParallelCluster は、 に必要なアクセス許可を自動的に追加しますParallelClusterInstancePolicy。カスタム IAM インスタンスポリシーを使用する場合は、「Amazon DCV 管理者ガイド」の「Amazon EC2 での Amazon DCV」で説明しているアクセス許可を使用します。

トラブルシューティングのヒントについては、「Amazon DCV の問題のトラブルシューティング」を参照してください。

pcluster update を使用する

AWS ParallelCluster バージョン 2.8.0 以降、 は現在のクラスターの作成に使用される設定と、設定ファイル内の設定で問題がないかpcluster updateを分析します。問題が発見された場合は報告され、問題を解決するための手順が表示されます。例えば、compute_instance_type 設定を異なるインスタンスタイプに変更した場合、アップデートを進める前にコンピューティングフリートを停止させる必要があります。この問題は、発見された時点で報告されます。ブロックの問題が報告されない場合、変更を適用するかどうかを確認するメッセージが表示されます。

各設定のドキュメントには、その設定の更新ポリシーが定義されています。

更新ポリシー: これらの設定は、更新中に変更できます。, 更新ポリシー: この設定は、更新中に変更できます。

> これらの設定は変更することができ、pcluster update を使用してクラスターを更新することができます。

更新ポリシー: この設定が変更された場合、更新は許可されません。

> これらの設定は、既存のクラスターが削除されていない場合は変更できません。変更を取り消すか、クラスターを削除して (pcluster delete を使用します)、古いクラスターの場所に新しいクラスターを作成する必要があります (pcluster create を使用します)。

更新ポリシー: この設定は、更新中には分析されません。

これらの設定は、pcluster update を使用して変更し、クラスターを更新することができます。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

これらの設定は、コンピューティングフリートが存在する間は変更できません。変更を元に戻すか、コンピューティング・フリートを停止 (pcluster stop を使用します) し、アップデート (pcluster update を使用します) し、新しいコンピューティングフリートを作成 (pcluster start を使用をします) する必要があります。

更新ポリシー: これらの設定は、更新中に減らすことはできません。

これらの設定は変更できますが、減らすことはできません。これらの設定を減らす必要がある場合は、クラスターを削除し (pcluster delete を使用します)、新しいクラスターを作成する必要があります (pcluster create を使用します)。

更新ポリシー: キューを現在のノード数以下に減らすには、まずコンピューティングフリートを停止させる必要があります。

これらの設定は変更可能ですが、もし変更によってキューのサイズが現在のサイズより小さくなる場合は、コンピューティングフリートを停止 (pcluster stop を使用します) し、アップデート (pcluster update を使用します) し、そして、新しいコンピューティングフリートを作成 (pcluster start を使用します) しなければなりません。

更新ポリシー: キュー内の静的ノードの数を減らすには、まずコンピューティングフリートを停止させる必要があります。

これらの設定は変更可能ですが、もし変更によってキュー内の静的ノードの数が現在のサイズより小さくなる場合は、コンピューティングフリートを停止 (pcluster stop を使用します) し、アップデート (pcluster update を使用します) し、そして、新しいコンピューティングフリートを作成 (pcluster start を使用します) しなければなりません。

更新ポリシー: この設定が変更された場合、更新は許可されません。この設定を強制的に更新することはできません。

これらの設定は、既存のクラスターが削除されていない場合は変更できません。変更を取り消すか、クラスターを削除して (pcluster delete を使用します)、古いクラスターの場所に新しいクラスターを作成する必要があります (pcluster create を使用します)。

更新ポリシー: AWS ParallelCluster マネージド Amazon FSx for Lustre ファイルシステムが設定で指定されていない場合、この設定は更新中に変更できます。

[cluster] fsx_settings が指定されていない場合、または [fsx fs] の fsx_settings と fsx-fs-id の両方が既存の外部 FSx for Lustre ファイルシステムのマウントに指定されている場合、この設定を変更できます。

この例では、更新をブロックするような変更が加えられた pcluster update をデモしています。

```
$ pcluster update
  Validating configuration file /home/username/.parallelcluster/config...
Retrieving configuration from CloudFormation for cluster test-1...
Found Changes:

#   section/parameter         old value                 new value
--  ------------------------  ------------------------  ------------------------
    [cluster default]
01* compute_instance_type     t2.micro                  c4.xlarge
02* ebs_settings              ebs2                      -

    [vpc default]
03  additional_sg             sg-0cd61884c4ad16341       sg-0cd61884c4ad11234

    [ebs ebs2]
04* shared_dir                shared                     my/very/very/long/sha...

Validating configuration update...
The requested update cannot be performed. Line numbers with an asterisk indicate
updates requiring additional actions. Please look at the details below:

#01
Compute fleet must be empty to update "compute_instance_type"
How to fix:
Make sure that there are no jobs running, then run the following command:
  pcluster stop -c $CONFIG_FILE $CLUSTER_NAME

#02
Cannot add/remove EBS Sections
How to fix:
Revert "ebs_settings" value to "ebs2"

#04
```

```
Cannot change the mount dir of an existing EBS volume
How to fix:
Revert "my/very/very/long/shared/dir" to "shared"

In case you want to override these checks and proceed with the update please
use the --force flag. Note that the cluster could end up in an unrecoverable
state.

Update aborted.
```

AMI のパッチ適用と EC2 インスタンスの交換

動的に起動されたすべてのクラスターコンピューティングノードが一貫した方法で動作するように、はクラスターインスタンスの自動 OS 更新 AWS ParallelCluster を無効にします。さらに、 のバージョン AWS ParallelCluster および関連する CLI ごとに特定の AWS ParallelCluster AMIs セットが構築されます。この特定の AMIs セットは変更されず、ビルドされた AWS ParallelCluster バージョンでのみサポートされます。リリースされたバージョンの AWS ParallelCluster AMIs は更新されません。

ただし、緊急のセキュリティ問題により、お客様はこれらの AMI にパッチを追加し、パッチが適用された AMI でクラスターを更新したい場合があります。これはAWS ParallelCluster 責任共有モデルと一致しています。

現在使用している CLI AWS ParallelCluster バージョンでサポートされている特定の AWS ParallelCluster AMIs のセットを表示するには、以下を実行します。

```
$ pcluster version
```

次に、GitHub リポジトリで amis.txt AWS ParallelClusterを表示します。

AWS ParallelCluster ヘッドノードは静的インスタンスであり、手動で更新できます。インスタンスタイプにインスタンスストアがない場合、ヘッドノードの再起動と再起動は AWS ParallelCluster バージョン 2.11 以降で完全にサポートされています。詳細については、「Linux インスタンス用 Amazon EC2 ユーザーガイド」の「Instance types with instance store volumes」を参照してください。既存のクラスター用の AMI を更新することはできません。

クラスターコンピューティングインスタンスの AMI 更新によるヘッドノードの再起動と再起動は、AWS ParallelCluster バージョン 3.0.0 以降で完全にサポートされています。これらの機能を使用するには、最新バージョンへのアップグレードを検討してください。

ヘッドノードインスタンスの更新または交換

状況によっては、ヘッドノードの再開または再起動が必要になることがあります。例えば、OS を手動で更新する場合や、ヘッドノードインスタンスの再起動を強制する AWS インスタンスのリタイアの予定がある場合などに必要です。

インスタンスにエフェメラルドライブがない場合は、いつでも停止して再び起動できます。リタイアが予定されている場合、停止したインスタンスを開始すると、新しいハードウェアを使用するように移行されます。

同様に、インスタンスストアがないインスタンスは手動で停止して起動できます。この場合や、エフェメラルボリュームのない他のインスタンスについては、クラスターのヘッドノードを停止して起動します。 に進んでください。

インスタンスにエフェメラルドライブがあり、停止されている場合、インスタンスストアのデータは失われます。ヘッドノードに使用されているインスタンスタイプにインスタンスストアがあるかどうかは、「Instance store volumes」にあるテーブルで確認できます。

以下のセクションでは、インスタンスストアボリュームでインスタンスを使用する際の制限について説明します。

インスタンスストアの制限事項

インスタンスストアで AWS ParallelCluster バージョン 2.11 およびインスタンスタイプを使用する場合の制限は次のとおりです。

- エフェメラルドライブが暗号化されていない場合 (`encrypted_ephemeral` パラメータが に設定されている`false`か、設定されていない場合) 、 AWS ParallelCluster インスタンスはインスタンスの停止後に起動できません。これは、存在しない古いエフェメラルに関する情報が `fstab` に書き込まれ、OS は存在しないストレージをマウントしようとするためです。
- エフェメラルドライブが暗号化されている場合 (`encrypted_ephemeral` パラメータが に設定されている場合`true`) 、 AWS ParallelCluster インスタンスは停止後に開始できますが、新しいエフェメラルドライブはセットアップ、マウント、または使用できません。
- エフェメラルドライブが暗号化されている場合、 AWS ParallelCluster インスタンスは再起動できますが、古いエフェメラルドライブ (インスタンスの再起動後も存続) にはアクセスできません。これは、暗号化キーが再起動時に失われたメモリに作成されるためです。

サポートされている唯一のケースは、エフェメラルドライブが暗号化されていない場合のインスタンスのリブートです。これは、ドライブがリブート後も存続し、fstab に書き込まれたエントリによって再びマウントされるためです。

インスタンスストアの制限回避策

まず、データを保存します。保存する必要のあるデータがあるかどうかを確認するには、ephemeral_dir フォルダの内容を表示します (デフォルトでは /scratch)。データは、ルートボリュームまたはクラスターにアタッチされた共有ストレージシステム (Amazon FSx、Amazon EFS、Amazon EBS など) に転送できます。リモートストレージへのデータ転送には追加コストが発生する可能性があることに注意してください。

制限の根本原因は、 がインスタンスストアボリュームのフォーマットとマウント AWS ParallelCluster に使用するロジックにあります。このロジックは、以下の形式の /etc/fstab にエントリを追加します。

```
$ /dev/vg.01/lv_ephemeral ${ephemeral_dir} ext4 noatime,nodiratime 0 0
```

${ephemeral_dir} は pcluster 設定ファイルの ephemeral_dir パラメータの値です (デフォルトは /scratch)。

この行を追加し、ノードがリブートされた場合、またはリブートされたときに、インスタンスストアボリュームが自動的に再マウントされるようにします。エフェメラルドライブ内のデータはリブート後も保持されるため、これは望ましいことです。ただし、エフェメラルドライブ上のデータは、開始サイクルまたは停止サイクル後は保持されません。つまり、データなしでフォーマットされマウントされるということです。

サポートされている唯一のケースは、エフェメラルドライブが暗号化されていない場合のインスタンスのリブートです。これは、ドライブがリブート後も存続し、fstab に書き込まれているため、マウントし直されるためです。

それ以外の場合でデータを保存するには、インスタンスを停止する前に論理ボリュームエントリを削除する必要があります。例えば、インスタンスを停止する前に /dev/vg.01/lv_ephemeral を /etc/fstab から削除します。この後、エフェメラルボリュームをマウントせずにインスタンスを起動します。ただし、インスタンスストアマウントは、インスタンスの停止または起動後に再び使用できなくなります。

データを保存して fstab エントリを削除したら、次のセクションに進みます。

クラスターのヘッドノードを停止して起動します。

> **Note**
>
> AWS ParallelCluster バージョン 2.11 以降、ヘッドノードの停止と開始は、インスタンスタイプにインスタンスストアがない場合にのみサポートされます。

1. クラスターに実行中のジョブがないことを確認します。

   Slurm スケジューラーを使用する場合。

   - `sbatch --no-requeue` オプションが指定されていない場合、実行中のジョブはキューに入れられます。
   - `--no-requeue` オプションを指定すると、実行中のジョブは失敗します。

2. クラスターコンピューティングフリートの停止を要求します。

```
$ pcluster stop cluster-name
Compute fleet status is: RUNNING. Submitting status change request.
Request submitted successfully. It might take a while for the transition to
 complete.
Please run 'pcluster status' if you need to check compute fleet status
```

3. コンピューティングフリートのステータスが `STOPPED` になるまで待ちます。

```
$ pcluster status cluster-name
...
ComputeFleetStatus: STOP_REQUESTED
$ pcluster status cluster-name
...
ComputeFleetStatus: STOPPED
```

4. OS の再起動またはインスタンスの再起動による手動更新の場合は、 AWS Management Console または を使用できます AWS CLI。以下に示しているのは、 AWS CLIを使用した例です。

```
$ aws ec2 stop-instances --instance-ids 1234567890abcdef0
{
  "StoppingInstances": [
    {
```

```
      "CurrentState": {
        "Name": "stopping"
        ...
      },
      "InstanceId": "i-1234567890abcdef0",
      "PreviousState": {
        "Name": "running"
        ...
      }
    }
  ]
}
$ aws ec2 start-instances --instance-ids 1234567890abcdef0
{
  "StartingInstances": [
    {
      "CurrentState": {
        "Name": "pending"
        ...
      },
      "InstanceId": "i-1234567890abcdef0",
      "PreviousState": {
        "Name": "stopped"
        ...
      }
    }
  ]
}
```

5. クラスターのコンピューティングフリートを起動します。

```
$ pcluster start cluster-name
Compute fleet status is: STOPPED. Submitting status change request.
Request submitted successfully. It might take a while for the transition to
 complete.
Please run 'pcluster status' if you need to check compute fleet status
```

AWS ParallelCluster CLI コマンド

pcluster および pcluster-configは AWS ParallelCluster CLI コマンドです。pcluster を使用して、 で HPC クラスターを起動および管理 AWS クラウド しpcluster-config、設定を更新します。

pcluster を使用するには、実行に必要なアクセス許可を持つ IAM ロールが必要です。

```
pcluster [ -h ] ( create | update | delete | start | stop | status | list |
                  instances | ssh | dcv | createami | configure | version ) ...
 pcluster-config [-h] (convert) ...
```

トピック

- pcluster
- pcluster-config

pcluster

pcluster はプライマリ AWS ParallelCluster CLI コマンドです。pcluster は、 AWS クラウド内の HPC クラスターを起動し、管理するために使用します。

```
pcluster [ -h ] ( create | update | delete | start | stop | status | list |
                  instances | ssh | dcv | createami | configure | version ) ...
```

引数

pcluster *command*

選択肢:
configure、create、createami、dcv、delete、instances、list、ssh、start、status、s

サブコマンド:

トピック

- pcluster configure
- pcluster create
- pcluster createami
- pcluster dcv
- pcluster delete
- pcluster instances
- pcluster list
- pcluster ssh
- pcluster start
- pcluster status
- pcluster stop
- pcluster update
- pcluster version

pcluster configure

AWS ParallelCluster 設定を開始します。詳細については、「の設定 AWS ParallelCluster」を参照してください。

```
pcluster configure [ -h ] [ -c CONFIG_FILE ] [ -r REGION ]
```

名前付き引数

-h, --help

pcluster configure のヘルプテキストを表示します。

-c *CONFIG_FILE*, --config *CONFIG_FILE*

使用する代替設定ファイルのフルパスを指定します。

デフォルトは ~/.parallelcluster/config です。

詳細については、「の設定 AWS ParallelCluster」を参照してください。

-r *REGION*, --region *REGION*

AWS リージョン 使用する を指定します。これを指定すると、設定は AWS リージョン 検出をスキップします。

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

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

スタックは、delete-stack コマンドで削除することができます。

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

pcluster configure が作成する VPC は CloudFormation のネットワークスタックには作成されません。その VPC はコンソールで手動で削除するか、または AWS CLIを使用して削除できます。

```
$ aws --region us-east-1 ec2 delete-vpc --vpc-id vpc-0b4ad9c4678d3c7ad
```

pcluster create

新しいクラスターを作成します。

```
pcluster create [ -h ] [ -c CONFIG_FILE ] [ -r REGION ] [ -nw ] [ -nr ]
                [ -u TEMPLATE_URL ] [ -t CLUSTER_TEMPLATE ]
                [ -p EXTRA_PARAMETERS ] [ -g TAGS ]
                cluster_name
```

位置引数

cluster_name

クラスターの名前を定義します。 AWS CloudFormation スタック名は ですparallelcluster-*cluster_name*。

名前付き引数

-h, --help

pcluster create のヘルプテキストを表示します。

-c *CONFIG_FILE*, --config *CONFIG_FILE*

使用する代替設定ファイルを指定します。

デフォルトは ~/.parallelcluster/config です。

-r *REGION*, --region *REGION*

AWS リージョン 使用する を指定します。新しいクラスター AWS リージョン の を選択するために使用される優先順位は次のとおりです。

1. -r または --region パラメータを pcluster create に渡す。
2. AWS_DEFAULT_REGION 環境変数
3. aws_region_name 設定ファイルの AWS ParallelCluster [aws]セクションで を設定する (デフォルトの場所は です) ~/.parallelcluster/config。pcluster configure コマンドで更新される場所です。
4. region 設定ファイル (~/.aws/config.) AWS CLI の [default]セクションで を設定する

-nw, --nowait

スタックコマンドの実行後にスタックイベントを待機しないことを示します。

デフォルトは False です。

-nr, --norollback

エラー時に スタックのロールバックを無効にします。

デフォルトは False です。

-u *TEMPLATE_URL*, --template-url *TEMPLATE_URL*

作成時に使用されたカスタム AWS CloudFormation テンプレートの URL を指定します。

-t *CLUSTER_TEMPLATE*, --cluster-template *CLUSTER_TEMPLATE*

使用するクラスターテンプレートを示します。

-p *EXTRA_PARAMETERS*, --extra-parameters *EXTRA_PARAMETERS*

スタック作成に追加のパラメータを追加します。

-g *TAGS*, --tags *TAGS*

スタックに追加するその他のタグを指定します。

コマンドが呼び出され、その呼び出しのステータスのポーリングが開始された場合は、「Ctrl-C」を使用して終了するのが安全です。`pcluster status mycluster` を呼び出すことで、現在のステータスの表示に戻ることができます。

AWS ParallelCluster バージョン 2.11.7 を使用する例：

```
$ pcluster create mycluster
    Beginning cluster creation for cluster: mycluster
Info: There is a newer version 3.1.4 of AWS ParallelCluster available.
Creating stack named: parallelcluster-mycluster
Status: ComputeFleetHITSubstack - CREATE_IN_PROGRESS
$ pcluster create mycluster --tags '{ "Key1" : "Value1" , "Key2" : "Value2" }'
```

pcluster createami

(Linux/macOS) 使用するカスタム AMI を作成します AWS ParallelCluster。

```
pcluster createami [ -h ] -ai BASE_AMI_ID -os BASE_AMI_OS
                   [ -i INSTANCE_TYPE ] [ -ap CUSTOM_AMI_NAME_PREFIX ]
                   [ -cc CUSTOM_AMI_COOKBOOK ] [--no-public-ip]
                   [ -post-install POST_INSTALL_SCRIPT ]
                   [ -c CONFIG_FILE ] [-t CLUSTER_TEMPLATE]
                   [--vpc-id VPC_ID] [--subnet-id SUBNET_ID]
                   [ -r REGION ]
```

必要な依存関係

AWS ParallelCluster CLI に加えて、 を実行するには次の依存関係が必要です`pcluster createami`。

- Packer: https://developer.hashicorp.com/packer/downloads から最新バージョンをダウンロードします。

> **Note**
>
> AWS ParallelCluster バージョン 2.8.0 以前は、 を使用するには Berkshelf (を使用してインストール`gem install berkshelf`) が必要でした`pcluster createami`。

名前付き引数

`-h, --help`

`pcluster createami` のヘルプテキストを表示します。

`-ai` *`BASE_AMI_ID`*, `--ami-id` *`BASE_AMI_ID`*

AMI の構築に使用するベース AMI AWS ParallelCluster を指定します。

`-os` *`BASE_AMI_OS`*, `--os` *`BASE_AMI_OS`*

基本 AMI の OS を指定します。有効なオプションは、`alinux2`、`ubuntu1804`、`ubuntu2004`、`centos7` です。

> **Note**
>
> OS はさまざまな AWS ParallelCluster バージョンで変更をサポートしています。
>
> - AWS ParallelCluster バージョン 2.10.4 で のサポート`centos8`が削除されました。
> - `centos8` のサポートが追加され、`centos6` のサポートが AWS ParallelCluster バージョン 2.10.0 で削除されました。
> - alinux2 のサポートが AWS ParallelCluster バージョン 2.6.0 で追加されました。
> - AWS ParallelCluster バージョン 2.5.0 で ubuntu1804 のサポートが追加されました。

`-i` *`INSTANCE_TYPE`*, `--instance-type` *`INSTANCE_TYPE`*

AMI の作成に使用するインスタンスタイプを指定します。

デフォルトは `t2.xlarge` です。

> **Note**
>
> AWS ParallelCluster バージョン 2.4.1 で `--instance-type`引数のサポートが追加されました。

`-ap` *`CUSTOM_AMI_NAME_PREFIX`*, `--ami-name-prefix` *`CUSTOM_AMI_NAME_PREFIX`*

結果の AMI AWS ParallelCluster のプレフィックス名を指定します。

デフォルトは `custom-ami-` です。

`-cc` *`CUSTOM_AMI_COOKBOOK`*, `--custom-cookbook` *`CUSTOM_AMI_COOKBOOK`*

AMI AWS ParallelCluster の構築に使用するクックブックを指定します。

`--post-install` *`POST_INSTALL_SCRIPT`*

ポストインストールスクリプトのパスを指定します。パスは、`s3://`、`https://`、`file://` のいずれかの URL スキームを使用する必要があります。次に例を示します。

- `https://`*`bucket-name`*`.s3.`*`region`*`.amazonaws.com/`*`path`*`/post_install.sh`
- `s3://`*`bucket-name`*`/post_install.sh`
- `file:///`*`opt/project`*`/post_install.sh`

> **Note**
>
> AWS ParallelCluster バージョン 2.10.0 で `--post-install`引数のサポートが追加されました。

`--no-public-ip`

AMI の作成に使用したインスタンスには、パブリック IP アドレスを関連付けないでください。デフォルトでは、パブリック IP アドレスがインスタンスに関連付けられます。

> **Note**
>
> AWS ParallelCluster バージョン 2.5.0 で `--no-public-ip`引数のサポートが追加されました。

-c *CONFIG_FILE*, --config *CONFIG_FILE*

使用する代替設定ファイルを指定します。

デフォルトは `~/.parallelcluster/config` です。

-t *CLUSTER_TEMPLATE*, --cluster-template *CLUSTER_TEMPLATE*

VPC およびサブネットの設定値を取得するために使用する *CONFIG_FILE* の [cluster] セクションを指定します。

> **Note**
> AWS ParallelCluster バージョン 2.4.0 で `--cluster-template`引数のサポートが追加されました。

--vpc-id *VPC_ID*

AMI の構築に使用する VPC の ID AWS ParallelCluster を指定します。

> **Note**
> AWS ParallelCluster バージョン 2.5.0 で `--vpc-id`引数のサポートが追加されました。

--subnet-id *SUBNET_ID*

AMI の構築に使用するサブネットの ID AWS ParallelCluster を指定します。

> **Note**
> AWS ParallelCluster バージョン 2.5.0 で `--vpc-id`引数のサポートが追加されました。

-r *REGION*, --region *#####*

AWS リージョン 使用する を指定します。デフォルトは、 `pcluster configure` コマンドを使用して AWS リージョン 指定された です。

pcluster dcv

ヘッドノードで実行している Amazon DCV サーバーとやり取りします。

```
pcluster dcv [ -h ] ( connect )
```

pcluster dcv *command*

選択肢: connect

> **Note**
>
> OS は、異なる AWS ParallelCluster バージョンの `pcluster dcv` コマンドの変更をサポートしています。
>
> - AWS ParallelCluster バージョン 2.10.0 で `centos8` での `pcluster dcv` コマンドをサポートしました。
> - Graviton AWS ベースのインスタンスでの `pcluster dcv` コマンドのサポートが AWS ParallelCluster バージョン 2.9.0 に追加されました。
> - AWS ParallelCluster バージョン 2.6.0 で `ubuntu1804` での `pcluster dcv` コマンドをサポートしました。
> - AWS ParallelCluster バージョン 2.5.0 で `centos7` での `pcluster dcv` コマンドをサポートしました。

名前付き引数

-h, --help

`pcluster dcv` のヘルプテキストを表示します。

サブコマンド

pcluster dcv connect

```
pcluster dcv connect [ -h ] [ -k SSH_KEY_PATH ] [ -r REGION ] cluster_name
```

> **Important**
>
> URL は、発行されてから 30 秒後に有効期限が切れます。URL の有効期限が切れる前に接続が確立されていない場合、`pcluster dcv connect` を再度実行して新しい URL を生成します。

位置引数

cluster_name

接続するクラスターの名前を指定します。

名前付き引数

`-h, --help`

`pcluster dcv connect` のヘルプテキストを表示します。

`-k` *`SSH_KEY_PATH`*, `--key-path` *`SSH_KEY_PATH`*

接続に使用する SSH キーのキーパス。

キーは、key_name 設定パラメータでクラスター作成時に指定したものである必要があります。この引数はオプションですが、指定しない場合は、SSH クライアントに対してデフォルトでそのキーが使用可能になる必要があります。例えば、`ssh-agent` で `ssh-add` に追加します。

`-r` *`REGION`*, `--region` *`#####`*

AWS リージョン 使用する を指定します。デフォルトは、 `pcluster configure` コマンドを使用して AWS リージョン 指定された です。

`-s, --show-url`

Amazon DCV セッションに接続するための 1 回限りの URL を表示します。このオプションを指定すると、デフォルトのブラウザは開きません。

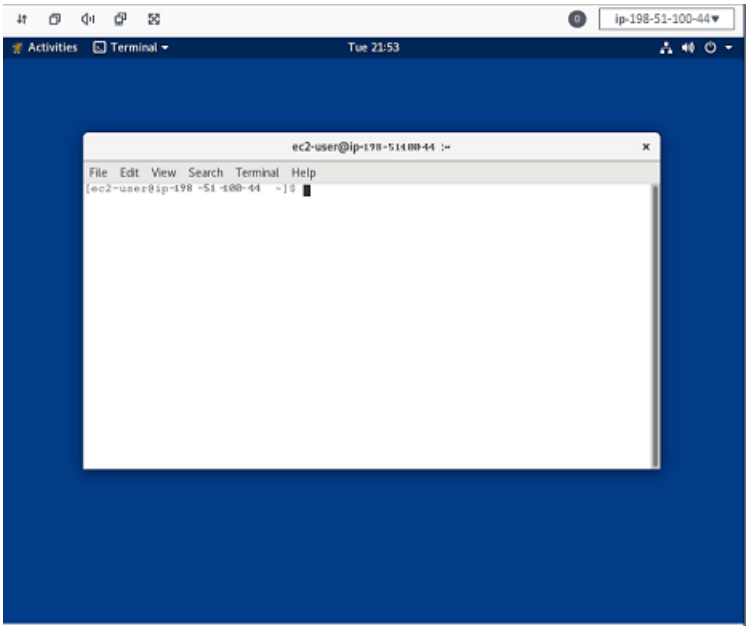
> **Note**
>
> AWS ParallelCluster バージョン 2.5.1 で `--show-url`引数のサポートが追加されました。

AWS ParallelCluster バージョン 2.11.7 の使用例：

```
$ pcluster dcv connect -k ~/.ssh/id_rsa mycluster
```

ヘッドノードで実行中の Amazon DCV セッションに接続するためのデフォルトブラウザを開きます。

新しい Amazon DCV セッションを作成します (まだ開始していない場合)。



pcluster delete

クラスターを削除します。

```
pcluster delete [ -h ] [ -c CONFIG_FILE ] [ -r REGION ] [ -nw ] cluster_name
```

位置引数

cluster_name

削除するクラスターの名前を指定します。

名前付き引数

`-h, --help`

`pcluster delete` のヘルプテキストを表示します。

`-c CONFIG_FILE, --config CONFIG_FILE`

使用する代替設定ファイルを指定します。

デフォルトは `~/.parallelcluster/config` です。

`--keep-logs`

クラスターを削除した後も CloudWatch Logs データを保持します。ロググループは手動で削除するまで残りますが、ログイベントは `retention_days` 設定に基づいて期限切れになります。この設定はデフォルトで 14 日です。

> **Note**
>
> AWS ParallelCluster バージョン 2.6.0 で **`--keep-logs`** 引数をサポートしました。

`-r REGION, --region REGION`

AWS リージョン 使用する を指定します。デフォルトは、 `pcluster configure` コマンドを使用して AWS リージョン 指定された です。

コマンドが呼び出され、その呼び出しのステータスのポーリングが開始された場合は、「Ctrl-C」を使用して終了するのが安全です。`pcluster status mycluster` を呼び出すことで、現在のステータスの表示に戻ることができます。

AWS ParallelCluster バージョン 2.11.7 の使用例：

```
$ pcluster delete -c path/to/config -r us-east-1 mycluster
Deleting: mycluster
Status: RootRole - DELETE_COMPLETE
Cluster deleted successfully.
```

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

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

スタックは、delete-stack コマンドで削除することができます。

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

pcluster configure が作成する VPC は CloudFormation のネットワークスタックには作成されません。その VPC はコンソールで手動で削除するか、または AWS CLIを使用して削除できます。

```
$ aws --region us-east-1 ec2 delete-vpc --vpc-id vpc-0b4ad9c4678d3c7ad
```

pcluster instances

クラスター内のすべてのインスタンスのリストが表示されます。

```
pcluster instances [ -h ] [ -c CONFIG_FILE ] [ -r REGION ] cluster_name
```

位置引数

cluster_name

指定された名前のクラスターのインスタンスを表示します。

名前付き引数

-h, --help

pcluster instances のヘルプテキストを表示します。

-c *CONFIG_FILE*, --config *CONFIG_FILE*

使用する代替設定ファイルを指定します。

デフォルトは ~/.parallelcluster/config です。

-r *REGION*, --region *REGION*

AWS リージョン 使用する を指定します。デフォルトは、 pcluster configure コマンドを使用して AWS リージョン 指定された です。

AWS ParallelCluster バージョン 2.11.7 の使用例：

```
$ pcluster instances -c path/to/config -r us-east-1 mycluster
MasterServer         i-1234567890abcdef0
ComputeFleet         i-abcdef01234567890
```

pcluster list

関連付けられたスタックのリストを表示します AWS ParallelCluster。

```
pcluster list [ -h ] [ -c CONFIG_FILE ] [ -r REGION ]
```

名前付き引数

-h, --help

pcluster list のヘルプテキストを表示します。

--color

クラスターのステータスを色で表示します。

デフォルトは False です。

-c *CONFIG_FILE*, --config *CONFIG_FILE*

使用する代替設定ファイルを指定します。

デフォルトは c です。

-r *REGION*, --region *REGION*

AWS リージョン 使用する を指定します。デフォルトは、 pcluster configure コマンドを使用して AWS リージョン 指定された です。

という名前の AWS CloudFormation スタックの名前を一覧表示しますparallelcluster-*。

AWS ParallelCluster バージョン 2.11.7 の使用例：

```
$ pcluster list -c path/to/config -r us-east-1
mycluster            CREATE_IN_PROGRESS  2.11.7
myothercluster       CREATE_IN_PROGRESS  2.11.7
```

pcluster ssh

事前に入力されているクラスターのユーザー名と IP アドレスを使用して ssh コマンドを実行します。任意の引数は ssh コマンドの末尾に付加されます。このコマンドは、設定ファイルのエイリアスセクションでカスタマイズできます。

```
pcluster ssh [ -h ] [ -d ] [ -r REGION ] cluster_name
```

位置引数

cluster_name

接続するクラスターの名前を指定します。

名前付き引数

-h, --help

pcluster ssh のヘルプテキストを表示します。

-d, --dryrun

実行するコマンドを出力して終了します。

デフォルトは False です。

-r *REGION*, --region *REGION*

AWS リージョン 使用する を指定します。pcluster configure を使用して、指定されたリージョンにデフォルトを設定します。

AWS ParallelCluster バージョン 2.11.7 を使用した例：

```
$ pcluster ssh -d mycluster -i ~/.ssh/id_rsa
```

```
SSH command: ssh ec2-user@1.1.1.1 -i /home/user/.ssh/id_rsa
```

```
$ pcluster ssh mycluster -i ~/.ssh/id_rsa
```

事前に入力されているクラスターのユーザー名と IP アドレスを使用して ssh コマンドを実行します。

```
ssh ec2-user@1.1.1.1 -i ~/.ssh/id_rsa
```

ssh コマンドは、[aliases] セクション のグローバル設定ファイルで定義されます。次のようにカスタマイズできます。

```
[ aliases ]
ssh = ssh {CFN_USER}@{MASTER_IP} {ARGS}
```

代入される変数:

CFN_USER

base_os のユーザー名が選択されています。

MASTER_IP

ヘッドノードの IP アドレス。

ARGS

ssh コマンドに渡すオプションの引数。

pcluster start

停止されているクラスターのコンピューティングフリートを開始します。

```
pcluster start [ -h ] [ -c CONFIG_FILE ] [ -r REGION ] cluster_name
```

位置引数

cluster_name

指定したクラスター名のコンピューティングフリートを起動します。

名前付き引数

-h, --help

pcluster start のヘルプテキストを表示します。

-c *CONFIG_FILE*, --config *CONFIG_FILE*

使用する代替設定ファイルを指定します。

デフォルトは ~/.parallelcluster/config です。

-r *REGION*, --region *REGION*

AWS リージョン 使用する を指定します。デフォルトは、 pcluster configure コマンドを使用して AWS リージョン 指定された です。

AWS ParallelCluster バージョン 2.11.7 の使用例：

```
$ pcluster start mycluster
Compute fleet status is: RUNNING. Submitting status change request.
Request submitted successfully. It might take a while for the transition to complete.
Please run 'pcluster status' if you need to check compute fleet status
```

このコマンドは、Auto Scaling グループのパラメータを次のいずれかに設定します。

- クラスターの作成に使用されたテンプレートの初期設定値 (max_queue_size および initial_queue_size)。
- 最初に作成してからクラスターの更新に使用された設定値。

pcluster status

クラスターの現在の状態をプルします。

```
pcluster status [ -h ] [ -c CONFIG_FILE ] [ -r REGION ] [ -nw ] cluster_name
```

位置引数

cluster_name

指定された名前のクラスターのステータスを表示します。

名前付き引数

-h, --help

pcluster status のヘルプテキストを表示します。

-c *CONFIG_FILE*, --config *CONFIG_FILE*

使用する代替設定ファイルを指定します。

デフォルトは ~/.parallelcluster/config です。

-r *REGION*, --region *REGION*

AWS リージョン 使用する を指定します。デフォルトは、 pcluster configure コマンドを使用して AWS リージョン 指定された です。

-nw, --nowait

スタックコマンド処理後に、スタックイベントを待たないことを示しています。

デフォルトは False です。

AWS ParallelCluster バージョン 2.11.7 の使用例：

```
$ pcluster status -c path/to/config -r us-east-1 mycluster
Status: ComputeFleetHITSubstack - CREATE_IN_PROGRESS
```

pcluster stop

コンピューティングフリートを停止し、ヘッドノードを実行中のままにします。

```
pcluster stop [ -h ] [ -c CONFIG_FILE ] [ -r REGION ] cluster_name
```

位置引数

cluster_name

指定したクラスター名のコンピューティングフリートを停止します。

AWS ParallelCluster バージョン 2.11.7 の使用例：

名前付き引数

-h, --help

pcluster stop のヘルプテキストを表示します。

-c *CONFIG_FILE*, --config *CONFIG_FILE*

使用する代替設定ファイルを指定します。

デフォルトは ~/.parallelcluster/config です。

-r *REGION*, --region *REGION*

AWS リージョン 使用する を指定します。デフォルトは、 pcluster configure コマンドを使用して AWS リージョン 指定された です。

```
$ pcluster stop mycluster
Compute fleet status is: STOPPED. Submitting status change request.
Request submitted successfully. It might take a while for the transition to complete.
Please run 'pcluster status' if you need to check compute fleet status
```

Auto Scaling グループパラメータを最小数/最大数/望ましい数 =0/0/0 に設定し、コンピューティングフリートを終了します。ヘッドは実行されたままです。すべての EC2 リソースを終了して EC2 の料金が発生するのを回避するには、クラスターを削除することを検討してください。

pcluster update

設定ファイルを解析し、クラスターを安全に更新できるかどうかを判断します。解析の結果、クラスターを更新できると判断された場合、変更を確認するメッセージが表示されます。解析の結果、クラスターが更新できない場合は、コンフリクトの原因となるコンフィギュレーション設定が詳細とともに列挙されます。詳細については、「pcluster update を使用する」を参照してください。

```
pcluster update [ -h ] [ -c CONFIG_FILE ] [ --force ] [ -r REGION ] [ -nr ]
                [ -nw ] [ -t CLUSTER_TEMPLATE ] [ -p EXTRA_PARAMETERS ] [ -rd ]
                [ --yes ] cluster_name
```

位置引数

cluster_name

更新するクラスターの名前を指定します。

名前付き引数

-h, --help

pcluster update のヘルプテキストを表示します。

-c *CONFIG_FILE*, --config *CONFIG_FILE*

使用する代替設定ファイルを指定します。

デフォルトは ~/.parallelcluster/config です。

--force

1 つ以上の設定がブロック変更されている場合や、アップデートを実行する前に未処理のアクション (コンピューティングフリートの停止など) が必要な場合でも、アップデートを有効にします。これは、--yes 引数と一緒にしてはいけません。

-r *REGION*, --region *REGION*

AWS リージョン 使用する を指定します。デフォルトは、 pcluster configure コマンドを使用して AWS リージョン 指定された です。

-nr, --norollback

エラー時の AWS CloudFormation スタックのロールバックを無効にします。

デフォルトは False です。

-nw, --nowait

スタックコマンド処理後に、スタックイベントを待たないことを示しています。

デフォルトは False です。

-t *CLUSTER_TEMPLATE*, --cluster-template *CLUSTER_TEMPLATE*

使用するクラスターテンプレートのセクションを指定します。

-p *EXTRA_PARAMETERS*, --extra-parameters *EXTRA_PARAMETERS*

スタック更新に追加のパラメータを追加します。

-rd, --reset-desired

Auto Scaling グループの現在の容量を初期設定値にリセットします。

デフォルトは False です。

--yes

すべてのプロンプトに対する回答が「はい」であると自動的に判断されます。これは、`--force` 引数と一緒にしてはいけません。

```
$ pcluster update -c path/to/config mycluster
Retrieving configuration from CloudFormation for cluster mycluster...
Validating configuration file .parallelcluster/config...
Found Configuration Changes:

#    parameter                   old value    new value
---  --------------------------  -----------  -----------
     [compute_resource default]
01   min_count                   1            2
02   max_count                   5            12

Validating configuration update...
Congratulations! The new configuration can be safely applied to your cluster.
Do you want to proceed with the update? - Y/N: Y
Updating: mycluster
Calling update_stack
Status: parallelcluster-mycluster - UPDATE_COMPLETE
```

コマンドが呼び出され、その呼び出しのステータスのポーリングが開始された場合は、「Ctrl-C」を使用して終了するのが安全です。`pcluster status mycluster` を呼び出すことで、現在のステータスの表示に戻ることができます。

pcluster version

AWS ParallelCluster バージョンを表示します。

```
pcluster version [ -h ]
```

コマンド固有のフラグについて、`pcluster [command] --help` を実行します。

名前付き引数

-h, --help

`pcluster version` のヘルプテキストを表示します。

コマンドが呼び出され、その呼び出しのステータスのポーリングが開始された場合は、「Ctrl-C」を使用して終了するのが安全です。`pcluster status mycluster` を呼び出すことで、現在のステータスの表示に戻ることができます。

```
$ pcluster version
2.11.7
```

pcluster-config

AWS ParallelCluster 設定ファイルを更新します。

```
pcluster-config [ -h ] [convert]
```

コマンド固有のフラグについて、`pcluster-config [command] -h` を実行します。

名前付き引数

-h, --help

`pcluster-config` のヘルプテキストを表示します。

> **Note**
>
> `pcluster-config` コマンドが AWS ParallelCluster バージョン 2.9.0 に追加されました。

サブコマンド

pcluster-config convert

```
pcluster-config convert [ -h ] [ -c CONFIG_FILE ] [ -t CLUSTER_TEMPLATE ]
                        [ -o OUTPUT_FILE ]
```

名前付き引数

-h, --help

`pcluster-config convert` のヘルプテキストを表示します。

-c *CONFIG_FILE*, --config-file *CONFIG_FILE*

読み込む設定ファイルのパスを指定します。

デフォルトは ~/.parallelcluster/config です。

詳細については、「の設定 AWS ParallelCluster」を参照してください。

-t *CLUSTER_TEMPLATE*, --cluster-template *CLUSTER_TEMPLATE*

使用する[cluster] セクションを示します。この引数が指定されない場合、pcluster-config convert は [global] セクション の cluster_template 設定を使用します。それが指定されていない場合は、[cluster default] セクションが使用されます。

-o *OUTPUT_FILE*, --output *OUTPUT_FILE*

変換後の設定ファイルを書き込むパスを指定します。デフォルトでは、出力は STDOUT に書き出されます。

例:

```
$ pcluster-config convert -t alpha -o ~/.parallelcluster/multiinstance
```

~/.parallelcluster/config の [cluster alpha] セクションで指定されたクラスター構成を変換し、変換後の構成ファイルを ~/.parallelcluster/multiinstance に書き込みます。

設定

デフォルトでは、 はすべての設定パラメータに `~/.parallelcluster/config` ファイル AWS ParallelCluster を使用します。`-c` または `--config` コマンドラインオプション、または `AWS_PCLUSTER_CONFIG_FILE` 環境変数を使用することで、カスタムコンフィグレーションファイルを指定することができます。

サンプル設定ファイルは、 の Python ディレクトリ AWS ParallelCluster に と共にインストールされます`site-packages/aws-parallelcluster/examples/config`。設定ファイルの例も GitHub (https://github.com/aws/aws-parallelcluster/blob/v2.11.9/cli/src/pcluster/examples/config) で入手できます。

Current AWS ParallelCluster 2 バージョン: 2.11.9。

トピック

- [レイアウト]
- [global] セクション
- [aws] セクション
- [aliases] セクション
- [cluster] セクション
- [compute_resource] セクション
- [cw_log] セクション
- [dashboard] セクション
- [dcv] セクション
- [ebs] セクション
- [efs] セクション
- [fsx] セクション
- [queue] セクション
- [raid] セクション
- [scaling] セクション
- [vpc] セクション
- 例

[レイアウト]

AWS ParallelCluster 設定は複数のセクションで定義されます。

以下のセクションが必要です。[global] セクションおよび [aws] セクション。

少なくとも、[cluster] セクションおよび [vpc] セクションを 1 つずつ含める必要があります。

セクションは、角括弧で囲まれたセクション名で始まり、パラメータと設定が続きます。

```
[global]
cluster_template = default
update_check = true
sanity_check = true
```

[global] セクション

pcluster に関連するグローバル設定オプションを指定します。

```
[global]
```

トピック

- cluster_template
- update_check
- sanity_check

cluster_template

デフォルトでクラスターに使用される cluster セクションの名前を定義します。cluster セクションの詳細については、「[cluster] セクション」を参照してください。クラスター名は、英字で始まり、60 文字以下で、英字、数字、ハイフン (-) のみで構成される必要があります。

例えば、[cluster default] で始まるセクションを指定する次の設定がデフォルトで使用されます。

```
cluster_template = default
```

更新ポリシー: この設定は、更新中には分析されません。

update_check

(オプション) pcluster の更新をチェックします。

デフォルト値は true です。

```
update_check = true
```

更新ポリシー: この設定は、更新中には分析されません。

sanity_check

(オプション) クラスターパラメーターで定義されているリソースの構成の検証を試行します。

デフォルト値は true です。

> **Warning**
> sanity_check を false に設定した場合、重要なチェックは省略されます。これにより、構成が意図したとおりに機能しなくなる可能性があります。

```
sanity_check = true
```

> **Note**
> AWS ParallelCluster バージョン 2.5.0 より前のバージョンでは、 はsanity_checkデフォルトで になりましたfalse。

更新ポリシー: この設定は、更新中には分析されません。

[aws] セクション

（オプション）を選択するために使用されます AWS リージョン。

クラスターの作成では、この優先順位を使用して、新しいクラスター AWS リージョン の を選択します。

1. `-r` または `--region` パラメータを `pcluster create` に渡す。
2. `AWS_DEFAULT_REGION` 環境変数
3. `aws_region_name` 設定ファイルの AWS ParallelCluster `[aws]`セクションで を設定する (デフォルトの場所は です) `~/.parallelcluster/config`。`pcluster configure` コマンドで更新される場所です。
4. `region` 設定ファイル (`~/.aws/config`.) AWS CLI の `[default]`セクションで を設定する

> **Note**
>
> AWS ParallelCluster バージョン 2.10.0 以前は、これらの設定は必須であり、すべてのクラスターに適用されていました。

認証情報を保存するには、環境、IAM ロールの Amazon EC2 または AWS CLI を使用します。 AWS ParallelCluster 設定ファイルに認証情報を保存することはありません。

```
[aws]
aws_region_name = Region
```

更新ポリシー: この設定は、更新中には分析されません。

[aliases] セクション

エイリアスを指定することで、ssh コマンドをカスタマイズできるようになります。

次のデフォルト設定を書き留めます。

- `CFN_USER` には、オペレーティングシステムのデフォルトのユーザー名が設定されます
- `MASTER_IP` には、ヘッドノードの IP アドレスが設定されます
- `ARGS` には、*`pcluster ssh cluster_name`* の後にユーザーが提供する任意の引数が設定されます

```
[aliases]
```

```
# This is the aliases section, you can configure
# ssh alias here
ssh = ssh {CFN_USER}@{MASTER_IP} {ARGS}
```

更新ポリシー: この設定は、更新中には分析されません。

[cluster] セクション

クラスターを作成するために使用できるクラスターテンプレートを定義します。設定ファイルには、複数の [cluster] セクションを含めることができます。

同じクラスターテンプレートを使用して、複数のクラスターを作成できます。

形式は [cluster *cluster-template-name*] です。[global] セクションの cluster_template 設定によって名付けられた[cluster] セクションがデフォルトで使用されますが、pcluster コマンドラインでオーバーライドすることができます。

cluster-template-name は、英字で始まり、30 文字以下で、英字、数字、ハイフン (-)、アンダーライン (_) のみで構成される必要があります。

```
[cluster default]
```

トピック

- additional_cfn_template
- additional_iam_policies
- base_os
- cluster_resource_bucket
- cluster_type
- compute_instance_type
- compute_root_volume_size
- custom_ami
- cw_log_settings
- dashboard_settings
- dcv_settings
- desired_vcpus
- disable_cluster_dns

- disable_hyperthreading
- ebs_settings
- ec2_iam_role
- efs_settings
- enable_efa
- enable_efa_gdr
- enable_intel_hpc_platform
- encrypted_ephemeral
- ephemeral_dir
- extra_json
- fsx_settings
- iam_lambda_role
- initial_queue_size
- key_name
- maintain_initial_size
- master_instance_type
- master_root_volume_size
- max_queue_size
- max_vcpus
- min_vcpus
- placement
- placement_group
- post_install
- post_install_args
- pre_install
- pre_install_args
- proxy_server
- queue_settings
- raid_settings
- s3_read_resource

- s3_read_write_resource
- scaling_settings
- scheduler
- shared_dir
- spot_bid_percentage
- spot_price
- tags
- template_url
- vpc_settings

additional_cfn_template

（オプション）クラスターとともに起動する追加の AWS CloudFormation テンプレートを定義します。この追加のテンプレートは、クラスターの外部にありながらクラスターのライフサイクルの一部であるリソースを作成するために使用されます。

この値は、すべてのパラメータが指定されているパブリックテンプレートへの HTTP URL である必要があります。

デフォルト値はありません。

```
additional_cfn_template = https://<bucket-name>.s3.amazonaws.com/my-cfn-template.yaml
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

additional_iam_policies

(オプション) Amazon EC2 の IAM ポリシーの Amazon リソースネーム (ARN) のリストを指定します。このリストは、カンマで区切られた AWS ParallelCluster で必要なアクセス権限に加えて、クラスターで使用されるルートロールにアタッチされます。IAM ポリシー名とその ARN が異なります。名前を `additional_iam_policies` の引数として使用することはできません。

クラスターノードのデフォルト設定にポリシーを追加する場合は、`ec2_iam_role` 設定を使用して特定の EC2 ポリシーを追加する代わりに、追加のカスタム IAM ポリシーを `additional_iam_policies` 設定と一緒に渡すことをお勧めします。これは、`additional_iam_policies` が AWS ParallelCluster に必要なデフォルトのアクセス許可に追加さ

れるためです。既存の ec2_iam_role には必要なアクセス許可がすべて含まれている必要があります。ただし、必要とされるアクセス許可は機能が追加されるたびに変更されることが多いため、既存の ec2_iam_role は古くなる可能性があります。

デフォルト値はありません。

```
additional_iam_policies = arn:aws:iam::123456789012:policy/CustomEC2Policy
```

> **Note**
>
> AWS ParallelCluster バージョン 2.5.0 で additional_iam_policies をサポートしました。

更新ポリシー: この設定は、更新中に変更できます。

base_os

(必須) クラスターで使用する OS タイプを指定します。

使用できるオプションは、以下のとおりです。

- alinux2
- centos7
- ubuntu1804
- ubuntu2004

> **Note**
>
> Graviton AWS ベースのインスタンスでは、alinux2、ubuntu1804、または のみがサポートubuntu2004されています。

> **Note**
>
> AWS ParallelCluster バージョン 2.11.4 で のサポートcentos8が削除されました。 AWS ParallelCluster バージョン 2.11.0 で ubuntu2004 のサポートが追加され、alinux お

よび ubuntu1604 のサポートが削除されました。のサポートcentos8が追加され、 のサポートcentos6が AWS ParallelCluster バージョン 2.10.0 で削除されました。 AWS ParallelCluster バージョン 2.6.0 で alinux2 をサポートしました。ubuntu1804 のサポートが追加され、ubuntu1404 のサポートが AWS ParallelCluster バージョン 2.5.0 で削除されました。

次の表で AWS リージョン 説明されている、 をサポートしていない特定の 以外centos7。他のすべての AWS 商用リージョンは、以下のオペレーティングシステムをすべてサポートしています。

パーティション (AWS リージョン)	**alinux2**	**centos7**	**ubuntu1804 および ubuntu2004**
商用 (特に記載 AWS リージョン されていないものすべて)	真	True	真
AWS GovCloud (米国東部) (us-gov-east-1)	真	False	真
AWS GovCloud (米国西部) (us-gov-west-1)	真	False	真
中国 (北京) (cn-north-1)	真	False	真
中国 (寧夏) (cn-northwest-1)	真	False	真

> **Note**
>
> base_os パラメータは、クラスターへのログインに使用されるユーザー名も決定します。

- centos7: centos
- ubuntu1804 および ubuntu2004: ubuntu
- alinux2: ec2-user

> **Note**
>
> AWS ParallelCluster バージョン 2.7.0 以前は、 base_osパラメータはオプションで、デフォルトは でしたalinux。 AWS ParallelCluster バージョン 2.7.0 以降は、base_os パラメータが必須です。

> **Note**
>
> scheduler パラメータが awsbatch の場合、alinux2 のみがサポートされます。

```
base_os = alinux2
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

cluster_resource_bucket

(オプション) クラスター作成時に生成されるリソースをホストするために使用される Amazon S3 バケットの名前を指定します。バケットはバージョニングを有効にしておく必要があります。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「Using versioning」(バージョニングの使用) を参照してください。このバケットは、複数のクラスターに使用することができます。バケットはクラスターと同じリージョンにあることが必要です。

このパラメータが指定されていない場合、クラスター作成時に新しいバケットが作成されます。新しいバケツは parallelcluster-*random_string* という名前になります。この名前では、*random_string* は英数字からなるランダムな文字列です。すべてのクラスターリソースは、 という形式のパスでこのバケットに保存されます*bucket_name*/*resource_directory*。 resource_directoryには という形式があります。ここで *stack_name*-*random_string*stack_name は、 で使用されるいずれかの AWS CloudFormation スタックの名前です AWS ParallelCluster。 *bucket_name* の値は、parallelcluster-*clustername* スタックの出力にある ResourcesS3Bucket 値で確認できます。*resource_directory* の値は、同じスタックから出力される ArtifactS3RootDirectory の値で確認することができます。

デフォルト値は parallelcluster-*random_string* です。

```
cluster_resource_bucket = amzn-s3-demo-bucket
```

> **Note**
> AWS ParallelCluster バージョン 2.10.0 で のサポートが追加されcluster_resource_bucketました。

更新ポリシー: この設定が変更された場合、更新は許可されません。この設定を強制的に更新することはできません。

cluster_type

(オプション) 起動するクラスターのタイプを定義します。queue_settings の設定が定義されている場合、この設定は [queue] セクションの compute_type の設定に置き換える必要があります。

有効なオプションは、ondemand と spot です。

デフォルト値は ondemand です。

スポットインスタンスの詳細については、「スポットインスタンスの操作」を参照してください。

> **Note**
> スポットインスタンスを使用するには、お客様のアカウントに AWSServiceRoleForEC2Spot サービスにリンクしたロールが必要です。を使用してアカウントにこのロールを作成するには AWS CLI、次のコマンドを実行します。
>
> ```
> aws iam create-service-linked-role --aws-service-name spot.amazonaws.com
> ```
>
> 詳細については、「Amazon EC2 ユーザーガイド」の「スポットインスタンスリクエスト向けのサービスにリンクされたロール」を参照してください。

```
cluster_type = ondemand
```

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

compute_instance_type

(オプション) クラスターコンピューティングノードに使用される Amazon EC2 インスタンスタイプを定義します。インスタンスタイプのアーキテクチャは、master_instance_type 設定に使用されるアーキテクチャと同じでなければなりません。queue_settings の設定が定義されている場合、この設定は [compute_resource] セクションの instance_type の設定に置き換える必要があります。

スawsbatchケジューラを使用している場合は、UI AWS Batch のコンピューティング環境の作成で、サポートされているインスタンスタイプのリストを参照してください。

スケジューラが awsbatch の場合、デフォルトは t2.micro、optimal です。

```
compute_instance_type = t2.micro
```

> **Note**
>
> AWS ParallelCluster バージョン AWS 2.8.0 では、Graviton ベースのインスタンス (A1 および C6gインスタンスを含む) のサポートが追加されました。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

compute_root_volume_size

(オプション) ComputeFleet のルートボリュームのサイズをジビバイト (GiB) 単位で指定します。AMI が growroot をサポートしている必要があります。

デフォルト値は 35 です。

> **Note**
>
> 2.5.0 から 2.10.4 までの AWS ParallelCluster バージョンでは、デフォルトは 25 でした。AWS ParallelCluster バージョン 2.5.0 以前は、デフォルトは 20 でした。

```
compute_root_volume_size = 35
```

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

custom_ami

(オプション) デフォルトの公開された AMI の代わりにヘッドノードおよびコンピューティングノードに使用するカスタム AMI の ID を指定します。詳細については、AMI を変更するまたはカスタム AMI AWS ParallelCluster の構築を参照してください。

デフォルト値はありません。

```
custom_ami = ami-00d4efc81188687a0
```

カスタム AMI の起動に追加のアクセス許可が必要な場合は、これらのアクセス許可をユーザーポリシーとヘッドノードポリシーの両方に追加する必要があります。

例えば、カスタム AMI に暗号化されたスナップショットが関連付けられている場合、ユーザーおよびヘッドノードポリシーの両方に次の追加のポリシーが必要です。

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
           "Effect": "Allow",
           "Action": [
               "kms:DescribeKey",
               "kms:ReEncrypt*",
               "kms:CreateGrant",
               "kms:Decrypt"
           ],
           "Resource": [
               "arn:aws:kms:<AWS_REGION>:<AWS_ACCOUNT_ID>:key/<AWS_KMS_KEY_ID>"
           ]
       }
   ]
}
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

cw_log_settings

(オプション) CloudWatch Logs の設定を持つ [cw_log] セクションを特定します。セクション名は、英字で始まり、30 文字以下で、英字、数字、ハイフン (-)、アンダーライン (_) のみで構成される必要があります。

詳しくは、「[cw_log] セクション」、「Amazon CloudWatch ダッシュボード」、「Amazon CloudWatch Logs との統合」を参照してください。

例えば、[cw_log custom-cw] で始まるセクションを指定する次の設定が CloudWatch Logs 設定で使用されます。

```
cw_log_settings = custom-cw
```

> **Note**
>
> AWS ParallelCluster バージョン 2.6.0 で のサポートが追加されcw_log_settingsました。

更新ポリシー: この設定が変更された場合、更新は許可されません。

dashboard_settings

(オプション) CloudWatch ダッシュボードの設定を持つ [dashboard] セクションを特定します。セクション名は、英字で始まり、30 文字以下で、英字、数字、ハイフン (-)、アンダーライン (_) のみで構成される必要があります。

詳細については、「[dashboard] セクション」を参照してください。

例えば、[dashboard custom-dashboard で始まるセクションを指定する次の設定が CloudWatch ダッシュボード 設定で使用されます。

```
dashboard_settings = custom-dashboard
```

> **Note**
>
> AWS ParallelCluster バージョン 2.10.0 で のサポートが追加されdashboard_settingsました。

更新ポリシー: この設定は、更新中に変更できます。

dcv_settings

(オプション) Amazon DCV 設定がある [dcv] セクションを識別します。セクション名は、英字で始まり、30 文字以下で、英字、数字、ハイフン (-)、アンダーライン (_) のみで構成される必要があります。

詳細については、「[dcv] セクション」を参照してください。

例えば、次の設定は、[dcv custom-dcv] を開始するセクションを Amazon DCV 設定で使用することを指定します。

```
dcv_settings = custom-dcv
```

> **Note**
>
> Graviton AWS ベースのインスタンスでは、Amazon DCV は でのみサポートされていますalinux2。

> **Note**
>
> AWS ParallelCluster バージョン 2.5.0 で のサポートが追加されdcv_settingsました。

更新ポリシー: この設定が変更された場合、更新は許可されません。

desired_vcpus

(オプション) コンピューティング環境で必要な vCPU 数を指定します。スケジューラが awsbatch の場合にのみ使用します。

デフォルト値は 4 です。

```
desired_vcpus = 4
```

更新ポリシー: この設定は、更新中には分析されません。

disable_cluster_dns

(オプション) クラスターの DNS エントリを作成しないようにするかどうかを指定します。デフォルトでは、 は Route 53 ホストゾーン AWS ParallelCluster を作成します。`disable_cluster_dns` が `true` に設定されている場合、ホストゾーンは作成されません。

デフォルト値は `false` です。

```
disable_cluster_dns = true
```

> **Warning**
> クラスターが正常に動作するためには、名前解決システムが必要です。`disable_cluster_dns` を `true` に設定した場合、追加の名前解決システムも提供する必要があります。

> **Important**
> `disable_cluster_dns` = `true` は、`queue_settings` 設定が指定された場合のみサポートされます。

> **Note**
> AWS ParallelCluster バージョン 2.9.1 で のサポートが追加され`disable_cluster_dns`ました。

更新ポリシー: この設定が変更された場合、更新は許可されません。

disable_hyperthreading

(Optional) ヘッドノードとコンピューティングノードでハイパースレッディングを無効にします。すべてのインスタンスタイプがハイパースレッディングを無効にできるわけではありません。ハイパースレッディングの無効化をサポートするインスタンスタイプの一覧については、「Amazon EC2 ユーザーガイド」の「インスタンスタイプ別の CPU コア数と CPU コアごとのスレッド数」を参照

してください。queue_settings の設定が定義されている場合、この設定を定義するか、[queue] セクションの disable_hyperthreading の設定を定義することができます。

デフォルト値は false です。

```
disable_hyperthreading = true
```

> **Note**
>
> disable_hyperthreading は scheduler = awsbatch の時、ヘッドノードにのみ影響します。

> **Note**
>
> AWS ParallelCluster バージョン 2.5.0 で disable_hyperthreading をサポートしました。

更新ポリシー: この設定が変更された場合、更新は許可されません。

ebs_settings

(オプション) ヘッドノードにマウントされている Amazon EBS ボリュームを持つ [ebs] セクションを識別します。複数の Amazon EBS ボリュームを使用する場合、これらのパラメータをそれぞれカンマで区切ってリストで入力してください。セクション名は、英字で始まり、30 文字以下で、英字、数字、ハイフン (-)、アンダーライン (_) のみで構成される必要があります。

最大 5 つの追加の Amazon EBS ボリュームがサポートされます。

詳細については、「[ebs] セクション」を参照してください。

例えば、以下の設定では、[ebs custom1] と [ebs custom2] で始まるセクションが Amazon EBS ボリューム使用されるように指定します。

```
ebs_settings = custom1, custom2
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

ec2_iam_role

(オプション) クラスター内のすべてのインスタンスにアタッチされる Amazon EC2 の既存の IAM ロールの名前を定義します。IAM ロール名とその Amazon リソースネーム (ARN) は異なります。ARN を `ec2_iam_role` の引数として使用することはできません 。

このオプションを指定すると、`additional_iam_policies` 設定は無視されます。クラスターノードのデフォルト設定にポリシーを追加する場合は、`ec2_iam_role` 設定を使用する代わりに、追加のカスタム IAM ポリシーを `additional_iam_policies` 設定と一緒に渡すことをお勧めします。

このオプションを指定しない場合、Amazon EC2 AWS ParallelCluster のデフォルトの IAM ロールが使用されます。詳細については、「AWS Identity and Access Management での ロール AWS ParallelCluster」を参照してください。

デフォルト値はありません。

```
ec2_iam_role = ParallelClusterInstanceRole
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

efs_settings

(オプション) Amazon EFS ファイルシステムに関する設定を指定します。セクション名は、英字で始まり、30 文字以下で、英字、数字、ハイフン (-)、アンダーライン (_) のみで構成される必要があります。

詳細については、「`[efs]` セクション」を参照してください。

例えば、`[efs customfs]` で始まるセクションを指定する次の設定が Amazon EFS ファイルシステム設定に使用されます。

```
efs_settings = customfs
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

enable_efa

(オプション) これが存在する場合、コンピューターノードで Elastic Fabric Adapter (EFA) を有効化することを指定します。EFA をサポートする EC2 インスタンスのリストを表示するに

は、「Linux インスタンス用の Amazon EC2 ユーザーガイド」の「サポートされるインスタンスタイプ」を参照してください。詳細については、「Elastic Fabric Adapter」を参照してください。queue_settings の設定が定義されている場合、この設定を定義するか、[queue] のセクションの enable_efa の設定を定義することができます。クラスタープレイスメントグループは、インスタンス間のレイテンシーを最小限に抑えるために使用する必要があります。詳細については、「placement」および「placement_group」を参照してください。

```
enable_efa = compute
```

> **Note**
>
> Arm ベースの Graviton2 インスタンスでの EFA のサポートが AWS ParallelCluster バージョン 2.10.1 に追加されました。

更新ポリシー: この設定が変更された場合、更新は許可されません。

enable_efa_gdr

（オプション）AWS ParallelCluster バージョン 2.11.3 以降、この設定は効果がありません。Elastic Fabric Adapter (EFA) の GPUDirect RDMA (リモートディレクトメモリーアクセス) のサポートは、インスタンスタイプとオペレーティングシステムの両方でサポートされていれば、常に有効です。

> **Note**
>
> AWS ParallelCluster バージョン 2.10.0 から 2.11.2: の場合compute、コンピューティングノードで GPUDirect RDMA (リモートダイレクトメモリアクセス) の Elastic Fabric Adapter (EFA) サポートが有効になっていることを指定します。この設定を compute にするためには、enable_efa の設定が compute に設定されている必要があります。GPUDirect RDMA の EFA は、特定のオペレーティングシステム (base_os は alinux2、centos7、ubuntu1804、または ubuntu2004) 上の特定のインスタンスタイプ (p4d.24xlarge) でサポートされています。queue_settings の設定が定義されている場合、この設定を定義するか、[queue]のセクションの enable_efa_gdr の設定を定義することができます。クラスタープレイスメントグループは、インスタンス間のレイテンシーを最小限に抑えるために使用する必要があります。詳細については、「placement」および「placement_group」を参照してください。

```
enable_efa_gdr = compute
```

> **Note**
>
> AWS ParallelCluster バージョン 2.10.0 で のサポートが追加されenable_efa_gdrました。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

enable_intel_hpc_platform

(オプション) 存在する場合は、インテル Parallel Studio のエンドユーザー使用許諾契約書が承認されたことを示します。これにより、インテル Parallel Studio がマスターノードにインストールされ、コンピューティングノードと共有されます。これにより、ヘッドノードのブートストラップに要する時間が数分長くなります。enable_intel_hpc_platform 設定は CentOS 7 (base_os = centos7) でのみサポートされています。

デフォルト値は false です。

```
enable_intel_hpc_platform = true
```

> **Note**
>
> enable_intel_hpc_platform パラメータは Graviton AWS ベースのインスタンスと互換性がありません。

> **Note**
>
> AWS ParallelCluster バージョン 2.5.0 で enable_intel_hpc_platform をサポートしました。

更新ポリシー: この設定が変更された場合、更新は許可されません。

encrypted_ephemeral

(オプション) LUKS (Linux Unified Key Setup) を使用して、回復不能なメモリ内キーにより一時的なインスタンスストアボリュームを暗号化します。

詳細については、「https://gitlab.com/cryptsetup/cryptsetup/blob/master/README.md」を参照してください。

デフォルト値は `false` です。

```
encrypted_ephemeral = true
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

ephemeral_dir

(オプション) インスタンスストアボリュームをマウントするパスを定義します (使用されている場合)。

デフォルト値は `/scratch` です。

```
ephemeral_dir = /scratch
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

extra_json

(オプション) Chef `dna.json` にマージされる追加の JSON を定義します。詳細については、「カスタム AMI AWS ParallelCluster の構築」を参照してください。

デフォルト値は `{}` です。

```
extra_json = {}
```

> **Note**
>
> AWS ParallelCluster バージョン 2.6.1 以降では、ノードの起動時にほとんどのインストールレシピがデフォルトでスキップされ、起動時間が短縮されます。起動時間の短縮よりも下位互換性の向上を優先させるために、すべてのインストールレシピを実行するに

は、extra_json 設定の cluster キーに "skip_install_recipes" : "no" を追加します。例:

```
extra_json = { "cluster" : { "skip_install_recipes" : "no" } }
```

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

fsx_settings

(オプション) FSx for Lustre 設定を定義するセクションを指定します。セクション名は、英字で始まり、30 文字以下で、英字、数字、ハイフン (-)、アンダーライン (_) のみで構成される必要があります。

詳細については、「[fsx] セクション」を参照してください。

例えば、[fsx fs] で始まるセクションを指定する次の設定が FSx for Lustre 設定で使用されます。

```
fsx_settings = fs
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

iam_lambda_role

（オプション）既存の AWS Lambda 実行ロールの名前を定義します。このロールは、クラスター内のすべての Lambda 関数にアタッチされます。詳細については、「AWS Lambda デベロッパーガイド」の「AWS Lambda execution role」を参照してください。

> **Note**
>
> バージョン 2.11.5 AWS ParallelCluster 以降、 は SGEまたは スTorqueケジューラの使用をサポートしていません。

IAM ロール名とその Amazon リソースネーム (ARN) は異なります。ARN を iam_lambda_role の引数として使用することはできません 。ec2_iam_role および iam_lambda_role の両方が定義されており、scheduler が sge、slurm または torque の場合、ロールは作成されません。scheduler が awsbatch であれば、pcluster start の

間にロールが作成されます。ポリシーの例ついては、「SGE、Slurm または Torque を使用する ParallelClusterLambdaPolicy」および「awsbatch を使用する ParallelClusterLambdaPolicy」を参照してください。

デフォルト値はありません。

```
iam_lambda_role = ParallelClusterLambdaRole
```

> **Note**
>
> AWS ParallelCluster バージョン 2.10.1 で のサポートが追加されiam_lambda_roleました。

更新ポリシー: この設定は、更新中に変更できます。

initial_queue_size

(オプション) クラスターでコンピューティングノードとして起動する Amazon EC2 インスタンスの最初の数を設定します。queue_settings の設定が定義されている場合、この設定は削除され、[compute_resource] セクションの initial_count の設定に置き換わります。

> **Note**
>
> バージョン 2.11.5 AWS ParallelCluster 以降、 は SGEまたは スTorqueケジューラの使用をサポートしていません。

この設定は、従来のスケジューラ (SGE、Slurm、Torque) にのみ適用されます。maintain_initial_size 設定が true の場合、initial_queue_size 設定は 1 以上でなければなりません。

スケジューラが awsbatch の場合は、min_vcpus を使用します。

デフォルトは 2 です。

```
initial_queue_size = 2
```

更新ポリシー: この設定は、更新中に変更できます。

key_name

(オプション) インスタンスへの SSH アクセスを有効にするための既存の Amazon EC2 キーペアを指定します。

```
key_name = mykey
```

> **Note**
> AWS ParallelCluster バージョン 2.11.0 以前は、 は必須の設定key_nameでした。

更新ポリシー: この設定が変更された場合、更新は許可されません。

maintain_initial_size

> **Note**
> バージョン 2.11.5 AWS ParallelCluster 以降、 は SGEまたは スTorqueケジューラの使用をサポートしていません。

(Optional) 従来のスケジューラ (SGE、Slurm および Torque) の Auto Scaling グループの最初のサイズを維持します。

スケジューラが `awsbatch` の場合は、`desired_vcpus` を使用します。

この設定は、ブーリアン型フラグです。`true` に設定した場合、Auto Scaling グループのメンバーは `initial_queue_size` の値より少なくなることはなく、`initial_queue_size` の値は 1 以上でなければなりません。クラスターは `max_queue_size` の値にスケールアップできます。`cluster_type = spot` の場合、Auto Scaling グループはインスタンスを中断し、`initial_queue_size` サイズを下回る可能性があります。

`false` に設定した場合、Auto Scaling グループは、不要なときにリソースがアイドル状態で待機するのを回避するため、ゼロ (0) メンバーにスケールダウンできます。

`queue_settings` の設定が定義されている場合、この設定は削除され、`[compute_resource]` セクションの `initial_count` および `min_count` の設定に置き換わります。

デフォルトは false です。

```
maintain_initial_size = false
```

更新ポリシー: この設定は、更新中に変更できます。

master_instance_type

(オプション) ヘッドノードに使用される Amazon EC2 インスタンスタイプを定義します。インスタンスタイプのアーキテクチャは、compute_instance_type 設定に使用されるアーキテクチャと同じでなければなりません。

無料利用枠 AWS リージョン がある では、 はデフォルトで無料利用枠インスタンスタイプ (t2.micro または) になりますt3.micro。無料利用枠 AWS リージョン がない では、 はデフォルトで になりますt3.micro。 AWS 無料利用枠の詳細については、AWS 「 無料利用枠に関する FAQs」を参照してください。

```
master_instance_type = t2.micro
```

> **Note**
>
> AWS ParallelCluster バージョン 2.10.1 より前では、 はデフォルトですべて t2.micro に設定されています AWS リージョン。 AWS ParallelCluster バージョン 2.10.0 では、ヘッドノードで p4d.24xlarge がサポートされていませんでした。 AWS ParallelCluster バージョン AWS 2.8.0 で Graviton ベースのインスタンス (A1や などC6g) のサポートが追加されました。

更新ポリシー: この設定が変更された場合、更新は許可されません。

master_root_volume_size

(オプション) ヘッドノードルートボリュームサイズをジビバイト (GiB) 単位で指定します。AMI が growroot をサポートしている必要があります。

デフォルト値は 35 です。

> **Note**
> 2.5.0 から 2.10.4 までの AWS ParallelCluster バージョンでは、デフォルトは 25 でした。AWS ParallelCluster バージョン 2.5.0 以前は、デフォルトは 20 でした。

```
master_root_volume_size = 35
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

max_queue_size

(オプション) クラスターで起動可能な最大数の Amazon EC2 インスタンスの最大数を設定します。queue_settings の設定が定義されている場合、この設定は削除され、[compute_resource] セクションの max_count の設定に置き換わります。

> **Note**
> バージョン 2.11.5 AWS ParallelCluster 以降、 は SGEまたは スTorqueケジューラの使用をサポートしていません。

この設定は、従来のスケジューラ (SGE、Slurm、Torque) にのみ適用されます。

スケジューラが awsbatch の場合は、max_vcpus を使用します。

デフォルトは 10 です。

```
max_queue_size = 10
```

更新ポリシー: この設定はアップデート中に変更することができますが、値が小さくなった場合はコンピューティングフリートを停止する必要があります。そうでなければ、既存のノードを終了させることができます。

max_vcpus

(オプション) コンピューティング環境での vCPU の最大数を指定します。スケジューラが awsbatch の場合にのみ使用します。

デフォルト値は 20 です。

```
max_vcpus = 20
```

更新ポリシー: これらの設定は、更新中に減らすことはできません。

min_vcpus

(オプション) awsbatch スケジューラの Auto Scaling グループの最初のサイズを維持します。

> **Note**
> バージョン 2.11.5 AWS ParallelCluster 以降、 は SGEまたは スTorqueケジューラの使用をサポートしていません。

スケジューラが SGE、Slurm、または Torque の場合は、maintain_initial_size を使用します。

コンピューティング環境のメンバー数が min_vcpus の値よりも少なくなることはありません。

デフォルトは 0 です。

```
min_vcpus = 0
```

更新ポリシー: この設定は、更新中に変更できます。

placement

(オプション) クラスターのプレイスメントグループのロジックを定義します。これにより、クラスター全体、またはクラスタープレイスメントグループを使用するコンピューティングインスタンスのみが有効になります。

queue_settings の設定が定義されている場合、この設定は削除され、[queue] セクションの placement_group の設定に置き換わります。異なるインスタンスタイプを同じプレイスメントグループで使用している場合、容量不足のエラーでリクエストが失敗する可能性が高くなります。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンス容量の不足」を参照してください。複数のキューがプレイスメントグループを共有できるのは、事前に作成され、各キューの placement_group 設定で構成された場合だけです。各 [queue] セクションが placement_group の設定を定義している場合、ヘッドノードはキューのプレイスメントグループに入ることができません。

有効なオプションは cluster または compute です。

このパラメータは、スケジューラが awsbatch の場合には使用されません。

デフォルト値は compute です。

```
placement = compute
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

placement_group

(オプション) クラスタープレイスメントグループを定義します。queue_settings の設定が定義されている場合、この設定は削除され、[queue] セクションの placement_group の設定に置き換わります。

有効なオプションは、次の値です。

- DYNAMIC
- 既存の Amazon EC2 クラスタープレイスメントグループ名

DYNAMIC に設定されている場合、一意のプレイスメントグループはクラスタースタックの一部として作成され、削除されます。

このパラメータは、スケジューラが awsbatch の場合には使用されません。

プレイスメントグループの詳細については、「Amazon EC2 ユーザーガイド」の「プレイスメントグループ」を参照してください。異なるインスタンスタイプを同じプレイスメントグループで使用している場合、容量不足のエラーでリクエストが失敗する可能性が高くなります。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンス容量の不足」を参照してください。

デフォルト値はありません。

すべてのインスタンスタイプがクラスタープレイスメントグループをサポートしているわけではありません。例えば、デフォルトのインスタンスタイプでは、クラスタープレイスメントグループは t3.micro によってサポートされません。クラスターのプレイスメントグループをサポートするインスタンスタイプのリストについては、「Amazon EC2 ユーザーガイド」の「クラスタープレイスメントグループのルールと制限」を参照してください。プレイスメントグループを使用する際のヒントについては、「プレイスメントグループとインスタンスの起動に関する問題」を参照してください。

```
placement_group = DYNAMIC
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

post_install

(オプション) すべてのノードブートストラップアクションが完了した後に実行されるポストインストールスクリプトへの URL を指定します。詳細については、「カスタムブートストラップアクション」を参照してください。

awsbatch をスケジューラとして使用する場合、ポストインストールスクリプトはヘッドノードでのみ実行されます。

パラメータの形式には、「http://*hostname/path/to/script.sh*」または「s3://*bucket-name/path/to/script.sh*」を指定できます。

デフォルト値はありません。

```
post_install = s3://<bucket-name>/my-post-install-script.sh
```

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

post_install_args

(オプション) ポストインストールスクリプトに渡される引用符で囲んだ引数のリストを指定します。

デフォルト値はありません。

```
post_install_args = "argument-1 argument-2"
```

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

pre_install

(オプション) ノードデプロイブートストラップアクションが開始される前に実行されるプレインストールスクリプトへの URL を指定します。詳細については、「カスタムブートストラップアクション」を参照してください。

awsbatch をスケジューラとして使用する場合、プレインストールスクリプトはマスターノードでのみ実行されます。

パラメータの形式には、「http://*hostname/path/to/script.sh*」または「s3://*bucket-name/path/to/script.sh*」を指定できます。

デフォルト値はありません。

```
pre_install = s3://bucket-name/my-pre-install-script.sh
```

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

pre_install_args

(オプション) プレインストールスクリプトに渡される引用符で囲んだ引数のリストを指定します。

デフォルト値はありません。

```
pre_install_args = "argument-3 argument-4"
```

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

proxy_server

(オプション) HTTP または HTTPS プロキシサーバーを定義します。通常は、http://*x.x.x.x:8080* です。

デフォルト値はありません。

```
proxy_server = http://10.11.12.13:8080
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

queue_settings

（オプション）クラスターが同種コンピューティングフリートの代わりにキューを使用するように指定し、どの[queue]セクションを使用するかを指定します。最初に表示された [queue] セクションは、デフォルトのスケジューラーキューです。queue セクション名は、英字の小文字で始まり、30 文字以下で、小文字の英字、数字、ハイフン (-) のみで構成される必要があります。

> **Important**
>
> queue_settings は、scheduler が slurm に設定されている場合にのみサポートされます。cluster_type、compute_instance_type、initial_queue_size、maintain_initial_ および spot_price の設定を指定してはいけません。disable_hyperthreading および enable_efa の設定は、[cluster] セクションか [queue] セクションのどちらかに指定できますが、両方に指定することはできません。

最大で 5 つの [queue] セクションに対応しています。

詳細については、「[queue] セクション」を参照してください。

例えば、以下の設定では、[queue q1] と [queue q2] で始まるセクションが使用されるように指定します。

```
queue_settings = q1, q2
```

> **Note**
>
> AWS ParallelCluster バージョン 2.9.0 で のサポートが追加されqueue_settingsました。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

raid_settings

(オプション) Amazon EBS ボリューム RAID 設定で [raid] セクションを識別します。セクション名は、英字で始まり、30 文字以下で、英字、数字、ハイフン (-)、アンダーライン (_) のみで構成される必要があります。

詳細については、「[raid] セクション」を参照してください。

例えば、[raid rs] で始まるセクションを指定する次の設定がオートスケーリング設定で使用されます。

```
raid_settings = rs
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

s3_read_resource

（オプション）AWS ParallelCluster ノードに読み取り専用アクセスを付与する Amazon S3 リソースを指定します。

例えば、`arn:aws:s3:::my_corporate_bucket*` は `my_corporate_bucket` というバケットと、そのバケット内のオブジェクトへの読み取り専用のアクセスを提供します。

形式の詳細については、「working with Amazon S3」(Amazon S3 の使用) を参照してください。

デフォルト値はありません。

```
s3_read_resource = arn:aws:s3:::my_corporate_bucket*
```

更新ポリシー: この設定は、更新中に変更できます。

s3_read_write_resource

(オプション) AWS ParallelCluster ノードに読み取りおよび書き込みアクセスを付与する Amazon S3 リソースを指定します。

例えば、`arn:aws:s3:::my_corporate_bucket/Development/*` は、`my_corporate_bucket` バケットの `Development` フォルダにあるすべてのオブジェクトに読み取り/書き込みアクセス許可を付与します。

形式の詳細については、「working with Amazon S3」(Amazon S3 の使用) を参照してください。

デフォルト値はありません。

```
s3_read_write_resource = arn:aws:s3:::my_corporate_bucket/*
```

更新ポリシー: この設定は、更新中に変更できます。

scaling_settings

オートスケーリングボリューム RAID 設定で `[scaling]` セクションを識別します。セクション名は、英字で始まり、30 文字以下で、英字、数字、ハイフン (-)、アンダーライン (_) のみで構成される必要があります。

詳細については、「[scaling] セクション」を参照してください。

例えば、[scaling custom] で始まるセクションを指定する次の設定がオートスケーリング設定で使用されます。

```
scaling_settings = custom
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

scheduler

(必須) クラスターのスケジューラを定義します。

有効なオプションは、次の値です。

awsbatch

AWS Batch

awsbatch スケジューラの詳細については、「ネットワークのセットアップ」および「AWS Batch (awsbatch)」を参照してください。

sge

> **Note**
> バージョン 2.11.5 AWS ParallelCluster 以降、 は SGEまたは スTorqueケジューラの使用をサポートしていません。

Son of Grid Engine (SGE)

slurm

Slurm Workload Manager (Slurm)

torque

> **Note**
> バージョン 2.11.5 AWS ParallelCluster 以降、 は SGEまたは スTorqueケジューラの使用をサポートしていません。

Torque Resource Manager (Torque)

> **Note**
>
> AWS ParallelCluster バージョン 2.7.0 以前は、 `scheduler`パラメータはオプションで、デフォルトは でした`sge`。 AWS ParallelCluster バージョン 2.7.0 以降では、 `scheduler`パラメータが必要です。

```
scheduler = slurm
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

shared_dir

(オプション) 共有 Amazon EBS ボリュームをマウントするパスを定義します。

このオプションを複数の Amazon EBS ボリュームと一緒に使用しないでください。代わりに、各 `[ebs]` セクションで `shared_dir` 値を指定します。

複数の Amazon EBS ボリュームを扱う場合の詳細は、「`[ebs]`」セクションを参照してください。

デフォルト値は `/shared` です。

`/myshared` の共有 Amazon EBS ボリュームを以下の例に示します。

```
shared_dir = myshared
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

spot_bid_percentage

(オプション) `awsbatch` がスケジューラである場合は、必要に応じて、ComputeFleet の最大スポット料金の計算に使用されるオンデマンド割合を設定します。

指定しない場合は、オンデマンド料金を上限として、現在のスポット市場価格が選択されます。

```
spot_bid_percentage = 85
```

更新ポリシー: この設定は、更新中に変更できます。

spot_price

> **Note**
> バージョン 2.11.5 AWS ParallelCluster 以降、 は SGEまたは スTorqueケジューラの使用をサポートしていません。

(オプション) 従来のスケジューラ (SGE、Slurm、および Torque) で必要に応じて ComputeFleet の最大スポット料金を設定します。`cluster_type` が `spot` に設定されている場合にのみ使用されます。値を指定しない場合、オンデマンド料金を上限とするスポット料金が課金されます。`queue_settings` の設定が定義されている場合、この設定は削除され、`[compute_resource] セクション`の `spot_price` の設定に置き換わります。

スケジューラが `awsbatch` の場合は、代わりに spot_bid_percentage を使用します。

ニーズに合ったスポットインスタンスを見つける方法については、「スポットインスタンスアドバイザー」を参照してください。

```
spot_price = 1.50
```

> **Note**
> AWS ParallelCluster バージョン 2.5.0 `spot_price`では、 `cluster_type = spot` が指定されていない場合、ComputeFleet のインスタンス起動は失敗します。これは AWS ParallelCluster バージョン 2.5.1 で修正されました。

更新ポリシー: この設定は、更新中に変更できます。

tags

（オプション）使用するタグを定義します AWS CloudFormation。

コマンドラインタグが --tags を介して指定されている場合は、設定タグとマージされます。

コマンドラインタグは、同じキーを持つタグ設定を上書きします。

タグは JSON 形式です。中括弧の外側に引用符を使用しないでください。

詳細については、「AWS CloudFormation ユーザーガイド」の「AWS CloudFormation リソースタグタイプ」を参照してください。

```
tags = {"key" : "value", "key2" : "value2"}
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

> **Note**
> 更新ポリシーでは、バージョン 2.8.0 からバージョン 2.9.1 までの AWS ParallelCluster の `tags` 設定の変更に対応していませんでした。
> バージョン 2.10.0 からバージョン 2.11.7 では、`tags` 設定の変更をサポートしている更新ポリシーのリストが正確ではありません。この設定を変更する際のクラスター更新はサポートされていません。

template_url

（オプション）クラスターの作成に使用される AWS CloudFormation テンプレートへのパスを定義します。

更新では、スタックの作成に使用した元のテンプレートが使用されます。

デフォルトは `https://aws_region_name-aws-parallelcluster.s3.amazonaws.com/templates/aws-parallelcluster-version.cfn.json` です。

> **Warning**
> これは上級者向けのパラメータです。この設定を変更する場合は、お客様の責任において行ってください。

```
template_url = https://us-east-1-aws-parallelcluster.s3.amazonaws.com/templates/aws-parallelcluster-2.11.9.cfn.json
```

更新ポリシー: この設定は、更新中には分析されません。

vpc_settings

(必須) クラスターがデプロイされる Amazon VPC 設定で [vpc] セクションを識別します。セクション名は、英字で始まり、30 文字以下で、英字、数字、ハイフン (-)、アンダーライン (_) のみで構成される必要があります。

詳細については、「[vpc] セクション」を参照してください。

例えば、[vpc public] で始まるセクションを指定する次の設定が Amazon VPC 設定で使用されます。

```
vpc_settings = public
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

[compute_resource] セクション

コンピューティングリソースのコンフィギュレーション設定を定義します。[compute_resource] のセクションは、[queue] セクション の compute_resource_settings の設定によって参照されます。[compute_resource] セクションは、scheduler が slurm に設定されている場合のみサポートされます。

形式は、[compute_resource *<compute-resource-name>*] です。*compute-resource-name* は、英字で始まり、30 文字以下で、英字、数字、ハイフン (-)、アンダーライン (_) のみで構成される必要があります。

```
[compute_resource cr1]
instance_type = c5.xlarge
min_count = 0
initial_count = 2
max_count = 10
spot_price = 0.5
```

> **Note**
>
> AWS ParallelCluster バージョン 2.9.0 で [compute_resource]セクションのサポートが追加されました。

トピック

- initial_count
- instance_type
- max_count
- min_count
- spot_price

initial_count

(オプション) このコンピューティングリソースを起動するための Amazon EC2 インスタンスの最初の数を設定します。クラスターの作成は、少なくともこのたくさんのノードがコンピューティングリソースに送信されるまで完了しません。キューの compute_type 設定が spot で、利用可能なスポットインスタンスが十分でない場合、クラスターの作成がタイムアウトして失敗することがあります。min_count の設定よりも大きいカウントは、scaledown_idletime の設定の対象となる動的容量です。この設定は、initial_queue_size 設定と置き換わります。

デフォルトは 0 です。

```
initial_count = 2
```

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

instance_type

(必須) このコンピューティングリソースに使用される Amazon EC2 インスタンスタイプを定義します。インスタンスタイプのアーキテクチャは、master_instance_type 設定に使用されるアーキテクチャと同じでなければなりません。instance_type の設定は、[queue] セクションで参照される[compute_resource] セクションごとに固有のものでなければなりません。この設定は、compute_instance_type 設定と置き換わります。

```
instance_type = t2.micro
```

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

max_count

(オプション) このコンピューティングリソースで起動可能な最大数の Amazon EC2 インスタンスの最大数を設定します。initial_count の設定値より大きいカウントは、パワーダウンモードで起動します。この設定は、max_queue_size 設定と置き換わります。

デフォルトは 10 です。

```
max_count = 10
```

更新ポリシー: キューを現在のノード数以下に減らすには、まずコンピューティングフリートを停止させる必要があります。

> **Note**
> 更新ポリシーでは、バージョン 2.0.0 から AWS ParallelCluster バージョン 2.9.1 のコンピューティングフリートが停止するまで、 max_count設定を変更することはサポートされていませんでした。

min_count

(オプション) このコンピューティングリソースで起動可能な最大数の Amazon EC2 インスタンスの最小数を設定します。これらのノードはすべて静的な容量です。クラスターの作成は、少なくともこの数のノードがコンピューティングリソースに送信されるまで完了しません。

デフォルトは 0 です。

```
min_count = 1
```

更新ポリシー: キュー内の静的ノードの数を減らすには、まずコンピューティングフリートを停止させる必要があります。

> **Note**
> 更新ポリシーでは、バージョン 2.0.0 から AWS ParallelCluster バージョン 2.9.1 のコンピューティングフリートが停止するまで、 min_count設定を変更することはサポートされていませんでした。

spot_price

(オプション) このコンピューティングリソースの最大スポット料金を設定します。このコンピューティングリソースを含むキューの compute_type 設定が spot に設定されている場合のみ使用されます。この設定は、spot_price 設定と置き換わります。

値を指定しない場合、オンデマンド料金を上限とするスポット料金が課金されます。

ニーズに合ったスポットインスタンスを見つける方法については、「Spot Instance advisor」(スポットインスタンスアドバイザー) を参照してください。

```
spot_price = 1.50
```

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

[cw_log] セクション

CloudWatch Logs の構成設定を定義します。

形式は、[cw_log *cw-log-name*] です。*cw-log-name* は、英字で始まり、30 文字以下で、英字、数字、ハイフン (-)、アンダーライン (_) のみで構成される必要があります。

```
[cw_log custom-cw-log]
enable = true
retention_days = 14
```

詳細についてはAmazon CloudWatch Logs との統合、Amazon CloudWatch ダッシュボード、およびAmazon CloudWatch Logs との統合を参照してください。

> **Note**
>
> AWS ParallelCluster バージョン 2.6.0 で のサポートが追加されcw_logました。

enable

(オプション) CloudWatch Logs ログ記録が有効かどうかを示します。

デフォルト値は true です。CloudWatch Logs を無効にするには false を使用します。

次の例では、CloudWatch Logs を有効にします。

```
enable = true
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

retention_days

(オプション) CloudWatch Logs が個々のログイベントを保持する日数を示します。

デフォルト値は 14 です。サポートされる値は
1、3、5、7、14、30、60、90、120、150、180、365、400、545、731、1827、3653 です。

次の例では、CloudWatch Logs が 30 日間ログイベントを保持するように設定しています。

```
retention_days = 30
```

更新ポリシー: この設定は、更新中に変更できます。

[dashboard] セクション

CloudWatch ダッシュボードの構成設定を定義します。

形式は、[dashboard *dashboard-name*] です。*dashboard-name* は、英字で始まり、30 文字以下で、英字、数字、ハイフン (-)、アンダーライン (_) のみで構成される必要があります。

```
[dashboard custom-dashboard]
enable = true
```

> **Note**
>
> AWS ParallelCluster バージョン 2.10.0 で のサポートが追加されdashboardました。

enable

(オプション) CloudWatch ダッシュボードログが有効かどうかを示します。

デフォルト値は true です。false を使用して、CloudWatch ダッシュボードを無効にします。

次の例では、CloudWatch ダッシュボードを有効にします。

```
enable = true
```

更新ポリシー: この設定は、更新中に変更できます。

[dcv] セクション

ヘッドノードで実行している Amazon DCV サーバーの構成設定を定義します。

Amazon DCV サーバを作成して設定するには、dcv セクションで定義した名前を使用してクラスター dcv_settings を指定し、enable を master に設定して、base_os を alinux2、centos7、ubuntu1804 または ubuntu2004 に設定します。ヘッドノードが ARM インスタンスの場合は、base_os を alinux2、centos7 または ubuntu1804 に設定します。

形式は、[dcv *dcv-name*] です。*dcv-name* は、英字で始まり、30 文字以下で、英字、数字、ハイフン (-)、アンダーライン (_) のみで構成される必要があります。

```
[dcv custom-dcv]
enable = master
port = 8443
access_from = 0.0.0.0/0
```

詳細については、Amazon DCV を介してヘッドノードに接続するを参照してください。

> **Important**
>
> デフォルトでは、 によって設定された Amazon DCV ポート AWS ParallelCluster はすべての IPv4 アドレスに対して開かれています。ただし、Amazon DCV ポートに接続できるのは、Amazon DCV セッションの URL があり、この URL が pcluster dcv connect から返されてから 30 秒以内に Amazon DCV セッションに接続する場合のみです。access_from 設定を使用して、CIDR 形式の IP 範囲で Amazon DCV ポートへのアクセスをさらに制限し、port 設定を使用して非標準ポートを設定します。

> **Note**
>
> AWS ParallelCluster バージョン 2.10.4 で、centos8 の [dcv] セクションのサポートが終了しました。 AWS ParallelCluster バージョン 2.10.0 で のセクションのサポートが追加され[dcv]centos8ました。 AWS ParallelCluster バージョン 2.9.0 AWS では、Graviton

ベースのインスタンスに関する [dcv]セクションのサポートが追加されました。alinux2 および の [dcv]セクションのサポートが AWS ParallelCluster バージョン 2.6.0 に追加ubuntu1804されました。 AWS ParallelCluster バージョン 2.5.0 で のセクションのサポートが追加され[dcv]centos7ました。

access_from

(オプション、推奨) Amazon DCV に接続するための CIDR 形式の IP 範囲を指定します。この設定は、 がセキュリティグループ AWS ParallelCluster を作成する場合にのみ使用されます。

デフォルト値は `0.0.0.0/0` で、すべてのインターネットアドレスからのアクセスを許可します。

```
access_from = 0.0.0.0/0
```

更新ポリシー: この設定は、更新中に変更できます。

enable

(必須) Amazon DCV がヘッドノードで有効になっているかどうかを示します。Amazon DCV をヘッドノードで有効にし、必要なセキュリティグループルールを設定するには、`enable` 設定を `master` に設定します。

次の例では、Amazon DCV をヘッドノードで有効にします。

```
enable = master
```

> **Note**
>
> Amazon DCV は、ヘッドノードで実行している Amazon DCV サーバーと Amazon DCV クライアント間のトラフィックを保護するための自己署名証明書を自動的に生成します。独自の証明書を設定するには、「Amazon DCV HTTPS 証明書」を参照してください。

更新ポリシー: この設定が変更された場合、更新は許可されません。

port

(オプション) Amazon DCV のポートを指定します。

デフォルト値は 8443 です。

```
port = 8443
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

[ebs] セクション

ヘッドノードにマウントされ、NFS を介してコンピューティングノードに共有されるボリュームの Amazon EBS ボリューム設定を定義します。

クラスター定義に Amazon EBS ボリュームを含める方法については、「[cluster] #### #/ebs_settings」を参照してください。

既存の Amazon EBS ボリュームを、クラスターのライフサイクルとは関係なく長期の永続ストレージとして使用するには、ebs_volume_id を指定してください。

を指定しない場合ebs_volume_id、 はクラスターの作成時に[ebs]設定から EBS ボリューム AWS ParallelCluster を作成し、クラスターの削除時にボリュームとデータを削除します。

詳細については、「ベストプラクティス: AWS ParallelCluster クラスターを新しいマイナーバージョンまたはパッチバージョンに移動する」を参照してください。

形式は、[ebs *ebs-name*] です。*ebs-name* は、英字で始まり、30 文字以下で、英字、数字、ハイフン (-)、アンダーライン (_) のみで構成される必要があります。

```
[ebs custom1]
shared_dir = vol1
ebs_snapshot_id = snap-xxxxx
volume_type = io1
volume_iops = 200
...

[ebs custom2]
shared_dir = vol2
...

...
```

トピック

- shared_dir
- ebs_kms_key_id
- ebs_snapshot_id
- ebs_volume_id
- encrypted
- volume_iops
- volume_size
- volume_throughput
- volume_type

shared_dir

(必須) 共有 Amazon EBS ボリュームをマウントするパスを指定します。

複数の Amazon EBS ボリュームを使用する場合、このパラメータは必須です。

1 つの Amazon EBS ボリュームを使用する場合は、このオプションは `[cluster]` セクションで指定された `shared_dir` を上書きします。以下の例では、ボリュームは `/vol1` にマウントされます。

```
shared_dir = vol1
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

ebs_kms_key_id

（オプション）暗号化に使用するカスタム AWS KMS キーを指定します。

このパラメータは、`encrypted = true` とともに使用する必要があります。また、カスタムの `ec2_iam_role` も含まれている必要があります。

詳細については、「カスタム KMS キーを使用したディスク暗号化」を参照してください。

```
ebs_kms_key_id = xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

ebs_snapshot_id

(オプション) ボリュームのソースとしてスナップショット ID を使用している場合は、Amazon EBS スナップショット ID を定義します。

デフォルト値はありません。

```
ebs_snapshot_id = snap-xxxxx
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

ebs_volume_id

(オプション) ヘッドノードにアタッチする既存の Amazon EBS ボリュームのボリューム ID を定義します。

デフォルト値はありません。

```
ebs_volume_id = vol-xxxxxx
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

encrypted

(オプション) Amazon EBS ボリュームが暗号化されているかどうかを指定します。注意: スナップショットでは使用しないでください。

デフォルト値は `false` です。

```
encrypted = false
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

volume_iops

(オプション) `io1`、`io2` および `gp3` タイプボリュームの IOPS の数を定義します。

デフォルト値、対応値、`volume_iops` 対 `volume_size` の比率は、`volume_type` と `volume_size` で異なります。

`volume_type = io1`

デフォルト `volume_iops = 100`

サポートする値 `volume_iops = 100 — 64000` †

最大 `volume_iops/volume_size` 比率 = 50 IOPS/GiB。5000 IOPS には、少なくとも 100 GiB の `volume_size` が必要です。

`volume_type = io2`

デフォルト `volume_iops = 100`

サポートする値 `volume_iops = 100 — 64000` (io2 Block Express ボリュームの場合は 256000) †

最大 `volume_iops/volume_size` 比率 = 500 IOPS/GiB。5000 IOPS には、少なくとも 10 GiB の `volume_size` が必要です。

`volume_type = gp3`

デフォルト `volume_iops = 3000`

サポートされる値 `volume_iops = 3000 — 16000`

最大 `volume_iops/volume_size` 比率 = 500 IOPS/GiB。5000 IOPS には、少なくとも 10 GiB の `volume_size` が必要です。

```
volume_iops = 200
```

更新ポリシー: この設定は、更新中に変更できます。

† 最大 IOPS は、32,000 IOPS 以上でプロビジョニングされた Nitro System で構築されたインスタンスにのみ保証されます。他のインスタンスは、最大 32,000 IOPS を保証します。ボリュームを変更しない限り、以前の io1 ボリュームは最大のパフォーマンスに達しない可能性があります。io2Block Express ボリュームは、最大 256,000 までの `volume_iops` 値をサポートします。詳細については、「Amazon EC2 ユーザーガイド」の「io2 Block Express ボリューム (プレビュー)」を参照してください。

volume_size

(オプション) 作成するボリュームのサイズ (GiB 単位) を指定します (スナップショットを使用していない場合)。

デフォルト値、対応値は、volume_type で異なります。

volume_type = standard

デフォルト値 volume_size = 20 GiB

サポートされる値 volume_size = 1 — 1024 GiB

volume_type = gp2、io1、io2 および gp3

デフォルト値 volume_size = 20 GiB

サポートされる値 volume_size = 1 — 16384 GiB

volume_type = sc1 および st1

デフォルト値 volume_size = 500 GiB

サポートされる値 volume_size = 500 — 16384 GiB

```
volume_size = 20
```

> **Note**
> AWS ParallelCluster バージョン 2.10.1 以前は、すべてのボリュームタイプのデフォルト値は 20 GiB でした。

更新ポリシー: この設定が変更された場合、更新は許可されません。

volume_throughput

(オプション) gp3 ボリュームタイプのスループットを MiB/秒 で定義します。

デフォルト値は 125 です。

サポートされる値 volume_throughput = 125 — 1000 MIB/秒

volume_throughput と volume_iops の比率は 0.25 以下にします。最大のスループットである 1000 MiB/秒を実現するためには、volume_iops の設定を 4000 以上にする必要があります。

```
volume_throughput = 1000
```

> **Note**
>
> AWS ParallelCluster バージョン 2.10.1 で のサポートが追加されvolume_throughputました。

更新ポリシー: この設定が変更された場合、更新は許可されません。

volume_type

(オプション) 起動するボリュームの Amazon EBS ボリュームタイプを指定します。

有効なオプションは次のボリュームタイプです。

gp2, gp3

汎用 SSD

io1, io2

プロビジョンド IOPS SSD

st1

スループット最適化 HDD

sc1

コールド HDD

standard

前世代のマグネット

詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EBS ボリュームの種類」を参照してください。

デフォルト値は gp2 です。

```
volume_type = io2
```

> **Note**
>
> gp3 および のサポートio2が AWS ParallelCluster バージョン 2.10.1 で追加されました。

更新ポリシー: この設定が変更された場合、更新は許可されません。

[efs] セクション

ヘッドインスタンスとコンピューティングインスタンスにマウントされる Amazon EFS の設定を定義します。詳細については、「Amazon FSx API リファレンス」の「CreateFileSystem」を参照してください。

Amazon EFS ファイルシステムをクラスター定義に含める方法については、「[cluster] #####」/「efs_settings」を参照してください。

クラスターのライフサイクルとは関係なく、既存の Amazon EFS ファイルシステムを長期の永続的なストレージとして使用するには、efs_fs_id を指定します。

を指定しない場合efs_fs_id、 はクラスターの作成時に[efs]設定から Amazon EFS ファイルシステム AWS ParallelCluster を作成し、クラスターの削除時にファイルシステムとデータを削除します。

詳細については、「ベストプラクティス: AWS ParallelCluster クラスターを新しいマイナーバージョンまたはパッチバージョンに移動する」を参照してください。

形式は、[efs *efs-name*] です。*efs-name* は、英字で始まり、30 文字以下で、英字、数字、ハイフン (-)、アンダーライン (_) のみで構成される必要があります。

```
[efs customfs]
shared_dir = efs
encrypted = false
performance_mode = generalPurpose
```

トピック

- efs_fs_id
- efs_kms_key_id
- encrypted
- performance_mode

- provisioned_throughput
- shared_dir
- throughput_mode

efs_fs_id

(オプション) 既存のファイルシステムの Amazon EFS ファイルシステム ID を定義します。

このオプションを指定すると、`shared_dir` を除く他の Amazon EFS オプションはすべて無効になります。

このオプションを設定した場合は、以下のタイプのファイルシステムのみサポートされます。

- スタックのアベイラビリティーゾーンにマウントターゲットがないファイルシステム
- スタックのアベイラビリティーゾーンに既存のマウントターゲットがあり、`0.0.0.0/0` からのインバウンドおよびアウトバウンドの NFS トラフィックが許可されているファイルシステム

`efs_fs_id` を検証するための健全性チェックには、IAM ロールに以下のアクセス許可が必要になります。

- `elasticfilesystem:DescribeMountTargets`
- `elasticfilesystem:DescribeMountTargetSecurityGroups`
- `ec2:DescribeSubnets`
- `ec2:DescribeSecurityGroups`
- `ec2:DescribeNetworkInterfaceAttribute`

エラーを回避するためには、これらのアクセス許可を IAM ロールに追加するか、`sanity_check = false` を設定します。

> **Important**
>
> から許可されるインバウンドおよびアウトバウンド NFS トラフィックを使用してマウントターゲットを設定すると`0.0.0.0/0`、マウントターゲットのアベイラビリティーゾーン内の任意の場所からの NFS マウントリクエストにファイルシステムが公開されます。スタックのアベイラビリティーゾーンにマウントターゲットを作成することはお勧め AWS しません。代わりに、 でこのステップ AWS を処理します。スタックのアベイラビリティーゾーン

にマウントターゲットが必要な場合は、[vpc] セクションで vpc_security_group_id オプションを指定して、カスタムセキュリティグループを使用することを検討してください。次に、そのセキュリティグループをマウントターゲットに追加し、sanity_check をオフにしてクラスターを作成します。

デフォルト値はありません。

```
efs_fs_id = fs-12345
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

efs_kms_key_id

（オプション）暗号化されたファイルシステムを保護するために使用する AWS Key Management Service（AWS KMS) カスタマーマネージドキーを識別します。これを設定する場合は、encrypted 設定を true に設定する必要があります。これは、「Amazon EFS API Reference」(Amazon EFS API リファレンス) の KmsKeyId パラメータに対応しています。

デフォルト値はありません。

```
efs_kms_key_id = 1234abcd-12ab-34cd-56ef-1234567890ab
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

encrypted

(オプション) ファイルシステムが暗号化されているかどうかを示します。これは、「Amazon EFS API Reference」(Amazon EFS API リファレンス) の Encrypted パラメータに対応しています。

デフォルト値は false です。

```
encrypted = true
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

performance_mode

(オプション) ファイル システムのパフォーマンスモードを定義します。これは、「Amazon EFS API Reference」(Amazon EFS API リファレンス) の PerformanceMode パラメータに対応しています。

有効なオプションは、次の値です。

- `generalPurpose`
- `maxIO`

いずれの値も、大文字と小文字が区別されます。

ほとんどのファイルシステムに `generalPurpose` パフォーマンスモードをお勧めします。

`maxIO` パフォーマンスモードを使用するファイルシステムでは、集計スループットと 1 秒あたりのオペレーション数をより高いレベルにスケーリングできます。ただし、ほとんどのファイルオペレーションでは、レイテンシーがわずかに長くなるというトレードオフがあります。

このパラメータは、ファイルシステムの作成後に変更することはできません。

デフォルト値は `generalPurpose` です。

```
performance_mode = generalPurpose
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

provisioned_throughput

(オプション) ファイルシステムのプロビジョニングされたスループット (MiB/秒で測定) を定義します。これは、「Amazon EFS API Reference」(Amazon EFS API リファレンス) の ProvisionedThroughputInMibps パラメータに対応しています。

このパラメータを使用する場合は、`throughput_mode` を `provisioned` に設定する必要があります。

スループットのクォータは `1024` MiB/秒です。クォータの引き上げをリクエストするには、 サポートにお問い合わせください。

最小値は `0.0` MiB/秒 です。

```
provisioned_throughput = 1024
```

更新ポリシー: この設定は、更新中に変更できます。

shared_dir

(必須) ヘッドノードとコンピューティングノードの Amazon EFS マウントポイントを定義します。

このパラメータは必須です。Amazon EFS セクションは、shared_dir が指定されている場合にのみ使用します。

共有ディレクトリとして /NONE または NONE を使用しないでください。

次の例では、/efs の Amazon EFS をマウントします。

```
shared_dir = efs
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

throughput_mode

(オプション) ファイルシステムのスループットモードを定義します。これは、「Amazon EFS API Reference」(Amazon EFS API リファレンス) の ThroughputMode パラメータに対応しています。

有効なオプションは、次の値です。

- bursting
- provisioned

デフォルト値は bursting です。

```
throughput_mode = provisioned
```

更新ポリシー: この設定は、更新中に変更できます。

[fsx] セクション

アタッチされた FSx for Lustre ファイルシステムの設定を定義します。詳細については、「Amazon FSx API リファレンス」の「Amazon FSx CreateFileSystem」を参照してください。

FSx for Lustre は，base_os が alinux2、centos7、ubuntu1804、または ubuntu2004 の場合に対応しています。

Amazon Linux を使用する場合、カーネルは `4.14.104-78.84.amzn1.x86_64` 以上である必要があります。手順については、「Amazon FSx for Lustre ユーザーガイド」の「Lustre クライアントをインストールする」を参照してください。

> **Note**
>
> `awsbatch` をスケジューラとして使用する場合、FSx for Lustre は現在サポートされていません。

> **Note**
>
> での FSx for Lustre のサポート`centos8`が AWS ParallelCluster バージョン 2.10.4 で削除されました。で FSx for Lustre のサポート`ubuntu2004`が AWS ParallelCluster バージョン 2.11.0 に追加されました。 AWS ParallelCluster バージョン 2.10.0 で `centos8` の FSx for Lustre をサポートしました。、`alinux2`、`ubuntu1604`および での FSx for Lustre のサポート`ubuntu1804`が AWS ParallelCluster バージョン 2.6.0 で追加されました。 AWS ParallelCluster バージョン 2.4.0 で `centos7` の FSx for Lustre をサポートしました。

既存のファイルシステムを使用する場合は、ポート 988 経由のインバウンドの TCP トラフィックを許可するセキュリティグループに関連付ける必要があります。セキュリティグループルールでソースを `0.0.0.0/0` に設定すると、そのルールのプロトコルとポート範囲に対する VPC セキュリティグループ内のすべての IP 範囲からのクライアントアクセスが可能になります。ファイルシステムへのアクセスをさらに制限するために、セキュリティグループルールには、より限定的なソースを使用することをお勧めします。例えば、より具体的な CIDR の範囲、IP アドレス、セキュリティグループ ID などを使用することができます。`vpc_security_group_id` を使用しない場合、これは自動的に行われます。

クラスターのライフサイクルとは関係なく、既存の Amazon FSx ファイルシステムを長期の永続的なストレージとして使用するには、`fsx_fs_id` を指定します。

を指定しない場合`fsx_fs_id`、 はクラスターの作成時に`[fsx]`設定から FSx for Lustre ファイルシステム AWS ParallelCluster を作成し、クラスターの削除時にファイルシステムとデータを削除します。

詳細については、「ベストプラクティス: AWS ParallelCluster クラスターを新しいマイナーバージョンまたはパッチバージョンに移動する」を参照してください。

形式は、[fsx *fsx-name*] です。*fsx-name* は、英字で始まり、30 文字以下で、英字、数字、ハイフン (-)、アンダーライン (_) のみで構成される必要があります。

```
[fsx fs]
shared_dir = /fsx
fsx_fs_id = fs-073c3803dca3e28a6
```

新しいファイルシステムを作成し、設定するには、次のパラメータを使用します。

```
[fsx fs]
shared_dir = /fsx
storage_capacity = 3600
imported_file_chunk_size = 1024
export_path = s3://bucket/folder
import_path = s3://bucket
weekly_maintenance_start_time = 1:00:00
```

トピック

- auto_import_policy
- automatic_backup_retention_days
- copy_tags_to_backups
- daily_automatic_backup_start_time
- data_compression_type
- deployment_type
- drive_cache_type
- export_path
- fsx_backup_id
- fsx_fs_id
- fsx_kms_key_id
- import_path
- imported_file_chunk_size
- per_unit_storage_throughput
- shared_dir
- storage_capacity

- storage_type
- weekly_maintenance_start_time

auto_import_policy

(オプション) FSx for Lustre ファイルシステムの作成に使用する S3 バケットに変更を反映させるための自動インポートポリシーを指定します。取り得る値には以下のものがあります。

NEW

FSx for Lustre は、リンクされた S3 バケットに追加された新しいオブジェクトのうち、現在 FSx for Lustre ファイルシステム内に存在しないオブジェクトのディレクトリリストを自動的にインポートします。

NEW_CHANGED

S3 バケットに追加された新しいオブジェクトや S3 バケットで変更された既存のオブジェクトのファイルとディレクトリのリストが、FSx for Lustre によって自動的にインポートされます。

これは、AutoImportPolicy プロパティに対応しています。詳細については、「Amazon FSx for Lustre ユーザーガイド」の「S3 バケットから更新を自動的にインポートする」を参照してください。auto_import_policy パラメータを指定する場合、automatic_backup_retention_days、copy_tags_to_backups、daily_automatic_backup_および fsx_backup_id パラメータを指定してはいけません。

auto_import_policy の設定が指定されていない場合、自動インポートは無効になります。FSx for Lustre は、ファイルシステムの作成時に、リンクされた S3 バケットのファイルとディレクトリのリストのみを更新します。

```
auto_import_policy = NEW_CHANGED
```

> **Note**
>
> AWS ParallelCluster バージョン 2.10.0 で のサポートが追加されauto_import_policyました。

更新ポリシー: この設定が変更された場合、更新は許可されません。

automatic_backup_retention_days

(オプション) 自動バックアップを保持する日数を指定します。これは、PERSISTENT_1 デプロイタイプでのみ有効です。automatic_backup_retention_days パラメータを指定する場合、auto_import_policy、export_path、import_path および imported_file_chunk_size パラメータを指定してはいけません。これは、AutomaticBackupRetentionDays プロパティに対応しています。

デフォルト値は 0 です。この設定は、自動バックアップを無効にします。指定できる値は 0 から 35 までの整数です。

```
automatic_backup_retention_days = 35
```

> **Note**
>
> AWS ParallelCluster バージョン 2.8.0 で automatic_backup_retention_days をサポートしました。

更新ポリシー: この設定は、更新中に変更できます。

copy_tags_to_backups

(オプション) ファイルシステムのタグをバックアップにコピーするかどうかを指定します。これは、PERSISTENT_1 デプロイタイプでのみ有効です。copy_tags_to_backups パラメータを指定した場合、automatic_backup_retention_days には 0 以上の値を指定し、auto_import_policy、export_path、import_path および imported_file_chunk_size パラメータは指定してはいけません。これは、CopyTagsToBackups プロパティに対応しています。

デフォルト値は false です。

```
copy_tags_to_backups = true
```

> **Note**
>
> AWS ParallelCluster バージョン 2.8.0 で のサポートが追加されcopy_tags_to_backupsました。

更新ポリシー: この設定が変更された場合、更新は許可されません。

daily_automatic_backup_start_time

(オプション) 自動バックアップを開始する時刻 (UTC) を指定します。これは、PERSISTENT_1 デプロイタイプでのみ有効です。daily_automatic_backup_start_time パラメータを指定した場合、automatic_backup_retention_days には 0 以上の値を指定し、auto_import_policy、export_path、import_path および imported_file_chunk_size パラメータは指定してはいけません。これは、DailyAutomaticBackupStartTime プロパティに対応しています。

フォーマットは HH:MM で、HH はゼロパディングされたその日の時間 (0～23)、MM はゼロパディングされたその日の分を表します。例えば、午前 1:03 UTC は次のようになります。

```
daily_automatic_backup_start_time = 01:03
```

初期値は 00:00～23:59 の間のランダムな時間です。

> **Note**
>
> AWS ParallelCluster バージョン 2.8.0 で daily_automatic_backup_start_time をサポートしました。

更新ポリシー: この設定は、更新中に変更できます。

data_compression_type

(オプション) FSx for Lustre のデータ圧縮タイプを指定します。これは、DataCompressionType プロパティに対応しています。詳細については、「Amazon FSx for Lustre ユーザーガイド」の「FSx for Lustre データ圧縮」を参照してください。

唯一の有効な値は LZ4 です。データ圧縮を無効にするには、data_compression_type パラメータを削除します。

```
data_compression_type = LZ4
```

> **Note**
>
> AWS ParallelCluster バージョン 2.11.0 で のサポートが追加されdata_compression_typeました。

更新ポリシー: この設定は、更新中に変更できます。

deployment_type

(オプション) FSx for Lustre のデプロイタイプを指定します。これは、DeploymentType プロパティに対応しています。詳細については、「Amazon FSx for Lustre ユーザーガイド」の「FSx for Lustre デプロイオプション」を参照してください。一時ストレージとデータの短期処理には、スクラッチデプロイタイプを選択します。SCRATCH_2 は、最新世代のスクラッチファイルシステムです。ベースラインスループットよりも高いバーストスループットと、データの転送中の暗号化を実現しています。

有効な値は SCRATCH_1、SCRATCH_2、および PERSISTENT_1 です。

SCRATCH_1

FSx for Lustre のデフォルトのデプロイタイプ。このデプロイタイプでは、storage_capacity 設定の可能な値は 1200、2400、および 3600 の倍数です。 AWS ParallelCluster バージョン 2.4.0 で のサポートが追加されSCRATCH_1ました。

SCRATCH_2

最新世代のスクラッチファイルシステム。スパイキーなワークロードに対して、ベースラインの最大 6 倍のスループットをサポートします。また、サポートされている AWS リージョンのサポートされているインスタンスタイプでは、データの転送中の暗号化もサポートされています。詳細については、「Amazon FSx for Lustre ユーザーガイド」の「転送中のデータの暗号化」を参照してください。このデプロイタイプでは、storage_capacity 設定の可能な値は 1200 および 2400 の倍数です。 AWS ParallelCluster バージョン 2.6.0 で SCRATCH_2 をサポートしました。

PERSISTENT_1

長期ストレージ用に設計されています。ファイルサーバーは高可用性であり、データはファイルシステムの AWS アベイラビリティーゾーン内で自動的にレプリケーションされます。サポートされているインスタンスタイプでは、データの転送中の暗号化もサポートされています。この

デプロイタイプでは、storage_capacity 設定の可能な値は 1200 および 2400 の倍数です。AWS ParallelCluster バージョン 2.6.0 で のサポートが追加されPERSISTENT_1ました。

デフォルト値は SCRATCH_1 です。

```
deployment_type = SCRATCH_2
```

> **Note**
>
> AWS ParallelCluster バージョン 2.6.0 で のサポートが追加されdeployment_typeました。

更新ポリシー: この設定が変更された場合、更新は許可されません。

drive_cache_type

(オプション) ファイルシステムに SSD ドライブキャッシュを搭載することを指定します。storage_type の設定が HDD になっている場合のみ設定可能です。これは、DriveCacheType プロパティに対応しています。詳細については、「Amazon FSx for Lustre ユーザーガイド」の「FSx for Lustre デプロイオプション」を参照してください。

唯一の有効な値は READ です。SSD ドライブのキャッシュを無効にするには、drive_cache_type の設定を指定しないでください。

```
drive_cache_type = READ
```

> **Note**
>
> AWS ParallelCluster バージョン 2.10.0 で のサポートが追加されdrive_cache_typeました。

更新ポリシー: この設定が変更された場合、更新は許可されません。

export_path

(オプション) ファイルシステムのルートがエクスポートされる Amazon S3 パスを指定します。export_path パラメータを指定する場

合、automatic_backup_retention_days、copy_tags_to_backups、daily_automatic_backup_
および fsx_backup_id パラメータを指定してはいけません。これは、ExportPath プロパティに対応しています。ファイルのデータやメタデータが自動的に export_path にエクスポートされません。データやメタデータのエクスポートについては、「Amazon FSx for Lustre ユーザーガイド」の「データリポジトリに変更をエクスポートする」を参照してください。

デフォルト値は s3://*import-bucket*/FSxLustre*[creation-timestamp]* です。ここで、*import-bucket* は import_path パラメータで指定されるバケットです。

```
export_path = s3://bucket/folder
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

fsx_backup_id

(オプション) 既存のバックアップからファイルシステムを復元する際に使用するバックアップの ID を指定します。fsx_backup_id パラメータが指定されている場合、auto_import_policy、deployment_type、export_path、fsx_kms_key_id、import_path、および per_unit_storage_throughput パラメータは指定できません。これらのパラメータはバックアップから読み込まれます。また、auto_import_policy、export_path、import_path および imported_file_chunk_size パラメータは指定できません。

これは、BackupId プロパティに対応しています。

```
fsx_backup_id = backup-fedcba98
```

> **Note**
>
> AWS ParallelCluster バージョン 2.8.0 で のサポートが追加されfsx_backup_idました。

更新ポリシー: この設定が変更された場合、更新は許可されません。

fsx_fs_id

(オプション) 既存の FSx for Lustre ファイルシステムをアタッチします。

このオプションを指定した場合、[fsx] セクションの shared_dir および fsx_fs_id の設定のみが使用され、[fsx] セクションのその他の設定は無視されます。

```
fsx_fs_id = fs-073c3803dca3e28a6
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

fsx_kms_key_id

（オプション）AWS Key Management Service（AWS KMS) カスタマーマネージドキーのキー ID を指定します。

このキーは、保管時のファイルシステムのデータを暗号化するために使用されます。

これは、カスタム ec2_iam_role と使用する必要があります。詳細については、「カスタム KMS キーを使用したディスク暗号化」を参照してください。これは、Amazon FSx API リファレンスの KmsKeyId パラメータに対応しています。

```
fsx_kms_key_id = xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

> **Note**
>
> AWS ParallelCluster バージョン 2.6.0 で のサポートが追加されfsx_kms_key_idました。

更新ポリシー: この設定が変更された場合、更新は許可されません。

import_path

(オプション) データをファイルシステムにロードし、エクスポートバケットとして機能する S3 バケットを指定します。詳細については、「export_path」を参照してください。import_path パラメータを指定する場合、automatic_backup_retention_days、copy_tags_to_backups、daily_automatic_backup_および fsx_backup_id パラメータを指定してはいけません。これは、Amazon FSx API リファレンスの ImportPath パラメータに対応しています。

インポートは、クラスター作成時に行われます。詳細については、「Amazon FSx for Lustre ユーザーガイド」の「データリポジトリからデータをインポートする」を参照してください。インポート時には、ファイルのメタデータ (名前、所有権、タイムスタンプ、パーミッション) のみがインポートされます。ファイルデータは、ファイルが最初にアクセスされるまで、S3 バケットからインポートされません。ファイルコンテンツの事前ロードの詳細については、「Amazon FSx for Lustre ユーザーガイド」の「ファイルシステムへのファイルの事前ロード」を参照してください。

値が指定されない場合、ファイルシステムは空になります。

```
import_path =  s3://bucket
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

imported_file_chunk_size

(オプション) データリポジトリからインポートされたファイルの場合 (import_path を使用)、この値がストライプカウントと 1 つの物理ディスクに保存されたファイルごとの最大データ量 (MiB 単位) を決定します。1 つのファイルにストライピングできるディスクの最大数は、ファイルシステムを構成するディスクの合計数によって制限されます。imported_file_chunk_size パラメータを指定する場合、automatic_backup_retention_days、copy_tags_to_backups、daily_automatic_backup_ および fsx_backup_id パラメータを指定してはいけません。これは、ImportedFileChunkSize プロパティに対応しています。

チャンクサイズのデフォルトは 1024 (1 GiB) であり、それは、512,000 MiB (500 GiB) まで高くすることができます。Amazon S3 オブジェクトの最大サイズは 5 TB です。

```
imported_file_chunk_size = 1024
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

per_unit_storage_throughput

(**PERSISTENT_1** デプロイタイプに必須) deployment_type = PERSISTENT_1 デプロイタイプの場合、1 テラバイト (TiB) のストレージごとの読み取りおよび書き込みスループットの量を MB/s/TiB 単位で取得します。ファイルシステムのスループット容量は、ファイルシステムのストレージ容量 (TiB) に per_unit_storage_throughput (MB/s/TiB) を掛けて計算されます。2.4 TiB のファイルシステムの場合、50 MB/s/TiB の per_unit_storage_throughput をプロビジョニングすると、ファイルシステムのスループットは 120 MB/s になります。プロビジョニングしたスループットに対して支払いが発生します。これは、PerUnitStorageThroughput プロパティに対応しています。

設定可能な値は、storage_type の設定値によって異なります。

storage_type = SSD

指定できる値は 50、100、200 です。

`storage_type` = HDD

指定できる値は 12、40 です。

```
per_unit_storage_throughput = 200
```

> **Note**
>
> AWS ParallelCluster バージョン 2.6.0 で `per_unit_storage_throughput` をサポートしました。

更新ポリシー: この設定が変更された場合、更新は許可されません。

shared_dir

(必須) ヘッドノードとコンピューティングノードで FSx for Lustre ファイルシステムのマウントポイントを定義します。

共有ディレクトリとして `/NONE` または `NONE` を使用しないでください。

次の例では、ファイルシステムを `/fsx` にマウントします。

```
shared_dir = /fsx
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

storage_capacity

(必須) ファイルシステムのストレージ容量 (GiB 単位) を指定します。これは、StorageCapacity プロパティに対応しています。

ストレージ容量の可能な値は `deployment_type` 設定によって異なります。

`SCRATCH_1`

可能な値は 1200、2400、および 3600 の倍数です。

`SCRATCH_2`

可能な値は 1200 および 2400 の倍数です。

PERSISTENT_1

設定可能な値は、他の設定の値によって異なります。

storage_type = SSD

可能な値は 1200 および 2400 の倍数です。

storage_type = HDD

設定可能な値は、per_unit_storage_throughput の設定内容によって異なります。

per_unit_storage_throughput = 12

可能な値は 6000 の倍数です。

per_unit_storage_throughput = 40

可能な値は 1800 の倍数です。

```
storage_capacity = 7200
```

> **Note**
>
> AWS ParallelCluster バージョン 2.5.0 および 2.5.1 では、 は 1200、2400、および 3600 の倍数の可能な値storage_capacityをサポートしています。バージョン 2.5.0 より前の AWS ParallelCluster バージョンでは、 の最小サイズは 3600 storage_capacityでした。

更新ポリシー: この設定が変更された場合、更新は許可されません。

storage_type

(オプション) フファイルシステムのストレージタイプを指定します。これは、StorageType プロパティに対応しています。指定できる値は SSD および HDD です。デフォルトは SSD です。

ストレージタイプによって、他の設定の可能な値が変わります。

storage_type = SSD

ソリッドステートドライブ (SSD) ストレージタイプを指定します。

storage_type = SSD は、他のいくつかの設定の可能な値を変更します。

drive_cache_type

この設定は指定できません。

deployment_type

この設定は、SCRATCH_1、SCRATCH_2、PERSISTENT_1 のいずれかに設定できます。

per_unit_storage_throughput

deployment_type を PERSISTENT_1 に設定する場合は、この設定を指定する必要があります。指定できる値は 50、100、200 です。

storage_capacity

この設定は必ず指定する必要があります。可能な値は、deployment_type に基づいて変化します。

deployment_type = SCRATCH_1

storage_capacity は 1200、2400、または 3600 の任意の倍数です。

deployment_type = SCRATCH_2 または deployment_type = PERSISTENT_1

storage_capacity は 1200、または 2400 の任意の倍数です。

storage_type = HDD

ハードディスクドライブ (HDD) のストレージタイプを指定します。

storage_type = HDD は、他の設定の可能な値を変更します。

drive_cache_type

この設定を指定することができます。

deployment_type

この設定は PERSISTENT_1 に設定する必要があります。

per_unit_storage_throughput

この設定は必ず指定する必要があります。指定できる値は 12、または、40 です。

storage_capacity

この設定は必ず指定する必要があります。設定可能な値は、per_unit_storage_throughput の設定によって異なります。

storage_capacity = 12

storage_capacity は 6000 の任意の倍数です。

storage_capacity = 40

storage_capacity は 1800 の任意の倍数です。

```
storage_type = SSD
```

> **Note**
>
> AWS ParallelCluster バージョン 2.10.0 で storage_type をサポートしました。

更新ポリシー: この設定が変更された場合、更新は許可されません。

weekly_maintenance_start_time

(オプション) 毎週のメンテナンスを実行する推奨時間 (UTC タイムゾーン) を指定します。これは、WeeklyMaintenanceStartTime プロパティに対応しています。

形式は [曜日]: [時間]: [分] です。例えば、月曜日の深夜は次のようになります。

```
weekly_maintenance_start_time = 1:00:00
```

更新ポリシー: この設定は、更新中に変更できます。

[queue] セクション

1 つのキューのコンフィギュレーション設定を定義します。[queue] セクションは、scheduler が slurm に設定されている場合のみサポートされます。

形式は、[queue *<queue-name>*] です。*queue-name* は、英字の小文字で始まり、30 文字以下で、小文字の英字、数字、ハイフン (-) のみで構成される必要があります。

```
[queue q1]
compute_resource_settings = i1,i2
placement_group = DYNAMIC
enable_efa = true
```

```
disable_hyperthreading = false
compute_type = spot
```

> **Note**
>
> AWS ParallelCluster バージョン 2.9.0 で [queue]セクションのサポートが追加されました。

トピック

- compute_resource_settings
- compute_type
- disable_hyperthreading
- enable_efa
- enable_efa_gdr
- placement_group

compute_resource_settings

(必須) このキューのコンピューティングリソースの設定を含む [compute_resource] セクションを指定します。セクション名は、英字で始まり、30 文字以下で、英字、数字、ハイフン (-)、アンダーライン (_) のみで構成される必要があります。

1 つの [queue] セクションに対して、最大 3 つの [compute_resource] セクションがサポートされています。

例えば、以下の設定では、[compute_resource cr1] と [compute_resource cr2] で始まるセクションが使用されるように指定します。

```
compute_resource_settings = cr1, cr2
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

compute_type

(オプション) このキューで起動するインスタンスの種類を定義します。この設定は、cluster_type 設定と置き換わります。

有効なオプションは、ondemand と spot です。

デフォルト値は ondemand です。

スポットインスタンスの詳細については、「スポットインスタンスの操作」を参照してください。

> **Note**
>
> スポットインスタンスを使用するには、お客様のアカウントに AWSServiceRoleForEC2Spot サービスにリンクしたロールが必要です。を使用してアカウントにこのロールを作成するには AWS CLI、次のコマンドを実行します。
>
> ```
> aws iam create-service-linked-role --aws-service-name spot.amazonaws.com
> ```
>
> 詳細については、「Amazon EC2 ユーザーガイド」の「スポットインスタンスリクエスト向けのサービスにリンクされたロール」を参照してください。

次の例では、このキューのコンピューティングノードに SpotInstances を使用しています。

```
compute_type = spot
```

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

disable_hyperthreading

(オプション) このキューに入っているノードのハイパースレッディングを無効にします。すべてのインスタンスタイプがハイパースレッディングを無効にできるわけではありません。ハイパースレッディングの無効化をサポートするインスタンスタイプの一覧については、「Amazon EC2 ユーザーガイド」のインスタンスタイプ別の CPU コア数と CPU コアごとのスレッド数を参照してください。[cluster] セクションの disable_hyperthreading 設定が定義されている場合は、この設定を定義することはできません。

デフォルト値は false です。

```
disable_hyperthreading = true
```

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

enable_efa

(オプション) true に設定されている場合、このキューのノードに対して Elastic Fabric Adapter (EFA) が有効であることを指定します。EFA をサポートする EC2 インスタンスのリストを確認するには、「Linux インスタンス用 Amazon EC2 ユーザーガイド」の「サポートされるインスタンスタイプ」を参照してください。[cluster] セクションの enable_efa 設定が定義されている場合は、この設定を定義することはできません。クラスタープレイスメントグループは、インスタンス間のレイテンシーを最小限に抑えるために使用する必要があります。詳細については、「placement」および「placement_group」を参照してください。

```
enable_efa = true
```

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

enable_efa_gdr

（オプション）AWS ParallelCluster バージョン 2.11.3 以降、この設定は効果がありません。GPUDirect RDMA (リモートダイレクトメモリアクセス) をサポートする Elastic Fabric Adapter (EFA) は、コンピューティングノードでは、インスタンスタイプでサポートされていれば常に有効です。

> **Note**
>
> AWS ParallelCluster バージョン 2.10.0 から 2.11.2: の場合true、このキュー内のノードに対して Elastic Fabric Adapter (EFA) GPUDirect RDMA (リモートダイレクトメモリアクセス) が有効になっていることを指定します。これを true に設定するには、enable_efa の設定を true にする必要があります。EFA GPUDirect RDMA は、これらのオペレーティングシステム (alinux2、centos7、ubuntu1804、または ubuntu2004) 上の以下のインスタンスタイプ (p4d.24xlarge) でサポートされています。[cluster] セクションの enable_efa_gdr 設定が定義されている場合は、この設定を定義することはできません。クラスタープレイスメントグループは、インスタンス間のレイテンシーを最小限に抑えるために使用する必要があります。詳細については、「placement」および「placement_group」を参照してください。

デフォルト値は false です。

```
enable_efa_gdr = true
```

> **Note**
>
> AWS ParallelCluster バージョン 2.10.0 で のサポートが追加されenable_efa_gdrました。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

placement_group

(オプション) 存在する場合、このキューのプレイスメントグループを定義します。この設定は、placement_group 設定と置き換わります。

有効なオプションは、次の値です。

- DYNAMIC
- 既存の Amazon EC2 クラスタープレイスメントグループ名

DYNAMIC に設定されている場合、このキューのための一意のプレイスメントグループはクラスタースタックの一部として作成され、削除されます。

プレイスメントグループの詳細については、「Amazon EC2 ユーザーガイド」の「プレイスメントグループ」を参照してください。異なるインスタンスタイプを同じプレイスメントグループで使用している場合、容量不足のエラーでリクエストが失敗する可能性が高くなります。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンス容量の不足」を参照してください。

デフォルト値はありません。

すべてのインスタンスタイプがクラスタープレイスメントグループをサポートしているわけではありません。例えば、t2.micro では、クラスタープレイスメントグループをサポートしていません。クラスターのプレイスメントグループをサポートするインスタンスタイプのリストについては、「Amazon EC2 ユーザーガイド」の「クラスタープレイスメントグループのルールと制限」を参照してください。プレイスメントグループを使用する際のヒントについては、「プレイスメントグループとインスタンスの起動に関する問題」を参照してください。

```
placement_group = DYNAMIC
```

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

[raid] セクション

多数の同一の Amazon EBS ボリュームから作成される RAID アレイの設定を定義します。RAID ドライブはヘッドノードにマウントされ、NFS でコンピューティングノードにエクスポートされます。

形式は、[raid *raid-name*] です。*raid-name* は、英字で始まり、30 文字以下で、英字、数字、ハイフン (-)、アンダーライン (_) のみで構成される必要があります。

```
[raid rs]
shared_dir = raid
raid_type = 1
num_of_raid_volumes = 2
encrypted = true
```

トピック

- shared_dir
- ebs_kms_key_id
- encrypted
- num_of_raid_volumes
- raid_type
- volume_iops
- volume_size
- volume_throughput
- volume_type

shared_dir

(必須) ヘッドノードとコンピューティングノードでの RAID アレイのマウントポイントを定義します。

RAID ドライブは、このパラメータが指定されている場合にのみ作成されます。

共有ディレクトリとして /NONE または NONE を使用しないでください。

次の例では、/raid のアレイをマウントします。

```
shared_dir = raid
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

ebs_kms_key_id

（オプション）暗号化に使用するカスタム AWS KMS キーを指定します。

このパラメータは、encrypted = true とともに使用し、カスタムの ec2_iam_role が含まれている必要があります。

詳細については、「カスタム KMS キーを使用したディスク暗号化」を参照してください。

```
ebs_kms_key_id = xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

encrypted

(オプション) ファイルシステムを暗号化するかどうかを指定します。

デフォルト値は false です。

```
encrypted = false
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

num_of_raid_volumes

(オプション) RAID アレイをアセンブルする Amazon EBS ボリュームの数を定義します。

ボリュームの最小数は 2 です。

ボリュームの最大数は 5 です。

デフォルト値は 2 です。

```
num_of_raid_volumes = 2
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

raid_type

(必須) RAID アレイの RAID タイプを定義します。

RAID ドライブは、このパラメータが指定されている場合にのみ作成されます。

有効なオプションは、次の値です。

- 0
- 1

RAID タイプの詳細ついては、「Amazon EC2 ユーザーガイド」の「RAID 情報」を参照してください。

以下の例では、RAID 0 アレイが作成されます。

```
raid_type = 0
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

volume_iops

(オプション) io1、io2 および gp3 タイプボリュームの IOPS の数を定義します。

デフォルト値、対応値、volume_iops 対 volume_size の比率は、volume_type と volume_size で異なります。

volume_type = io1

デフォルト volume_iops = 100

サポートする値 volume_iops = 100 — 64000 †

最大 volume_iops/volume_size 比率 = 50 IOPS/GiB。5000 IOPS には、少なくとも 100 GiB の volume_size が必要です。

volume_type = io2

デフォルト volume_iops = 100

サポートする値 volume_iops = 100 — 64000 (io2 Block Express ボリュームの場合は 256000) †

最大 volume_iops/volume_size 比率 = 500 IOPS/GiB。5000 IOPS には、少なくとも 10 GiB の volume_size が必要です。

volume_type = gp3

デフォルト volume_iops = 3000

サポートされる値 volume_iops = 3000 — 16000

最大 volume_iops/volume_size 比率 = 500 IOPS/GiB。5000 IOPS には、少なくとも 10 GiB の volume_size が必要です。

```
volume_iops = 3000
```

更新ポリシー: この設定は、更新中に変更できます。

† 最大 IOPS は、32,000 IOPS 以上でプロビジョニングされた Nitro System で構築されたインスタンスにのみ保証されます。他のインスタンスは、最大 32,000 IOPS を保証します。ボリュームを変更しない限り、古い io1 ボリュームはパフォーマンスが完全にはならないことがあります。io2Block Express ボリュームは、最大 256000 までの volume_iops 値をサポートします。詳細については、「Amazon EC2 ユーザーガイド」の「io2 Block Express ボリューム (プレビュー)」を参照してください。

volume_size

(オプション) 作成するボリュームのサイズを定義します (GiB 単位)。

デフォルト値、対応値は、volume_type で異なります。

volume_type = standard

デフォルト値 volume_size = 20 GiB

サポートされる値 volume_size = 1 — 1024 GiB

volume_type = gp2、io1、io2 および gp3

デフォルト値 volume_size = 20 GiB

サポートされる値 volume_size = 1 — 16384 GiB

`volume_type` = `sc1` および `st1`

デフォルト値 `volume_size` = 500 GiB

サポートされる値 `volume_size` = 500 — 16384 GiB

```
volume_size = 20
```

> **Note**
>
> AWS ParallelCluster バージョン 2.10.1 以前は、すべてのボリュームタイプのデフォルト値は 20 GiB でした。

更新ポリシー: この設定が変更された場合、更新は許可されません。

volume_throughput

(オプション) gp3 ボリュームタイプのスループットを MiB/秒 で定義します。

デフォルト値は 125 です。

サポートされる値 `volume_throughput` = 125 — 1000 MIB/秒

`volume_throughput` と `volume_iops` の比率は 0.25 以下にします。最大のスループットである 1000 MiB/秒を実現するためには、`volume_iops` の設定を 4000 以上にする必要があります。

```
volume_throughput = 1000
```

> **Note**
>
> AWS ParallelCluster バージョン 2.10.1 で のサポートが追加されvolume_throughputました。

更新ポリシー: この設定が変更された場合、更新は許可されません。

volume_type

(オプション) 構築するボリュームのタイプを定義します。

有効なオプションは、次の値です。

`gp2, gp3`

汎用 SSD

`io1, io2`

プロビジョンド IOPS SSD

`st1`

スループット最適化 HDD

`sc1`

コールド HDD

`standard`

前世代のマグネット

詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EBS ボリュームの種類」を参照してください。

デフォルト値は gp2 です。

```
volume_type = io2
```

> **Note**
>
> gp3 および のサポートio2が AWS ParallelCluster バージョン 2.10.1 で追加されました。

更新ポリシー: この設定が変更された場合、更新は許可されません。

[scaling] セクション

トピック

- scaledown_idletime

コンピューティングノードがスケールする方法を定義する設定を指定します。

形式は、[scaling *scaling-name*] です。*scaling-name* は、英字で始まり、30 文字以下で、英字、数字、ハイフン (-)、アンダーライン (_) のみで構成される必要があります。

```
[scaling custom]
scaledown_idletime = 10
```

scaledown_idletime

(オプション) ジョブがない場合にコンピューティングノードが終了する時間 (単位: 分) を指定します。

このパラメータは、awsbatch がスケジューラの場合には使用されません。

デフォルト値は 10 です。

```
scaledown_idletime = 10
```

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

[vpc] セクション

Amazon VPC の構成設定を指定します。VPC の詳細については、「Amazon VPC ユーザーガイド」の「Amazon VPC とは?」と「VPC 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。

形式は、[vpc *vpc-name*] です。*vpc-name* は、英字で始まり、30 文字以下で、英字、数字、ハイフン (-)、アンダーライン (_) のみで構成される必要があります。

```
[vpc public]
vpc_id = vpc-xxxxxx
master_subnet_id = subnet-xxxxxx
```

トピック

- additional_sg
- compute_subnet_cidr
- compute_subnet_id
- master_subnet_id
- ssh_from

- use_public_ips
- vpc_id
- vpc_security_group_id

additional_sg

(オプション) すべてのインスタンス用の追加の Amazon VPC セキュリティグループ ID を指定します。

デフォルト値はありません。

```
additional_sg = sg-xxxxxx
```

compute_subnet_cidr

(オプション) クラスレスのドメイン間ルーティング (CIDR) ブロックを指定します。コンピューティングサブネット AWS ParallelCluster を作成する場合は、このパラメータを使用します。

```
compute_subnet_cidr = 10.0.100.0/24
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

compute_subnet_id

(オプション) コンピューティングノードをプロビジョンする既存のサブネットの ID を指定します。

指定しない場合は、compute_subnet_id は master_subnet_id の値を使用します。

サブネットがプライベートの場合は、ウェブアクセスに NAT を設定する必要があります。

```
compute_subnet_id = subnet-xxxxxx
```

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

master_subnet_id

(必須) ヘッドノードをプロビジョンする既存のサブネットの ID を指定します。

```
master_subnet_id = subnet-xxxxxx
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

ssh_from

(オプション) SSH アクセスを許可する CIDR 形式の IP 範囲を指定します。

このパラメータは、 がセキュリティグループ AWS ParallelCluster を作成する場合にのみ使用されます。

デフォルト値は `0.0.0.0/0` です。

```
ssh_from = 0.0.0.0/0
```

更新ポリシー: この設定は、更新中に変更できます。

use_public_ips

(オプション) コンピューティングインスタンスにパブリック IP アドレスを割り当てるかどうかを定義します。

`true` に設定すると、Elastic IP アドレスがヘッドノードに関連付けられます。

`false` に設定した場合、ヘッドインスタンスには、「Auto-assign Public IP」(自動割り当てパブリック IP) サブネット設定パラメータの値に応じてパブリック IP が設定されます (または設定されません)。

例については、「networking configuration」(ネットワーク設定) を参照してください。

デフォルト値は `true` です。

```
use_public_ips = true
```

> **Important**
>
> デフォルトでは、それぞれ 5 つの (5) Elastic IP アドレスに制限 AWS アカウント されています AWS リージョン。詳細については、「Amazon EC2 ユーザーガイド」の「Elastic IP アドレスの制限」を参照してください。

更新ポリシー: この設定を更新で変更するためには、コンピューティングフリートを停止する必要があります。

vpc_id

(必須) クラスターをプロビジョンする Amazon VPC の ID を指定します。

```
vpc_id = vpc-xxxxxx
```

更新ポリシー: この設定が変更された場合、更新は許可されません。

vpc_security_group_id

(オプション) すべてのインスタンスに既存のセキュリティグループを使用するよう指定します。

デフォルト値はありません。

```
vpc_security_group_id = sg-xxxxxx
```

によって作成されたセキュリティグループは、 `ssh_from`設定で指定されたアドレスからポート 22 を使用して SSH アクセス AWS ParallelCluster を許可し、 `ssh_from` 設定が指定されていない場合はすべての IPv4 アドレス (`0.0.0.0/0`) を許可します。Amazon DCV が有効になっている場合、セキュリティグループでは、`access_from` 設定で指定したアドレスまたは (`access_from` 設定の指定がない場合は) すべての IPv4 アドレス (`0.0.0.0/0`) から、ポート 8443 (または `port` 設定で指定された任意のポート) を使用して、Amazon DCV にアクセスすることを許可します。

> **Warning**
>
> `[cluster]` `fsx_settings` が指定されていない場合、または `fsx_settings` と外部に存在する FSx for Lustre ファイルシステムの両方が `[fsx fs]` の `fsx-fs-id` に指定されている場合は、このパラメータの値を変更してクラスターを更新できます。
> AWS ParallelCluster マネージド FSx for Lustre ファイルシステムが `fsx_settings`および で指定されている場合、このパラメータの値を変更することはできません`[fsx fs]`。

更新ポリシー: AWS ParallelCluster マネージド Amazon FSx for Lustre ファイルシステムが設定で指定されていない場合、この設定は更新中に変更できます。

例

次の設定例はSlurm、、Torque、および ス AWS Batch ケジューラを使用した AWS ParallelCluster 設定を示しています。

> **Note**
>
> バージョン 2.11.5 AWS ParallelCluster 以降、 は SGEまたは スTorqueケジューラの使用をサポートしていません。

目次

- Slurm Workload Manager (slurm)
- Son of Grid Engine (sge) および Torque Resource Manager (torque)
- AWS Batch (awsbatch)

Slurm Workload Manager (**`slurm`**)

次の例では、`slurm` スケジューラでクラスターを起動します。この例では、1 つのクラスターを 2 つのジョブキューで起動します。最初のキュー `spot` には、初期状態で 2 つの `t3.micro` スポットインスタンスが用意されています。最大で 10 インスタンスまで拡張でき、10 分間ジョブが実行されなかった場合には最小で 1 インスタンスまで拡張することができます (`scaledown_idletime` 設定で調整可能)。2 番目のキューである ondemand は、インスタンスがない状態からスタートし、最大で 5 台の `t3.micro` オンデマンドインスタンスまで拡張することができます。

```
[global]
update_check = true
sanity_check = true
cluster_template = slurm

[aws]
aws_region_name = <your AWS #####>

[vpc public]
master_subnet_id = <your subnet>
vpc_id = <your VPC>

[cluster slurm]
```

```
key_name = <your EC2 keypair name>
base_os = alinux2                   # optional, defaults to alinux2
scheduler = slurm
master_instance_type = t3.micro     # optional, defaults to t3.micro
vpc_settings = public
queue_settings = spot,ondemand

[queue spot]
compute_resource_settings = spot_i1
compute_type = spot                 # optional, defaults to ondemand

[compute_resource spot_i1]
instance_type = t3.micro
min_count = 1                       # optional, defaults to 0
initial_count = 2                   # optional, defaults to 0

[queue ondemand]
compute_resource_settings = ondemand_i1

[compute_resource ondemand_i1]
instance_type = t3.micro
max_count = 5                       # optional, defaults to 10
```

Son of Grid Engine (**sge**) および Torque Resource Manager (**torque**)

> **Note**
>
> この例では、 AWS ParallelCluster バージョン 2.11.4 以前のバージョンにのみ適用されます。バージョン 2.11.5 以降は、 AWS ParallelCluster では SGE または Torque スケジューラの使用はサポートしていません。

次の例では、`torque` または `sge` のスケジューラでクラスターを起動します。SGE を使用するには、`scheduler = torque` を `scheduler = sge` に変更します。サンプルの設定では、最大 5 つの同時ノード が許可され、10 分間ジョブが実行されない場合、2 つにスケールダウンします。

```
[global]
update_check = true
sanity_check = true
cluster_template = torque
```

```
[aws]
aws_region_name = <your AWS #####>

[vpc public]
master_subnet_id = <your subnet>
vpc_id = <your VPC>

[cluster torque]
key_name = <your EC2 keypair name>but they aren't eligible for future updates
base_os = alinux2                   # optional, defaults to alinux2
scheduler = torque                  # optional, defaults to sge
master_instance_type = t3.micro     # optional, defaults to t3.micro
vpc_settings = public
initial_queue_size = 2              # optional, defaults to 0
maintain_initial_size = true        # optional, defaults to false
max_queue_size = 5                  # optional, defaults to 10
```

> **Note**
>
> バージョン 2.11.5 AWS ParallelCluster 以降、 は SGEまたは スTorqueケジューラの使用をサポートしていません。これらのバージョンを使用する場合は、引き続き使用するか、AWS サービスおよびサポートチームによる AWS サポートのトラブルシューティングを行うことができます。

AWS Batch (**awsbatch**)

次の例では、awsbatch スケジューラでクラスターを起動します。ジョブリソースのニーズに基いて、より良いインスタンスタイプを選択するように設定されています。

サンプルの設定では、最大 40 の同時 vCPU が許可され、10 分間 (scaledown_idletime 設定で調整可能) ジョブが実行されない場合、ゼロにスケールダウンします。

```
[global]
update_check = true
sanity_check = true
cluster_template = awsbatch

[aws]
aws_region_name = <your AWS #####>
```

```
[vpc public]
master_subnet_id = <your subnet>
vpc_id = <your VPC>

[cluster awsbatch]
scheduler = awsbatch
compute_instance_type = optimal # optional, defaults to optimal
min_vcpus = 0                   # optional, defaults to 0
desired_vcpus = 0               # optional, defaults to 4
max_vcpus = 40                  # optional, defaults to 20
base_os = alinux2               # optional, defaults to alinux2, controls the base_os
 of
                                # the head node and the docker image for the compute
 fleet
key_name = <your EC2 keypair name>
vpc_settings = public
```

の AWS ParallelCluster 仕組み

AWS ParallelCluster は、クラスターを管理する方法としてだけでなく、 AWS サービスを使用して HPC 環境を構築する方法のリファレンスとして構築されました。

トピック

- AWS ParallelCluster プロセス
- AWS が使用する サービス AWS ParallelCluster
- AWS ParallelCluster Auto Scaling

AWS ParallelCluster プロセス

このセクションは、サポートされている従来のジョブスケジューラ (SGE、Slurm、Torque) のいずれかを使用してデプロイされた HPC クラスターにのみ適用されます。これらのスケジューラとともに使用すると、 は Auto Scaling グループと基盤となるジョブスケジューラの両方とやり取りすることで、コンピューティングノードのプロビジョニングと削除 AWS ParallelCluster を管理します。

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

> **Note**
>
> バージョン 2.11.5 以降、 は SGEまたは スTorqueケジューラの使用をサポート AWS ParallelCluster していません。2.11.4 までのバージョンで引き続き使用できますが、 AWS サービスおよび AWS サポートチームによる今後の更新やトラブルシューティングのサポートを受けることはできません。

トピック

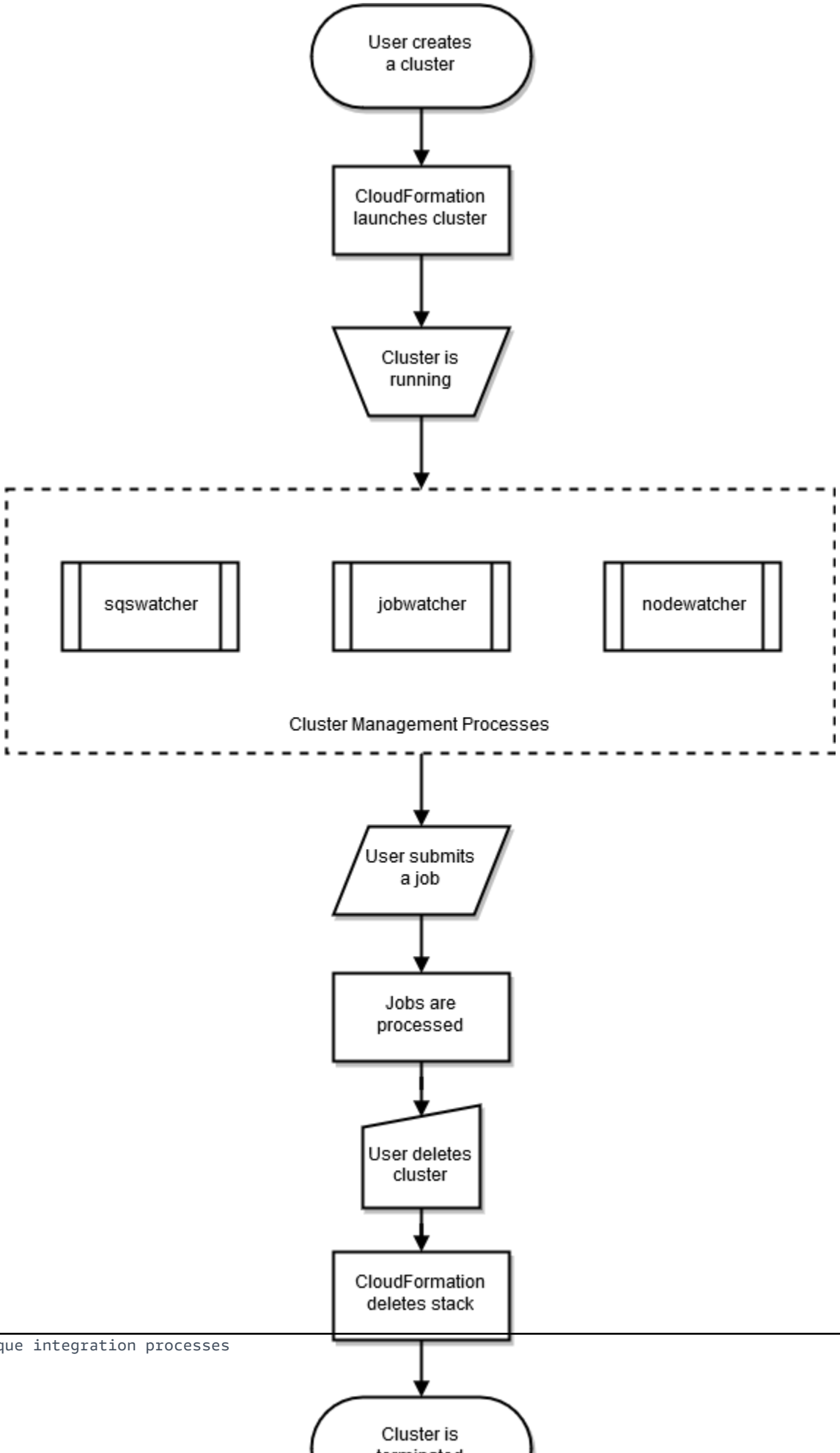
- SGE and Torque integration processes
- Slurm integration processes

SGE and Torque integration processes

> **Note**
>
> このセクションは、 AWS ParallelCluster バージョン 2.11.4 以前のバージョンにのみ適用されます。バージョン 2.11.5 以降は、 AWS ParallelCluster では SGE および Torque スケジューラ、Amazon SNS、Amazon SQS の使用はサポートしていません。

全般的な概要

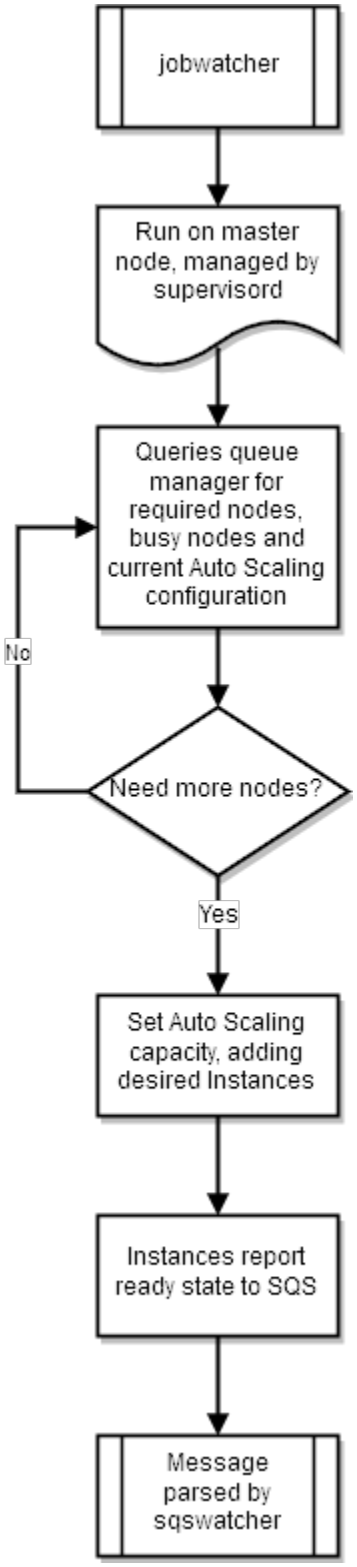
クラスターのライフサイクルは、ユーザーによって作成された後に始まります。通常、クラスターはコマンドラインインターフェイス (CLI) から実行されます。作成後、クラスターは削除されるまで存在します。 AWS ParallelCluster デーモンは、主に HPC クラスターの伸縮性を管理するために、クラスターノードで実行されます。次の図は、ユーザーのワークフローとクラスターのライフサイクルを示します。以下のセクションでは、クラスターの管理に使用される AWS ParallelCluster デーモンについて説明します。


User creates a cluster
CloudFormation launches cluster
Cluster is running
sqswatcher
jobwatcher
nodewatcher
Cluster Management Processes
User submits a job
Jobs are processed
User deletes cluster
CloudFormation deletes stack
Cluster is

SGE および スTorqueケジューラでは、、`jobwatcher`、および `nodewatchersqswatcher`プロセス AWS ParallelCluster を使用します。

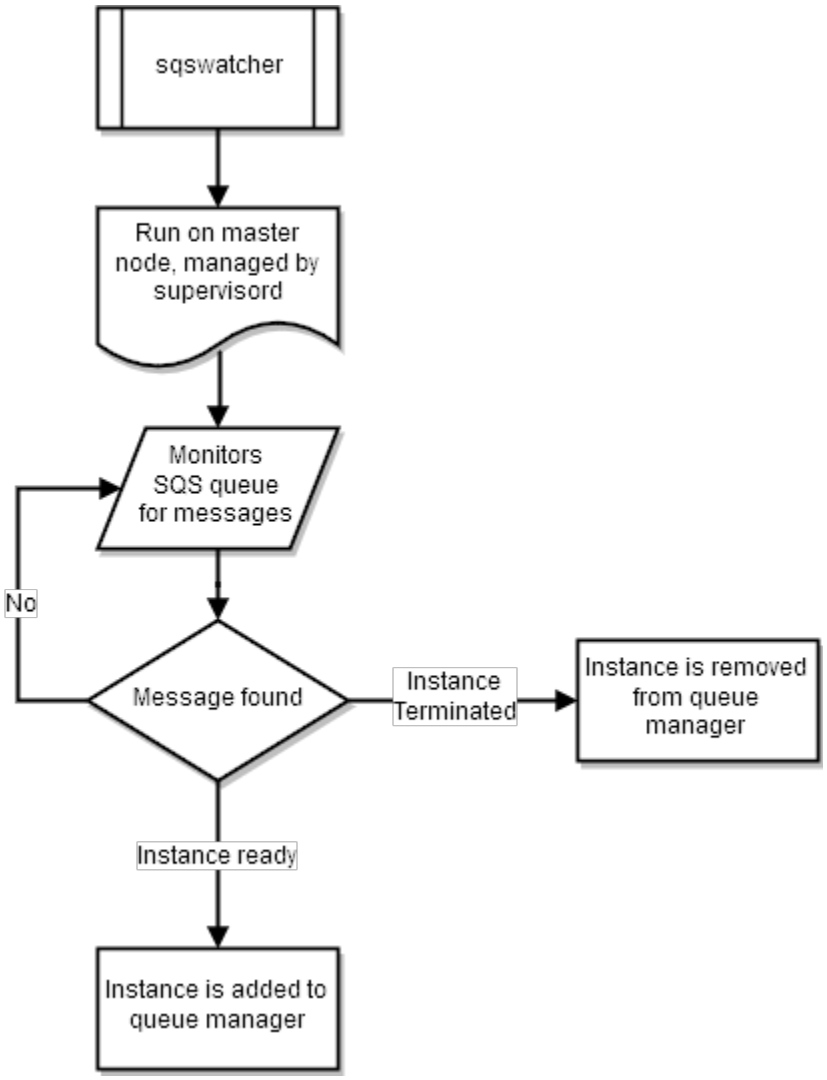
`jobwatcher`

クラスターが動作しているとき、ルートユーザーが所有するプロセスは、設定されたスケジューラ (SGE または Torque) をモニタリングします。毎分、キューを評価し、いつスケールアップするかを決定します。


jobwatcher
Run on master node, managed by supervisord
Queries queue manager for required nodes, busy nodes and current Auto Scaling configuration
No
Need more nodes?
Yes
Set Auto Scaling capacity, adding desired Instances
Instances report ready state to SQS
Message parsed by sqswatcher

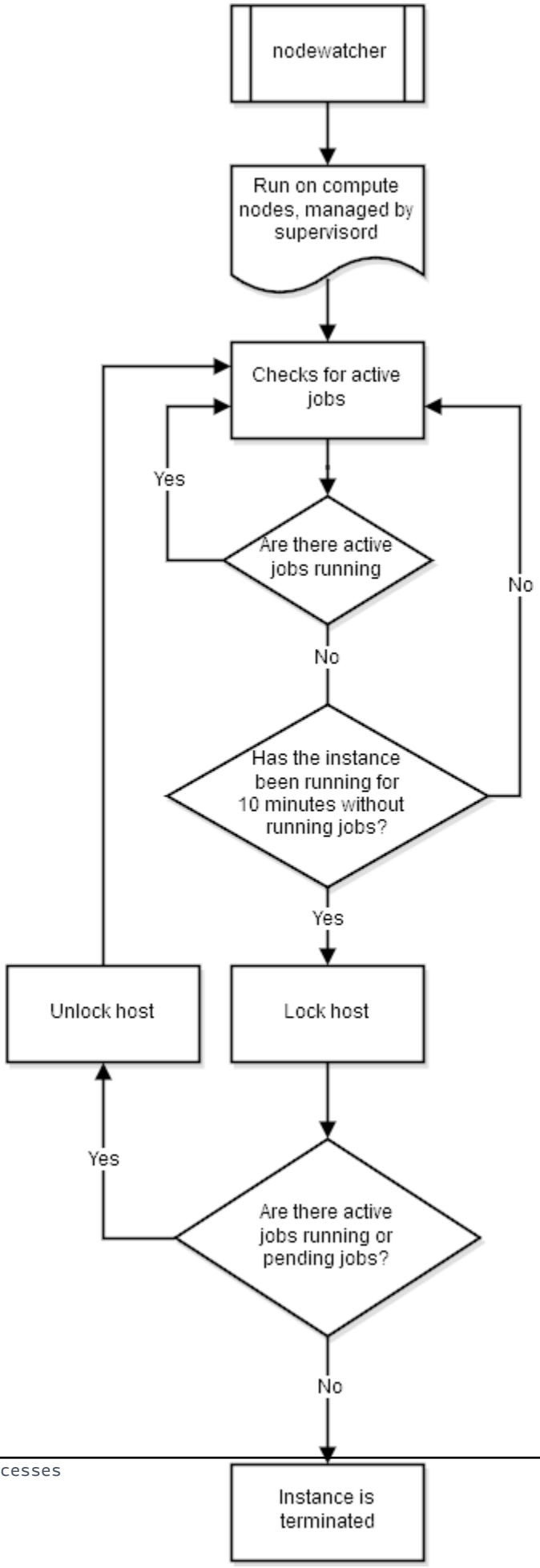
sqswatcher

sqswatcher プロセスでは、オートスケーリングによって送信される Amazon SQS メッセージをモニタリングして、クラスター内のステータスの変化を通知します。インスタンスがオンラインになると、「instance ready」メッセージが Amazon SQS に送信されます。このメッセージは、ヘッドノード上で実行されている sqs_watcher に送信されます。これらのメッセージは、新しいインスタンスがオンラインになったときや終了したときにキューマネージャーに通知するために使用されるため、それらをキューに追加したり、キューから削除したりすることができます。



nodewatcher

nodewatcher プロセスは、コンピューティングシステムの各ノード上で実行されます。ユーザーが定義した scaledown_idletime 期間が過ぎると、インスタンスは終了します。


nodewatcher
Run on compute nodes, managed by supervisord
Checks for active jobs
Yes
Are there active jobs running
No
No
Has the instance been running for 10 minutes without running jobs?
Yes
Unlock host
Lock host
Yes
Are there active jobs running or pending jobs?
No
Instance is terminated

Slurm integration processes

スSlurmケジューラでは、 `clustermgtd`と `computemgt`プロセス AWS ParallelCluster を使用します。

clustermgtd

ヘテロジニアスモード (`queue_settings` 値を指定した場合) で動作するクラスターには、ヘッドノード上で動作するクラスター管理デーモン (`clustermgtd`) プロセスがあります。これらのタスクはクラスター管理デーモンが行います。

- 非アクティブなパーティションのクリーンアップ
- 静的容量管理: 静的な容量が常に稼働していることを確認します
- スケジューラを Amazon EC2 と同期します。
- 孤立したインスタンスのクリーンアップ
- 中断したワークフローの外で発生した Amazon EC2 の終了時にスケジューラーノードの状態を復元します
- 異常のある Amazon EC2 インスタンスの管理 (Amazon EC2 のヘルスチェックの失敗)
- スケジュールされたメンテナンスイベントの管理
- 異常のあるスケジューラノードの管理 (スケジューラのヘルスチェックの失敗)

computemgtd

ヘテロジニアスモード (`queue_settings` 値を指定した場合) で動作するクラスターには、コンピューティングノード上で動作するコンピューティング管理デーモン (`computemgtd`) プロセスがあります。5 分ごとに、コンピューティング管理デーモンはヘッドノードに到達できること、および正常であることを確認します。5 分が経過し、ヘッドノードに到達できない、または正常でない場合、コンピューティングノードはシャットダウンされます。

AWS が使用する サービス AWS ParallelCluster

次の Amazon Web Services (AWS) サービスが使用されています AWS ParallelCluster。

トピック

- AWS Auto Scaling
- AWS Batch

- AWS CloudFormation
- Amazon CloudWatch
- Amazon CloudWatch Logs
- AWS CodeBuild
- Amazon DynamoDB
- Amazon Elastic Block Store
- Amazon Elastic Compute Cloud
- Amazon Elastic Container Registry
- Amazon EFS
- Amazon FSx for Lustre
- AWS Identity and Access Management
- AWS Lambda
- Amazon DCV
- Amazon Route 53
- Amazon Simple Notification Service
- Amazon Simple Queue Service
- Amazon Simple Storage Service
- Amazon VPC

AWS Auto Scaling

> **Note**
>
> このセクションは、 AWS ParallelCluster バージョン 2.11.4 以前のバージョンにのみ適用されます。バージョン 2.11.5 以降、 AWS ParallelCluster は の使用をサポートしていません AWS Auto Scaling。

AWS Auto Scaling は、アプリケーションをモニタリングし、特定の変化するサービス要件に基づいて容量を自動的に調整するサービスです。このサービスは、ComputeFleet インスタンスを Auto Scaling グループとして管理します。このグループは、変化するワークロードによって伸縮自在に動作することも、初期のインスタンス構成によって静的に固定することもできます。

AWS Auto Scaling は ComputeFleet インスタンスで使用されますが、 AWS Batch クラスターでは使用されません。

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

AWS Batch

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

AWS Batch は AWS Batch クラスターでのみ使用されます。

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

AWS CloudFormation

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

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

Amazon CloudWatch

Amazon CloudWatch (CloudWatch) は、データと実用的なインサイトを提供するモニタリングおよびオブザーバビリティサービスです。これらのインサイトは、アプリケーションのモニタリング、パフォーマンスの変化やサービスの例外への対応、リソースの使用状況の最適化に利用できま

す。CloudWatch は AWS ParallelCluster、Docker イメージのビルドステップと AWS Batch ジョブの出力をモニタリングおよびログに記録するダッシュボードに使用されます。

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

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

Amazon CloudWatch Logs

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

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

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

AWS CodeBuild

AWS CodeBuild (CodeBuild) は、ソースコードに準拠し、テストを実行し、すぐにデプロイできるソフトウェアパッケージを生成する AWS マネージド型の継続的統合サービスです。では AWS ParallelCluster、CodeBuild を使用して、クラスターの作成時に Docker イメージを自動的かつ透過的に構築します。

CodeBuild は AWS Batch クラスターでのみ使用されます。

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

Amazon DynamoDB

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

DynamoDB は AWS Batch クラスターでは使用されません。

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

Amazon Elastic Block Store

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

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

Amazon Elastic Compute Cloud

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

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

Amazon Elastic Container Registry

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

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

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

Amazon EFS

Amazon Elastic File System (Amazon EFS)は、 AWS クラウド サービスやオンプレミスのリソースで使用できる、シンプルでスケーラブルな、フルマネージドされた伸縮自在な NFS ファイルシステ

ムを提供します。Amazon EFS は、`efs_settings` 設定が指定され、「`[efs]` セクション」を参照している場合に使用されます。 AWS ParallelCluster バージョン 2.1.0 で Amazon EFS のサポートが追加されました。

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

Amazon FSx for Lustre

FSx for Lustre は、オープンソースの Lustre ファイルシステムを使用した高性能なファイルシステムを提供します。FSx for Lustre は、`fsx_settings` 設定が指定され、「`[fsx]` セクション」を参照する場合に使用されます。 AWS ParallelCluster バージョン 2.2.1 で FSx for Lustre をサポートしました。

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

AWS Identity and Access Management

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

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

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

AWS Lambda

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

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

Amazon DCV

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

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

Amazon Route 53

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

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

Amazon Simple Notification Service

> **Note**
>
> このセクションは、 AWS ParallelCluster バージョン 2.11.4 以前のバージョンにのみ適用されます。バージョン 2.11.5 以降は、 AWS ParallelCluster では Amazon Simple Notification Service の使用はサポートしていません。

Amazon Simple Notification Service (Amazon SNS) は、オートスケーリングからの通知を受け取ります。これらのイベントは、ライフサイクルイベントと呼ばれ、Auto Scaling グループでインスタンスが起動または終了したときに生成されます。内では AWS ParallelCluster、Auto Scaling グループの Amazon SNS トピックが Amazon SQS キューにサブスクライブされます。

Amazon SNS は AWS Batch クラスターでは使用されません。

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

Amazon Simple Queue Service

> **Note**
>
> このセクションは、 AWS ParallelCluster バージョン 2.11.4 以前のバージョンにのみ適用されます。バージョン 2.11.5 以降では、 AWS ParallelCluster は Amazon Simple Queue Service の使用はサポートしていません。

Amazon Simple Queue Service (Amazon SQS) は、オートスケーリングから送られてくる通知、Amazon SNS を介して送られてくる通知、およびコンピューティングノードから送られてくる通知を保持します。Amazon SQS は、通知の送信と通知の受信を分離します。これにより、ヘッドノードはポーリングプロセスによって通知を処理することができます。このプロセスでは、ヘッドノードが Amazon SQSwatcher を実行し、キューをポーリングします。オートスケーリングとコンピューティングノードは、キューにメッセージを投稿します。

Amazon SQS は AWS Batch クラスターでは使用されません。

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

Amazon Simple Storage Service

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

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

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

Amazon VPC

Amazon VPC は、クラスター内のノードが使用するネットワークを定義します。クラスターの VPC 設定は `[vpc]` セクションで定義されています。

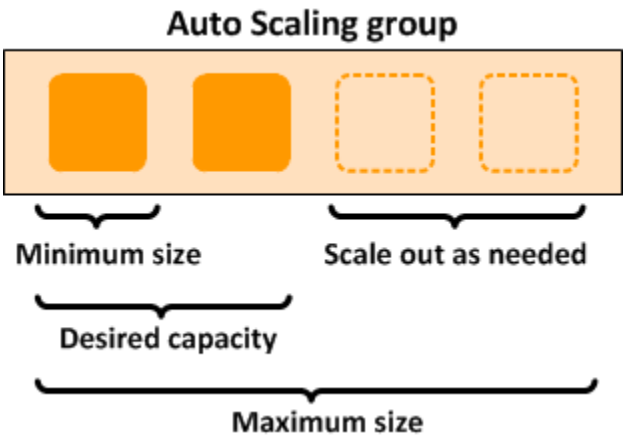
Amazon VPC の詳細については、https://aws.amazon.com/vpc/ および https://docs.aws.amazon.com/vpc/ を参照してください。

AWS ParallelCluster Auto Scaling

> **Note**
>
> このセクションは、 AWS ParallelCluster バージョン 2.11.4 以前のバージョンにのみ適用されます。バージョン 2.11.5 以降、 AWS ParallelCluster は SGE または Torque スケジューラの使用をサポートしていません。2.11.4 までのバージョンで引き続き使用できますが、サービスおよび AWS サポートチームによる AWS 今後の更新やトラブルシューティングのサポートを受けることはできません。
> AWS ParallelCluster バージョン 2.9.0 以降、Auto Scaling は Slurm Workload Manager () での使用はサポートされていませんSlurm。Slurm と複数のキュースケーリングの詳細については、「マルチキューモードのチュートリアル」を参照してください。

このトピックで説明するオートスケーリング戦略は、Son of Grid Engine (SGE) または Torque Resource Manager (Torque) のいずれかで展開される HPC クラスターに適用されます。これらのスケジューラのいずれかを使用してデプロイすると、 AWS ParallelCluster は、コンピューティングノードの Auto Scaling グループを管理し、必要に応じてスケジューラ構成を変更することで、スケーリング機能を実装します。に基づく HPC クラスターの場合 AWS Batch、 AWS ParallelCluster は AWS マネージドジョブスケジューラによって提供される Elastic Scaling 機能に依存します。詳細については、「Amazon EC2 Auto Scaling User Guide」(Amazon EC2 Auto Scaling ユーザーガイド) の「What is Amazon EC2 Auto Scaling」(Amazon EC2 Auto Scaling とは) を参照してください。

でデプロイされたクラスター AWS ParallelCluster は、いくつかの点で伸縮自在です。`initial_queue_size` を設定すると、ComputeFleet Auto Scaling グループの最小サイズの値と、必要な容量の値が指定されます。`max_queue_size` を設定すると、ComputeFleet Auto Scaling グループの最大サイズの値が指定されます。



スケールアップ

毎分、ヘッドノードでは `jobwatcher` というプロセスが実行されます。キュー内の保留中のジョブに必要な現在のインスタンス数を評価します。ビジー状態のノードとリクエストされたノードの合計数が、Auto Scaling グループで現在必要な値より大きい場合、さらにインスタンスを追加します。さらにジョブを送信した場合、キューが再評価され、Auto Scaling グループは、指定した `max_queue_size` まで更新されます。

SGE スケジューラでは、実行する多数のスロットがジョブごとに必要です (1 つのスロットが 1 つの処理ユニット (例: vCPU) に対応します)。現在保留中のジョブを処理するために必要なインスタンスの数を評価する場合、`jobwatcher` は 1 つのコンピューティングノードの容量でリクエストされたスロットの合計数を除算します。利用可能な vCPU の数に相当するコンピューティングノードの容量は、クラスター設定で指定されている Amazon EC2 インスタンスタイプによって異なります。

Slurm (AWS ParallelCluster バージョン 2.9.0 より前) および スTorqueケジューラでは、状況に応じて、各ジョブで各ノードにノード数とスロット数の両方が必要になる場合があります。`jobwatcher` では、新しい計算要件を満たすために必要なコンピューティングノードの数をリクエストごとに決定します。例えば、あるクラスターのコンピューティングインスタンスタイプが `c5.2xlarge` (8 vCPU) であり、3 つの 保留中のキューに登録されたジョブの要件が以下であると仮定します。

- job1: 2 ノード/4 スロット
- job2: 3 ノード/2 スロット
- job3: 1 ノード/4 スロット

この例の `jobwatcher` では、3 つのジョブを処理するために、Auto Scaling グループに 3 つの新しいコンピューティングインスタンスが必要です。

現在の制限: オートスケールアップロジックは、部分的にロードされたビジーノードを考慮しません。例えば、ジョブを実行しているノードでは、スロットが空いていてもビジーとみなされます。

スケールダウン

各コンピューティングノードで `nodewatcher` と呼ばれるプロセスを実行し、ノードのアイドル時間を評価します。インスタンスは、次の条件のいずれも満たされた場合に終了します。

- インスタンスで `scaledown_idletime` より長い期間ジョブが実行されていない (デフォルト設定は 10 分)
- クラスター内に保留中のジョブがない

インスタンスを終了するために、`nodewatcher` は、TerminateInstanceInAutoScalingGroup API オペレーションを呼び出します。その結果、Auto Scaling グループのサイズが Auto Scaling グループの最小サイズを上回ると、インスタンスは削除されます。このプロセスでは、実行中のジョブに影響を及ぼすことなく、クラスターをスケールダウンすることができます。また、固定ベース数のインスタンスにより伸縮自在なクラスターを有効にします。

静的クラスター

スケールの値は、HPC の場合も他のワークロードと同じです。ここでの唯一の違いは、 AWS ParallelCluster には、具体的にインテリジェントな方法で操作させるコードがあることです。例えば、静的クラスターが必要な場合は、`initial_queue_size` パラメータおよび `max_queue_size` パラメータを必要なクラスターの正確なサイズに設定し、その後 `maintain_initial_size` パラメータを true に設定します。これにより、ComputeFleet Auto Scaling グループの最小容量、最大容量、および必要な容量に対して同じ値が設定されます。

チュートリアル

以下のチュートリアルでは、 の使用を開始する方法を示し AWS ParallelCluster、いくつかの一般的なタスクのベストプラクティスガイダンスを提供します。

トピック

- で最初のジョブを実行する AWS ParallelCluster
- カスタム AMI AWS ParallelCluster の構築
- AWS ParallelCluster とスawsbatchケジューラを使用した MPI ジョブの実行
- カスタム KMS キーを使用したディスク暗号化
- マルチキューモードのチュートリアル

で最初のジョブを実行する AWS ParallelCluster

このチュートリアルでは、最初の Hello World ジョブを実行する方法について説明します AWS ParallelCluster。

前提条件

- AWS ParallelCluster がインストールされます。
- AWS CLI がインストールされ、設定されています。
- EC2 キーペアがある。
- `pcluster` CLI の実行に必要なアクセス許可を持つ IAM ロールがある。

インストールを確認する

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

```
$ pcluster version
```

これにより、実行中の のバージョンが返されます AWS ParallelCluster。設定に関するメッセージが出力に表示されたら、 AWS ParallelClusterを設定するために次のコマンドを実行する必要があります。

```
$ pcluster configure
```

初めてクラスターを作成する

では、最初のクラスターを作成していきましょう。このチュートリアルのワークロードのパフォーマンス負荷は高くないため、デフォルトのインスタンスサイズ `t2.micro` を使います。(本稼働ワークロードの場合は、ニーズに最適なインスタンスサイズを選択します)

クラスター hello-world を呼び出してみましょう。

```
$ pcluster create hello-world
```

クラスターが作成されると、次のような出力が表示されます。

```
Starting: hello-world
Status: parallelcluster-hello-world - CREATE_COMPLETE
MasterPublicIP = 54.148.x.x
ClusterUser: ec2-user
MasterPrivateIP = 192.168.x.x
GangliaPrivateURL = http://192.168.x.x/ganglia/
GangliaPublicURL = http://54.148.x.x/ganglia/
```

メッセージ `CREATE_COMPLETE` は、クラスターが正しく作成されたことを示します。また、この出力には、ヘッドノードのパブリック IP アドレスとプライベート IP アドレスも表示されています。ログインするにはこの IP が必要です。

ヘッドノードにログインする

OpenSSH pem ファイルを使用してヘッドノードにログインします。

```
pcluster ssh hello-world -i /path/to/keyfile.pem
```

ログインしたら、`qhost` コマンドを実行して、コンピューティングノードがセットアップおよび設定されていることを確認します。

```
$ qhost
HOSTNAME                ARCH         NCPU NSOC NCOR NTHR  LOAD  MEMTOT  MEMUSE  SWAPTO
 SWAPUS
----------------------------------------------------------------------------------------------
global                  -               -    -    -    -     -       -       -       -
      -
ip-192-168-1-125        lx-amd64        2    1    2    2  0.15    3.7G  130.8M 1024.0M
    0.0
```

```
ip-192-168-1-126        lx-amd64        2    1    2    2  0.15    3.7G  130.8M 1024.0M
    0.0
```

クラスターに 2 つのコンピューティングノードがあり、どちらも 2 つのスレッドを使用できることが出力に示されます。

SGE を使用して最初のジョブを実行する

> **Note**
>
> この例では、 AWS ParallelCluster バージョン 2.11.4 以前のバージョンにのみ適用されます。バージョン 2.11.5 以降は、 AWS ParallelCluster は SGE または Torque スケジューラの使用はサポートしていません。

次に、しばらくの間スリープしてから、独自のホスト名を出力するジョブを作成します。

`hellojob.sh` というファイルを次の内容で作成します。

```
#!/bin/bash
sleep 30
echo "Hello World from $(hostname)"
```

次に、qsub を使用してジョブを送信し、実行されることを確認します。

```
$ qsub hellojob.sh
Your job 1 ("hellojob.sh") has been submitted
```

これで、キューを表示してジョブのステータスを確認できます。

```
$ qstat
job-ID  prior   name       user         state submit/start at     queue
          slots ja-task-ID
-----------------------------------------------------------------------------------------------
      1 0.55500 hellojob.s ec2-user     r     03/24/2015 22:23:48
 all.q@ip-192-168-1-125.us-west     1
```

ジョブが現在実行ステータスであることが出力に示されます。ジョブが終了するまで 30 秒間待ってから、もう一度 `qstat` を実行します。

```
$ qstat
$
```

キューにはジョブがないため、現在のディレクトリで出力を確認できます。

```
$ ls -l
total 8
-rw-rw-r-- 1 ec2-user ec2-user 48 Mar 24 22:34 hellojob.sh
-rw-r--r-- 1 ec2-user ec2-user  0 Mar 24 22:34 hellojob.sh.e1
-rw-r--r-- 1 ec2-user ec2-user 34 Mar 24 22:34 hellojob.sh.o1
```

ジョブスクリプトの e1 および o1 ファイルが出力で確認できます。e1 ファイルは空であるため、stderr への出力はありません。o1 ファイルを表示すると、ジョブの出力が表示されます。

```
$ cat hellojob.sh.o1
Hello World from ip-192-168-1-125
```

また、出力には、ジョブがインスタンス ip-192-168-1-125 上で正常に実行されていることも示されています。

クラスターの作成と使用の詳細については、「ベストプラクティス」を参照してください。

カスタム AMI AWS ParallelCluster の構築

> **Important**
>
> AWS ParallelClusterをカスタマイズするためのアプローチとして、カスタム AMI を構築することはお勧めしません。
> これは、独自の AMI を構築した後、 の今後のリリースで更新やバグ修正を受け取らなくなるためです AWS ParallelCluster。さらに、カスタム AMI を構築する場合は、新しい AWS ParallelCluster リリースごとにカスタム AMI の作成に使用した手順を繰り返す必要があります。

さらに読む前に、まずカスタムブートストラップアクションセクションをチェックして、変更をスクリプト化して今後の AWS ParallelCluster リリースでサポートできるかどうかを判断することをお勧めします。

カスタム AMI の構築は理想的ではありませんが (前述の理由により) 、 のカスタム AMI の構築 AWS ParallelCluster が必要なシナリオはまだあります。このチュートリアルでは、これらのシナリオに対応するカスタム AMI を構築する手順を説明します。

> **Note**
>
> AWS ParallelCluster バージョン 2.6.1 以降では、ノードの起動時にほとんどのインストールレシピがデフォルトでスキップされます。これは、起動時間を短縮するためです。起動時間の短縮よりも下位互換性の向上を優先させるために、すべてのインストールレシピを実行するには、extra_json 設定の cluster キーに "skip_install_recipes" : "no" を追加します。例 :
>
> ```
> extra_json = { "cluster" : { "skip_install_recipes" : "no" } }
> ```

前提条件

- AWS ParallelCluster がインストールされます。
- AWS CLI がインストールされ、設定されています。
- EC2 キーペアがある。
- pcluster CLI の実行に必要なアクセス許可を持つ IAM ロールがある。

AMI AWS ParallelCluster をカスタマイズする方法

カスタム AMI AWS ParallelCluster を使用するには、次のセクションで説明する 3 つの方法があります。これら 3 つの方法のうち、2 つの方法では、 AWS アカウントアカウントで使用できる新しい AMI を構築する必要があります。3 番目の方法 (実行時にカスタム AMI を使用する) では、事前に何かを構築する必要はありませんが、デプロイにはリスクが伴います。お客様のニーズに最適なものを選択してください。

AMI を変更する

これが最も安全で推奨される方法です。ベース AMI AWS ParallelCluster は新しいリリースで更新されることが多いため、この AMI には、インストールおよび設定時に が機能 AWS ParallelCluster するために必要なすべてのコンポーネントがあります。これは、ベースとして起動できます。

New EC2 console

1. AWS ParallelCluster AMI リストで、使用する特定の に対応する AMI を見つけ AWS リージョン ます。選択した AMI リストは、 AWS ParallelCluster 使用する のバージョンと一致する必要があります。`pcluster version` を実行して、バージョンを確認します。 AWS ParallelCluster バージョン 2.11.9 については、https://github.com/aws/aws-parallelcluster/blob/v2.11.9/amis.txt を参照してください。別のバージョンを選択するには、同じリンクを使用して [Tag: 2.11.9] ボタンを選択し、[タグ] タブを選択して、適切なバージョンを選択します。
2. にサインイン AWS Management Console し、https://console.aws.amazon.com/ec2/ で Amazon EC2 コンソールを開きます。
3. [Amazon EC2 ダッシュボード] で、[インスタンスを起動] を選択します。
4. アプリケーションイメージと OS イメージで、より多くの AMIs を参照し、コミュニティ AMIs に移動し、 の AWS ParallelCluster AMI ID AWS リージョン を検索ボックスに入力します。
5. AMI を[選択]し、[インスタンスタイプ] とプロパティを選択して、[キーペア] と [インスタンスを起動] を選択します。
6. OS ユーザーと SSH キーを使用してインスタンスにログインします。詳細については、[インスタンス] に移動し、新しいインスタンス、[接続] の順に選択します。
7. 必要に応じてインスタンスをカスタマイズします。
8. 次のコマンドを実行して、インスタンスを AMI 作成用に準備します。

```
sudo /usr/local/sbin/ami_cleanup.sh
```

9. [インスタンス] に移動し、新しいインスタンスを選択し、[インスタンスの状態]、[インスタンスの停止] の順に選択します。
10. EC2 コンソールまたは AWS CLI create-image を使用して、インスタンスから新しい AMI を作成します。

    EC2 コンソールから

    a. ナビゲーションペインで、[Instances (インスタンス)] を選択します。

    b. 作成および変更したインスタンスを選択します。

    c. [アクション] で、[イメージとテンプレート]、[イメージの作成] の順に選択します。

    d. [Create Image] を選択します。
11. クラスター設定の [custom_ami] フィールドに AMI ID を入力します。

Old EC2 console

1. AWS ParallelCluster AMI リストで、使用する特定の に対応する AMI を見つけ AWS リージョン ます。選択した AMI リストは、 AWS ParallelCluster 使用する のバージョンと一致する必要があります。`pcluster version` を実行して、バージョンを確認します。 AWS ParallelCluster バージョン 2.11.9 については、https://github.com/aws/aws-parallelcluster/blob/v2.11.9/amis.txt を参照してください。別のバージョンを選択するには、同じリンクを使用して [Tag: 2.11.9] ボタンを選択し、[タグ] タブを選択して、適切なバージョンを選択します。
2. にサインイン AWS Management Console し、https://console.aws.amazon.com/ec2/ で Amazon EC2 コンソールを開きます。
3. [Amazon EC2 ダッシュボード] で、[インスタンスを起動] を選択します。
4. コミュニティ AMIs を選択し、AMI ID AWS ParallelCluster を検索して選択します。
5. インスタンスタイプを選択し、[次へ: インスタンスの詳細の設定] または [確認と起動] を選択してインスタンスを起動します。
6. [起動] を選択し、[キーペア] を選択して、[インスタンスを起動] を選択します。
7. OS ユーザーと SSH キーを使用してインスタンスにログインします。詳細については、[インスタンス] に移動し、新しいインスタンス、[接続] の順に選択します。
8. 必要に応じてインスタンスをカスタマイズします。
9. 次のコマンドを実行して、インスタンスを AMI 作成用に準備します。

```
sudo /usr/local/sbin/ami_cleanup.sh
```

10. [インスタンス] に移動して、新しいインスタンスを選択し、[インスタンスの状態]、[停止] の順に選択します。
11. EC2 コンソールまたは AWS CLI create-image を使用して、インスタンスから新しい AMI を作成します。

    EC2 コンソールから

    a. ナビゲーションペインで、[Instances (インスタンス)] を選択します。

    b. 作成および変更したインスタンスを選択します。

    c. [アクション] で、[イメージ]、[イメージの作成] の順に選択します。

    d. [Create Image] を選択します。
12. クラスター設定の [custom_ami] フィールドに AMI ID を入力します。

カスタム AMI AWS ParallelCluster の構築

カスタマイズされた AMI とソフトウェアがすでに導入されている場合は、 AWS ParallelCluster で必要な変更を適用することができます。

1. AWS ParallelCluster CLI とともに、ローカルシステムに以下をインストールします。
   - Packer: Packer ウェブサイトから OS の最新バージョンを探してインストールします。 バージョンは 1.4.0 以上である必要がありますが、最新のバージョンを推奨します。`packer` コマンドがパスにあるかどうかを確認します。

   > **Note**
   > AWS ParallelCluster バージョン 2.8.0 以前は、 を使用するには Berkshelf (を使用してインストール`gem install berkshelf`) が必要でした`pcluster createami`。

2. Packer がユーザーに代わって AWS API オペレーションを呼び出すことができるように、 AWS アカウント 認証情報を設定します。Packer が動作するために必要な最小限のパーミッションのセットは、Packer ドキュメントの「Amazon AMI Builder」の「IAM Task or Instance Role」(トピックの IAM タスクまたはインスタンスのロール) セクションに記載されています。
3. CLI AWS ParallelCluster `createami`の コマンドを使用して、ベースとして指定したものから AMI AWS ParallelCluster を構築します。

   ```
   pcluster createami --ami-id <BASE_AMI> --os <BASE_AMI_OS>
   ```

   > **Important**
   > `createami` コマンドには、実行中のクラスターの AWS ParallelCluster AMI を *<BASE_AMI>* として使用しないでください。そうしなければ、コマンドは失敗します。

   その他のパラメータについては、「`pcluster createami`」を参照してください。
4. ステップ 4 のコマンドは Packer を実行し、具体的には以下の処理を実行します。
   a. 提供されているベース AMI を使用してインスタンスを起動します。
   b. ク AWS ParallelCluster ックブックをインスタンスに適用して、関連するソフトウェアをインストールし、その他の必要な設定タスクを実行します。
   c. インスタンスを停止します。

d. インスタンスから新しい AMI を作成します。

e. AMI の作成後にインスタンスを終了します。

f. クラスターの作成に使用する新しい AMI ID 文字列を出力します。

5. クラスターを作成するには、クラスター設定内の [custom_ami] フィールドに AMI ID を入力します。

> **Note**
>
> カスタム AMI AWS ParallelCluster の構築に使用されるインスタンスタイプは ですt2.xlarge。このインスタンスタイプは AWS 無料利用枠の対象ではないため、この AMI の構築時に作成されたインスタンスに対して課金されます。

実行時にカスタム AMI を使用する

> **Warning**
>
> と互換性のない AMI を使用するリスクを回避するには AWS ParallelCluster、この方法を使用しないことをお勧めします。
> ランタイムにテストされていない可能性のある AMIs を使用してコンピューティングノードを起動すると、 AWS ParallelClusterの必要なソフトウェアのランタイムインストールと互換性がない AWS ParallelCluster と、 が動作を停止する可能性があります。

事前に何も作成しない場合は、AMI を使用して、その AMI AWS ParallelCluster から を作成できます。

この方法では、クラスターの作成 AWS ParallelCluster 時に に必要なすべてのソフトウェアをインストールする必要があるため、 の作成 AWS ParallelCluster に時間がかかります。さらに、スケールアップにも時間がかかります。

- クラスター設定内の [custom_ami] フィールドに AMI ID を入力します。

AWS ParallelCluster とス**awsbatch**ケジューラを使用した MPI ジョブの実行

このチュートリアルでは awsbatch をスケジューラとして使用して MPI ジョブを実行する方法を説明します。

前提条件

- AWS ParallelCluster がインストールされます。
- AWS CLI がインストールされ、設定されています。
- EC2 キーペアがある。
- pcluster CLI の実行に必要なアクセス許可を持つ IAM ロールがある。

クラスターの作成

まず、スケジューラとして awsbatch を使用するクラスター用の設定を作成しましょう。vpc セクションと key_name フィールドの不足しているデータを、設定時に作成したリソースに挿入してください。

```
[global]
sanity_check = true

[aws]
aws_region_name = us-east-1

[cluster awsbatch]
base_os = alinux
# Replace with the name of the key you intend to use.
key_name = key-#######
vpc_settings = my-vpc
scheduler = awsbatch
compute_instance_type = optimal
min_vcpus = 2
desired_vcpus = 2
max_vcpus = 24

[vpc my-vpc]
# Replace with the id of the vpc you intend to use.
vpc_id = vpc-#######
```

```
# Replace with id of the subnet for the Head node.
master_subnet_id = subnet-#######
# Replace with id of the subnet for the Compute nodes.
# A NAT Gateway is required for MNP.
compute_subnet_id = subnet-#######
```

これでクラスターの作成を開始できます。クラスターの名前を *awsbatch-tutorial* とします。

```
$ pcluster create -c /path/to/the/created/config/aws_batch.config -t awsbatch awsbatch-tutorial
```

クラスターが作成されると、次のような出力が表示されます。

```
Beginning cluster creation for cluster: awsbatch-tutorial
Creating stack named: parallelcluster-awsbatch
Status: parallelcluster-awsbatch - CREATE_COMPLETE
MasterPublicIP: 54.160.xxx.xxx
ClusterUser: ec2-user
MasterPrivateIP: 10.0.0.15
```

ヘッドノードにログインする

AWS ParallelCluster Batch CLI コマンドはすべて、 AWS ParallelCluster がインストールされているクライアントマシンで使用できます。ただし、ヘッドノードに SSH 接続し、そこからジョブを送信します。これにより、ヘッドと AWS Batch ジョブを実行するすべての Docker インスタンス間で共有される NFS ボリュームを活用できます。

SSH pem ファイルを使用してヘッドノードにログインします。

```
$ pcluster ssh awsbatch-tutorial -i /path/to/keyfile.pem
```

ログインしたら、 コマンド`awsbqueues`と を実行して`awsbhosts`、設定された AWS Batch キューと実行中の Amazon ECS インスタンスを表示します。

```
[ec2-user@ip-10-0-0-111 ~]$ awsbqueues
jobQueueName                      status
---------------------------------  --------
parallelcluster-awsbatch-tutorial  VALID
```

```
[ec2-user@ip-10-0-0-111 ~]$ awsbhosts
ec2InstanceId        instanceType    privateIpAddress    publicIpAddress
 runningJobs
-------------------  --------------  ------------------  -----------------
 -------------
i-0d6a0c8c560cd5bed  m4.large        10.0.0.235          34.239.174.236
 0
```

出力からわかるように、1 つの実行中のホストを選択します。これは、設定で `min_vcpus` に選択した値が原因です。 AWS Batch キューとホストに関する追加の詳細を表示する場合は、 コマンドに -dフラグを追加します。

を使用して最初のジョブを実行する AWS Batch

MPI に移行する前に、しばらくの間スリープしてから独自のホスト名を出力し、パラメータとして渡された名前にあいさつをするダミージョブを作成しましょう。

以下の内容の「hellojob.sh」という名前のファイルを作成します。

```
#!/bin/bash

sleep 30
echo "Hello $1 from $HOSTNAME"
echo "Hello $1 from $HOSTNAME" > "/shared/secret_message_for_${1}_by_
${AWS_BATCH_JOB_ID}"
```

次に、awsbsub を使用してジョブを送信し、実行されることを確認します。

```
$ awsbsub -jn hello -cf hellojob.sh Luca
Job 6efe6c7c-4943-4c1a-baf5-edbfeccab5d2 (hello) has been submitted.
```

キューを表示してジョブのステータスを確認します。

```
$ awsbstat
jobId                                jobName      status    startedAt
 stoppedAt    exitCode
------------------------------------  -----------  --------  -------------------
 -----------  ----------
6efe6c7c-4943-4c1a-baf5-edbfeccab5d2  hello        RUNNING   2018-11-12 09:41:29  -
        -
```

ジョブの詳細情報が出力に示されます。

```
$ awsbstat 6efe6c7c-4943-4c1a-baf5-edbfeccab5d2
jobId                    : 6efe6c7c-4943-4c1a-baf5-edbfeccab5d2
jobName                  : hello
createdAt                : 2018-11-12 09:41:21
startedAt                : 2018-11-12 09:41:29
stoppedAt                : -
status                   : RUNNING
statusReason             : -
jobDefinition            : parallelcluster-exampleBatch:1
jobQueue                 : parallelcluster-exampleBatch
command                  : /bin/bash -c 'aws s3 --region us-east-1 cp
 s3://amzn-s3-demo-bucket/batch/job-hellojob_sh-1542015680924.sh /tmp/batch/job-
hellojob_sh-1542015680924.sh; bash /tmp/batch/job-hellojob_sh-1542015680924.sh Luca'
exitCode                 : -
reason                   : -
vcpus                    : 1
memory[MB]               : 128
nodes                    : 1
logStream                : parallelcluster-exampleBatch/default/c75dac4a-5aca-4238-
a4dd-078037453554
log                      : https://console.aws.amazon.com/cloudwatch/home?region=us-
east-1#logEventViewer:group=/aws/batch/job;stream=parallelcluster-exampleBatch/default/
c75dac4a-5aca-4238-a4dd-078037453554
--------------------------
```

ジョブの現在の状態は、RUNNING です。ジョブが終了するまで 30 秒間待ってから、もう一度 awsbstat を実行します。

```
$ awsbstat
jobId                                    jobName      status    startedAt
 stoppedAt    exitCode
------------------------------------  -----------  --------  -------------------
 -----------  ----------
```

これで、ジョブが SUCCEEDED ステータスになりました。

```
$ awsbstat -s SUCCEEDED
jobId                                    jobName      status     startedAt
 stoppedAt              exitCode
```

```
------------------------------------  -----------  ---------  -------------------
 -------------------  ----------
6efe6c7c-4943-4c1a-baf5-edbfeccab5d2  hello        SUCCEEDED  2018-11-12 09:41:29
 2018-11-12 09:42:00           0
```

現在キューにはジョブがないため、awsbout コマンドで出力を確認できます。

```
$ awsbout 6efe6c7c-4943-4c1a-baf5-edbfeccab5d2
2018-11-12 09:41:29: Starting Job 6efe6c7c-4943-4c1a-baf5-edbfeccab5d2
download: s3://amzn-s3-demo-bucket/batch/job-hellojob_sh-1542015680924.sh to tmp/batch/
job-hellojob_sh-1542015680924.sh
2018-11-12 09:42:00: Hello Luca from ip-172-31-4-234
```

ジョブはインスタンス「ip-172-31-4-234」で正常に実行されたことがわかります。

/shared ディレクトリを調べると、秘密のメッセージが表示されます。

このチュートリアルに含まれない利用可能な機能をすべて確認するには、「AWS ParallelCluster バッチ CLI のドキュメント」を参照してください。チュートリアルを進める準備ができたら、MPI ジョブを送信する方法を確認しましょう。

マルチノード並列環境で MPI ジョブを実行する

ヘッドノードにログインしたまま、mpi_hello_world.c という名前のファイルを /shared ディレクトリに作成します。次の MPI プログラムをファイルに追加します。

```
// Copyright 2011 www.mpitutorial.com
//
// An intro MPI hello world program that uses MPI_Init, MPI_Comm_size,
// MPI_Comm_rank, MPI_Finalize, and MPI_Get_processor_name.
//
#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();
}
```

次のコードを submit_mpi.sh として保存します。

```
#!/bin/bash
echo "ip container: $(/sbin/ip -o -4 addr list eth0 | awk '{print $4}' | cut -d/ -f1)"
echo "ip host: $(curl -s "http://169.254.169.254/latest/meta-data/local-ipv4")"

# get shared dir
IFS=',' _shared_dirs=(${PCLUSTER_SHARED_DIRS})
_shared_dir=${_shared_dirs[0]}
_job_dir="${_shared_dir}/${AWS_BATCH_JOB_ID%#*}-${AWS_BATCH_JOB_ATTEMPT}"
_exit_code_file="${_job_dir}/batch-exit-code"

if [[ "${AWS_BATCH_JOB_NODE_INDEX}" -eq  "${AWS_BATCH_JOB_MAIN_NODE_INDEX}" ]]; then
    echo "Hello I'm the main node $HOSTNAME! I run the mpi job!"

    mkdir -p "${_job_dir}"

    echo "Compiling..."
    /usr/lib64/openmpi/bin/mpicc -o "${_job_dir}/mpi_hello_world" "${_shared_dir}/
mpi_hello_world.c"

    echo "Running..."
```

```
    /usr/lib64/openmpi/bin/mpirun --mca btl_tcp_if_include eth0 --allow-run-as-root --
machinefile "${HOME}/hostfile" "${_job_dir}/mpi_hello_world"

    # Write exit status code
    echo "0" > "${_exit_code_file}"
    # Waiting for compute nodes to terminate
    sleep 30
else
    echo "Hello I'm the compute node $HOSTNAME! I let the main node orchestrate the mpi
 processing!"
    # Since mpi orchestration happens on the main node, we need to make sure the
 containers representing the compute
    # nodes are not terminated. A simple trick is to wait for a file containing the
 status code to be created.
    # All compute nodes are terminated by AWS Batch if the main node exits abruptly.
    while [ ! -f "${_exit_code_file}" ]; do
        sleep 2
    done
    exit $(cat "${_exit_code_file}")
fi
```

これで、最初の MPI ジョブを送信して 3 つのノードで同時に実行する準備ができました。

```
$ awsbsub -n 3 -cf submit_mpi.sh
```

ジョブのステータスをモニタリングし、それが RUNNING ステータスになるまで待ちます。

```
$ watch awsbstat -d
```

ジョブが RUNNING ステータスになったら、その出力を確認することができます。メインモードの出力を表示するには、#0 をジョブ ID に追加します。コンピューティングノードの出力を表示するには、#1 および #2 を使用します。

```
[ec2-user@ip-10-0-0-111 ~]$ awsbout -s 5b4d50f8-1060-4ebf-ba2d-1ae868bbd92d#0
2018-11-27 15:50:10: Job id: 5b4d50f8-1060-4ebf-ba2d-1ae868bbd92d#0
2018-11-27 15:50:10: Initializing the environment...
2018-11-27 15:50:10: Starting ssh agents...
2018-11-27 15:50:11: Agent pid 7
2018-11-27 15:50:11: Identity added: /root/.ssh/id_rsa (/root/.ssh/id_rsa)
2018-11-27 15:50:11: Mounting shared file system...
2018-11-27 15:50:11: Generating hostfile...
```

```
2018-11-27 15:50:11: Detected 1/3 compute nodes. Waiting for all compute nodes to
 start.
2018-11-27 15:50:26: Detected 1/3 compute nodes. Waiting for all compute nodes to
 start.
2018-11-27 15:50:41: Detected 1/3 compute nodes. Waiting for all compute nodes to
 start.
2018-11-27 15:50:56: Detected 3/3 compute nodes. Waiting for all compute nodes to
 start.
2018-11-27 15:51:11: Starting the job...
download: s3://amzn-s3-demo-bucket/batch/job-submit_mpi_sh-1543333713772.sh to tmp/
batch/job-submit_mpi_sh-1543333713772.sh
2018-11-27 15:51:12: ip container: 10.0.0.180
2018-11-27 15:51:12: ip host: 10.0.0.245
2018-11-27 15:51:12: Compiling...
2018-11-27 15:51:12: Running...
2018-11-27 15:51:12: Hello I'm the main node! I run the mpi job!
2018-11-27 15:51:12: Warning: Permanently added '10.0.0.199' (RSA) to the list of known
 hosts.
2018-11-27 15:51:12: Warning: Permanently added '10.0.0.147' (RSA) to the list of known
 hosts.
2018-11-27 15:51:13: Hello world from processor ip-10-0-0-180.ec2.internal, rank 1 out
 of 6 processors
2018-11-27 15:51:13: Hello world from processor ip-10-0-0-199.ec2.internal, rank 5 out
 of 6 processors
2018-11-27 15:51:13: Hello world from processor ip-10-0-0-180.ec2.internal, rank 0 out
 of 6 processors
2018-11-27 15:51:13: Hello world from processor ip-10-0-0-199.ec2.internal, rank 4 out
 of 6 processors
2018-11-27 15:51:13: Hello world from processor ip-10-0-0-147.ec2.internal, rank 2 out
 of 6 processors
2018-11-27 15:51:13: Hello world from processor ip-10-0-0-147.ec2.internal, rank 3 out
 of 6 processors

[ec2-user@ip-10-0-0-111 ~]$ awsbout -s 5b4d50f8-1060-4ebf-ba2d-1ae868bbd92d#1
2018-11-27 15:50:52: Job id: 5b4d50f8-1060-4ebf-ba2d-1ae868bbd92d#1
2018-11-27 15:50:52: Initializing the environment...
2018-11-27 15:50:52: Starting ssh agents...
2018-11-27 15:50:52: Agent pid 7
2018-11-27 15:50:52: Identity added: /root/.ssh/id_rsa (/root/.ssh/id_rsa)
2018-11-27 15:50:52: Mounting shared file system...
2018-11-27 15:50:52: Generating hostfile...
2018-11-27 15:50:52: Starting the job...
download: s3://amzn-s3-demo-bucket/batch/job-submit_mpi_sh-1543333713772.sh to tmp/
batch/job-submit_mpi_sh-1543333713772.sh
```

```
2018-11-27 15:50:53: ip container: 10.0.0.199
2018-11-27 15:50:53: ip host: 10.0.0.227
2018-11-27 15:50:53: Compiling...
2018-11-27 15:50:53: Running...
2018-11-27 15:50:53: Hello I'm a compute node! I let the main node orchestrate the mpi
 execution!
```

これでジョブが正常に完了したことを確認できるようになりました。

```
[ec2-user@ip-10-0-0-111 ~]$ awsbstat -s ALL
jobId                                   jobName        status     startedAt
 stoppedAt            exitCode
------------------------------------  -------------  ---------  -------------------
 -------------------  ----------
5b4d50f8-1060-4ebf-ba2d-1ae868bbd92d  submit_mpi_sh  SUCCEEDED  2018-11-27 15:50:10
 2018-11-27 15:51:26  -
```

注: ジョブを終了する場合は、終了前に `awsbkill` コマンドを使用します。

カスタム KMS キーを使用したディスク暗号化

AWS ParallelCluster では、設定オプション (`ebs_kms_key_id` および `fsx_kms_key_id`) をサポートしています。これらのオプションを使用すると、Amazon EBS ディスク暗号化または FSx for Lustre のカスタム AWS KMS キーを提供できます。これらを使用するには、`ec2_iam_role` を指定します。

クラスターを作成するには、 AWS KMS キーがクラスターのロールの名前を知っている必要があります。これにより、カスタムの `ec2_iam_role` が必要になるため、クラスターの作成で作成されたロールを使用できなくなります。

前提条件

- AWS ParallelCluster がインストールされます。
- AWS CLI がインストールされ、設定されています。
- EC2 キーペアがある。
- `pcluster` CLI の実行に必要なアクセス許可を持つ IAM ロールがある。

ロールの作成

まず、次のようにポリシーを作成します。

1. IAM コンソール (https://console.aws.amazon.com/iam/home) に移動します。
2. [Policy] (ポリシー) の [Create policy] (ポリシーの作成) で、[JSON] タブをクリックします。
3. ポリシーの本文として、[Instance Policy] (インスタンスポリシー) に貼り付けます。*<AWS ACCOUNT ID>* と *<REGION>* をすべて置き換えます。
4. ポリシー `ParallelClusterInstancePolicy` に名前を付け、[Create Policy] (ポリシーの作成) をクリックします。

次にロールを作成します。

1. Roles] (ロール) で、ロールを作成します。
2. 信頼されたエンティティとして [EC2] をクリックします
3. [Permissions] (アクセス許可) で、先ほど作成した `ParallelClusterInstancePolicy` ロールを検索してアタッチします。
4. ロール `ParallelClusterInstanceRole` に名前を付け、[Create Role] (ロールの作成) をクリックします。

キーのアクセス許可を付与する

AWS KMS コンソール > カスタマーマネージドキー > キーのエイリアスまたはキー ID をクリックします。

[Key policy] (キーポリシー) タブの下にある[Key users] (キーユーザー) ボックスの [ADD] (追加) ボタンをクリックし、先ほど作成した [ParallelClusterInstanceRole] を検索します。アタッチします。

クラスターの作成

これで、クラスターが作成されます。以下は、暗号化された `Raid 0` ドライブを含むクラスターの例です。

```
[cluster default]
...
raid_settings = rs
```

```
ec2_iam_role = ParallelClusterInstanceRole

[raid rs]
shared_dir = raid
raid_type = 0
num_of_raid_volumes = 2
volume_size = 100
encrypted = true
ebs_kms_key_id = xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

以下は，FSx for Lustre ファイルシステムを用いた例です。

```
[cluster default]
...
fsx_settings = fs
ec2_iam_role = ParallelClusterInstanceRole

[fsx fs]
shared_dir = /fsx
storage_capacity = 3600
imported_file_chunk_size = 1024
export_path = s3://bucket/folder
import_path = s3://bucket
weekly_maintenance_start_time = 1:00:00
fsx_kms_key_id = xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

同様の構成は、Amazon EBS および Amazon FSx ベースのファイルシステムにも適用されます。

マルチキューモードのチュートリアル

マルチキューモードで AWS ParallelCluster でジョブを実行する

このチュートリアルでは、 AWS ParallelCluster を使用して で最初の Hello World ジョブを実行する手順を説明しますマルチキューモード。

前提条件

- AWS ParallelCluster がインストールされます。
- AWS CLI がインストールされ、設定されています。

- EC2 キーペアがある。
- `pcluster` CLI の実行に必要なアクセス許可を持つ IAM ロールがある。

> **Note**
> マルチキューモードは AWS ParallelCluster 、バージョン 2.9.0 以降でのみサポートされています。

クラスターを設定する

まず、次のコマンドを実行して AWS ParallelCluster 、 が正しくインストールされていることを確認します。

```
$ pcluster version
```

`pcluster version` の詳細については、「`pcluster version`」を参照してください。

このコマンドは、実行中の のバージョンを返します AWS ParallelCluster。

次に、`pcluster configure` を実行して、基本的な設定ファイルを生成します。このコマンドに続くすべてのプロンプトに従います。

```
$ pcluster configure
```

`pcluster configure` コマンドの詳細については、「`pcluster configure`」を参照してください。

この手順が完了すると、`~/.parallelcluster/config` の下に基本的な設定ファイルが作成されます。このファイルには、基本的なクラスター構成と VPC セクションが含まれています。

このチュートリアルの次の部分では、新しく作成した設定を変更して、複数のキューを持つクラスターを起動する方法について説明します。

> **Note**
> このチュートリアルで使用される一部のインスタンスは、無料利用枠の対象外です。

このチュートリアルでは、以下の構成を使用します。

```
[global]
update_check = true
sanity_check = true
cluster_template = multi-queue

[aws]
aws_region_name = <Your AWS #####>

[scaling demo]
scaledown_idletime = 5              # optional, defaults to 10 minutes

[cluster multi-queue-special]
key_name = < Your key name >
base_os = alinux2                   # optional, defaults to alinux2
scheduler = slurm
master_instance_type = c5.xlarge    # optional, defaults to t2.micro
vpc_settings = <Your VPC section>
scaling_settings = demo             # optional, defaults to no custom scaling settings
queue_settings = efa,gpu

[cluster multi-queue]
key_name = <Your SSH key name>
base_os = alinux2                   # optional, defaults to alinux2
scheduler = slurm
master_instance_type = c5.xlarge    # optional, defaults to t2.micro
vpc_settings = <Your VPC section>
scaling_settings = demo
queue_settings = spot,ondemand

[queue spot]
compute_resource_settings = spot_i1,spot_i2
compute_type = spot                 # optional, defaults to ondemand

[compute_resource spot_i1]
instance_type = c5.xlarge
min_count = 0                       # optional, defaults to 0
max_count = 10                      # optional, defaults to 10

[compute_resource spot_i2]
instance_type = t2.micro
min_count = 1
initial_count = 2
```

```
[queue ondemand]
compute_resource_settings = ondemand_i1
disable_hyperthreading = true       # optional, defaults to false

[compute_resource ondemand_i1]
instance_type = c5.2xlarge
```

クラスターを作成する

ここでは、マルチキューモードのクラスターを作成する方法について詳しく説明します。

まず、クラスターに multi-queue-hello-world という名前を付け、前のセクションで定義した multi-queue クラスターのセクションに従ってクラスタを作成します。

```
$ pcluster create multi-queue-hello-world -t multi-queue
```

pcluster create の詳細については、「pcluster create」を参照してください。

クラスターが作成されると、次の出力が表示されます。

```
Beginning cluster creation for cluster: multi-queue-hello-world
Creating stack named: parallelcluster-multi-queue-hello-world
Status: parallelcluster-multi-queue-hello-world - CREATE_COMPLETE
MasterPublicIP: 3.130.xxx.xx
ClusterUser: ec2-user
MasterPrivateIP: 172.31.xx.xx
```

メッセージ CREATE_COMPLETE は、クラスターが正しく作成されたことを示します。また、ヘッドノードのパブリックおよびプライベート IP アドレスも出力されます。

ヘッドノードにログインする

プライベート SSH キーファイルを使用して、ヘッドノードにログインします。

```
$ pcluster ssh multi-queue-hello-world -i ~/path/to/keyfile.pem
```

pcluster ssh の詳細については、「pcluster ssh」を参照してください。

ログインしたら、sinfo スケジューラキューがセットアップされ設定されていることを確認します。

sinfo の詳細については、「Slurm ドキュメンテーション」の「sinfo」を参照してください。

```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
ondemand     up   infinite     10  idle~ ondemand-dy-c52xlarge-[1-10]
spot*        up   infinite     18  idle~ spot-dy-c5xlarge-[1-10],spot-dy-t2micro-[2-9]
spot*        up   infinite      2   idle spot-dy-t2micro-1,spot-st-t2micro-1
```

この出力では、idle 状態の t2.micro コンピューティングノードが 2 台、クラスター内で利用可能であることがわかります。

> **Note**
>
> - spot-st-t2micro-1 は名前に st を持つ静的ノードです。このノードは常に利用可能で、クラスター構成の min_count = 1 に対応しています。
> - spot-dy-t2micro-1 は名前に dy を持つ動的ノードです。このノードは、クラスター構成上、initial_count - min_count = 1 に対応しているため、現在利用可能です。このノードは、あなたが設定した scaledown_idletime の 5 分後にスケールダウンします。

他のノードは、ノードの状態に ~ サフィックスが表示されてすべて省電力状態になり、EC2 インスタンスはそれらのノードをサポートしません。デフォルトのキューは、キュー名の後に * というサフィックスが付いていますので、spot がデフォルトのジョブキューとなります。

マルチキューモードでジョブを実行する

次に、しばらくの間、ジョブをスリープ状態にして実行してみます。このジョブは、後で自分のホスト名を出力します。このスクリプトが現在のユーザーで実行できることを確認します。

```
$ cat hellojob.sh
#!/bin/bash
sleep 30
echo "Hello World from $(hostname)"

$ chmod +x hellojob.sh
$ ls -l hellojob.sh
-rwxrwxr-x 1 ec2-user ec2-user 57 Sep 23 21:57 hellojob.sh
```

sbatch コマンドを使用してジョブを送信します。このジョブに対して -N 2 オプションで 2 つのノードを要求し、ジョブが正常に送信されることを確認します。sbatch の詳細については、「Slum ドキュメンテーション」の「sbatch」を参照してください。

```
$ sbatch -N 2 --wrap "srun hellojob.sh"
Submitted batch job 2
```

squeue コマンドでは、キューの表示やジョブの状態を確認することができます。なお、特定のキューを指定していないため、デフォルトのキュー (spot) が使用されます。squeue の詳細については、「Slurm ドキュメンテーション」の「squeue」を参照してください。

```
$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                 2      spot     wrap ec2-user  R       0:10      2 spot-dy-
t2micro-1,spot-st-t2micro-1
```

ジョブが現在実行ステータスであることが出力に示されます。ジョブが終了するまで 30 秒間待ってから、もう一度 squeue を実行します。

```
$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
```

キュー内のジョブがすべて終了したので、カレントディレクトリにある出力ファイル slurm-2.out を探します。

```
$ cat slurm-2.out
Hello World from spot-dy-t2micro-1
Hello World from spot-st-t2micro-1
```

この出力では、spot-st-t2micro-1 と spot-st-t2micro-2 の各ノードでジョブが正常に実行されたことも示されています。

次のコマンドで特定のインスタンスに制約条件を指定して、同じジョブを送信します。

```
$ sbatch -N 3 -p spot -C "[c5.xlarge*1&t2.micro*2]" --wrap "srun hellojob.sh"
Submitted batch job 3
```

これらのパラメータを sbatch に使用しました。

- -N 3 — 3 つのノードを要求します

- `-p spot` — ジョブを spot キューへ送信します また、`-p ondemand` を指定して ondemand キューにジョブを送信することもできます。
- `-C "[c5.xlarge*1&t2.micro*2]"` — このジョブの特定のノード制約を指定します。これは、このジョブに使用される 1 台の `c5.xlarge` ノードと 2 台の `t2.micro` ノードを要求します。

`sinfo` コマンドを実行して、ノードとキューを表示します。(のキュー AWS ParallelCluster は、 のパーティションと呼ばれます) Slurm。

```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
ondemand     up   infinite     10  idle~ ondemand-dy-c52xlarge-[1-10]
spot*        up   infinite      1   mix# spot-dy-c5xlarge-1
spot*        up   infinite     17  idle~ spot-dy-c5xlarge-[2-10],spot-dy-t2micro-[2-9]
spot*        up   infinite      2  alloc spot-dy-t2micro-1,spot-st-t2micro-1
```

ノードの電源が入っています。これは、ノードの状態に # というサフィックスが付いていることで示されます。squeue コマンドを実行して、クラスター内のジョブに関する情報を表示します。

```
$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                 3      spot     wrap ec2-user CF       0:04      3 spot-dy-
c5xlarge-1,spot-dy-t2micro-1,spot-st-t2micro-1
```

ジョブは `CF` (CONFIGURING) の状態で、インスタンスがスケールアップしてクラスターに参加するのを待っています。

約 3 分後、ノードが利用可能になり、ジョブは `R` (RUNNING) の状態になります。

```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
ondemand     up   infinite     10  idle~ ondemand-dy-c52xlarge-[1-10]
spot*        up   infinite     17  idle~ spot-dy-c5xlarge-[2-10],spot-dy-t2micro-[2-9]
spot*        up   infinite      1    mix spot-dy-c5xlarge-1
spot*        up   infinite      2  alloc spot-dy-t2micro-1,spot-st-t2micro-1
$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                 3      spot     wrap ec2-user  R       0:04      3 spot-dy-
c5xlarge-1,spot-dy-t2micro-1,spot-st-t2micro-1
```

ジョブが終了すると、3 つのノードはすべて `idle` の状態になります。

```
$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
ondemand     up   infinite     10  idle~ ondemand-dy-c52xlarge-[1-10]
spot*        up   infinite     17  idle~ spot-dy-c5xlarge-[2-10],spot-dy-t2micro-[2-9]
spot*        up   infinite      3   idle spot-dy-c5xlarge-1,spot-dy-t2micro-1,spot-st-
t2micro-1
```

そして、キューにジョブが残っていない状態になってから、ローカルディレクトリに slurm-3.out があるかどうかを確認します。

```
$ cat slurm-3.out
Hello World from spot-dy-c5xlarge-1
Hello World from spot-st-t2micro-1
Hello World from spot-dy-t2micro-1
```

また、対応するノードでジョブが正常に実行されたことも出力されています。

スケールダウンのプロセスを監視することができます。カスタム scaledown_idletime を 5 分に指定したクラスター構成では アイドル状態で 5 分経過すると、動的ノード spot-dy-c5xlarge-1 と spot-dy-t2micro-1 は自動的にスケールダウンし、POWER_DOWN モードになります。なお、静的ノード spot-st-t2micro-1 はスケールダウンしません。

```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
ondemand     up   infinite     10  idle~ ondemand-dy-c52xlarge-[1-10]
spot*        up   infinite      2  idle% spot-dy-c5xlarge-1,spot-dy-t2micro-1
spot*        up   infinite     17  idle~ spot-dy-c5xlarge-[2-10],spot-dy-t2micro-[2-9]
spot*        up   infinite      1   idle spot-st-t2micro-1
```

上記のコードから、spot-dy-c5xlarge-1 と spot-dy-t2micro-1 が POWER_DOWN モードになっていることがわかります。これは % というサフィックスで示されます。対応するインスタンスは直ちに終了しますが、ノードは POWER_DOWN 状態のままで、120 秒 (2 分) は使用できません。この時間が経過すると、ノードはパワーセーブ状態で復帰し、再び使用できるようになります。詳細については、「マルチキューモードの Slurm ガイド」を参照してください。

これがクラスターの最終的な状態です。

```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
```

```
ondemand     up   infinite     10  idle~ ondemand-dy-c52xlarge-[1-10]
spot*        up   infinite     19  idle~ spot-dy-c5xlarge-[1-10],spot-dy-t2micro-[1-9]
spot*        up   infinite      1   idle spot-st-t2micro-1
```

クラスターからログオフした後、pcluster delete を実行してクリーンアップすることができます。pcluster list および pcluster delete についての詳細については、「pcluster list」および「pcluster delete」を参照してください。

```
$ pcluster list
multi-queue CREATE_COMPLETE 2.11.9
$ pcluster delete multi-queue
Deleting: multi-queue
...
```

EFA と GPU インスタンスを備えたクラスター上でのジョブ実行

チュートリアルのこの部分では、EFA ネットワークと GPU リソースを持つインスタンスを含む複数のキューを使用して構成を変更し、クラスターを起動する方法について説明します。なお、このチュートリアルで使用するインスタンスは料金の高いインスタンスです。

このチュートリアルで説明している手順を実行する前に、アカウントの制限を確認し、これらのインスタンスを使用する権限があることを確認してください。

以下の方法で設定ファイルを変更します。

```
[global]
update_check = true
sanity_check = true
cluster_template = multi-queue-special

[aws]
aws_region_name = <Your AWS #####>

[scaling demo]
scaledown_idletime = 5

[cluster multi-queue-special]
key_name = <Your SSH key name>
base_os = alinux2                   # optional, defaults to alinux2
scheduler = slurm
master_instance_type = c5.xlarge    # optional, defaults to t2.micro
vpc_settings = <Your VPC section>
```

```
scaling_settings = demo
queue_settings = efa,gpu

[queue gpu]
compute_resource_settings = gpu_i1
disable_hyperthreading = true       # optional, defaults to false

[compute_resource gpu_i1]
instance_type = g3.8xlarge

[queue efa]
compute_resource_settings = efa_i1
enable_efa = true
placement_group = DYNAMIC           # optional, defaults to no placement group settings

[compute_resource efa_i1]
instance_type = c5n.18xlarge
max_count = 5
```

クラスターを作成します

```
$ pcluster create multi-queue-special -t multi-queue-special
```

クラスターが作成されたら、プライベートの SSH キーファイルを使ってヘッドノードにログインします。

```
$ pcluster ssh multi-queue-special -i ~/path/to/keyfile.pem
```

これがクラスターの初期状態です。

```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
efa*         up   infinite      5  idle~ efa-dy-c5n18xlarge-[1-5]
gpu          up   infinite     10  idle~ gpu-dy-g38xlarge-[1-10]
```

ここでは、ノードに EFA や GPU のリソースがあるかどうかを確認するために、いくつかのジョブを送信する方法を説明します。

まず、ジョブスクリプトを書き込みます。`efa_job.sh` は 30 秒間スリープします。その後、`lspci` コマンドの出力で EFA を確認してください。`gpu_job.sh` は 30 秒間スリープします。その後、`nvidia-smi` を実行すると、そのノードの GPU 情報が表示されます。

```
$ cat efa_job.sh
#!/bin/bash

sleep 30
lspci | grep "EFA"

$ cat gpu_job.sh
#!/bin/bash

sleep 30
nvidia-smi

$ chmod +x efa_job.sh
$ chmod +x gpu_job.sh
```

sbatch でジョブを送信します、

```
$ sbatch -p efa --wrap "srun efa_job.sh"
Submitted batch job 2
$ sbatch -p gpu --wrap "srun gpu_job.sh" -G 1
Submitted batch job 3
$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                 2       efa     wrap ec2-user CF       0:32      1 efa-dy-
c5n18xlarge-1
                 3       gpu     wrap ec2-user CF       0:20      1 gpu-dy-g38xlarge-1
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
efa*         up   infinite      1   mix# efa-dy-c5n18xlarge-1
efa*         up   infinite      4  idle~ efa-dy-c5n18xlarge-[2-5]
gpu          up   infinite      1   mix# gpu-dy-g38xlarge-1
gpu          up   infinite      9  idle~ gpu-dy-g38xlarge-[2-10]
```

数分後、ノードがオンラインになり、ジョブが実行されているのが確認できるはずです。

```
[ec2-user@ip-172-31-15-251 ~]$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
efa*         up   infinite      4  idle~ efa-dy-c5n18xlarge-[2-5]
efa*         up   infinite      1    mix efa-dy-c5n18xlarge-1
gpu          up   infinite      9  idle~ gpu-dy-g38xlarge-[2-10]
gpu          up   infinite      1    mix gpu-dy-g38xlarge-1
[ec2-user@ip-172-31-15-251 ~]$ squeue
```

```
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                 4       gpu     wrap ec2-user  R       0:06      1 gpu-dy-g38xlarge-1
                 5       efa     wrap ec2-user  R       0:01      1 efa-dy-
c5n18xlarge-1
```

ジョブが完了したら、出力を確認します。slurm-2.out ファイルの出力から、efa-dy-c5n18xlarge-1 ノードに EFA が存在していることを確認できます。slurm-3.out ファイルの出力から、nvidia-smi 出力には gpu-dy-g38xlarge-1 ノードの GPU 情報が含まれていることがわかります。

```
$ cat slurm-2.out
00:06.0 Ethernet controller: Amazon.com, Inc. Elastic Fabric Adapter (EFA)

$ cat slurm-3.out
Thu Oct  1 22:19:18 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 450.51.05    Driver Version: 450.51.05    CUDA Version: 11.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla M60           Off  | 00000000:00:1D.0 Off |                    0 |
| N/A   28C    P0    38W / 150W |      0MiB /  7618MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   1  Tesla M60           Off  | 00000000:00:1E.0 Off |                    0 |
| N/A   36C    P0    37W / 150W |      0MiB /  7618MiB |     98%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

スケールダウンのプロセスを監視することができます。クラスター構成では、以前、カスタム scaledown_idletime を 5 分に指定しました。その結果、アイドル状態が 5 分経過すると、動的ノードである spot-dy-c5xlarge-1 と spot-dy-t2micro-1 が自動的にスケールダウン

し、POWER_DOWN モードになります。最終的にノードは省電力モードになり、再び使用できるようになります。

```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
efa*         up   infinite      1  idle% efa-dy-c5n18xlarge-1
efa*         up   infinite      4  idle~ efa-dy-c5n18xlarge-[2-5]
gpu          up   infinite      1  idle% gpu-dy-g38xlarge-1
gpu          up   infinite      9  idle~ gpu-dy-g38xlarge-[2-10]

# After 120 seconds
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
efa*         up   infinite      5  idle~ efa-dy-c5n18xlarge-[1-5]
gpu          up   infinite     10  idle~ gpu-dy-g38xlarge-[1-10]
```

クラスターからログオフした後、pcluster delete *<cluster name>* を実行してクリーンアップすることができます。

```
$ pcluster list
multi-queue-special CREATE_COMPLETE 2.11.9
$ pcluster delete multi-queue-special
Deleting: multi-queue-special
...
```

詳細については、「マルチキューモードの Slurm ガイド」を参照してください。

開発

以下のセクションを使用して、 の開発を開始できます AWS ParallelCluster。

> **Important**
>
> 以下のセクションでは、クックブックのレシピのカスタムバージョンと、カスタムの AWS ParallelCluster ノードパッケージを使用する手順を示しています。この情報では、高度なカスタマイズ方法について説明し AWS ParallelCluster、デバッグが困難な問題が発生する可能性があります。インストール後のフックは一般的にデバッグが容易で、 のリリース間で移植性が高いため、 AWS ParallelCluster チームはカスタマイズにカスタムブートストラップアクションのスクリプトを使用することを強くお勧めします AWS ParallelCluster。

トピック

- カスタム AWS ParallelCluster クックブックのセットアップ
- カスタム AWS ParallelCluster ノードパッケージのセットアップ

カスタム AWS ParallelCluster クックブックのセットアップ

> **Important**
>
> ク AWS ParallelCluster ックブックレシピのカスタムバージョンを使用する手順は次のとおりです。これは高度なカスタマイズ方法であり AWS ParallelCluster、デバッグが難しい問題が発生する可能性があります。インストール後のフックは一般的にデバッグしやすく、 のリリース間で移植性が高いため、 AWS ParallelCluster チームはカスタマイズにカスタムブートストラップアクションのスクリプトを使用することを強くお勧めします AWS ParallelCluster。

ステップ

1. AWS ParallelCluster クックブックコードをクローンしたAWS ParallelCluster クックブック作業ディレクトリを特定します。

```
_cookbookDir=<path to cookbook>
```

2. ク AWS ParallelCluster ックブックの最新バージョンを検出します。

```
_version=$(grep version ${_cookbookDir}/metadata.rb|awk '{print $2}'| tr -d \')
```

3. AWS ParallelCluster クックブックのアーカイブを作成し、md5 を計算します。

```
cd "${_cookbookDir}"
_stashName=$(git stash create)
git archive --format tar --prefix="aws-parallelcluster-cookbook-${_version}/"
 "${_stashName:-HEAD}" | gzip > "aws-parallelcluster-cookbook-${_version}.tgz"
md5sum "aws-parallelcluster-cookbook-${_version}.tgz" > "aws-parallelcluster-
cookbook-${_version}.md5"
```

4. Amazon S3 バケットを作成し、アーカイブ、その md5、およびその最終更新日をバケットにアップロードします。パブリック読み取り ACL を使用して、パブリック読み取り可能なアクセス許可を付与します。

```
_bucket=<the bucket name>
aws s3 cp --acl public-read aws-parallelcluster-cookbook-${_version}.tgz s3://
${_bucket}/cookbooks/aws-parallelcluster-cookbook-${_version}.tgz
aws s3 cp --acl public-read aws-parallelcluster-cookbook-${_version}.md5 s3://
${_bucket}/cookbooks/aws-parallelcluster-cookbook-${_version}.md5
aws s3api head-object --bucket ${_bucket} --key cookbooks/aws-parallelcluster-
cookbook-${_version}.tgz --output text --query LastModified > aws-parallelcluster-
cookbook-${_version}.tgz.date
aws s3 cp --acl public-read aws-parallelcluster-cookbook-${_version}.tgz.date s3://
${_bucket}/cookbooks/aws-parallelcluster-cookbook-${_version}.tgz.date
```

5. [cluster] セクションの下にある AWS ParallelCluster 設定ファイルに次の変数を追加します。

```
custom_chef_cookbook = https://${_bucket}.s3.<the bucket region>.amazonaws.com/
cookbooks/aws-parallelcluster-cookbook-${_version}.tgz
extra_json = { "cluster" : { "skip_install_recipes" : "no" } }
```

> **Note**
>
> AWS ParallelCluster バージョン 2.6.1 以降では、起動時間を短縮するために、ノードの起動時にほとんどのインストールレシピがデフォルトでスキップされます。下位互換性の

向上よりも起動時間の短縮を優先させるために、ほとんどのインストールレシピをスキップするには、extra_json 設定の cluster キーから "skip_install_recipes" : "no" を削除します。

カスタム AWS ParallelCluster ノードパッケージのセットアップ

> **Warning**
>
> AWS ParallelCluster ノードパッケージのカスタムバージョンを使用する手順は次のとおりです。これは高度なカスタマイズ方法であり AWS ParallelCluster、デバッグが難しい問題が発生する可能性があります。インストール後のフックは一般的にデバッグしやすく、 のリリース間で移植性が高いため、 AWS ParallelCluster チームはカスタマイズにカスタムブートストラップアクションのスクリプトを使用することを強くお勧めします AWS ParallelCluster。

ステップ

1. AWS ParallelCluster ノードコードをクローンした AWS ParallelCluster ノード作業ディレクトリを特定します。

```
_nodeDir=<path to node package>
```

2. AWS ParallelCluster ノードの最新バージョンを検出します。

```
_version=$(grep "version = \"" ${_nodeDir}/setup.py |awk '{print $3}' | tr -d \")
```

3. AWS ParallelCluster ノードのアーカイブを作成します。

```
cd "${_nodeDir}"
_stashName=$(git stash create)
git archive --format tar --prefix="aws-parallelcluster-node-${_version}/"
 "${_stashName:-HEAD}" | gzip > "aws-parallelcluster-node-${_version}.tgz"
```

4. Amazon S3 バケットを作成して、アーカイブをそのバケットにアップロードします。パブリック読み取り ACL を使用して、パブリック読み取り可能なアクセス許可を付与します。

```
_bucket=<the bucket name>
```

```
aws s3 cp --acl public-read aws-parallelcluster-node-${_version}.tgz s3://${_bucket}/
node/aws-parallelcluster-node-${_version}.tgz
```

5. 次の変数を [cluster]セクションの設定 AWS ParallelCluster ファイルに追加します。

```
extra_json = { "cluster" : { "custom_node_package" : "https://${_bucket}.s3.<the
 bucket region>.amazonaws.com/node/aws-parallelcluster-node-${_version}.tgz",
 "skip_install_recipes" : "no" } }
```

> **Note**
>
> AWS ParallelCluster バージョン 2.6.1 以降では、起動時間を短縮するために、ノードの起動時にほとんどのインストールレシピがデフォルトでスキップされます。下位互換性の向上よりも起動時間の短縮を優先させるために、ほとんどのインストールレシピをスキップするには、extra_json 設定の cluster キーから "skip_install_recipes" : "no" を削除します。

AWS ParallelCluster トラブルシューティング

AWS ParallelCluster コミュニティは、AWS ParallelCluster GitHub Wiki で多くのトラブルシューティングのヒントを提供する Wiki ページを保持しています。既知の問題のリストは、「Known issues」(既知の問題) を参照してください。

トピック

- ログの取得と保存
- スタックデプロイ時のトラブルシューティング
- マルチキューモードクラスターでの問題のトラブルシューティング
- シングルキューモードのクラスターにおける問題のトラブルシューティング
- プレイスメントグループとインスタンスの起動に関する問題
- 置き換えられないディレクトリ
- Amazon DCV の問題のトラブルシューティング
- AWS Batch 統合されたクラスターにおける問題のトラブルシューティング
- リソースの作成に失敗したときのトラブルシューティング
- IAM ポリシーのサイズに関する問題のトラブルシューティング
- 追加のサポート

ログの取得と保存

ログは問題を解決するための有用なリソースです。ログを使用して AWS ParallelCluster リソースの問題をトラブルシューティングする前に、まずクラスターログのアーカイブを作成する必要があります。AWS ParallelCluster GitHub Wiki のクラスターのログのアーカイブを作成する トピックで説明されている手順に従って、このプロセスを開始します。

稼働中のクラスターの 1 つに問題が発生した場合、トラブルシューティングを開始する前に `pcluster stop <cluster_name>` コマンドを実行してクラスターを STOPPED 状態にしてください。これにより、予想外のコストが発生することを防ぐことができます。

`pcluster` が機能しなくなったクラスターを削除したい場合は、`pcluster delete –keep-logs <cluster_name>` コマンドを実行します。このコマンドを実行すると、クラスターは削除されますが、Amazon CloudWatch に保存されているロググループは保持されます。このコマンドの詳細については、「`pcluster delete`」を参照してください。

スタックデプロイ時のトラブルシューティング

クラスターの作成に失敗し、スタックの作成がロールバックされる場合は、以下のログファイルを参照して問題を診断します。これらのログの中から ROLLBACK_IN_PROGRESS の出力を探します。失敗した場合のは次のように表示されます。

```
$ pcluster create mycluster
Creating stack named: parallelcluster-mycluster
Status: parallelcluster-mycluster - ROLLBACK_IN_PROGRESS
Cluster creation failed.  Failed events:
  - AWS::EC2::Instance MasterServer Received FAILURE signal with UniqueId
 i-07af1cb218dd6a081
```

この問題を診断するには、`--norollback` フラグを含む `pcluster create` を使用してクラスターを再度作成します。次に、クラスターに SSH 接続します。

```
$ pcluster create mycluster --norollback
...
$ pcluster ssh mycluster
```

ヘッドノードにログインすると、エラーの原因を突き止めるための 3 つの主要なログファイルが見つかります。

- `/var/log/cfn-init.log` は `cfn-init` のスクリプトのログです。最初に、このログを確認してください。このログには `Command chef failed` のようなエラーが表示される可能性があります。エラーメッセージの詳細については、この行の直前の行を参照してください。詳細については、「cfn-init」を参照してください。
- `/var/log/cloud-init.log` は cloud-init のログです。`cfn-init.log` に何も表示されない場合は、次にこのログを確認してください。
- `/var/log/cloud-init-output.log` は cloud-init で実行されたコマンドの出力です。これには `cfn-init` からの出力も含まれます。通常、この種の問題のトラブルシューティングには、このログを見る必要はありません。

マルチキューモードクラスターでの問題のトラブルシューティング

このセクションは、Slurmジョブスケジューラで AWS ParallelCluster バージョン 2.9.0 以降を使用してインストールされたクラスターに関連しています。マルチキューモードの詳細については、「マルチキューモード」を参照してください。

トピック

- キーログ
- ノードの初期化に関する問題のトラブルシューティング
- 予期しないノードの置換や終了のトラブルシューティング
- 問題のあるインスタンスやノードの置換、終了、電源オフ
- その他の既知のノードやジョブの問題のトラブルシューティング

キーログ

次の表は、ヘッドノードのキーログの概要を示したものです。

`/var/log/cfn-init.log`

これは init AWS CloudFormation ログです。インスタンスのセットアップ時に実行されたすべてのコマンドが含まれています。初期化時の問題のトラブルシューティングに役立ちます。

`/var/log/chef-client.log`

これは Chef クライアントログです。Chef/CINC で実行されたすべてのコマンドが含まれています。初期化時の問題のトラブルシューティングに役立ちます。

`/var/log/parallelcluster/slurm_resume.log`

これは `ResumeProgram` ログです。動的ノードのインスタンスを起動し、動的ノードの起動に関する問題のトラブルシューティングに役立ちます。

`/var/log/parallelcluster/slurm_suspend.log`

これが `SuspendProgram` のログです。動的ノードのインスタンスが終了する際に呼び出されます。動的ノードの終了時の問題をトラブルシューティングするのに役立ちます。このログを確認する際には、`clustermgtd` のログも確認する必要があります。

`/var/log/parallelcluster/clustermgtd`

これが `clustermgtd` のログです。ほとんどのクラスターオペレーションのアクションを管理する集中型デーモンとして動作します。起動、終了、クラスターの動作の問題のトラブルシューティングに役立ちます。

`/var/log/slurmctld.log`

これはSlurmコントロールデーモン log. AWS ParallelCluster does がスケーリングを決定しません。むしろ、Slurm の要求を満たすためにリソースの起動を試みます。スケーリングや割り当ての問題、ジョブ関連の問題、スケジューラ関連の起動や終了の問題などに有効です。

コンピュートノードのキーノートは次のとおりです。

`/var/log/cloud-init-output.log`

これは cloud-init のログです。インスタンスのセットアップ時に実行されたすべてのコマンドが含まれています。初期化時の問題のトラブルシューティングに役立ちます。

`/var/log/parallelcluster/computemgtd`

これが `computemgtd` のログです。各コンピュートノード上で動作し、ヘッドノード上の `clustermgtd` デーモンがオフラインになるレアなケースでノードをモニタリングします。予期せぬ終了の問題のトラブルシューティングに役立ちます。

`/var/log/slurmd.log`

これは Slurm コンピュートデーモンのログです。初期化やコンピュートの不具合に関するトラブルシューティングに役立ちます。

ノードの初期化に関する問題のトラブルシューティング

このセクションでは、ノードの初期化に関するトラブルを解決する方法について説明します。これには、ノードが起動しない、電源が入らない、またはクラスターが参加できない、などの問題が含まれます。

ヘッドノード:

該当するログ:

- `/var/log/cfn-init.log`
- `/var/log/chef-client.log`
- `/var/log/parallelcluster/clustermgtd`
- `/var/log/parallelcluster/slurm_resume.log`
- `/var/log/slurmctld.log`

/var/log/cfn-init.log と /var/log/chef-client.log のログを確認します。これらのログには、ヘッドノードのセットアップ時に実行されたすべてのアクションが含まれています。セットアップ中に発生するほとんどのエラーは、/var/log/chef-client.log ログにエラーメッセージが表示されます。クラスターの構成でプリインストールまたはポストインストールのスクリプトが指定されている場合、スクリプトが正常に実行されていることをログメッセージで再確認します。

クラスターが作成されると、ヘッドノードはコンピュートノードがクラスターに参加するのを待ってからクラスターに参加する必要があります。そのため、コンピュートノードがクラスターに参加できないと、ヘッドノードも失敗してしまいます。このような問題を解決するためには、使用しているコンピュートノートの種類に応じて、次の手順のいずれかを実行します。

動的コンピュートノード:

- ResumeProgram ログ (/var/log/parallelcluster/slurm_resume.log) でコンピューティングノード名を検索し、そのノードで ResumeProgram が呼び出されたことがあるかどうかを調べます (ResumeProgramが呼び出されていない場合は、 slurmctldログ (/var/log/slurmctld.log) をチェックして、ノードResumeProgramで を呼び出そうとしたSlurmかどうかを判断できます。)
- なお、ResumeProgram のパーミッションが正しくないと、ResumeProgram がバックグラウンドで失敗する可能性があります。ResumeProgram の設定を変更したカスタム AMI を使用している場合は、ResumeProgram の所有者が slurm ユーザーであり、744 (rwxr--r--) の許可を持っていることを確認してください。
- ResumeProgram が呼ばれた場合、そのノードのインスタンスが起動しているかどうかを確認します。インスタンスが起動しなかった場合は、起動の失敗を示すエラーメッセージが表示されます。
- インスタンスが起動する場合は、セットアップ時に問題が発生した可能性があります。ResumeProgram のログに対応するプライベート IP アドレスとインスタンス ID が表示されます。さらに、特定のインスタンスに対応するセットアップログを確認することができます。コンピューティングノードのセットアップエラーのトラブルシューティングについては、次のセクションを参照してください。

静的コンピューティングノード:

- clustermgtd (/var/log/parallelcluster/clustermgtd) ログをチェックして、ノードに対してインスタンスが起動されたかどうかを確認します。起動されていない場合は、起動失敗の詳細を示す明確なエラーメッセージが表示されるはずです。

- インスタンスが起動した場合、セットアップ中に何らかの問題が発生します。ResumeProgram のログに対応するプライベート IP アドレスとインスタンス ID が表示されます。さらに、特定のインスタンスに対応するセットアップログを確認することができます。

- コンピュートノード
  - 該当するログ:
    - /var/log/cloud-init-output.log
    - /var/log/slurmd.log
  - コンピュートノードが起動した場合、まず /var/log/cloud-init-output.log を確認します。ヘッドノードの /var/log/chef-client.log ログと同様のセットアップログが含まれているはずです。セットアップ中に発生するほとんどのエラーは、/var/log/cloud-init-output.log ログにエラーメッセージが表示されます。クラスター構成でプレインストールまたはポストインストールスクリプトが指定されている場合、それらが正常に実行されたかどうかを確認します。
  - Slurm の設定を変更したカスタム AMI を使用している場合は、Slurm 関連のエラーが発生し、コンピュートノードがクラスターに参加できない可能性があります。スケジューラ関連のエラーについては、/var/log/slurmd.log のログを確認してください。

予期しないノードの置換や終了のトラブルシューティング

このセクションでは、ノードに関連する問題、特にノードが予期せず置換されたり終了したりした場合のトラブルシューティング方法について引き続き説明します。

- 該当するログ:
  - /var/log/parallelcluster/clustermgtd (ヘッドノード)
  - /var/log/slurmctld.log (ヘッドノード)
  - /var/log/parallelcluster/computemgtd (コンピューティングノード)
- 予期せず置換または終了したノード
  - clustermgtd のログ (/var/log/parallelcluster/clustermgtd) で、clustermgtd がノードの交換や終了のアクションをとったかどうかを確認します。なお、clustermgtd は通常のノードメンテナンスのアクションをすべて行います。
  - clustermgtd がノードを置換または終了させた場合、なぜそのアクションがノードに対して行われたのかを詳細に説明するメッセージが必要です。スケジューラ関連の理由 (例えば、ノードが DOWN にあるため) の場合は、slurmctld ログで確認してください。理由が Amazon EC2 に

関連するものであれば、置換を必要とする Amazon EC2 に関連する問題の詳細を示す情報メッセージが表示されるはずです。

- clustermgtd がノードを終了しなかった場合は、まず、これが Amazon EC2 による予想される終了、具体的にはスポット終了であるかどうかを確認します。Compute ノードでcomputemgtd実行されている は、 clustermgtdが異常であると判断された場合にノードを終了するアクションを実行することもできます。computemgtd のログ (/var/log/parallelcluster/computemgtd) を確認し、computemgtd がノードを終了したかどうかを確認します。

- ノードが失敗しました
  - slurmctld ログ (/var/log/slurmctld.log) で、ジョブやノードが失敗した理由を確認します。なお、ノードに障害が発生した場合、ジョブは自動的に再キューされます。
  - slurm_resume がノードの起動を報告し、clustermgtd が数分後にそのノードに対応するインスタンスが Amazon EC2 に存在しないと報告した場合、ノードはセットアップ中に失敗する可能性があります。コンピュート (/var/log/cloud-init-output.log) からログを取得するには、次の手順で行います。
    - Slurm が新しいノードを立ち上げるためのジョブを送信します。
    - ノードが起動したら、このコマンドで終了保護を有効にします。

      ```
      aws ec2 modify-instance-attribute --instance-id i-xyz --disable-api-termination
      ```

    - このコマンドで、ノードのコンソール出力を取得します。

      ```
      aws ec2 get-console-output --instance-id i-xyz --output text
      ```

問題のあるインスタンスやノードの置換、終了、電源オフ

- 該当するログ:
  - /var/log/parallelcluster/clustermgtd (ヘッドノード)
  - /var/log/parallelcluster/slurm_suspend.log (ヘッドノード)
- 通常、clustermgtd は期待されるすべてのインスタンス終了アクションを処理します。clustermgtd ログを見て、ノードの置換または終了に失敗した理由を確認します。
- scaledown_idletime に失敗した動的ノードの場合、SuspendProgram ログで SuspendProgram が slurmctld から特定のノードを引数として呼び出されたかどうかを確認します。なお、SuspendProgram は実際には何のアクションもしません。呼び出されたときだけ口

グに記録されます。インスタンスの終了や NodeAddr のリセットは全て clustermgtd が行います。Slurm は SuspendTimeout の後、自動的にノードを POWER_SAVING 状態に戻します。

その他の既知のノードやジョブの問題のトラブルシューティング

もう 1 つの既知の問題は、ジョブの割り当てやスケーリングの決定に失敗 AWS ParallelCluster する可能性があることです。このタイプの問題では、 はSlurm指示に従って AWS ParallelCluster リソースを起動、終了、または維持します。これらの問題については、slurmctld ログを確認してトラブルシューティングを行ってください。

シングルキューモードのクラスターにおける問題のトラブルシューティング

> **Note**
>
> バージョン 2.11.5 AWS ParallelCluster 以降、 は SGEまたは スTorqueケジューラの使用をサポートしていません。

ここでは、次の 2 つの構成のうち、マルチキューモードを持たないクラスターに適用します。

- 2.9.0 より前の AWS ParallelCluster バージョンと、SGE、Torque、または Slurmジョブスケジューラを使用して起動しました。
- AWS ParallelCluster バージョン 2.9.0 以降および SGEまたは Torque ジョブスケジューラを使用して起動しました。

トピック

- キーログ
- 起動および参加オペレーションの失敗のトラブルシューティング
- スケーリング問題のトラブルシューティング
- 他のクラスター関連の問題のトラブルシューティング

キーログ

次のログファイルは、ヘッドノードのキーログです。

AWS ParallelCluster バージョン 2.9.0 以降の場合：

`/var/log/chef-client.log`

これは CINC (chef) のクライアントログです。CINC で実行されたすべてのコマンドが含まれています。初期化時の問題のトラブルシューティングに役立ちます。

すべての AWS ParallelCluster バージョン：

`/var/log/cfn-init.log`

これが `cfn-init` のログです。インスタンスのセットアップ時に実行されたすべてのコマンドが含まれているため、初期化に関する問題のトラブルシューティングに役立ちます。詳細については、「cfn-init」を参照してください。

`/var/log/clustermgtd.log`

これは Slurm スケジューラの `clustermgtd` のログです。`clustermgtd` はほとんどのクラスターオペレーションのアクションを管理する集中型デーモンとして動作します。起動、終了、クラスターの動作の問題のトラブルシューティングに役立ちます。

`/var/log/jobwatcher`

これは、SGE およびTorque スケジューラの `jobwatcher` のログです。`jobwatcher` はスケジューラキューをモニターし、Auto Scaling グループを更新します。ノードのスケールアップに関する問題のトラブルシューティングに役立ちます。

`/var/log/sqswatcher`

これは SGE および Torque スケジューラの `sqswatcher` のログです。`sqswatcher` は、初期化に成功したコンピューティングインスタンスから送られてくるインスタンス準備イベントを処理します。また、スケジューラの構成にコンピュートノードを追加します。このログは、ノードがクラスターへの参加に失敗した場合のトラブルシューティングに役立ちます。

以下は、コンピュートノードのキーログです。

AWS ParallelCluster バージョン 2.9.0 以降

`/var/log/cloud-init-output.log`

これは Cloud init のログです。インスタンスのセットアップ時に実行されたすべてのコマンドが含まれています。初期化時の問題のトラブルシューティングに役立ちます。

AWS ParallelCluster 2.9.0 より前のバージョン

`/var/log/cfn-init.log`

これは CloudFormation のログです。インスタンスのセットアップ時に実行されたすべてのコマンドが含まれています。初期化に関する問題のトラブルシューティングに役立ちます。

すべてのバージョン

`/var/log/nodewatcher`

これは `nodewatcher` のログです。SGE および Torque スケジューラを使用している場合に各コンピューティングノード上で動作する `nodewatcher` デーモンです。アイドル状態のノードはスケールダウンします。このログは、リソースのスケールダウンに関する問題に役立ちます。

起動および参加オペレーションの失敗のトラブルシューティング

- 該当するログ:
  - `/var/log/cfn-init-cmd.log` (ヘッドノードとコンピュートノード)
  - `/var/log/sqswatcher` (ヘッドノード)
- ノードの起動に失敗した場合は、`/var/log/cfn-init-cmd.log` ログで特定のエラーメッセージを確認します。通常、ノードの起動の失敗はセットアップの失敗が原因です。
- スケジューラの設定に成功したにもかかわらず、コンピュートノードがスケジューラの設定に参加できなかった場合、`/var/log/sqswatcher` ログを確認し、`sqswatcher` がイベントを処理したかどうかを確認します。通常、これらの問題は、`sqswatcher` がイベントを処理していないことが原因です。

スケーリング問題のトラブルシューティング

- 該当するログ:
  - `/var/log/jobwatcher` (ヘッドノード)
  - `/var/log/nodewatcher` (コンピュートノード)
- スケールアップの問題:ヘッドノードの場合、`/var/log/jobwatcher` ログを確認し、`jobwatcher` デーモンが必要なノード数を適切に計算し、Auto Scaling グループを更新したかどうかを確認します。なお、`jobwatcher` はスケジューラのキューをモニターし、Auto Scaling グループを更新します。

- スケールダウンの問題:コンピュートノードの場合、問題のノードの `/var/log/nodewatcher` ログを確認し、ノードがスケールダウンした理由を確認します。`nodewatcher` デーモンは、コンピュートノードがアイドル状態になるとスケールダウンします。

他のクラスター関連の問題のトラブルシューティング

ラージスケールのクラスター、特に 500 台以上のコンピュートノードを持つクラスターでは、コンピュートノートがランダムに故障するという問題があります。この問題は、シングルキュークラスターのスケーリングアーキテクチャの制限に関連しています。大規模なクラスターを使用する場合、AWS ParallelCluster バージョン v2.9.0 以降を使用している場合、 を使用している場合Slurm、およびこの問題を回避するには、複数のキューモードがサポートされているクラスターにアップグレードして切り替える必要があります。`pcluster-config convert` を実行することで可能になります。

ultra-large-scaleクラスターの場合、システムに追加のチューニングが必要になる場合があります。詳細については、 にお問い合わせください サポート。

プレイスメントグループとインスタンスの起動に関する問題

ノード間のレイテンシーを最小にするには、プレースメントグループを使用します。プレイスメントグループにより、確実にインスタンスが同じネットワーキングバックボーンに配置されます。リクエスト時に十分な数のインスタンスが用意されていない場合は、`InsufficientInstanceCapacity` エラーが返ります。クラスタープレイスメントグループを使用する際にこのエラーが発生する可能性を減らすために、`placement_group` パラメータを `DYNAMIC` に、`placement` パラメータを `compute` に設定します。

高性能な共有ファイルシステムが必要な場合は、FSx for Lustre の使用をご検討ください。

ヘッドノードがプレースメントグループに含まれている必要がある場合は、ヘッドノードとすべてのコンピュートノードに同じインスタンスタイプとサブネットを使用します。これにより、`compute_instance_type` パラメータは `master_instance_type` パラメータと同じ値になり、`placement` パラメータは `cluster` に設定され、`compute_subnet_id` パラメータは指定されません。この設定では、`master_subnet_id` パラメータの値がコンピュートノードに使用されます。

詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスの起動に関する問題のトラブルシューティング」と「プレイスメントグループの役割と制限」を参照してください。

置き換えられないディレクトリ

以下のディレクトリはノード間で共有され、置き換えることはできません。

`/home`

これには、デフォルトユーザーのホームフォルダ (Amazon Linux では `/home/ec2_user`、CentOS では `/home/centos`、Ubuntu では `/home/ubuntu`) が含まれます。

`/opt/intel`

これには、Intel MPI、Intel Parallel Studio、および関連ファイルが含まれます。

`/opt/sge`

> **Note**
>
> バージョン 2.11.5 AWS ParallelCluster 以降、 は SGEまたは スTorqueケジューラの使用をサポートしていません。

これには、Son of Grid Engine および関連ファイルが含まれます。(条件付き、`scheduler = sge` の場合のみ)

`/opt/slurm`

これには、Slurm Workload Manager および関連ファイルが含まれます。(条件付き、`scheduler = slurm` の場合のみ)

`/opt/torque`

> **Note**
>
> バージョン 2.11.5 AWS ParallelCluster 以降、 は SGEまたは スTorqueケジューラの使用をサポートしていません。

これには、Torque Resource Manager および関連ファイルが含まれます。(条件付き、`scheduler = torque` の場合のみ)

Amazon DCV の問題のトラブルシューティング

トピック

- Amazon DCV のログ
- Amazon DCV インスタンスタイプメモリ
- Ubuntu の Amazon DCV の問題

Amazon DCV のログ

Amazon DCV のログは `/var/log/dcv/` ディレクトリのファイルに書き込まれます。これらのログを確認することは、問題の解決に役立ちます。

Amazon DCV インスタンスタイプメモリ

インスタンスタイプには、Amazon DCV を実行するために、少なくとも 1.7 ギビバイト (GiB) の RAM が必要です。Nano および micro のインスタンスタイプには、Amazon DCV を実行するのに十分なメモリがありません。

Ubuntu の Amazon DCV の問題

Ubuntu の DCV セッションで Gnome ターミナルを実行する場合、ログインシェルを介して AWS ParallelCluster が利用できるユーザー環境に自動的にアクセスできない場合があります。ユーザー環境には、openmpi や intelmpi などの環境モジュールやその他のユーザー設定が用意されています。

Gnome ターミナルのデフォルト設定では、シェルをログインシェルとして起動することはできません。つまり、シェルプロファイルは自動的にソース化されず、 AWS ParallelCluster ユーザー環境はロードされません。

シェルプロファイルを適切にソースし、 AWS ParallelCluster ユーザー環境にアクセスするには、次のいずれかを実行します。

- デフォルトのターミナル設定を変更します。
  1. Gnome ターミナルで [編集] メニューを選択します。
  2. [設定]、[プロファイル] の順に選択します。
  3. [コマンド] を選択し、[ログインシェルとしてコマンドを実行] を選択します。
  4. [新しいターミナル] を開きます。
- コマンドラインを使用して、使用可能なプロファイルを取得します。

```
$ source /etc/profile && source $HOME/.bashrc
```

AWS Batch 統合されたクラスターにおける問題のトラブルシューティング

このセクションは、ス AWS Batch ケジューラ統合のクラスターに関連しています。

ヘッドノードの問題

ヘッドノードに関連するセットアップの問題は、シングルキューのクラスターと同様にトラブルシューティングが可能です。これらの問題を解決する方法の詳細については、「シングルキューモードのクラスターにおける問題のトラブルシューティング」を参照してください。

AWS Batch マルチノード並列ジョブの送信に関する問題

をジョブスケジューラ AWS Batch として使用するときにマルチノード並列ジョブの送信に問題がある場合は、 AWS ParallelCluster バージョン 2.5.0 にアップグレードする必要があります。それが不可能な場合は、トピック「Self patch a cluster used for submitting multi-node parallel jobs through AWS Batch」で紹介されている回避策を使用できます。

コンピューティングの問題

AWS Batch は、サービスのスケーリングとコンピューティングの側面を管理します。コンピューティング関連の問題が発生した場合は、 AWS Batch トラブルシューティングドキュメントを参照してください。

ジョブの失敗

ジョブが失敗した場合は、`awsbout` コマンドを実行してジョブの出力を取得することができます。また、`awsbstat -d` コマンドを実行して、Amazon CloudWatch が保存しているジョブログへのリンクを取得することもできます。

リソースの作成に失敗したときのトラブルシューティング

このセクションは、クラスターリソースが作成に失敗した場合に関連しています。

リソースの作成に失敗すると、ParallelCluster は次のようなエラーメッセージを返します。

```
pcluster create -c config my-cluster
Beginning cluster creation for cluster: my-cluster
WARNING: The instance type 'p4d.24xlarge' cannot take public IPs. Please make sure that
 the subnet with
id 'subnet-1234567890abcdef0' has the proper routing configuration to allow private IPs
 reaching the
Internet (e.g. a NAT Gateway and a valid route table).
WARNING: The instance type 'p4d.24xlarge' cannot take public IPs. Please make sure that
 the subnet with
id 'subnet-1234567890abcdef0' has the proper routing configuration to allow private IPs
 reaching the Internet
(e.g. a NAT Gateway and a valid route table).
Info: There is a newer version 3.0.3 of AWS ParallelCluster available.
Creating stack named: parallelcluster-my-cluster
Status: parallelcluster-my-cluster - ROLLBACK_IN_PROGRESS
Cluster creation failed.  Failed events:
- AWS::CloudFormation::Stack MasterServerSubstack Embedded stack
arn:aws:cloudformation:region-id:123456789012:stack/parallelcluster-my-cluster-
MasterServerSubstack-ABCDEFGHIJKL/a1234567-b321-c765-d432-dcba98766789
was not successfully created:
The following resource(s) failed to create: [MasterServer].
- AWS::CloudFormation::Stack parallelcluster-my-cluster-MasterServerSubstack-
ABCDEFGHIJKL The following resource(s) failed to create: [MasterServer].
- AWS::EC2::Instance MasterServer You have requested more vCPU capacity than your
 current vCPU limit of 0 allows for the instance bucket that the
specified instance type belongs to. Please visit http://aws.amazon.com/contact-us/ec2-
request to request an adjustment to this limit.
(Service: AmazonEC2; Status Code: 400; Error Code: VcpuLimitExceeded; Request ID:
 a9876543-b321-c765-d432-dcba98766789; Proxy: null)
}
```

たとえば、前のコマンドレスポンスにステータスメッセージが表示される場合は、現在の vCPU 制限を超えないインスタンスタイプを使用するか、vCPU 容量の追加をリクエストする必要があります。

CloudFormation コンソールを使用して `"Cluster creation failed"` ステータスに関する情報を確認することもできます。

コンソールから CloudFormation のエラーメッセージを表示します。

1. にログイン AWS Management Console し、https://console.aws.amazon.com/cloudformation に移動します。

2. スタック名は parallelcluster-*cluster_name* を選択します。
3. [イベント] タブを選択します。
4. [論理 ID] でリソースイベントのリストをスクロールして、作成に失敗したリソースの [ステータス] を確認します。サブタスクの作成に失敗した場合は、失敗したリソースイベントを見つけるために逆算します。
5. AWS CloudFormation エラーメッセージの例：

```
2022-02-07 11:59:14 UTC-0800 MasterServerSubstack CREATE_FAILED Embedded stack
arn:aws:cloudformation:region-id:123456789012:stack/parallelcluster-my-cluster-
MasterServerSubstack-ABCDEFGHIJKL/a1234567-b321-c765-d432-dcba98766789
was not successfully created: The following resource(s) failed to create:
 [MasterServer].
```

IAM ポリシーのサイズに関する問題のトラブルシューティング

ロールにアタッチされた管理ポリシーのクォータを確認するには、IAM と AWS STS クォータ、名前の要件、および文字制限を参照してください。管理ポリシーのサイズがクォータを超える場合は、ポリシーを 2 つ以上のポリシーに分割してください。IAM ロールにアタッチされたポリシー数のクォータを超えた場合は、追加のロールを作成し、そのロール間でポリシーを配布してクォータを満たすようにします。

追加のサポート

既知の問題点の一覧は、GitHub Wiki のメインページまたは「問題」ページを参照してください。緊急の問題については、 に問い合わせる サポート か、新しい GitHub の問題を開きます。

AWS ParallelCluster サポートポリシー

AWS ParallelCluster は複数のリリースを同時にサポートします。すべての AWS ParallelCluster リリースには、サポート終了日 (EOSL) がスケジュールされています。EOSL の日付を過ぎると、そのリリースに対するサポートやメンテナンスは提供されません。

AWS ParallelCluster は`major.minor.`*`patch`*バージョンスキームを使用します。最新のメジャーバージョンリリースの新しいマイナーバージョンリリースには、新機能、パフォーマンスの向上、セキュリティアップデート、バグ修正が含まれます。マイナーバージョンには、メジャーバージョン内での下位互換性があります。重大な問題については、 AWS はパッチリリースを通じて修正を提供しますが、EOSL に達していないリリースの最新のマイナーバージョンに限られます。新しいバージョンリリースのアップデートを使用する場合は、新しいマイナーバージョンまたはパッチバージョンにアップグレードする必要があります。

AWS ParallelCluster バージョン	サポート終了日(EOSL)
2.10.4 以前	12/31/2021
2.11。*x*	12/31/2022

のセキュリティ AWS ParallelCluster

のクラウドセキュリティが最優先事項 AWS です。お客様は AWS 、最もセキュリティの影響を受けやすい組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャを活用できます。

セキュリティは、 AWS とお客様の間の責任共有です。責任共有モデル、、は、これをクラウドのセキュリティ、およびクラウド内のセキュリティとして説明しています。

- クラウドのセキュリティ – AWS クラウドで AWS サービスを実行するインフラストラクチャを保護する AWS 責任があります。 AWS また、 では、安全に使用できるサービスも提供しています。サードパーティーの監査者は、AWS コンプライアンスプログラムコンプライアンスプログラムの一環として、当社のセキュリティの有効性を定期的にテストおよび検証。が適用されるコンプライアンスプログラムの詳細については AWS ParallelCluster、「コンプライアンスプログラムAWS による対象範囲内のサービスコンプライアンスプログラム」を参照してください。
- クラウド内のセキュリティ – お客様の責任は、使用する特定の AWS サービスによって決まります。また、お客様は、データの機密性、企業の要件、および適用される法律や規制など、その他のいくつかの関連要素についても責任を負います。

このドキュメントでは、 を使用する際の責任共有モデルの適用方法について説明します AWS ParallelCluster。以下のトピックでは、セキュリティとコンプライアンスの目的 AWS ParallelCluster を達成するために を設定する方法を示します。また、 AWS リソースのモニタリングと保護に役立つ AWS ParallelCluster 方法で を使用する方法についても説明します。

トピック

- が使用する サービスのセキュリティ情報 AWS ParallelCluster
- でのデータ保護 AWS ParallelCluster
- の Identity and Access Management AWS ParallelCluster
- のコンプライアンス検証 AWS ParallelCluster
- TLS の最小バージョン 1.2 の指定

が使用する サービスのセキュリティ情報 AWS ParallelCluster

- Amazon EC2 でのセキュリティ
- Amazon API Gateway でのセキュリティ

- のセキュリティ AWS Batch
- AWS CloudFormationでのセキュリティ
- Amazon CloudWatch でのセキュリティ
- AWS CodeBuildでのセキュリティ
- Amazon DynamoDB でのセキュリティ
- Amazon ECR でのセキュリティ
- Amazon ECS でのセキュリティ
- Amazon EFS でのセキュリティ
- FSx for Lustre でのセキュリティー
- AWS Identity and Access Management (IAM) のセキュリティ
- EC2 Image Builder でのセキュリティ
- のセキュリティ AWS Lambda
- Amazon Route 53 でのセキュリティ
- Amazon SNS でのセキュリティ
- Amazon SQS のセキュリティ（AWS ParallelCluster バージョン 2.x の場合）
- Amazon S3 でのセキュリティ
- Amazon VPC でのセキュリティ

でのデータ保護 AWS ParallelCluster

責任 AWS 共有モデル、 でのデータ保護に適用されます AWS ParallelCluster。このモデルで説明されているように、 AWS はすべての を実行するグローバルインフラストラクチャを保護する責任があります AWS クラウド。ユーザーは、このインフラストラクチャでホストされるコンテンツに対する管理を維持する責任があります。また、使用する「 AWS のサービス 」のセキュリティ設定と管理タスクもユーザーの責任となります。データプライバシーの詳細については、データプライバシーに関するよくある質問を参照してください。欧州でのデータ保護の詳細については、AWS セキュリティブログに投稿された AWS 責任共有モデルおよび GDPR のブログ記事を参照してください。

データ保護の目的で、認証情報を保護し AWS アカウント 、 AWS IAM Identity Center または AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定することをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- で API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。CloudTrail 証跡を使用して AWS アクティビティをキャプチャする方法については、「 AWS CloudTrail ユーザーガイド」のCloudTrail 証跡の使用」を参照してください。
- AWS 暗号化ソリューションと、その中のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。
- Amazon Macie などの高度な管理されたセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API AWS を介して にアクセスするときに FIPS 140-3 検証済み暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-3」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報を、タグ、または [名前] フィールドなどの自由形式のテキストフィールドに含めないことを強くお勧めします。これは、コンソール AWS ParallelCluster 、API、または SDK を使用して AWS CLIまたは他の AWS のサービス を使用する場合も同様です。 AWS SDKs タグ、または名前に使用される自由記述のテキストフィールドに入力したデータは、請求または診断ログに使用される場合があります。外部サーバーに URL を提供する場合、そのサーバーへのリクエストを検証できるように、認証情報を URL に含めないことを強くお勧めします。

データの暗号化

セキュリティで保護されたサービスの重要な特徴として、情報はアクティブに使用されていないときに暗号化されます。

保管中の暗号化

AWS ParallelCluster 自体は、ユーザーに代わって AWS サービスとやり取りするために必要な認証情報以外の顧客データを保存しません。

クラスター内のノード上のデータについては、保管時に暗号化できます。

Amazon EBS ボリュームの場合、暗号化は AWS ParallelCluster バージョン 2.x の `[ebs]`セクション`ebs_kms_key_id`の設定を使用して設定されます) 。詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EBS の暗号化」を参照してください。

Amazon EFS ボリュームの場合、暗号化は AWS ParallelCluster バージョン 2.x) の [efs]セクションの encryptedおよび efs_kms_key_id設定を使用して設定されます。詳細については、「Amazon Elastic File System User Guide」(Amazon Elastic File System ユーザーガイド) の「How encryption at rest works」(保管時のデータの暗号化) を参照してください。

FSx for Lustre ファイルシステムでは、Amazon FSx ファイルシステムの作成時に、保管中のデータの暗号化が自動的に有効になります。詳細については、「FSx for Lustre ユーザーガイド」の「保管中のデータの暗号化」を参照してください。

NVMe ボリュームを持つインスタンスタイプでは、NVMe インスタンスストアボリューム内のデータは、インスタンスのハードウェアモジュールに実装されている XTS-AES-256 暗号を使用して暗号化されます。暗号化キーはハードウェアモジュールで作成され、NVMe インスタンスストレージデバイスごとに固有です。すべての暗号化キーはインスタンスが停止または終了して復元できないときに破棄されます。この暗号化を無効にしたり、独自の暗号キーを指定したりすることはできません。詳細については、「Amazon EC2 ユーザーガイド」の「保管中の暗号化」を参照してください。

AWS ParallelCluster を使用して、カスタマーデータをローカルコンピュータに送信して保存する AWS サービスを呼び出す場合は、そのサービスのユーザーガイドの「セキュリティとコンプライアンス」の章を参照して、データの保存、保護、暗号化の方法を確認してください。

転送中の暗号化

デフォルトでは、 AWS ParallelCluster および AWS サービスエンドポイントを実行しているクライアントコンピュータから送信されるすべてのデータは、HTTPS/TLS 接続を介してすべてを送信することで暗号化されます。クラスター内のノード間のトラフィックは、選択したインスタンスタイプに応じて自動的に暗号化することができます。詳細については、「Amazon EC2 ユーザーガイド」の「転送中の暗号化」を参照してください。

関連情報

- Amazon EC2 でのデータ保護
- EC2 Image Builder でのデータ保護
- でのデータ保護 AWS CloudFormation
- Amazon EFS でのデータ保護
- Amazon S3 でのデータ保護
- Amazon FSx for Lustre のデータ保護

の Identity and Access Management AWS ParallelCluster

AWS ParallelCluster は、 ロールを使用して AWS リソースとそのサービスにアクセスします。 AWS ParallelCluster がアクセス許可を付与するために使用するインスタンスポリシーとユーザーポリシーについては、「」を参照してください[AWS Identity and Access Management での ロール AWS ParallelCluster](#)。

唯一の大きな違いは、標準の ユーザーと長期の認証情報を使用する場合の認証方法です。ユーザーが AWS サービスのコンソールにアクセスするにはパスワードが必要ですが、同じユーザーが を使用して同じオペレーションを実行するにはアクセスキーペアが必要です AWS ParallelCluster。他のすべての短期の認証情報については、使用方法がコンソールと同じです。

で使用される認証情報 AWS ParallelCluster はプレーンテキストファイルに保存され、暗号化されません。

- `$HOME/.aws/credentials` には、 AWS リソースにアクセスするために必要な長期の認証情報が保存されます。これには、アクセスキー ID とシークレットアクセスキーが含まれます。
- 引き受けるロールの認証情報やサービス用の認証情報などの短期認証情報は AWS IAM Identity Center 、それぞれ `$HOME/.aws/cli/cache`フォルダと $HOME/.aws/sso/cache フォルダに保存されます。

リスクの軽減

- `$HOME/.aws` フォルダとその子フォルダおよびファイルに対して、許可されたユーザーにのみアクセスを制限するようにファイルシステムのアクセス許可を設定することを強くお勧めします。
- 一時的な認証情報を持つロールをできるだけ使用し、認証情報が漏洩した場合の損害の可能性を減らします。長期の認証情報は、短期のロールの認証情報を要求および更新する場合にのみ使用します。

のコンプライアンス検証 AWS ParallelCluster

サードパーティーの監査者は、複数の コンプライアンスプログラムの一環として AWS サービスのセキュリティと AWS コンプライアンスを評価します。 AWS ParallelCluster を使用してサービスにアクセスしても、そのサービスのコンプライアンスは変更されません。

特定のコンプライアンスプログラムの対象となる AWS サービスのリストについては、「コンプライアンスAWS プログラムによる対象範囲内のサービスコンプライアンス」を参照してください。一般的な情報については、AWS 「 コンプライアンスプログラム」を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、「AWS Artifactでレポートをダウンロードする」を参照してください。

を使用する際のお客様のコンプライアンス責任 AWS ParallelCluster は、お客様のデータの機密性、貴社のコンプライアンス目的、適用される法律および規制によって決まります。 では、コンプライアンスに役立つ以下のリソース AWS を提供しています。

- セキュリティとコンプライアンスのクイックスタートガイド – これらのデプロイガイドでは、アーキテクチャ上の考慮事項について説明し、セキュリティとコンプライアンスに重点を置いたベースライン環境をデプロイする手順を示します AWS。
- Amazon Web Services での HIPAA セキュリティとコンプライアンスのためのアーキテクチャ設計 AWS ホワイトペーパー – このホワイトペーパーでは、企業が AWS を使用して HIPAA 準拠のアプリケーションを作成する方法について説明します。
- AWS コンプライアンスリソース – このワークブックとガイドのコレクションは、お客様の業界や地域に適用される場合があります。
- 「 デベロッパーガイド」の「ルールによるリソースの評価」 – この AWS Config サービスは、リソース設定が社内プラクティス、業界ガイドライン、および規制にどの程度準拠しているかを評価します。 AWS Config
- AWS Security Hub – この AWS サービスは、 内のセキュリティ状態を包括的に把握 AWS し、セキュリティ業界標準とベストプラクティスへの準拠を確認するのに役立ちます。

TLS の最小バージョン 1.2 の指定

AWS サービスと通信する際のセキュリティを強化するには、TLS 1.2 以降を使用するように を設定 AWS ParallelCluster する必要があります。を使用する場合 AWS ParallelCluster、Python を使用して TLS バージョンを設定します。

が TLS 1.2 より前の TLS バージョン AWS ParallelCluster を使用しないようにするには、OpenSSL を再コンパイルしてこの最小値を適用し、新しく構築された OpenSSL を使用するように Python OpenSSL を再コンパイルする必要があります。

現在サポートされているプロトコルの確認

まず、OpenSSL を使用して、テストサーバーと Python SDK に使用する自己署名証明書を作成します。

```
$ openssl req -subj '/CN=localhost' -x509 -newkey rsa:4096 -nodes -keyout key.pem -out
 cert.pem -days 365
```

次に、OpenSSL を使用してテストサーバーをスピンアップします。

```
$ openssl s_server -key key.pem -cert cert.pem -www
```

新しいターミナルウィンドウで仮想環境を作成し、Python SDK をインストールします。

```
$ python3 -m venv test-env
source test-env/bin/activate
pip install botocore
```

SDK の基になる HTTP ライブラリを使用する、`check.py` という名前の新しい Python スクリプトを作成します。

```
$ import urllib3
URL = 'https://localhost:4433/'

http = urllib3.PoolManager(
ca_certs='cert.pem',
cert_reqs='CERT_REQUIRED',
)
r = http.request('GET', URL)
print(r.data.decode('utf-8'))
```

新しいスクリプトを実行します。

```
$ python check.py
```

確立された接続に関する詳細が表示されます。出力で「Protocol:」を検索します。出力が「TLSv1.2」以降の場合、SDK のデフォルト設定は TLS v1.2 以降です。それ以前のバージョンの場合は、OpenSSL を再コンパイルして Python を再コンパイルする必要があります。

ただし、Python のインストールがデフォルトで TLS v1.2 以降に設定されている場合でも、サーバーが TLS v1.2 以降をサポートしていないと、Python は TLS v1.2 より前のバージョンに再ネゴシエートする可能性があります。Python が以前のバージョンに自動的に再ネゴシエートしないことを確認するには、次のようにしてテストサーバーを再起動します。

```
$ openssl s_server -key key.pem -cert cert.pem -no_tls1_3 -no_tls1_2 -www
```

以前のバージョンの OpenSSL を使用している場合は、`-no_tls_3` フラグが使用できない可能性があります。この場合は、使用している OpenSSL のバージョンが TLS v1.3 をサポートしていないため、フラグを削除します。次に、Python スクリプトを実行します。

```
$ python check.py
```

Python のインストールが TLS 1.2 より前のバージョンに対して正しく再ネゴシエートしない場合は、SSL エラーが表示されます。

```
$ urllib3.exceptions.MaxRetryError: HTTPSConnectionPool(host='localhost',
 port=4433): Max retries exceeded with url: / (Caused by SSLError(SSLError(1, '[SSL:
 UNSUPPORTED_PROTOCOL] unsupported protocol (_ssl.c:1108)')))
```

接続を確立できる場合は、OpenSSL と Python を再コンパイルして、TLS v1.2 より前のプロトコルのネゴシエーションを無効にする必要があります。

OpenSSL と Python のコンパイル

AWS ParallelCluster が TLS 1.2 より前のものについてネゴシエートしないようにするには、OpenSSL と Python を再コンパイルする必要があります。これを行うには、次のコンテンツをコピーしてスクリプトを作成し、実行します。

```
#!/usr/bin/env bash
set -e

OPENSSL_VERSION="1.1.1d"
OPENSSL_PREFIX="/opt/openssl-with-min-tls1_2"
PYTHON_VERSION="3.8.1"
PYTHON_PREFIX="/opt/python-with-min-tls1_2"



curl -O "https://www.openssl.org/source/openssl-$OPENSSL_VERSION.tar.gz"
```

```
tar -xzf "openssl-$OPENSSL_VERSION.tar.gz"
cd openssl-$OPENSSL_VERSION
./config --prefix=$OPENSSL_PREFIX no-ssl3 no-tls1 no-tls1_1 no-shared
make > /dev/null
sudo make install_sw > /dev/null


cd /tmp
curl -O "https://www.python.org/ftp/python/$PYTHON_VERSION/Python-$PYTHON_VERSION.tgz"
tar -xzf "Python-$PYTHON_VERSION.tgz"
cd Python-$PYTHON_VERSION
./configure --prefix=$PYTHON_PREFIX --with-openssl=$OPENSSL_PREFIX --disable-shared > /
dev/null
make > /dev/null
sudo make install > /dev/null
```

これにより、静的にリンクされた OpenSSL を持つ Python のバージョンがコンパイルされます。このバージョンは、TLS 1.2 より前のバージョンは自動的にネゴシエートしません。また、`/opt/openssl-with-min-tls1_2` ディレクトリに OpenSSL がインストールされ、`/opt/python-with-min-tls1_2` ディレクトリに Python がインストールされます。このスクリプトを実行した後、新しいバージョンの Python のインストールを確認します。

```
$ /opt/python-with-min-tls1_2/bin/python3 --version
```

これにより、以下が出力されます。

```
Python 3.8.1
```

この新しいバージョンの Python が TLS 1.2 より前のバージョンをネゴシエートしないことを確認するには、新しくインストールされた Python バージョン (つまり現在サポートされているプロトコルの確認) を使用する手順 `/opt/python-with-min-tls1_2/bin/python3` を再実行します。

リリースノートとドキュメント履歴

次の表は、「AWS ParallelCluster ユーザーガイド」の主な更新や新機能の一覧です。また、お客様からいただいたフィードバックに対応するために、ドキュメントを頻繁に更新しています。

変更	説明	日付
ドキュメントのみリリース	AWS ParallelCluster バージョン 2 固有のユーザーガイドが公開されました。 ドキュメントのみのリリース: • AWS ParallelCluster バージョン 2 には独自のユーザーガイドがあります。	2023 年 7 月 17 日
AWS ParallelCluster バージョン 2.11.9 をリリース	AWS ParallelCluster バージョン 2.11.9 がリリースされました。 バグ修正: • マネージド FSx for Lustre ファイルシステムの置き換えや、`vpc_security_group_id` への変更を含むクラスター更新でのデータの損失を防止します。 変更点の詳細については、GitHub に掲載されている aws-parallelcluster パッケージ	2022 年 12 月 2 日

の CHANGELOG ファイルを参照してください。

AWS ParallelCluster バージョン 2.11.8 をリリース	AWS ParallelCluster バージョン 2.11.8 がリリースされました。
	
	変更:
	
	• インテル MPI はバージョン 2021 アップデート 6 に更新されました (バージョン 2021 アップデート 4 から更新)。詳細については、「Intel® MPI Library 2021 Update 6」を参照してください。
	• EFA インストーラ 1.19.0 のアップグレード
	  • Efa-driver: `efa-1.16.0-1`
	  • Efa-config: `efa-config-1.11-1` (`efa-config-1.9-1` から)
	  • Efa-profile: `efa-profile-1.5-1` (変更なし)
	  • Libfabric-aws: `libfabric-aws-1.16.0-1` (`libfabric-1.13.2` から)
	  • Rdma-core: `rdma-core-41.0-2` (`rdma-core-37.0` から)
	  • Open MPI: `openmpi40-aws-4.1.4-3` (`openmpi40-aws-4.1.1-2` から)
		2022 年 11 月 14 日

	• AWS Batch 統合で Lambda 関数で使用される Python ランタイムを python3.9 にアップグレードします。 バグ修正: • クラスタータグはサポートされていないため、更新中にクラスタータグが変更されないようにします。 変更点の詳細については、GitHub の aws-parallelcluster パッケージの `CHANGELOG` ファイルを参照してください。	
AWS ParallelCluster バージョン 2.11.7 をリリース	AWS ParallelCluster バージョン 2.11.7 がリリースされました。 変更: • Slurm をバージョン 20.11.9 へアップグレードします。 変更点の詳細については、GitHub の aws-parallelcluster パッケージの `CHANGELOG` ファイルを参照してください。	2022 年 5 月 13 日

AWS ParallelCluster バージョン 2.11.6 をリリース	AWS ParallelCluster バージョン 2.11.6 がリリースされました。 機能強化: • ネットワークがない場合の例外管理を改善します。 変更: • OS パッケージの更新とセキュリティ修正。 変更点の詳細については、GitHub の aws-parallelcluster パッケージの CHANGELOG ファイルを参照してください。	2022 年 4 月 19 日

AWS ParallelCluster バージョン 2.11.5 をリリース	AWS ParallelCluster バージョン 2.11.5 がリリースされました。 機能強化: • FSx for Lustre `AutoImportPolicy` オプションの値として `NEW_CHANGED_DELETED` のサポートを追加します。 • SGE と Torque スケジューラのサポートを削除しました。 • パフォーマンスが低下する可能性を避けるため、Amazon Linux で `log4j-cve-2021-44228-hotpatch` のサービスを無効にしてください。 変更: • NVIDIA ドライバーをバージョン `470.82.01` から `470.103.01` にアップグレードします。 • NVIDIA ファブリックマネージャをバージョン `470.103.01` (`470.82.01` から) にアップグレードします。	2022 年 3 月 1 日

- CUDA ライブラリをバージョン 11.4.4 から 11.4.3 にアップグレードします。
- Intel MPI はバージョン 2021 アップデート 4 に更新されました (バージョン 2019 アップデート 8 から更新)。詳細については、「Intel® MPI Library 2021 Update 4」を参照してください。
- ヘッドノード作成のタイムアウトを 1 時間に延長します。

バグ修正:

- ブラウザ経由の DCV 接続を修正します。
- YAML の引用を修正して、カスタムタグが数値として解析されないようにします。

変更点の詳細については、GitHub の aws-parallelcluster パッケージの CHANGELOG ファイルを参照してください。

AWS ParallelCluster バージョン 2.11.4 をリリース	AWS ParallelCluster バージョン 2.11.4 がリリースされました。 変更点は次の通りです。 • CentOS 8 サポートが削除されました。CentOS 8 は 2021 年 12 月 31 日にサポート終了 (EOL) を迎えます。 • Slurm Workload Manager をバージョン 20.11.8 にアップグレードしました • Cinc クライアントを `17.2.29` にアップグレードします。 • Amazon DCV を Amazon DCV 2021.2-11190 に更新しました。詳細については、「Amazon DCV 管理者ガイド」の「DCV 2021.2-11190 — 2021 年 10 月 11 日」を参照してください。 • NVIDIA ドライバーをバージョン `460.73.01` から `470.82.01` にアップグレードします。 • CUDA ライブラリをバージョン `11.4.3` から `11.3.0` にアップグレードします。 • NVIDIA ファブリックマネージャを `470.82.01` にアップグレードします。	2021 年 12 月 20 日

- Amazon Linux 2 では、インスタンス起動時にパッケージ更新を無効にします。
- Ubuntu および Amazon Linux 2 の無人パッケージ更新を無効にします。
- Python 3 バージョンのAWS CloudFormation ヘルパースクリプトを CentOS 7 と Ubuntu 18.04 にインストールします。(これらは既に Amazon Linux 2 と Ubuntu 20.04 で使用されていました)。

修正点には以下が含まれます。

- `ec2_iam_role` パラメータの更新を無効にしました。
- T2 インスタンスの起動テンプレートの `CpuOptions` 設定を修正しました。

変更点の詳細については、GitHub に掲載されている aws-parallelcluster、aws-parallelcluster-cookbook、aws-parallelcluster-node の各パッケージの `CHANGELOG` ファイルを参照してください。

AWS ParallelCluster バージョン 2.11.3 をリリース	AWS ParallelCluster バージョン 2.11.3 がリリースされました。 • `arc.liv.ac.uk` で Son of Grid Engine ソースが利用できないことによる `pcluster createami` の不具合を修正しました。 Elastic Fabric Adapter インストーラーを 1.13.0 から 1.14.1 にアップグレードします • EFA 設定: `efa-config-1.9-1` (`efa-config-1.9` から) • EFA プロファイル: `efa-profile-1.5-1` (変更なし) • EFA カーネルモジュール: `efa-1.14.2` (`efa-1.13.0` から) • RDMA コア: `rdma-core-37.0` (`rdma-core-35.0amzn` から) • Libfabric: `libfabric-1.13.2` (`libfabric-1.13.0amzn1.0` から) • Open MPI: `openmpi40-aws-4.1.1-2` (変更なし)	2021 年 11 月 3 日

GPUDirect RDMA は、インスタンスタイプでサポートされていれば常に有効です。

- `enable_efa_gdr` および `enable_efa_gdr` の設定オプションは効果がありません。

変更点の詳細については、GitHub に掲載されている aws-parallelcluster、aws-parallelcluster-cookbook、aws-parallelcluster-node の各パッケージの `CHANGELOG` ファイルを参照してください。

AWS ParallelCluster バージョン 2.11.2 をリリース	AWS ParallelCluster バージョン 2.11.2 がリリースされました。
	
	変更点は次の通りです。
	
	• EFA がベース AMI にインストールされている場合、ブートストラップ時に GPUDirect RDMA (GDR) を有効にして EFA をインストールしないでください。
	• `nvidia-fabricmanager` パッケージのバージョンをロックして、 によってインストールされた NVIDIA ドライバーのバージョンと同期したままにします AWS ParallelCluster。
	• Slurm: ノードの起動中にクラスターを停止して再起動した場合に発生する問題を修正しました。
	• Elastic Fabric Adapter インストーラーを 1.13.0 に更新しました。
	  • EFA 設定: `efa-config-1.9` (変更なし)
	  • EFA プロファイル: `efa-profile-1.5-1` (変更なし)
	  • EFA カーネルモジュール: `efa-1.13.0` (変更なし)
		2021 年 8 月 27 日

  - RDMA コア: `rdma-core-35.0amzn` (`rdma-core-32.1amzn` から)
  - Libfabric: `libfabric-1.13.0amzn1.0` (`libfabric-1.11.2amzn1.1` から)
  - Open MPI: `openmpi40-aws-4.1.1-2` (変更なし)
- プリインストールされた EFA パッケージでカスタム AMI を使用する場合、ノードのブートストラップ時に EFA に変更は加えられません。オリジナル EFA パッケージのデプロイを保持しています。

変更点の詳細については、GitHub に掲載されている aws-parallelcluster、aws-parallelcluster-cookbook の各 CHANGELOG ファイルを参照してください。

AWS ParallelCluster バージョン 2.11.1 をリリース	AWS ParallelCluster バージョン 2.11.1 がリリースされました。
	
	変更点は次の通りです。
	
	• noatime マウントオプションを使用してファイルシステムをマウントすると、ファイルの読み取り時に最終アクセス時刻を記録しなくなります。これにより、リモートファイルシステムのパフォーマンスが向上します。
	• Elastic Fabric Adapter インストーラーを 1.12.3 に更新しました。
	  • EFA 設定: efa-config-1.9 (efa-config-1.8-1 から)
	  • EFA プロファイル: efa-profile-1.5-1 (変更なし)
	  • EFA カーネルモジュール: efa-1.13.0 (efa-1.12.3 から)
	  • RDMA コア: rdma-core-32.1amzn (変更なし)
	  • Libfabric: libfabric-1.11.2amzn1.1 (変更なし)
	  • Open MPI: openmpi40-aws-4.1.1-2 (変更なし)
		2021 年 7 月 23 日

- スケジューラ AWS Batch として を使用する場合、ヘッドノードへの`aws-parallelcluster` パッケージのインストールを再試行します。
- 31 個以上の vCPU を持つインスタンスタイプで SGE を構築する場合、障害が発生しないようになりました。
- バージョン 1.247348.0 で見つかった問題を回避するために、Amazon CloudWatch エージェントのバージョン 1.247347.6 にピンしました。

変更点の詳細については、GitHub に掲載されている aws-parallelcluster、aws-parallelcluster-cookbook の各 CHANGELOG ファイルを参照してください。

AWS ParallelCluster バージョン 2.11.0 をリリース	AWS ParallelCluster バージョン 2.11.0 がリリースされました。 変更点は次の通りです。 • Ubuntu 20.04 (`ubuntu2004`) のサポートを追加し、Ubuntu 16.04 (`ubuntu1604`) および Amazon Linux (`alinux`) のサポートを削除しました。Amazon Linux 2 (`alinux2`) は引き続き完全にサポートされています。詳細については、「`base_os`」を参照してください。 • Python 3.6 以下のバージョンのサポートを終了しました。 • デフォルトのルートボリュームのサイズが 35 ギビバイト (GiB) に変更されました。詳細については、「`compute_root_volume_size`」および「`master_root_volume_size`」を参照してください。 • Elastic Fabric Adapter インストーラーを 1.12.2 に更新しました。 • EFA 設定: `efa-config-1.8-1` (`efa-config-1.7` から)	2021 年 7 月 1 日

  - EFA プロファイル:`efa-profile-1.5-1` (`efa-profile-1.4` から)
  - EFA カーネルモジュール: `efa-1.12.3` (`efa-1.10.2` から)
  - RDMA コア: `rdma-core-32.1amzn` (`rdma-core-31.2amzn` から)
  - Libfabric: `libfabric-1.11.2amzn1.1` (`libfabric-1.11.1amzn1.0` から)
  - Open MPI: `openmpi40-aws-4.1.1-2` (`openmpi40-aws-4.1.0` から)
- Slurm をバージョン `20.02.7` から `20.11.7` にアップグレードしました。
- `centos7` と `centos8` に SSM Agent をインストールします (SSM エージェントは `alinux2`、`ubuntu1804` 、および にプリインストールされています) `ubuntu2004` 。
- SGE: `qstat` でのホスト名フィルターには、必ずショートネームを使用してください。
- インスタンスのメタデータを取得するには、インスタンスメタデータサービスの

バージョン 1 (IMDSv1) ではなく、インスタンスメタデータサービスのバージョン 2 (IMDSv2) を使用してください。詳細については「アマゾン EC2 ユーザーガイド」の「Instance metadata and user data」(インスタンスメタデータとユーザーデータ) を参照してください。

- NVIDIA ドライバーをバージョン `450.80.02` から `460.73.01` にアップグレードします。
- CUDA ライブラリをバージョン `11.3.0` から `11.0` にアップグレードします。
- NVIDIA ファブリックマネージャを `nvidia-fabricmanager-460` にアップグレードします。
- AWS ParallelCluster virtualenvs で使用される Python を `3.7.10` (から) にアップグレードします`3.6.13`。
- Cinc クライアントを `16.13.16` にアップグレードします。
- aws-parallelcluster-cookbook のサードパーティーの依存関係をアップグレードします。

- apt-7.4.0 (apt-7.3.0 から)。
- iptables-8.0.0 (iptables-7.1.0 から)。
- line-4.0.1 (line-2.9.0 から)。
- openssh-2.9.1 (openssh-2.8.1 から)。
- pyenv-3.4.2 (pyenv-3.1.1 から)。
- selinux-3.1.1 (selinux-2.1.1 から)。
- ulimit-1.1.1 (ulimit-1.0.0 から)。
- yum-6.1.1 (yum-5.1.0 から)。
- yum-epel-4.1.2 (yum-epel-3.3.0 から)。

変更点の詳細については、GitHub に掲載されている aws-parallelcluster、aws-parallelcluster-cookbook、aws-parallelcluster-node の各パッケージのファイルを参照してください。

AWS ParallelCluster バージョン 2.10.4 をリリース	AWS ParallelCluster バージョン 2.10.4 がリリースされました。 変更点は次の通りです。 • Slurm をバージョン `20.02.4` から `20.02.7` にアップグレードしました。 変更点の詳細については、GitHub に掲載されている aws-parallelcluster のパッケージのファイルを参照してください。	2021 年 5 月 15 日

AWS ParallelCluster バージョン 2.10.3 をリリース	AWS ParallelCluster バージョン 2.10.3 がリリースされました。
	
	変更点は次の通りです。
	
	• AWS 中国および Ubuntu の Arm ベースの Graviton インスタンスで 18.04 および Amazon Linux 2 AWS のサポートが追加されました AWS GovCloud (US) AWS リージョン。
	• Elastic Fabric Adapter インストーラーを 1.11.2 に更新しました。
	  • EFA 設定: `efa-config-1.7` (変更なし)
	  • EFA プロファイル:`efa-profile-1.4` (`efa-profile-1.3` から)
	  • EFA カーネルモジュール: `efa-1.10.2` (変更なし)
	  • RDMA コア: `rdma-core-31.2amzn` (変更なし)
	  • Libfabric: `libfabric-1.11.1amzn1.0` (変更なし)
	  • Open MPI: `openmpi40-aws-4.1.0` (変更なし)
	
	変更点の詳細については、GitHub に掲載されている aws-parallelcluster のパッケー
		2021 年 3 月 18 日

ジのファイルを参照してください。

AWS ParallelCluster バージョン 2.10.2 をリリース	AWS ParallelCluster バージョン 2.10.2 がリリースされました。 変更点は次の通りです。 • `--dry-run` モードで Amazon EC2 RunInstances API オペレーションを呼び出したときに、クラスターターゲット AMI を使用するためのクラスター設定検証を改善しました。 • AWS ParallelCluster 仮想環境で使用されている Python バージョンを 3.6.13 に更新します。 • Arm インスタンスタイプの `sanity_check` を修正しました。 • Slurm スケジューラまたは Arm インスタンスタイプで `centos8` を使用する場合の `enable_efa` を修正しました。 • `apt update` を非対話型モード (`-y`) で実行します。 • `alinux2` と `centos8` で `encrypted_ephemeral` = true を固定します。 変更点の詳細については、GitHub に掲載されている aws-parallelcluster のパッケー	2021 年 3 月 2 日

ジのファイルを参照してください。

AWS ParallelCluster バージョン 2.10.1 をリリース	AWS ParallelCluster バージョン 2.10.1 がリリースされました。
	
	変更点は次の通りです。
	
	• アフリカ (ケープタウン) (af-south-1) 、欧州 (ミラノ) ()、me-south-1 中東 (バーレーン) (me-south-1) のサポートが追加されました AWS リージョン。起動時は、以下のようにサポートが制限されています。
	  • FSx for Lustre と Arm-based の Graviton インスタンスは、これらの AWS リージョンのいずれにも対応していません。
	  • AWS Batch はアフリカ (ケープタウン) ではサポートされていません。
	  • Amazon EBS io2とgp3ボリュームタイプは、アフリカ (ケープタウン) および欧州 (ミラノ) ではサポートされていません AWS リージョン。
	• Amazon EBS io2 および gp3 ボリュームタイプのサポートを追加しました。詳細については、[ebs] セクションおよび [raid] セクションを参照してください。
		2020 年 12 月 22 日

- alinux2、ubuntu1804 、ubuntu2004 が動作する Arm-based の Graviton2 インスタンスで Elastic Fabric Adapter のサポートを追加しました。詳細については、「Elastic Fabric Adapter」を参照してください。
- Arm パフォーマンスライブラリ 20.2.1 を Arm の AMI (alinux2、centos8、ubuntu1804) にインストールします。詳細については、「Arm パフォーマンスライブラリ」を参照してください。
- インテル MPI はバージョン 2019 アップデート 8 に更新されました (バージョン 2019 アップデート 7 から更新)。詳細については、「Intel® MPI Library 2019 Update 8」(インテル MPI ライブラリー 2019 アップデート 8) を参照してください。
- によるスロットリングによって発生するジョブの失敗を終了するために、Docker AWS Batch エントリポイントから API オペレーション呼び出しを削除 AWS CloudFormation DescribeStacks しました AWS CloudFormation。

- クラスター構成を検証する際の Amazon EC2 `DescribeInstanceTypes` API オペレーションコールの呼び出しを改善しました。
- Amazon Linux 2 の Docker イメージは、`awsbatch` スケジューラの Docker イメージを構築する際に、Amazon ECR Public から引き出されます。
- デフォルトのインスタンスタイプが、ハードコードされた`t2.micro`インスタンスタイプから、インスタンス`t3.micro`タイプにデフォルトで無料利用枠がない の無料利用枠インスタンスタイプ AWS リージョン（`t2.micro` に応じて`t3.micro`または AWS リージョン）AWS リージョンに変更されました。
- Elastic Fabric Adapter インストーラーを 1.11.1 に更新しました。
  - EFA 設定: `efa-config-1.7` (`efa-config-1.5` から)
  - EFA プロファイル:`efa-profile-1.3` (`efa-profile-1.1` から)
  - EFA カーネルモジュール: `efa-1.10.2` (変更なし)

  - RDMA コア: `rdma-core-31.2amzn` (`rdma-core-31.amzn0` から)
  - Libfabric: `libfabric-1.11.1amzn1.0` (`libfabric-1.10.1amzn1.1` から)
  - Open MPI: `openmpi40-aws-4.1.0` (`openmpi40-aws-4.0.5` から)
- `vpc_settings`、`vpc_id`、`master_subnet_id` の各パラメータが必要になりました。
- ヘッドノードの `nfsd` デーモンは、最低でも 8 スレッドを使用するように設定されました。8 つ以上のコアがある場合は、コアの数だけスレッドを使用します。`ubuntu1604` を使用した場合、設定はノードの再起動後にのみ変更されます。
- Amazon DCV を Amazon DCV 2020.2-9662 に更新しました。詳細については、「Amazon DCV 管理者ガイド」の「DCV 2020.2-9662 — 2020 年 12 月 4 日」を参照してください。
- の Intel MPI および HPC パッケージ AWS ParallelCluster は Amazon S3 から

プルされます。インテルの yum repos からは引き出されなくなりました。

- officialAMI の作成中に、すべての OSs `multi-user.target` でデフォルトのsystemd実行レベルを に変更しました。 AWS ParallelCluster AMIs DCV が有効な場合のみ、ヘッドノードのランレベルが `graphical.target` に設定されます。これにより、グラフィカルなサービス (`x/gdm` など) が必要のないときに実行されるのを防ぐことができます。
- ヘッドノードの `p4d.24xlarge` インスタンスのサポートを有効にします。
- Amazon Route 53 で Slurm ノードを登録する際の最大再試行回数を増やします。

変更点の詳細については、GitHub に掲載されている aws-parallelcluster、aws-parallelcluster-cookbook、aws-parallelcluster-node の各パッケージのファイルを参照してください。

AWS ParallelCluster バージョン 2.10.0 をリリース	AWS ParallelCluster バージョン 2.10.0 がリリースされました。
	
	変更点は次の通りです。
	
	• すべての AWS リージョン（ AWS 中国および AWS GovCloud (米国) リージョン以外) CentOS で 8 のサポートを追加しました。CentOS 6 のサポートを削除しました。
	• コンピューティングノードの `p4d.24xlarge` インスタンスのサポートを追加しました。
	• 新しい `enable_efa_gdr` 設定を使用することにより、EFA での NVIDIA GPUDirect RDMA のサポートを追加しました。
	• Amazon FSx for Lustre 機能のサポートを追加しました。
	  • Amazon FSx for Lustre ファイルシステムで、`auto_import_policy` 設定を使用してプリファレンスをインポートするように設定します。
	  • `storage_type` および `drive_cache_type` 設定を使用した HDD ベースの Amazon FSx for Lustre
		2020 年 11 月 18 日

ファイルシステムのサポートを追加しました。

- ヘッドノードのメトリクスを含む Amazon CloudWatch ダッシュボードを追加し、クラスターログに簡単にアクセスできるようになりました。詳細については、「Amazon CloudWatch ダッシュボード」を参照してください。
- `cluster_resource_bucket` 設定を使用して、クラスター構成情報を保存するため、既存の Amazon S3 バケットを使用できるようになりました。
- `pcluster createami` コマンドを強化しました。
  - AMI のビルド時にポストインストールスクリプトを使用するための `--post-install` パラメータを追加しました。
  - 別のバージョンの によって作成されたベース AMI を使用する場合に失敗する検証ステップを追加しました AWS ParallelCluster。
  - 選択したオペレーティングシステムがベース AMI のオペレーティングシステムと異なる場合に失敗

する検証ステップを追加しました。

  - AWS ParallelCluster ベース AMI の使用のサポートが追加されました。

- pcluster update コマンドを強化しました。
  - アップデート時に tags の設定を変更できるようになりました。
  - アップデートの際、コンピューティングフリートを停止することなくキューのサイズを変更できるようになりました。
- slurm_resume スクリプトの all_or_nothing_batch 設定パラメータを追加しました。True、slurm_resume の場合、Slurm で保留されているすべてのジョブに必要なすべてのインスタンスが利用可能である場合にのみ成功します。詳細については、GitHub の AWS ParallelCluster Wiki の all_or_nothing_batch 「起動の紹介」を参照してください。
- Elastic Fabric Adapter インストーラーを 1.10.1 に更新しました。

  - EFA 設定: `efa-config-1.5` (`efa-config-1.4` から)
  - EFA プロファイル:`efa-profile-1.1` (`efa-profile-1.0.0` から)
  - EFA カーネルモジュール: `efa-1.10.2` (`efa-1.6.0` から)
  - RDMA コア: `rdma-core-31.amzn0` (`rdma-core-28.amzn0` から)
  - Libfabric: `libfabric-1.11.1amzn1.0` (`libfabric-1.10.1amzn1.1` から)
  - Open MPI: `openmpi40-aws-4.0.5` (`openmpi40-aws-4.0.3` から)
- AWS GovCloud (US) リージョンで、Amazon DCV とのサポートを有効にします AWS Batch。
- AWS 中国リージョンで、Amazon FSx for Lustre のサポートを有効にします。
- NVIDIA ドライバをバージョン 450.51.05 から 450.80.02 にアップグレードします。
- NVIDIA Fabric Manager をインストールして、サポートされているプ

ラットフォームで NVIDIA NVSwitch を有効にします。

- AWS リージョン のデフォルト を削除しましたus-east-1 。デフォルトでは、この検索順序を使用します。
  - AWS リージョン -rまたは --region引数で指定します。
  - AWS_DEFAULT_REGION 環境変数
  - aws_region_name AWS ParallelCluster 設定ファイルの [aws]セクションの 設定 (デフォルトは ~/.parallelcluster/config) 。
  - region AWS CLI 設定ファイルの [default] セクションの 設定 (デフォルトは ~/aws/config) 。

変更点の詳細については、GitHub に掲載されている aws-parallelcluster、aws-parallelcluster-cookbook、aws-parallelcluster-node の各パッケージのファイルを参照してください。

AWS ParallelCluster バージョン 2.9.0 をリリース	AWS ParallelCluster バージョン 2.9.0 がリリースされました。 変更点は次の通りです。 • Slurm Workload Manager と併用することで、コンピューティングフリートに複数のキューと複数のインスタンスタイプのサポートを追加しました。キューを使用する場合、Slurm では Auto Scaling グループが使用されなくなります。Amazon Route 53 のホストゾーンがクラスターとともに作成され、Slurm スケジューラが使用されているときにコンピューティングノードの DNS 解決に使用されるようになりました。詳細については、「マルチキューモード」を参照してください。 • Arm ベースの Graviton ベースのインスタンスでの Amazon DCV AWS のサポートが追加されました。 • 起動テンプレートで CPU オプションをサポートしていないインスタンスタイプ (`*.metal` インスタンスタイプなど) で、ハイパースレッディングを無効にする機能を追加しました。	2020 年 9 月 11 日

- ヘッドノードから共有されるファイルシステムの NFS 4 へのサポートを追加しました。
- 多数のノードがクラスターに参加するときのスロットリングを避けるため、コンピューティングノードをブートストラップ AWS CloudFormation するときの cfn-init への依存関係を削除しました。
- Elastic Fabric Adapter インストーラーを 1.9.5 に更新しました。
  - EFA 設定: `efa-config-1.4` (`efa-config-1.3` から)
  - EFA のプロファイル: `efa-profile-1.0.0` (新規)
  - カーネルモジュール: `efa-1.6.0` (変更なし)
  - RDMA コア: `rdma-core-28.amzn0` (変更なし)
  - Libfabric: `libfabric-1.10.1amzn1.1` (変更なし)
  - Open MPI: `openmpi40-aws-4.0.3` (変更なし)
- Slurm をバージョン `19.05.5` から `20.02.4` にアップグレードしました。

- Amazon DCV を Amazon DCV 2020.1-9012 に更新しました。詳細については、「Amazon DCV 管理者ガイド」の「DCV 2020.1-9012 — 2020 年 8 月 24 日リリースノート」を参照してください。
- 共有 NFS ドライブをマウントする際には、ホスト名ではなくヘッドノードのプライベート IP アドレスを使用します。
- CloudWatch Logs に新しいログストリームを追加しました (`chef-client` 、`clustermgtd` 、`computemgtd` 、`slurm_resume` および `slurm_suspend`)。
- インストール前およびインストール後のスクリプトで、キュー名のサポートを追加しました。
- で AWS GovCloud (US) AWS リージョン、Amazon DynamoDB オンデマンド請求オプションを使用します。詳細については、「Amazon DynamoDB Developer Guide」(Amazon DynamoDB デベロッパーガイド) の「On-Demand Mode」(オンデマンドモード) を参照してください。

	変更点の詳細については、GitHub に掲載されている aws-parallelcluster、aws-parallelcluster-cookbook、aws-parallelcluster-node の各パッケージのファイルを参照してください。	
AWS ParallelCluster バージョン 2.8.1 をリリース	AWS ParallelCluster バージョン 2.8.1 がリリースされました。 変更点は次の通りです。 • Amazon DCV セッションの画面ロックを無効にして、ユーザーがロックアウトされないようにしました。 • Arm-based AWS Graviton-based のインスタンスタイプを含む場合の `pcluster configure` を修正します。 変更点の詳細については、GitHub に掲載されている aws-parallelcluster、aws-parallelcluster-cookbook、aws-parallelcluster-node の各パッケージのファイルを参照してください。	2020 年 8 月 4 日

AWS ParallelCluster バージョン 2.8.0 をリリース	AWS ParallelCluster バージョン 2.8.0 がリリースされました。 変更点は次の通りです。 • Arm ベースの AWS Graviton ベースのインスタンス (A1や などC6g) のサポートが追加されました。 • Amazon FSx for Lustre の自動デイリーバックアップ機能のサポートを追加しました。詳細については、「automatic_backup_retention_days 」、「copy_tags_to_backups 」、「daily_automatic_backup_start_time 」、および「fsx_backup_id 」を参照してください。 • pcluster createami から Berkshelf への依存を削除しました。 • pcluster update の堅牢性とユーザーエクスペリエンスを向上させました。詳細については、「pcluster update を使用する」を参照してください。 • Elastic Fabric Adapter インストーラーを 1.9.4 に更新しました。	2020 年 7 月 23 日

  - カーネルモジュール: `efa-1.6.0` (`efa-1.5.1` から更新)
  - RDMA コア: `rdma-core-28.amzn0` (`rdma-core-25.0` から更新)
  - Libfabric: `libfabric-1.10.1amzn1.1` (`libfabric-aws-1.9.0amzn1.1` から更新)
  - Open MPI: `openmpi40-aws-4.0.3` (変更なし)
- NVIDIA ドライバを、CentOS 6 では Tesla バージョン 440.95.01 に、その他のディストリビューションではバージョン 450.51.05 にアップグレードします。
- CentOS 6 以外のディストリビューションでは、CUDA ライブラリをバージョン 11.0 にアップグレードします。

変更点の詳細については、GitHub に掲載されている aws-parallelcluster、aws-parallelcluster-cookbook、aws-parallelcluster-node の各パッケージのファイルを参照してください。

AWS ParallelCluster バージョン 2.7.0 をリリース	AWS ParallelCluster バージョン 2.7.0 がリリースされました。 変更点は次の通りです。 • `base_os` は必須パラメータになりました。 • `scheduler` は必須パラメータになりました。 • Amazon DCV を Amazon DCV 2020.0 に更新しました。詳細については、「Amazon DCV がサラウンドサウンド 7.1 とスタイラスをサポートするバージョン 2020.0 をリリース」を参照してください。 インテル MPI はバージョン 2019 アップデート 7 に更新されました (バージョン 2019 アップデート 6 から更新)。詳細については、「Intel® MPI Library 2019 Update 7」(インテル MPI ライブラリ 2019 アップデート 7) を参照してください。 Elastic Fabric Adapter インストーラーを 1.8.4 に更新しました。 • カーネルモジュール: `efa-1.5.1` (変更なし) • RDMA コア: `rdma-core-25.0` (変更なし)	2020 年 5 月 19 日

	• Libfabric: `libfabric-aws-1.9.0amzn1.1` (変更なし) • Open MPI: `openmpi40-aws-4.0.3` (`openmpi40-aws-4.0.2` から更新) • CentOS 7 AMI をバージョン 7.8-2003 にアップグレード (7.7-1908 から更新)。詳細については、「CentOS-7 (2003) Release Notes」を参照してください。	
AWS ParallelCluster バージョン 2.6.1 をリリース	AWS ParallelCluster バージョン 2.6.1 がリリースされました。 変更点は次の通りです。 • Amazon CloudWatch Logs に保存されているログから `cfn-init-cmd` と `cfn-wire` を削除しました。詳細については、「Amazon CloudWatch Logs との統合」を参照してください。	2020 年 4 月 17 日

AWS ParallelCluster バージョン 2.6.0 をリリース	AWS ParallelCluster バージョン 2.6.0 がリリースされました。 変更点は次の通りです。 • Amazon Linux 2 のサポートを追加しました。 • クラスターとスケジューラのログの収集に Amazon CloudWatch Logs が使用されるようになりました。詳細については、「Amazon CloudWatch Logs との統合」を参照してください。 • 新しい Amazon FSx for Lustre のデプロイタイプ `SCRATCH_2` および `PERSISTENT_1` のサポートを追加しました。Ubuntu 18.04 および Ubuntu 16.04 の FSx for Lustre をサポートしました。詳細については、「fsx」を参照してください。 • Ubuntu 18.04 で Amazon DCV のサポートを追加しました。詳細については、「Amazon DCV を介してヘッドノードに接続する」を参照してください。	2020 年 2 月 27 日
AWS ParallelCluster バージョン 2.5.1 をリリース	AWS ParallelCluster バージョン 2.5.1 がリリースされました。	2019 年 12 月 13 日

AWS ParallelCluster バージョン 2.5.0 をリリース	AWS ParallelCluster バージョン 2.5.0 がリリースされました。	2019 年 11 月 18 日
AWS ParallelCluster が Intel MPI のサポートを導入	AWS ParallelCluster バージョン 2.4.1 では、Intel MPI のサポートが導入されています。	2019 年 7 月 29 日
AWS ParallelCluster が EFA のサポートを導入	AWS ParallelCluster バージョン 2.4.0 では、Elastic Fabric Adapter (EFA) のサポートが導入されています。	2019 年 6 月 11 日
AWS ParallelCluster ドキュメントサイトでリリースされた AWS ドキュメント	AWS ParallelCluster ドキュメントが 10 の言語で、HTML 形式と PDF 形式の両方で利用可能になりました。	2018 年 5 月 24 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。