AWS ParallelCluster 사용 설명서(v2)

AWS ParallelCluster

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

AWS ParallelCluster: AWS ParallelCluster 사용 설명서(v2)

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

Amazon의 상표 및 트레이드 드레스는 Amazon 외 제품 또는 서비스와 함께, Amazon 브랜드 이미지를 떨어뜨리거나 고객에게 혼동을 일으킬 수 있는 방식으로 사용할 수 없습니다. Amazon이 소유하지 않은 기타 모든 상표는 Amazon과 제휴 관계이거나 관련이 있거나 후원 관계와 관계없이 해당 소유자의 자산입니다.

Table of Contents

정의 AWS ParallelCluster 1
요금 1
설 AWS ParallelCluster정 2
설치 AWS ParallelCluster 2
AWS ParallelCluster 가상 환경에 설치(권장) 2
pip AWS ParallelCluster 를 사용하여 비가상 환경에 설치 2
설치 후 수행할 단계 3
각 환경에 대한 상세 지침 3
가상 환경 4
Linux 6
macOS 10
Windows 13
구성 AWS ParallelCluster 15
모범 사례 22
모범 사례: 마스터 인스턴스 유형 선택 22
모범 사례: 네트워크 성능 23
모범 사례: 예산 알림 24
모범 사례: 클러스터를 새 AWS ParallelCluster 마이너 또는 패치 버전으로 이동 24
CfnCluster에서 로 이동 AWS ParallelCluster 25
지원되는 리전 26
사용 AWS ParallelCluster 29
네트워크 구성 29
AWS ParallelCluster 단일 퍼블릭 서브넷의 30
AWS ParallelCluster 서브넷 2개 사용 31
AWS ParallelCluster 를 사용하여 연결된 단일 프라이빗 서브넷의 AWS Direct Connect 32
AWS ParallelClusterawsbatch 스케줄러 사용 33
사용자 지정 부트스트랩 작업 34
구성 35
인수 36
예제 36
Amazon S3 작업 38
예제 38
스팟 인스턴스 작업 38
시나리오 1: 실행 중인 작업이 없는 스팟 인스턴스가 중단됨 39

시나리오 2: 단일 노드 작업을 실행하는 스팟 인스턴스가 중단됨 39
시나리오 3: 다중 노드 작업을 실행하는 스팟 인스턴스가 중단됨 40
AWS Identity and Access Management 의 역할 AWS ParallelCluster 42
클러스터 생성을 위한 기본 설정 42
Amazon EC2의 기존 IAM 역할 사용 42
AWS ParallelCluster 인스턴스 및 사용자 정책 예제 43
에서 지원하는 스케줄러 AWS ParallelCluster 84
Son of Grid Engine 85
Slurm Workload Manager 85
Torque Resource Manager 97
AWS Batch 97
태그 지정 104
Amazon CloudWatch 대시보드 108
Amazon CloudWatch Logs와 통합 110
Elastic Fabric Adapter 112
Intel 셀렉트 솔루션 113
Intel MPI 활성화 115
Intel HPC 플랫폼 사양 116
Arm 퍼포먼스 라이브러리 117
Amazon DCV를 통해 헤드 노드에 연결합니다. 118
Amazon DCV HTTPS 인증서 119
Amazon DCV 라이선싱 119
`pcluster update` 사용하기 120
AMI 패치 및 EC2 인스턴스 교체 122
헤드 노드 인스턴스 업데이트 또는 교체 123
인스턴스 스토어 제한 123
인스턴스 스토어 제한 해결 방법 124
클러스터의 헤드 노드 중지 및 시작 125
AWS ParallelCluster CLI 명령 127
`pcluster` 127
인수 127
하위 명령: 127
`pcluster configure` 128
`pcluster create` 129
`pcluster createami` 131
`pcluster dcv` 134

pcluster delete ... 137
pcluster instances ... 139
pcluster list ... 140
pcluster ssh ... 140
pcluster start ... 142
pcluster status ... 143
pcluster stop ... 144
pcluster update ... 145
pcluster version ... 147
pcluster-config ... 147
이름 지정된 인수 ... 148
구성 ... 150
레이아웃 ... 151
[global] 섹션 ... 151
cluster_template ... 151
update_check ... 152
sanity_check ... 152
[aws] 섹션 ... 152
[aliases] 섹션 ... 153
[cluster] 섹션 ... 154
additional_cfn_template ... 156
additional_iam_policies ... 156
base_os ... 157
cluster_resource_bucket ... 159
cluster_type ... 159
compute_instance_type ... 160
compute_root_volume_size ... 161
custom_ami ... 161
cw_log_settings ... 162
dashboard_settings ... 162
dcv_settings ... 163
desired_vcpus ... 163
disable_cluster_dns ... 164
disable_hyperthreading ... 164
ebs_settings ... 165
ec2_iam_role ... 166

efs_settings 166
enable_efa 166
enable_efa_gdr 167
enable_intel_hpc_platform 168
encrypted_ephemeral 168
ephemeral_dir 169
extra_json 169
fsx_settings 169
iam_lambda_role 170
initial_queue_size 170
key_name 171
maintain_initial_size 171
master_instance_type 172
master_root_volume_size 173
max_queue_size 173
max_vcpus 174
min_vcpus 174
placement 174
placement_group 175
post_install 176
post_install_args 176
pre_install 176
pre_install_args 177
proxy_server 177
queue_settings 177
raid_settings 178
s3_read_resource 178
s3_read_write_resource 179
scaling_settings 179
scheduler 179
shared_dir 180
spot_bid_percentage 181
spot_price 181
tags 182
template_url 182
vpc_settings 183

[compute_resource] 섹션 ... 183
initial_count ... 184
instance_type ... 184
max_count ... 185
min_count ... 185
spot_price ... 186
[cw_log] 섹션 ... 186
enable ... 186
retention_days ... 187
[dashboard] 섹션 ... 187
enable ... 187
[dcv] 섹션 ... 188
access_from ... 189
enable ... 189
port ... 189
[ebs] 섹션 ... 190
shared_dir ... 191
ebs_kms_key_id ... 191
ebs_snapshot_id ... 191
ebs_volume_id ... 192
encrypted ... 192
volume_iops ... 192
volume_size ... 193
volume_throughput ... 194
volume_type ... 195
[efs] 섹션 ... 195
efs_fs_id ... 196
efs_kms_key_id ... 197
encrypted ... 198
performance_mode ... 198
provisioned_throughput ... 199
shared_dir ... 199
throughput_mode ... 199
[fsx] 섹션 ... 200
auto_import_policy ... 202
automatic_backup_retention_days ... 203

copy_tags_to_backups 203
daily_automatic_backup_start_time 204
data_compression_type 204
deployment_type 205
drive_cache_type 206
export_path 206
fsx_backup_id 207
fsx_fs_id 207
fsx_kms_key_id 208
import_path 208
imported_file_chunk_size 209
per_unit_storage_throughput 209
shared_dir 210
storage_capacity 210
storage_type 211
weekly_maintenance_start_time 213
[queue] 섹션 213
compute_resource_settings 214
compute_type 214
disable_hyperthreading 215
enable_efa 215
enable_efa_gdr 216
placement_group 216
[raid] 섹션 217
shared_dir 218
ebs_kms_key_id 218
encrypted 218
num_of_raid_volumes 219
raid_type 219
volume_iops 219
volume_size 220
volume_throughput 221
volume_type 222
[scaling] 섹션 223
scaledown_idletime 223
[vpc] 섹션 223

additional_sg ... 224
compute_subnet_cidr ... 224
compute_subnet_id ... 224
master_subnet_id ... 225
ssh_from ... 225
use_public_ips ... 225
vpc_id ... 226
vpc_security_group_id ... 226
예시 ... 38
Slurm 예 ... 227
SGE 및 Torque 예제 ... 228
AWS Batch 예제 ... 229
AWS ParallelCluster 작동 방식 ... 231
AWS ParallelCluster 프로세스 ... 231
SGE and Torque integration processes ... 232
Slurm integration processes ... 238
AWS 에서 사용하는 서비스 AWS ParallelCluster ... 238
AWS Auto Scaling ... 239
AWS Batch ... 240
AWS CloudFormation ... 240
Amazon CloudWatch ... 240
Amazon CloudWatch Logs ... 241
AWS CodeBuild ... 241
Amazon DynamoDB ... 241
Amazon Elastic Block Store ... 241
Amazon Elastic Compute Cloud ... 242
Amazon Elastic 컨테이너 레지스트리 ... 242
Amazon EFS ... 242
Amazon FSx for Lustre ... 242
AWS Identity and Access Management ... 243
AWS Lambda ... 243
Amazon DCV ... 243
Amazon Route 53 ... 243
Amazon Simple Notification Service ... 244
Amazon Simple Queue Service ... 244
Amazon Simple Storage Service(S3) ... 245

Amazon VPC 245
AWS ParallelCluster Auto Scaling 245
확장 246
축소 247
정적 클러스터 247
자습서 248
에서 첫 번째 작업 실행 AWS ParallelCluster 248
설치 확인 248
첫 번째 클러스터 생성 249
헤드 노드에 로그인 249
SGE를 사용하여 첫 번째 작업 실행 250
사용자 지정 AWS ParallelCluster AMI 빌드 251
AWS ParallelCluster AMI를 사용자 지정하는 방법 252
AMI 수정 252
사용자 지정 AWS ParallelCluster AMI 빌드 254
런타임 시 사용자 지정 AMI 사용 256
AWS ParallelCluster 및 awsbatch 스케줄러를 사용하여 MPI 작업 실행 256
클러스터 생성 257
헤드 노드에 로그인 249
를 사용하여 첫 번째 작업 실행 AWS Batch 259
다중 노드 병렬 환경에서 MPI 작업 실행 261
사용자 지정 KMS 키를 사용한 디스크 암호화 265
역할 만들기 265
키 권한을 부여하세요. 266
클러스터 생성 257
다중 대기열 모드 자습서 267
여러 대기열 모드로 AWS ParallelCluster 에서 작업 실행 267
개발 279
사용자 지정 AWS ParallelCluster 쿡북 설정 279
단계 279
사용자 지정 AWS ParallelCluster 노드 패키지 설정 281
단계 281
문제 해결 283
로그 검색 및 보존 283
스택 배포 문제 해결 284
다중 대기열 모드 클러스터의 문제 해결 284

키 로그 ... 285
노드 초기화 문제 해결 ... 286
예상치 못한 노드 교체 및 종료 문제 해결 ... 288
문제가 있는 인스턴스 및 노드 교체, 종료 또는 전원 끄기 ... 289
기타 알려진 노드 및 작업 문제 해결 ... 289
단일 대기열 모드 클러스터의 문제 해결 ... 289
키 로그 ... 290
시작 및 조인 작업 실패 문제 해결 ... 291
규모 조정 문제 해결 ... 292
기타 클러스터 관련 문제 해결 ... 292
배치 그룹 및 인스턴스 시작 문제 ... 292
교체할 수 없는 디렉터리 ... 293
Amazon DCV의 문제 해결 ... 294
Amazon DCV용 로그 ... 294
Amazon DCV 인스턴스 유형 메모리 ... 294
Ubuntu Amazon DCV 문제 ... 294
AWS Batch 통합을 통한 클러스터의 문제 해결 ... 295
헤드 노드 문제 ... 295
AWS Batch 다중 노드 병렬 작업 제출 문제 ... 295
컴퓨팅 문제 ... 295
작업 실패 ... 295
리소스 생성 실패 시 문제 해결 ... 296
IAM 정책 크기 문제 해결 ... 297
추가 지원 ... 297
AWS ParallelCluster 지원 정책 ... 298
보안 ... 299
에서 사용하는 서비스에 대한 보안 정보 AWS ParallelCluster ... 299
데이터 보호 ... 300
데이터 암호화 ... 301
다음 사항도 참조하세요. ... 302
ID 및 액세스 관리 ... 302
규정 준수 확인 ... 303
TLS 1.2 적용 ... 304
현재 지원되는 프로토콜 확인 ... 304
OpenSSL 및 Python 컴파일 ... 305
릴리스 정보 및 문서 기록 ... 307

..... cccxlvii

정의 AWS ParallelCluster

AWS ParallelCluster 는에서 고성능 컴퓨팅(HPC) 클러스터를 배포하고 관리하는 데 도움이 되는 AWS 지원되는 오픈 소스 클러스터 관리 도구입니다 AWS 클라우드. 필요한 컴퓨팅 리소스, 스케줄러와 공유 파일 시스템을 자동으로 설정합니다. AWS Batch 및 Slurm 스케줄러와 AWS ParallelCluster 함께를 사용할 수 있습니다.

를 AWS ParallelCluster사용하면 개념 증명 및 프로덕션 HPC 컴퓨팅 환경을 빠르게 구축하고 배포할 수 있습니다. 또한 전체 DNA 시퀀싱 워크플로를 자동화하는 유전체학 포털 AWS ParallelCluster과 같은 상위 수준 워크플로를 빌드하고 배포할 수 있습니다.

요금

AWS ParallelCluster 명령줄 인터페이스(CLI) 또는 API를 사용하는 경우 AWS ParallelCluster 이미지 및 클러스터를 생성하거나 업데이트할 때 생성된 AWS 리소스에 대해서만 비용을 지불합니다. 자세한 내용은 AWS 에서 사용하는 서비스 AWS ParallelCluster 단원을 참조하십시오.

설 AWS ParallelCluster정

주제

- 설치 AWS ParallelCluster
- 구성 AWS ParallelCluster
- 모범 사례
- CfnCluster에서 로 이동 AWS ParallelCluster
- 지원되는 리전

설치 AWS ParallelCluster

AWS ParallelCluster 는 Python 패키지로 배포되며 Python 패키지 관리자pip인를 사용하여 설치됩니다. Python 패키지 설치에 대한 자세한 내용은 Python 패키징 사용 설명서의 패키지 설치를 참조하세요.

설치 방법 AWS ParallelCluster:

- 가상 환경 사용(권장)
- `pip` 사용

최신 CLI의 버전 번호는 GitHub의 릴리스 페이지에서 확인할 수 있습니다.

이 안내서의 명령 예제에서는 Python v3이 설치되어 있다고 가정합니다. `pip` 명령 예제에는 `pip3` 버전이 사용됩니다.

AWS ParallelCluster 가상 환경에 설치(권장)

가상 환경에 AWS ParallelCluster 를 설치하는 것이 좋습니다. AWS ParallelCluster 를 사용하여를 설치하려고 `pip3`할 때 문제가 발생하면 가상 환경에 AWS ParallelCluster 를 설치하여 도구와 해당 종속성을 격리할 수 있습니다. 일반적으로 사용하는 것과 다른 버전의 Python을 사용할 수 있습니다.

pip AWS ParallelCluster 를 사용하여 비가상 환경에 설치

Linux, Windows 및 macOS AWS ParallelCluster 에서의 기본 배포 방법은 Python의 패키지 관리자pip인 입니다. pip는 Python 패키지 및 해당 종속 항목을 쉽게 설치, 업그레이드 및 제거하는 방법을 제공합니다.

현재 AWS ParallelCluster 버전

AWS ParallelCluster 는 정기적으로 업데이트됩니다. 최신 버전을 사용하는지 확인하려면 GitHub의 릴리스 페이지를 참조하세요.

pip 및 지원되는 버전의 Python이 이미 있는 경우 다음 명령을 AWS ParallelCluster 사용하여를 설치할 수 있습니다. Python 버전 3 이상이 설치되어 있으면 **pip3** 명령을 사용하는 것이 좋습니다.

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

설치 후 수행할 단계

설치 후 실행 파일 경로를 PATH 변수에 추가해야 할 AWS ParallelCluster수 있습니다. 플랫폼별 지침은 다음 주제를 참조하세요.

- Linux – 명령줄 경로에 AWS ParallelCluster 실행 파일 추가
- macOS – 명령줄 경로에 AWS ParallelCluster 실행 파일 추가
- Windows – 명령줄 경로에 AWS ParallelCluster 실행 파일 추가

를 실행하여가 올바르게 AWS ParallelCluster 설치되었는지 확인할 수 있습니다`pcluster version`.

```
$ pcluster version
2.11.9
```

AWS ParallelCluster 는 정기적으로 업데이트됩니다. 의 최신 버전으로 업데이트하려면 설치 명령을 다시 AWS ParallelCluster실행합니다. 최신 버전의에 대한 자세한 내용은 AWS ParallelCluster 릴리스 정보를 AWS ParallelCluster참조하세요.

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

제거하려면를 AWS ParallelCluster사용합니다`pip uninstall`.

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

Python 및 pip가 없는 경우 사용 환경에 해당하는 절차를 사용합니다.

각 환경에 대한 상세 지침

- AWS ParallelCluster 가상 환경에 설치(권장)

- Linux AWS ParallelCluster 에 설치
- macOS AWS ParallelCluster 에 설치
- Windows AWS ParallelCluster 에 설치

AWS ParallelCluster 가상 환경에 설치(권장)

요구 사항 버전이 다른 pip 패키지와 충돌하지 않도록 가상 환경에 AWS ParallelCluster 를 설치하는 것이 좋습니다.

사전 조건

- pip 및 Python이 설치되었는지 확인합니다. pip3 및 Python 3 버전 3.8을 권장합니다. Python 2를 사용하는 경우 pip3 대신 pip를, venv 대신 virtualenv를 사용합니다.

가상 환경에 AWS ParallelCluster 를 설치하려면

1. virtualenv를 설치하지 않은 경우 pip3을 사용하여 virtualenv를 설치합니다. python3 -m virtualenv help에 도움말 정보가 표시되면 2단계로 이동합니다.

   Linux, macOS, or Unix

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

   exit를 실행하여 현재 터미널 창에서 나가서 새 터미널 창을 열어 환경에 변경 사항을 적용합니다.

   Windows

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

   exit를 실행하여 현재 명령 프롬프트에서 나가서 새 명령 프롬프트를 열어 환경에 변경 사항을 적용합니다.
2. 가상 환경을 생성하고 이름을 지정합니다.

   Linux, macOS, or Unix

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

또는 -p 옵션을 사용하여 특정 Python 버전을 지정할 수 있습니다.

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

Windows

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

3. 새 가상 환경을 활성화합니다.

   Linux, macOS, or Unix

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

   Windows

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

4. 가상 환경에 AWS ParallelCluster 를 설치합니다.

   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. AWS ParallelCluster 가 올바르게 설치되었는지 확인합니다.

   Linux, macOS, or Unix

   ```
   $ pcluster version
   2.11.9
   ```

   Windows

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

`deactivate` 명령을 사용하여 가상 환경을 종료할 수 있습니다. 새 세션을 시작할 때마다 환경을 다시 활성화해야 합니다.

의 최신 버전으로 업그레이드하려면 설치 명령을 다시 AWS ParallelCluster실행합니다.

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 에 설치

Python용 패키지 관리자pip인를 사용하여 대부분의 Linux 배포에 AWS ParallelCluster 및 해당 종속성을 설치할 수 있습니다. 먼저 Python 및 pip가 설치되었는지 확인합니다.

1. 해당 버전의 Linux에 Python 및 pip가 포함되어 있는지 확인하려면 `pip --version`을 실행합니다.

   ```
   $ pip --version
   ```

   를 pip 설치한 경우 pip를 사용하여 설치 주제 AWS ParallelCluster 로 이동합니다. 그렇지 않은 경우 2단계로 진행합니다.
2. Python이 설치되었는지 확인하려면 `python --version`을 실행합니다.

   ```
   $ python --version
   ```

   Python 3 버전 3.6 이상 또는 Python 2 버전 2.7이 설치되어 있는 경우 pip를 AWS ParallelCluster 사용하여 설치 주제로 이동합니다. 그렇지 않은 경우 Python을 설치한 다음 이 절차로 돌아와서 pip를 설치합니다.
3. Python Packaging Authority에서 제공하는 스크립트를 사용하여 pip를 설치합니다.
4. `curl` 명령을 사용하여 설치 스크립트를 다운로드합니다.

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

5. Python으로 스크립트를 실행하여 pip 및 기타 필요한 지원 패키지의 최신 버전을 다운로드하여 설치합니다.

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

or

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

--user 스위치를 포함한 경우 스크립트는 pip를 ~/.local/bin 경로에 설치합니다.

6. pip가 포함된 폴더가 PATH 변수의 일부인지 확인하려면 다음을 수행하세요.

a. 사용자 폴더에서 셸의 프로파일 스크립트를 찾습니다. 어떤 셸을 가지고 있는지 잘 모르는 경우 basename $SHELL을 실행합니다.

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

- Bash – .bash_profile, .profile 또는 .bash_login
- Zsh – .zshrc
- Tcsh – .tcshrc, .cshrc 또는 .login

b. 다음 예제와 유사한 프로필 스크립트 끝에 내보내기 명령을 추가합니다.

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

내보내기 명령은 경로(이 예제에서 ~/.local/bin)를 기존 PATH 변수 앞에 삽입합니다.

c. 해당 변경 사항을 적용하려면 현재 세션에 프로필을 다시 로드합니다.

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

7. pip가 올바르게 설치되었는지 확인합니다.

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

Sections

- AWS ParallelCluster 를 사용하여 설치 pip
- 명령줄 경로에 AWS ParallelCluster 실행 파일 추가
- Linux에 Python 설치

AWS ParallelCluster 를 사용하여 설치 **pip**

를 사용하여 pip를 설치합니다 AWS ParallelCluster.

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

--user 스위치를 사용하면 pip가 AWS ParallelCluster 를 ~/.local/bin에 설치합니다.

가 올바르게 AWS ParallelCluster 설치되었는지 확인합니다.

```
$ pcluster version
2.11.9
```

최신 버전으로 업그레이드하려면 설치 명령을 다시 실행합니다.

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

명령줄 경로에 AWS ParallelCluster 실행 파일 추가

pip를 사용하여 설치한 후 pcluster 실행 파일을 운영 체제의 PATH 환경 변수에 추가해야 할 수 있습니다.

가 pip 설치된 폴더를 확인하려면 다음 명령을 AWS ParallelCluster실행합니다.

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

설치할 때 --user 스위치를 생략한 경우 AWS ParallelCluster실행 파일이 Python 설치 bin 폴더에 있을 수 있습니다. Python 설치 위치를 모르는 경우, 다음 명령을 실행하세요.

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

실제 실행 파일이 아니라 symlink 경로가 출력될 수 있습니다. symlink가 가리키는 위치를 보려면 `ls -al`을 실행합니다.

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

설치 AWS ParallelCluster의 3단계에서 경로에 추가한 것과 동일한 폴더인 경우 설치가 완료된 것입니다. 그렇지 않으면 3a - 3c의 동일한 단계를 다시 수행하여 이 폴더를 경로에 추가합니다.

Linux에 Python 설치

배포가 Python과 함께 제공되지 않았거나 이전 버전과 함께 제공된 경우 `pip` 및를 설치하기 전에 Python을 설치합니다 AWS ParallelCluster.

Linux에 Python 3를 설치하려면

1. Python이 이미 설치되어 있는지 확인합니다.

   ```
   $ python3 --version
   ```

   or

   ```
   $ python --version
   ```

   > **Note**
   >
   > Linux 배포에 Python이 포함되어 있는 경우 Python 개발자 패키지를 설치해야 할 수도 있습니다. 개발자 패키지에는 확장을 컴파일하고 AWS ParallelCluster를 설치하는 데 필요한 헤더와 라이브러리가 포함되어 있습니다. 패키지 관리자를 사용하여 개발자 패키지를 설치합니다. 이 파일은 일반적으로 `python-dev` 또는 `python-devel`이라는 이름이 지정되어 있습니다.

2. Python 2.7 이상이 설치되어 있지 않은 경우 배포의 패키지 관리자를 사용하여 Python을 설치합니다. 다음과 같이 명령과 패키지 이름이 다릅니다.

   - Ubuntu와 같은 Debian 계열 시스템에는 `apt`를 사용합니다.

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

- Red Hat 및 계열 시스템에는 yum을 사용합니다.

```
$ sudo yum install python3
```

- SUSE 및 계열 시스템에는 zypper를 사용합니다.

```
$ sudo zypper install python3
```

3. Python이 올바르게 설치되었는지 확인하려면 명령 프롬프트 또는 셸을 열고 다음 명령을 실행합니다.

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

macOS AWS ParallelCluster 에 설치

Sections

- 사전 조건
- pip를 사용하여 macOS AWS ParallelCluster 에 설치
- 명령줄 경로에 AWS ParallelCluster 실행 파일 추가

사전 조건

- Python 3 버전 3.7+ 또는 Python 2 버전 2.7

Python 설치를 확인합니다.

```
$ python --version
```

컴퓨터에 아직 Python이 설치되지 않았거나 다른 버전의 Python을 설치하려는 경우 Linux AWS ParallelCluster 에 설치의 절차를 수행하세요.

pip를 사용하여 macOS AWS ParallelCluster 에 설치

를 pip 직접 사용하여를 설치할 수도 있습니다 AWS ParallelCluster. pip가 없는 경우 기본 설치 주제의 지침을 따릅니다. pip3 --version을 실행하여 macOS 버전에 Python과 pip3이 이미 포함되어 있는지 확인합니다.

```
$ pip3 --version
```

macOS AWS ParallelCluster 에를 설치하려면

1. Python.org의 다운로드 페이지에서 Python의 최신 버전을 다운로드하고 설치합니다.
2. Python Packaging Authority에서 제공하는 pip3 설치 스크립트를 다운로드하고 실행합니다.

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

3. 새로 설치된를 사용하여 pip3를 설치합니다 AWS ParallelCluster. Python 버전 3 이상을 사용하는 경우에는 pip3 명령을 사용하는 것이 좋습니다.

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

4. AWS ParallelCluster 가 올바르게 설치되었는지 확인합니다.

   ```
   $ pcluster version
   2.11.9
   ```

   프로그램을 찾을 수 없는 경우 프로그램을 명령줄 경로에 추가합니다.

최신 버전으로 업그레이드하려면 설치 명령을 다시 실행합니다.

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

명령줄 경로에 AWS ParallelCluster 실행 파일 추가

pip를 사용하여 설치한 후 pcluster 프로그램을 운영 체제의 PATH 환경 변수에 추가해야 할 수 있습니다. 프로그램의 위치는 Python 설치 위치에 따라 달라집니다.

Example AWS ParallelCluster 설치 위치 - Python 3.6 및 **pip** (사용자 모드)를 사용하는 macOS

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

앞의 예제에 나온 버전을 현재 가지고 있는 Python 버전으로 대체합니다.

Python 설치 위치를 모르는 경우, which python을 실행하세요.

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

실제 프로그램 경로가 아니라 symlink 경로가 출력될 수 있습니다. `ls -al`을 실행하여 어디를 가리키는지 확인합니다.

```
$ 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
```

pip는 Python 애플리케이션이 있는 것과 동일한 폴더에 프로그램을 설치합니다. 이 폴더를 PATH 변수에 추가합니다.

PATH 변수를 수정하려면(Linux, macOS 또는 Unix)

1. 사용자 폴더에서 셸의 프로파일 스크립트를 찾습니다. 어떤 셸을 가지고 있는지 잘 모르는 경우 `echo $SHELL`을 실행합니다.

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

   - Bash – `.bash_profile`, `.profile` 또는 `.bash_login`
   - Zsh – `.zshrc`
   - Tcsh – `.tcshrc`, `.cshrc` 또는 `.login`
2. 내보내기 명령을 프로필 스크립트에 추가하세요.

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

   이 명령은 이 예제의 `~/.local/bin` 경로를 현재 PATH 변수에 추가합니다.
3. 프로필을 현재 세션에 로드합니다.

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

Windows AWS ParallelCluster 에 설치

Python용 패키지 관리자pip인를 사용하여 Windows AWS ParallelCluster 에를 설치할 수 있습니다. pip가 이미 있는 경우 기본 설치 주제의 지침을 따릅니다.

Sections

- Windowspip에서 Python 및를 AWS ParallelCluster 사용하여 설치
- 명령줄 경로에 AWS ParallelCluster 실행 파일 추가

Windows**pip**에서 Python 및를 AWS ParallelCluster 사용하여 설치

Python Software Foundation은 pip가 포함된 Windows용 설치 관리자를 제공합니다.

Python 및 **pip**를 설치하려면(Windows)

1. Python.org의 다운로드 페이지에서 Python Windows x86-64 설치 관리자를 다운로드합니다.
2. 설치 관리자를 실행합니다.
3. Add Python 3 to PATH(PATH에 Python 3 추가)를 선택합니다.
4. Install Now(지금 설치)를 선택합니다.

설치 관리자가 사용자 폴더에 Python을 설치하고, 사용자 경로에 프로그램 폴더를 추가합니다.

AWS ParallelCluster 를 사용하여를 설치하려면**pip3**(Windows)

Python 버전 3 이상을 사용하는 경우에는 pip3 명령을 사용하는 것이 좋습니다.

1. 시작 메뉴에서 Windows 명령 프롬프트를 엽니다.
2. 다음 명령을 사용하여 Python과 pip가 모두 올바르게 설치되었는지 확인합니다.

```
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 를 사용하여를 설치합니다pip.

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

4. AWS ParallelCluster 가 올바르게 설치되었는지 확인합니다.

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

최신 버전으로 업그레이드하려면 설치 명령을 다시 실행합니다.

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

명령줄 경로에 AWS ParallelCluster 실행 파일 추가

AWS ParallelCluster 를 사용하여 설치한 후 운영 체제의 PATH 환경 변수에 `pcluster` 프로그램을 `pip`추가합니다.

다음 명령을 실행하여 `pcluster` 프로그램이 어디에 설치되어 있는지 확인할 수 있습니다.

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

해당 명령이 어떤 결과도 반환하지 않는다면 수동으로 경로를 추가해야 합니다. 명령줄 또는 Windows 탐색기를 사용하여 프로그램이 컴퓨터에서 어디에 설치되어 있는지 검색합니다. 일반적인 경로는 다음과 같습니다.

- Python 3 및 **`pip3`** – `C:\Python38\Scripts\`
- Python 3 및 **`pip3`** --사용자 옵션 – `%APPDATA%\Python\Python38\Scripts`

> **Note**
>
> 버전 번호를 포함한 폴더 이름은 달라질 수 있습니다. 앞의 예제에서는 Python38을 보여 줍니다. 필요한 경우 사용 중인 버전 번호로 바꿉니다.

PATH 변수를 수정하려면(Windows)

1. Windows 키를 누르고 **`environment variables`**를 입력하세요.
2. 계정의 환경 변수 편집을 선택합니다.
3. 경로를 선택한 다음 편집을 선택합니다.

4. Variable value(변수 값) 필드에 경로를 추가합니다. 예: ***C:\new\path***
5. 확인을 두 번 선택하여 새 설정을 적용합니다.
6. 실행 중인 명령 프롬프트를 모두 닫았다가 명령 프롬프트 창을 다시 엽니다.

구성 AWS ParallelCluster

설치 후 다음 구성 단계를 AWS ParallelCluster완료합니다.

AWS 계정에 `pcluster` CLI를 실행하는 데 필요한 권한이 포함된 역할이 있는지 확인합니다. 자세한 내용은 AWS ParallelCluster 인스턴스 및 사용자 정책 예제 단원을 참조하십시오.

자격 AWS 증명을 설정합니다. 자세한 내용은AWS CLI 사용 설명서의 AWS CLI구성을 참조하세요.

```
$ 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]:
```

클러스터가 시작 AWS 리전 되는 에는 Amazon EC2 키 페어가 하나 이상 있어야 합니다. 자세한 내용을 알아보려면 Amazon EC2 key pairs의 Amazon EC2 키 페어를 참조하세요.

```
$ pcluster configure
```

구성 마법사는 클러스터를 생성하는 데 필요한 모든 정보를 요구합니다. 를 스케줄러 AWS Batch 로 사용할 때와를 사용할 때의 시퀀스 세부 정보는 다릅니다Slurm. 클러스터 구성에 대한 자세한 정보는 구성 섹션을 참조하세요.

> **Note**
>
> 버전 2.11.5부터 SGE 또는 Torque 스케줄러 사용을 지원하지 AWS ParallelCluster 않습니다. 2.11.4 이하의 버전에서는 계속 사용할 수 있지만 AWS 서비스 및 AWS 지원 팀의 향후 업데이트 또는 문제 해결 지원을 받을 수 없습니다.

Slurm

유효한 AWS 리전 식별자 목록에서 클러스터를 실행할 AWS 리전 를 선택합니다.

> **Note**
>
> AWS 리전 표시된 목록은 계정의 파티션을 기반으로 하며 계정에 대해 AWS 리전 활성화된 만 포함합니다. 계정 활성화에 AWS 리전 대한 자세한 내용은의 관리를 AWS 리전 참조하세요AWS 일반 참조. 표시된 예제는 AWS 글로벌 파티션에서 가져온 것입니다. 계정이 AWS GovCloud (US) 파티션에 있는 경우 해당 파티션 AWS 리전 의 만 나열됩니다(gov-us-east-1 및 gov-us-west-1). 마찬가지로 계정이 AWS 중국 파티션에 있는 경우 cn-north-1 및 만 cn-northwest-1 표시됩니다. 에서 AWS 리전 지원하는 전체 목록은 단원을 AWS ParallelCluster참조하십시오지원되는 리전.

```
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]:
```

클러스터와 함께 사용할 스케줄러를 선택합니다.

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

운영 체제를 선택합니다.

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

> **Note**
>
> AWS ParallelCluster 버전 2.6.0에서에 대한 지원이 `alinux2` 추가되었습니다.

컴퓨팅 노드 클러스터의 최소 크기와 최대 크기가 입력됩니다. 이 값은 인스턴스 수로 측정됩니다.

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

헤드 및 컴퓨팅 노드 인스턴스 유형이 입력됩니다. 인스턴스 유형의 경우 계정 인스턴스 한도는 요구 사항을 충족할 만큼 충분히 큽니다. 자세한 내용은 Amazon EC2 사용 설명서에서 온디맨드 인스턴스 제한을 참조하세요.

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

키 페어는 선택한 AWS 리전에서 Amazon EC2로 등록된 키 페어 중에 선택됩니다.

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

이전 단계를 완료한 후 기존 VPC를 사용할지 아니면가 자동으로 VPC를 AWS ParallelCluster 생성하도록 할지 결정합니다. 올바르게 구성된 VPC가 없는 경우에서 새 VPC를 생성할 수 AWS ParallelCluster 있습니다. 동일한 퍼블릭 서브넷의 헤드 및 컴퓨팅 노드를 모두 사용하거나 모든 노드가 프라이빗 서브넷에 있는 퍼블릭 서브넷의 헤드 노드만 사용합니다. 의 VPCs 있습니다 AWS 리전. 기본 한도는 각 VPCs AWS 리전. 이 제한값 및 증가 요청 방법에 대한 자세한 내용은 Amazon VPC 사용 설명서의 VPC 및 서브넷을 참조하세요.

에서 VPC를 AWS ParallelCluster 생성하도록 허용하는 경우 모든 노드가 퍼블릭 서브넷에 있어야 하는지 여부를 결정해야 합니다.

> **Important**
>
> 에서 생성한 VPCs 기본적으로 VPC 흐름 로그를 활성화하지 AWS ParallelCluster 않습니다. VPC 흐름 로그를 사용하여 VPC의 네트워크 인터페이스에서 송수신되는 IP 트래픽에 대한 정보를 캡처할 수 있습니다. 자세한 내용은 Amazon VPC 사용 설명서의 VPC 흐름 로그를 참조하세요.

> **Note**
>
> `1. Master in a public subnet and compute fleet in a private subnet` 항목을 선택하면 AWS ParallelCluster 항목은 프리 티어 리소스를 지정하더라도 추가 비용이 발생하는 NAT 게이트웨이를 생성합니다.

```
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
```

새 VPC를 생성하지 않는 경우 기존 VPC를 선택해야 합니다.

VPC를 AWS ParallelCluster 생성하도록 선택한 경우 나중에를 사용하여 삭제할 수 있도록 VPC ID AWS CLI 를 기록해 둡니다.

```
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
```

VPC를 선택한 후 기존 서브넷을 사용할지 아니면 새 서브넷을 생성할지를 결정해야 합니다.

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

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

AWS Batch

유효한 AWS 리전 식별자 목록에서 클러스터를 실행할 AWS 리전 를 선택합니다.

```
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]:
```

클러스터와 함께 사용할 스케줄러를 선택합니다.

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

`awsbatch`을 스케줄러로 선택한 경우 `alinux2`가 운영 체제로 사용됩니다.

컴퓨팅 노드 클러스터의 최소 크기와 최대 크기가 입력됩니다. 이 값은 vCPU에서 측정됩니다.

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

헤드 노드 인스턴스 유형이 입력됩니다. awsbatch 스케줄러를 사용할 때 컴퓨팅 노드는 인스턴스 유형인 optimal을 사용합니다.

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

Amazon EC2 키 페어는 선택한 AWS 리전에서 Amazon EC2로 등록된 키 페어 중에 선택됩니다.

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

기존 VPCs 아니면에서 VPCs AWS ParallelCluster 자동으로 생성할지 결정합니다. 올바르게 구성된 VPC가 없는 경우에서 새 VPC를 생성할 수 AWS ParallelCluster 있습니다. 동일한 퍼블릭 서브넷의 헤드 및 컴퓨팅 노드를 모두 사용하거나 모든 노드가 프라이빗 서브넷에 있는 퍼블릭 서브넷의 헤드 노드만 사용합니다. 의 VPCs 있습니다 AWS 리전. 기본 VPC 수는 5개입니다. 이 제한값 및 증가 요청 방법에 대한 자세한 내용은 Amazon VPC 사용 설명서의 VPC 및 서브넷을 참조하세요.

> **Important**
>
> 에서 생성한 VPCs 기본적으로 VPC 흐름 로그를 활성화하지 AWS ParallelCluster 않습니다. VPC 흐름 로그를 사용하여 VPC의 네트워크 인터페이스에서 송수신되는 IP 트래픽에 대한 정보를 캡처할 수 있습니다. 자세한 내용은 Amazon VPC 사용 설명서의 VPC 흐름 로그를 참조하세요.

에서 VPC를 AWS ParallelCluster 생성하도록 허용하는 경우 모든 노드가 퍼블릭 서브넷에 있어야 하는지 여부를 결정합니다.

> **Note**
>
> 1. Master in a public subnet and compute fleet in a private subnet
> 항목을 선택하면 AWS ParallelCluster 항목은 프리 티어 리소스를 지정하더라도 추가 비용이 발생하는 NAT 게이트웨이를 생성합니다.

```
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
```

새 VPC를 생성하지 않는 경우 기존 VPC를 선택해야 합니다.

VPC를 AWS ParallelCluster 생성하도록 선택한 경우 나중에를 사용하여 삭제할 수 있도록 VPC ID AWS CLI 를 기록해 둡니다.

```
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
```

VPC를 선택한 후 기존 서브넷을 사용할지 아니면 새 서브넷을 생성할지를 결정합니다.

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

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

이전 단계를 완료하면 간단한 클러스터가 VPC로 시작됩니다. VPC는 퍼블릭 IP 주소를 지원하는 기존 서브넷을 사용합니다. 서브넷의 라우팅 테이블은 `0.0.0.0/0 => igw-xxxxxx`입니다. 다음 조건을 알아 두세요.

- VPC에는 `DNS Resolution = yes` 및 `DNS Hostnames = yes`가 있어야 합니다.
- 또한 VPC에는 AWS 리전에 대한 올바른 `domain-name`가 있는 DHCP 옵션이 있어야 합니다. 기본 DHCP 옵션 세트는 이미 필수 AmazonProvidedDNS를 지정합니다. 도메인 이름 서버를 두 개 이상 지정하는 경우 Amazon VPC 사용 설명서의 DHCP 옵션 세트를 참조하세요. 프라이빗 서브넷을 사용하는 경우 NAT 게이트웨이 또는 내부 프록시를 사용하여 컴퓨팅 노드에 대한 웹 액세스를 활성화하세요. 자세한 내용은 네트워크 구성 단원을 참조하십시오.

모든 설정에 올바른 값이 포함되면 생성 명령을 실행하여 클러스터를 시작할 수 있습니다.

```
$ pcluster create mycluster
```

클러스터가 "CREATE_COMPLETE" 상태에 도달하면 정상 SSH 클라이언트 설정을 사용하여 연결할 수 있습니다. Amazon EC2 인스턴스에 연결하는 방법에 대한 자세한 내용은 Amazon EC2 사용 설명서의 EC2 사용 설명서를 참조하세요.

다음 명령을 실행하여 클러스터를 삭제합니다.

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

VPC에서 네트워크 리소스를 삭제하려면 CloudFormation 네트워킹 스택을 삭제하면 됩니다. 스택 이름은 “parallelclusternetworking-”으로 시작하며 “YYYYMMDDHHMMSS” 형식의 생성 시간을 포함합니다. list-stacks 명령을 사용하여 스택을 나열할 수 있습니다.

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

스택은 delete-stack 명령을 사용하여 삭제할 수 있습니다.

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

pcluster configure가 사용자를 위해 생성하는 VPC는 CloudFormation 네트워킹 스택에서 생성되지 않습니다. 콘솔에서 또는 AWS CLI를 사용하여 해당 VPC를 수동으로 삭제할 수 있습니다.

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

모범 사례

모범 사례: 마스터 인스턴스 유형 선택

프라이머리 노드는 어떤 작업도 실행하지 않지만, 프라이머리 노드의 기능과 크기 조정은 클러스터의 전체 성능에 매우 중요합니다.

프라이머리 노드에 사용할 인스턴스 유형을 선택할 때는 다음 항목을 평가해야 합니다.

- 클러스터 크기: 프라이머리 노드는 클러스터의 규모 조정 로직을 관리하고 새 노드를 스케줄러에 연결하는 역할을 합니다. 상당한 양의 노드로 구성된 클러스터를 스케일 업 또는 스케일 다운해야 하는 경우 프라이머리 노드에 추가 컴퓨팅 파워를 제공하는 것이 좋습니다.
- 공유 파일 시스템: 공유 파일 시스템을 사용하여 컴퓨팅 노드와 프라이머리 노드 간에 아티팩트를 공유하는 경우 프라이머리가 NFS 서버를 노출하는 노드라는 점을 고려하세요. 이러한 이유로 네트워크 대역폭과 워크플로를 처리하기에 충분한 전용 Amazon EBS 대역폭을 갖춘 인스턴스 유형을 선택하는 것이 좋습니다.

모범 사례: 네트워크 성능

네트워크 통신을 개선할 수 있는 모든 가능성을 포괄하는 세 가지 힌트가 있습니다.

- 배치 그룹: 클러스터 배치 그룹은 단일 가용 영역 내에 있는 인스턴스의 논리적 그룹입니다. 배치 그룹에 대한 자세한 내용은 Amazon EC2 사용 설명서의 배치 그룹을 참조하세요. `placement_group = your-placement-group-name` 항목과 함께 자체 배치 그룹을 사용하기 위해 클러스터를 구성하거나 AWS ParallelCluster 가 `placement_group = DYNAMIC`의 `"compute"` 전략을 사용하여 배치 그룹을 생성하게 할 수 있습니다. 자세한 내용은 다중 대기열 모드인 경우 `placement_group` 항목을 단일 대기열 모드인 경우 `placement_group` 항목을 참조하세요.
- 향상된 네트워킹: 향상된 네트워킹을 지원하는 인스턴스 유형을 선택하는 것이 좋습니다. 자세한 내용은 Amazon EC2 사용 설명서의 Linux에서 향상된 네트워킹을 참조하세요.
- Elastic Fabric Adapter: 높은 수준의 규모 조정 가능한 인스턴스 간 통신을 지원하려면 네트워크에 맞는 EFA 네트워크 인터페이스를 선택하는 것이 좋습니다. EFA의 사용자 지정 운영 체제(OS) 우회 하드웨어는 AWS 클라우드의 온디맨드 탄력성과 유연성을 통해 인스턴스 간 통신을 개선합니다. EFA를 사용하도록 단일 Slurm 클러스터 대기열을 구성하려면 `enable_efa = true`를 설정합니다. 에서 EFA를 사용하는 방법에 대한 자세한 내용은 Elastic Fabric Adapter 및 섹션을 AWS ParallelCluster참조하세요`enable_efa`. EFA에 대한 자세한 정보는 Linux 인스턴스용 Amazon EC2 사용자 설명서의 Elastic Fabric Adapter를 참조하세요.
- 인스턴스 대역폭: 대역폭은 인스턴스 규모에 따라 조정되므로 필요에 더 적합한 인스턴스 유형을 선택하는 것이 좋습니다. Amazon EC2 사용 설명서의 Amazon EBS 최적화 인스턴스 및 Amazon EBS 볼륨 유형을 참조하세요.

모범 사례: 예산 알림

AWS ParallelCluster 리소스 비용을 관리하려면 AWS Budgets 작업을 사용하여 선택한 AWS 리소스에 대한 예산 및 정의된 예산 임계값 알림을 생성하는 것이 좋습니다. 자세한 내용은AWS Budgets 사용 설명서의 예산 작업 구성을 참조하세요. 또한 Amazon CloudWatch를 사용하여 결제 경보를 생성할 수도 있습니다. 자세한 내용은 예상 AWS 요금을 모니터링하기 위한 결제 경보 생성을 참조하세요.

모범 사례: 클러스터를 새 AWS ParallelCluster 마이너 또는 패치 버전으로 이동

현재 각 AWS ParallelCluster 마이너 버전은 `pcluster` CLI와 함께 독립형입니다. 클러스터를 새 마이너 또는 패치 버전으로 옮기려면 새 버전의 CLI를 사용하여 클러스터를 다시 생성해야 합니다.

클러스터를 새 마이너 버전으로 옮기는 프로세스를 최적화하거나 다른 이유로 공유 스토리지 데이터를 저장하려면 다음 모범 사례를 사용하는 것이 좋습니다.

- Amazon EFS 및 FSx for Lustre와 같은 외부 볼륨에 개인 데이터를 저장합니다. 이렇게 하면 한 클러스터에서 다른 클러스터로 데이터를 쉽게 옮길 수 있습니다.
- AWS CLI 또는 AWS Management Console를 사용하여 아래 나열된 유형의 공유 스토리지 시스템을 생성합니다.
  - `[ebs]` 섹션
  - `[efs]` 섹션
  - `[fsx]` 섹션

  그것들을 새 클러스터 구성에 기존 파일 시스템으로 추가합니다. 이렇게 하면 클러스터를 삭제해도 파일이 보존되고 새 클러스터에 연결할 수 있습니다. 공유 스토리지 시스템은 일반적으로 클러스터에 연결되든 분리되었든 관계없이 요금이 부과됩니다.

  Amazon EFS 또는 Amazon FSx for Lustre 파일 시스템을 사용하는 것이 좋습니다. 이러한 파일 시스템은 동시에 여러 클러스터에 연결할 수 있고 이전 클러스터를 삭제하기 전에 새 클러스터에 연결할 수 있기 때문입니다. 자세한 내용은 Amazon EFS 사용 설명서의 Amazon EFS 파일 시스템 탑재 및 Amazon FSx for Lustre 사용 설명서의 FSx for Lustre 파일 시스템을 참조하세요.
- 사용자 지정 AMI 대신 사용자 지정 부트스트랩 작업을 사용하여 인스턴스를 사용자 지정합니다. 이렇게 하면 새 버전마다 새 사용자 지정 AMI를 새로 생성할 필요가 없으므로 생성 프로세스가 최적화됩니다.
- 권장 시퀀스

  1. 기존 파일 시스템 정의를 사용하도록 클러스터 구성을 업데이트하세요.

2. `pcluster` 버전을 확인하고 필요한 경우 업데이트하세요.
3. 새 클러스터를 만들고 테스트합니다.
   - 새 클러스터에서 데이터를 사용할 수 있는지 확인합니다.
   - 애플리케이션이 새 클러스터에서 작동하는지 확인합니다.
4. 새 클러스터를 완전히 테스트한 후 운영 중이며, 이전 클러스터를 사용하지 않을 것이 확실하다면 이전 클러스터를 삭제하세요.

CfnCluster에서 로 이동 AWS ParallelCluster

AWS ParallelCluster 는 CfnCluster의 향상된 버전입니다.

현재 CfnCluster를 사용하는 경우 AWS ParallelCluster 대신를 사용하고 이를 사용하여 새 클러스터를 생성하는 것이 좋습니다. CfnCluster를 계속 사용할 수는 있지만, CfnCluster는 더 이상 개발되지 않으며 새로운 특성이 추가되지 않습니다.

CfnCluster와 간의 주요 차이점 AWS ParallelCluster 은 다음 단원에서 설명합니다.

AWS ParallelCluster CLI는 다른 클러스터 세트를 관리합니다.

`cfncluster` CLI에서 생성된 클러스터는 `pcluster` CLI에서 관리할 수 없습니다. 다음 명령은 CfnCluster에서 생성된 클러스터에서 작동하지 않습니다.

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

CfnCluster로 생성한 클러스터를 관리하려면 `cfncluster` CLI를 사용해야 합니다.

기존 클러스터를 관리하기 위해 CfnCluster 패키지가 필요한 경우 Python 가상 환경에서 해당 패키지를 설치하고 사용하는 것이 좋습니다.

AWS ParallelCluster 및 CfnCluster는 서로 다른 IAM 사용자 지정 정책을 사용합니다.

이전에 CfnCluster 클러스터 생성에 사용된 사용자 지정 IAM 정책은 AWS ParallelCluster와 함께 사용할 수 없습니다. 에 대한 사용자 지정 정책이 필요한 경우 새 정책을 생성 AWS ParallelCluster해야 합니다. AWS ParallelCluster 안내서를 참조하세요.

AWS ParallelCluster 및 CfnCluster는 서로 다른 구성 파일을 사용합니다.

AWS ParallelCluster 구성 파일은 `~/.parallelcluster` 폴더에 있습니다. CfnCluster 구성 파일은 `~/.cfncluster` 폴더에 있습니다.

기존 CfnCluster 구성 파일을와 함께 사용하려면 다음 작업을 완료해야 AWS ParallelCluster합니다.

1. `~/.cfncluster/config`에서 `~/.parallelcluster/config`로 구성 파일을 이동합니다.
2. `extra_json` 구성 파라미터를 사용하는 경우 표시된 대로 변경합니다.

CfnCluster 설정:

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

AWS ParallelCluster 설정:

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

에서 AWS ParallelCluster신경절은 기본적으로 비활성화되어 있습니다.

에서 AWS ParallelCluster신경절은 기본적으로 비활성화되어 있습니다. Ganglia를 활성화하려면 다음 단계를 완료합니다.

1. `extra_json` 파라미터를 표시된 대로 설정합니다.

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

2. 또한 포트 80에 대한 연결을 허용하도록 헤드 보안 그룹을 변경합니다.

   퍼블릭 IP에서 포트 80에 대한 인바운드 연결을 허용하는 새 보안 그룹 규칙을 추가하여 `parallelcluster-<CLUSTER_NAME>-MasterSecurityGroup-<xxx>` 보안 그룹을 수정해야 합니다. 자세한 내용은 Amazon EC2 사용 설명서의 보안 그룹에 규칙 추가를 참조하세요.

지원되는 리전

AWS ParallelCluster 버전 2.x는 AWS 리전다음에서 사용할 수 있습니다.

리전 이름	리전
미국 동부(오하이오)	us-east-2

리전 이름	리전
미국 동부(버지니아 북부)	us-east-1
미국 서부(캘리포니아 북부)	us-west-1
미국 서부(오레곤)	us-west-2
아프리카(케이프타운)	af-south-1
아시아 태평양(홍콩)	ap-east-1
아시아 태평양(뭄바이)	ap-south-1
아시아 태평양(서울)	ap-northeast-2
아시아 태평양(싱가포르)	ap-southeast-1
아시아 태평양(시드니)	ap-southeast-2
아시아 태평양(도쿄)	ap-northeast-1
캐나다(중부)	ca-central-1
중국(베이징)	cn-north-1
중국(닝샤)	cn-northwest-1
유럽(프랑크푸르트)	eu-central-1
유럽(아일랜드)	eu-west-1
유럽(런던)	eu-west-2
유럽(밀라노)	eu-south-1
유럽(파리)	eu-west-3
유럽(스톡홀름)	eu-north-1
중동(바레인)	me-south-1

리전 이름	리전
남아메리카(상파울루)	sa-east-1
AWS GovCloud(미국 동부)	us-gov-east-1
AWS GovCloud(미국 서부)	us-gov-west-1

사용 AWS ParallelCluster

주제

- 네트워크 구성
- 사용자 지정 부트스트랩 작업
- Amazon S3 작업
- 스팟 인스턴스 작업
- AWS Identity and Access Management 의 역할 AWS ParallelCluster
- 에서 지원하는 스케줄러 AWS ParallelCluster
- AWS ParallelCluster 리소스 및 태그 지정
- Amazon CloudWatch 대시보드
- Amazon CloudWatch Logs와 통합
- Elastic Fabric Adapter
- Intel 셀렉트 솔루션
- Intel MPI 활성화
- Intel HPC 플랫폼 사양
- Arm 퍼포먼스 라이브러리
- Amazon DCV를 통해 헤드 노드에 연결합니다.
- pcluster update 사용하기
- AMI 패치 및 EC2 인스턴스 교체

네트워크 구성

AWS ParallelCluster 는 네트워킹에 Amazon Virtual Private Cloud(VPC)를 사용합니다. VPC는 클러스터를 배포하기 위한 유연하고 구성 가능한 네트워킹 플랫폼을 제공합니다.

VPC에는 리전의 도메인 이름이 올바른 및 `DNS Resolution = yes` `DNS Hostnames = yes` DHCP 옵션이 있어야 합니다. 기본 DHCP 옵션 세트는 이미 필수 AmazonProvidedDNS를 지정합니다. 도메인 이름 서버를 두 개 이상 지정하는 경우 Amazon VPC 사용 설명서의 DHCP 옵션 세트를 참조하세요.

AWS ParallelCluster 는 다음과 같은 상위 수준 구성을 지원합니다.

- 헤드 노드와 컴퓨팅 노드 모두를 위한 하나의 서브넷
- 퍼블릭 서브넷에 헤드 노드가 있고 프라이빗 서브넷에 컴퓨팅 노드가 있는 두 개의 서브넷 서브넷은 새 서브넷이거나 기존 서브넷일 수 있습니다.

이러한 모든 구성은 퍼블릭 IP 주소 지정 여부에 관계없이 작동할 수 AWS ParallelCluster 있습니다. 또한 모든 AWS 요청에 HTTP 프록시를 사용하도록 배포할 수 있습니다. 이러한 구성을 조합하면 많은 배포 시나리오가 가능해집니다. 예를 들어 인터넷을 통해 모든 액세스가 가능한 단일 퍼블릭 서브넷을 구성할 수 있습니다. 또는 모든 트래픽에 대해 AWS Direct Connect 및 HTTP 프록시를 사용하여 완전 프라이빗 네트워크를 구성할 수 있습니다.

이러한 시나리오의 일부를 보려면 다음 아키텍처 다이어그램을 참조하세요.

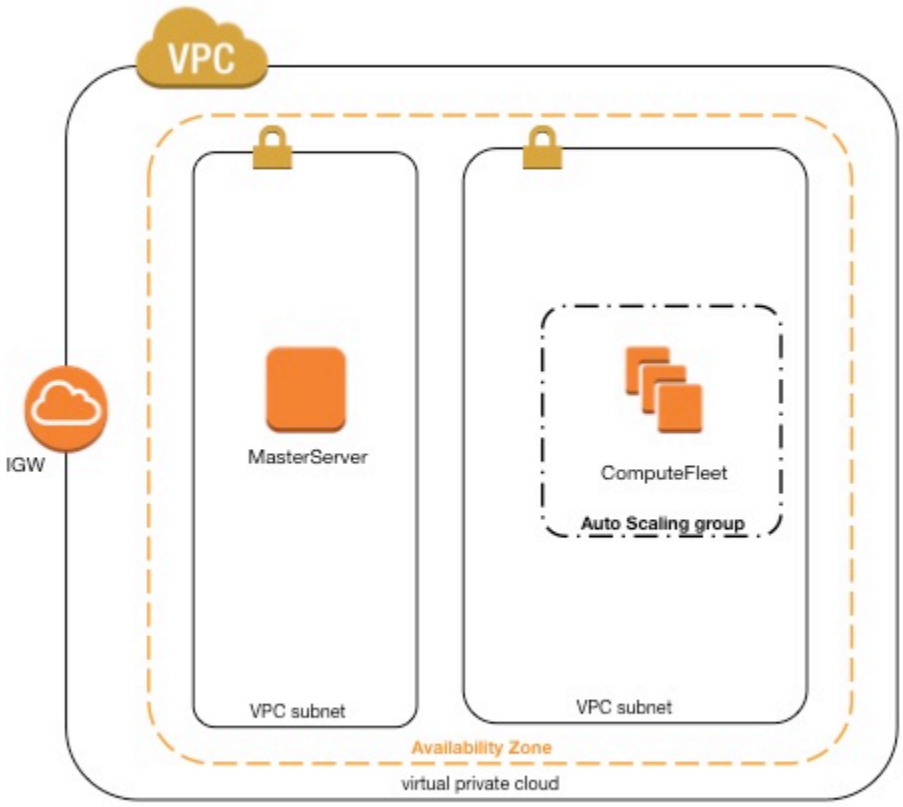
AWS ParallelCluster 단일 퍼블릭 서브넷의

이 아키텍처의 구성에는 다음 설정이 필요합니다.

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

인터넷 게이트웨이를 사용하려면 모든 인스턴스에 전역적으로 고유한 IP 주소가 있어야 하므로 use_public_ips 설정을 false로 설정할 수 없습니다. 자세한 내용은 Amazon VPC 사용 설명서의 인터넷 액세스 활성화를 참조하세요.

AWS ParallelCluster 서브넷 2개 사용



컴퓨팅 인스턴스에 대한 새 프라이빗 서브넷을 생성하기 위한 구성에는 다음 설정이 필요합니다.

모든 값은 예시로 제공될 뿐입니다.

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

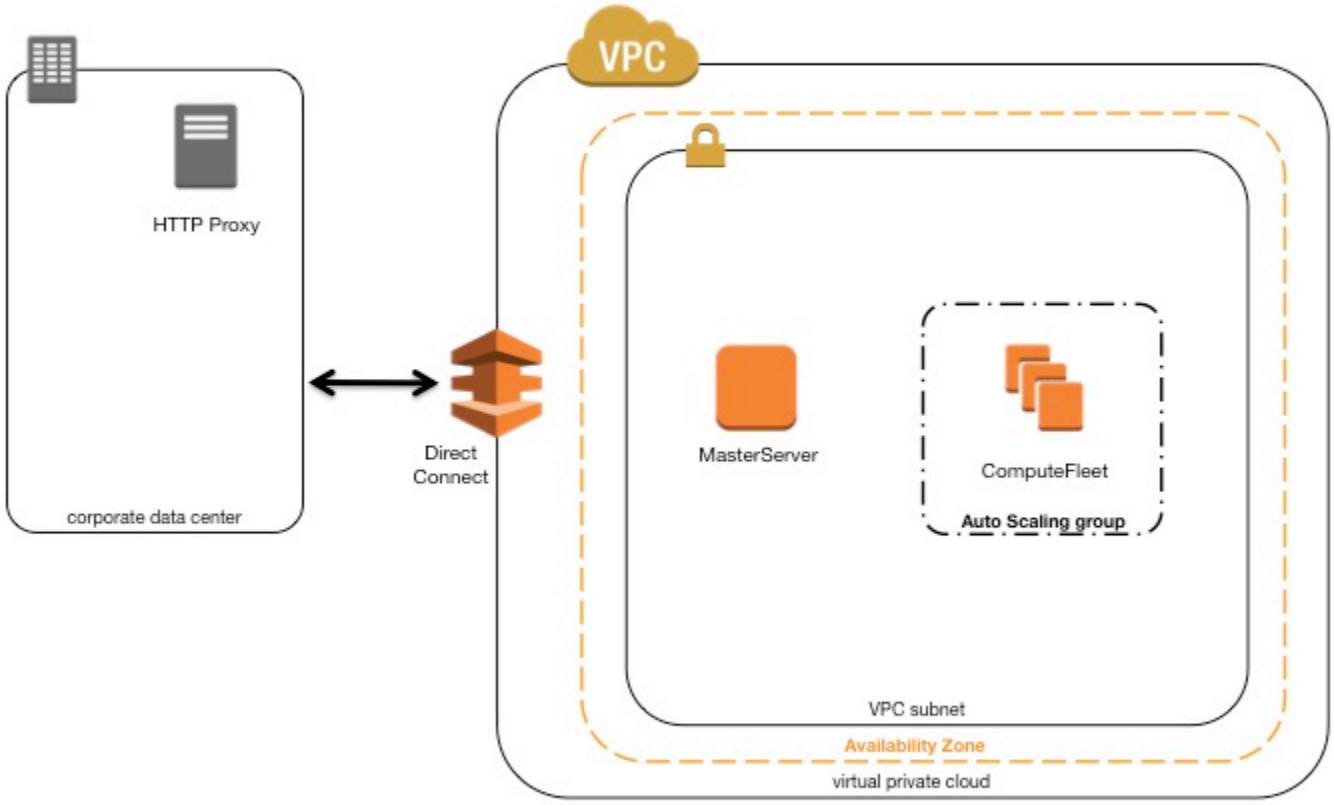
기존 프라이빗 네트워크를 사용하기 위한 구성에는 다음 설정이 필요합니다.

```
[vpc public-private-existing]
vpc_id = vpc-xxxxxx
```

```
master_subnet_id = subnet-<public>
compute_subnet_id = subnet-<private>
```

이러한 두 가지 구성에서 컴퓨팅 인스턴스에 대한 웹 액세스를 활성화하려면 NAT 게이트웨이 또는 내부 PROXY가 있어야 합니다.

AWS ParallelCluster 를 사용하여 연결된 단일 프라이빗 서브넷의 AWS Direct Connect



이 아키텍처의 구성에는 다음 설정이 필요합니다.

```
[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_ips를 false로 설정한 경우 모든 트래픽에 프록시를 사용하려면 VPC를 올바르게 설정해야 합니다. 헤드 노드와 컴퓨팅 노드 모두에 웹 액세스가 필요합니다.

AWS ParallelClusterawsbatch 스케줄러 사용

를 스케줄러 유형awsbatch으로 사용하면가 AWS Batch 관리형 컴퓨팅 환경을 AWS ParallelCluster 생성합니다. 환경은에서 시작되는 Amazon Elastic Container Service(Amazon ECS) 컨테이너 인스턴스를 AWS Batch 관리합니다compute_subnet. AWS Batch 가 올바르게 작동하려면 Amazon ECS 컨테이너 인스턴스가 Amazon ECS 서비스 엔드포인트와 통신하기 위해 외부 네트워크 액세스 권한이 필요합니다. 이 경우는 다음 시나리오로 전환됩니다.

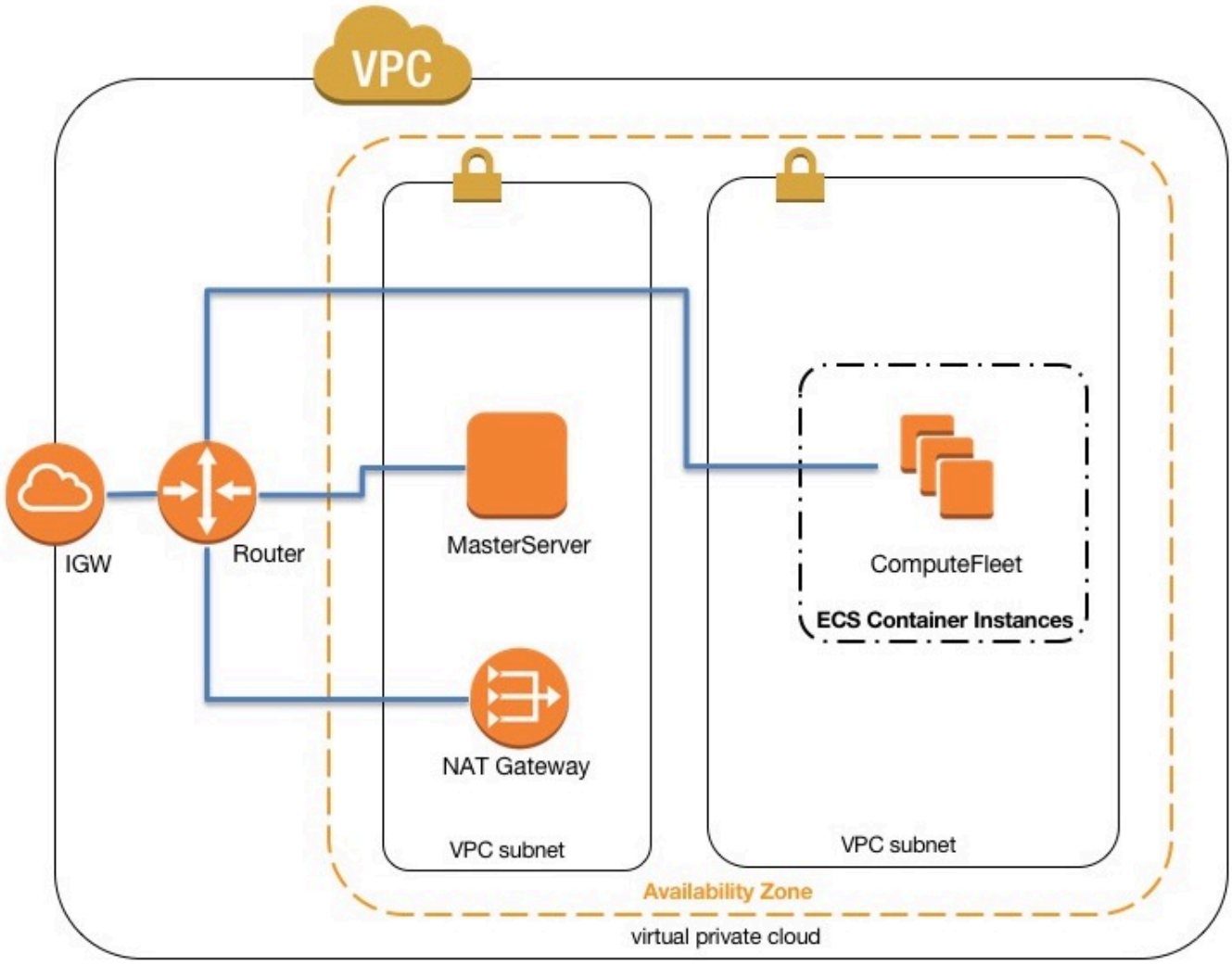
- compute_subnet은 NAT 게이트웨이를 사용하여 인터넷에 액세스합니다. 이 방법이 권장 방법입니다.
- compute_subnet에서 시작된 인스턴스는 퍼블릭 IP 주소를 가지고 있고 인터넷 게이트웨이를 통해 인터넷에 연결할 수 있습니다.

또한 다중 노드 병렬 작업에 관심이 있는 경우(AWS Batch 설명서에 따라):

AWS Batch 다중 노드 병렬 작업은 다중 노드 병렬 작업 컨테이너에 Amazon EC2 인스턴스와 동일한 네트워킹 속성을 제공하는 Amazon ECS awsvpc 네트워크 모드를 사용합니다. 각 다중 노드 병렬 작업 컨테이너는 고유의 탄력적 네트워크 인터페이스, 기본 프라이빗 IP 주소, 내부 DNS 호스트 이름을 가져옵니다. 네트워크 인터페이스는 호스트 컴퓨팅 리소스와 동일한 Amazon VPC 서브넷에서 생성됩니다. 컴퓨팅 리소스에 적용되는 모든 보안 그룹은 네트워크 인터페이스에도 적용됩니다.

Amazon ECS 작업 네트워킹을 사용할 경우 awsvpc 네트워크 모드는 Amazon EC2 시작 유형을 사용하는 작업에 대한 퍼블릭 IP 주소를 탄력적 네트워크 인터페이스에 제공하지 않습니다. 인터넷에 액세스하려면 Amazon EC2 시작 유형을 사용하는 작업이 NAT 게이트웨이를 사용하도록 구성된 프라이빗 서브넷에서 시작되어야 합니다.

클러스터가 다중 노드 병렬 작업을 실행할 수 있도록 하려면 NAT 게이트웨이를 구성해야 합니다.



자세한 정보는 다음의 주제를 참조하세요.

- AWS Batch 관리형 컴퓨팅 환경
- AWS Batch 다중 노드 병렬 작업
- Amazon ECS awsvpc 작업 모드를 이용한 태스크 네트워킹

사용자 지정 부트스트랩 작업

AWS ParallelCluster 는 클러스터가 생성될 때 기본 부트스트랩 작업 이전(설치 전) 또는 이후(설치 후)에 임의의 코드를 실행할 수 있습니다. 대부분의 경우 이 코드는 Amazon Simple Storage Service(S3)에 저장되며 HTTPS 연결을 통해 액세스됩니다. 이 코드는 루트로 실행되며 클러스터 OS에서 지원되

는 어떠한 스크립트 언어로든 작성될 수 있습니다. 코드는 일반적으로 Bash 또는 Python으로 작성됩니다.

사전 설치 작업은 NAT 구성, Amazon Elastic Block Store(Amazon EBS) 또는 스케줄러 구성과 같은 클러스터 배포 부트스트랩 전에 호출됩니다. 일부 사전 설치 작업에는 스토리지 수정, 사용자 및 패키지 추가가 포함될 수 있습니다.

설치 후 작업은 클러스터 부트스트랩 프로세스가 완료된 후에 호출됩니다. 설치 후 작업은 인스턴스가 완전히 구성되고 완료된 것으로 간주되기 전에 발생하는 마지막 작업입니다. 일부 사후 설치 작업에는 스케줄러 설정 변경, 스토리지 및 패키지 수정이 포함될 수 있습니다.

구성 중에 인수를 지정하여 인수를 스크립트에 전달할 수 있습니다. 이를 위해 인수는 따옴표로 묶여서 사전 설치 또는 사후 설치 작업에 전달됩니다.

사전 설치 또는 사후 설치 작업이 실패하면 인스턴스 부트스트랩도 실패합니다. 성공할 경우 종료 코드 0으로 표시됩니다. 다른 종료 코드는 인스턴스 부트스트랩이 실패했음을 나타냅니다.

실행 중인 헤드 노드와 컴퓨팅 노드를 구분할 수 있습니다. `/etc/parallelcluster/cfnconfig` 파일을 소싱하고 헤드 및 컴퓨팅 노드에 대해 가능한 값이 각각 “`MasterServer`” 및 “`ComputeFleet`”인 `cfn_node_type` 환경 변수를 평가합니다.

```
#!/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
```

구성

다음 구성 설정은 사전 설치 및 사후 설치 작업과 인수를 정의하는 데 사용됩니다.

```
# 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
```

인수

처음 두 개의 인수 `$0` 및 `$1`은 스크립트 이름과 URL에 예약됩니다.

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

예제

다음은 클러스터에 R 패키지를 설치하는 간단한 사후 설치 스크립트를 생성하는 단계입니다.

1. 스크립트를 생성합니다.

   ```
   #!/bin/bash

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

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

2. 올바른 권한을 사용하여 스크립트를 Amazon S3에 업로드합니다. 퍼블릭 읽기 권한이 적절하지 않은 경우, s3_read_resource 또는 s3_read_write_resource 파라미터를 사용하여 액세스 권한을 부여하세요. 자세한 내용은 Amazon S3 작업 단원을 참조하십시오.

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

> **Important**
>
> Windows에서 스크립트를 편집한 경우 스크립트를 Amazon S3에 업로드하기 전에 줄 끝을 CRLF에서 LF로 변경해야 합니다.

3. 새 설치 후 작업을 포함하도록 AWS ParallelCluster 구성을 업데이트합니다.

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

버킷에 public-read 권한이 없는 경우 s3을 URL 프로토콜로 사용합니다.

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

4. 클러스터를 시작합니다.

```
$ pcluster create mycluster
```

5. 출력을 확인합니다.

```
$ 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 작업

클러스터 리소스에 Amazon S3 버킷에 액세스할 수 있는 권한을 제공하려면 AWS ParallelCluster 구성의 s3_read_resource 및 s3_read_write_resource 파라미터에 버킷 ARNs을 지정합니다. 를 사용한 액세스 제어에 대한 자세한 내용은 섹션을 AWS ParallelCluster참조하세요AWS Identity and Access Management 의 역할 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/*
```

두 개의 파라미터는 모두 * 또는 유효한 Amazon S3 ARN을 허용합니다. Amazon S3 ARN 지정에 대한 자세한 내용은AWS 일반 참조의 Amazon S3 ARN 형식을 참조하세요.

예제

다음 예제는 Amazon S3 버킷 my_corporate_bucket에 있는 객체에 대한 읽기 액세스를 제공합니다.

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

다음 예제에서는 버킷에 대한 읽기 액세스 권한을 제공하지만 버킷에서 항목을 읽을 수는 없습니다.

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

이 마지막 예제에서는 버킷 및 버킷에 저장된 항목에 대한 읽기 액세스 권한을 제공합니다.

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

스팟 인스턴스 작업

AWS ParallelCluster 클러스터 구성이 cluster_type = 스팟으로 설정된 경우는 스팟 인스턴스를 사용합니다. 스팟 인스턴스는 온디맨드 인스턴스보다 비용 효율적이지만 중단될 수 있습니다. 중단이 미치는 영향은 사용되는 스케줄러에 따라 다릅니다. Amazon EC2가 스팟 인스턴스를 중지하거나 종료하

기 2분 전에 경고를 제공하는 스팟 인스턴스 중단 공지를 활용하는 데 도움이 될 수 있습니다. 자세한 내용은 Amazon EC2 사용 설명서의 스팟 인스턴스 중단을 참조하세요. 다음 섹션에서는 스팟 인스턴스가 중단될 수 있는 세 가지 시나리오를 설명합니다.

> **Note**
>
> 스팟 인스턴스를 사용하려면 계정에 `AWSServiceRoleForEC2Spot` 서비스 연결 역할이 있어야 합니다. 를 사용하여 계정에서이 역할을 생성하려면 다음 명령을 AWS CLI실행합니다.
>
> ```
> aws iam create-service-linked-role --aws-service-name spot.amazonaws.com
> ```
>
> 자세한 내용은 Amazon EC2 사용 설명서에서 스팟 인스턴스 요청을 위한 서비스 연결 역할을 참조하세요.

시나리오 1: 실행 중인 작업이 없는 스팟 인스턴스가 중단됨

이 중단이 발생하면 스케줄러 대기열에 추가 인스턴스가 필요한 보류 중인 작업이 있거나 활성 인스턴스 수가 `initial_queue_size` 설정보다 적은 경우 인스턴스를 교체하려고 AWS ParallelCluster 시도합니다. AWS ParallelCluster 에서 새 인스턴스를 프로비저닝할 수 없는 경우 새 인스턴스에 대한 요청이 주기적으로 반복됩니다.

시나리오 2: 단일 노드 작업을 실행하는 스팟 인스턴스가 중단됨

이 중단 동작은 사용 중인 스케줄러에 따라 다릅니다.

Slurm

상태 코드 `NODE_FAIL`으로 작업이 실패하고, 작업이 다시 대기열에 들어갑니다(작업 제출 시 `--no-requeue`이 지정되지 않은 한). 노드가 정적 노드인 경우 해당 노드가 교체됩니다. 노드가 동적 노드인 경우 노드가 종료되고 재설정됩니다. `--no-requeue` 파라미터를 포함한 `sbatch`에 관한 자세한 내용은 Slurm 설명서의 sbatch를 참조하세요.

> **Note**
>
> 이 동작은 AWS ParallelCluster 버전 2.9.0에서 변경되었습니다. 이전 버전에서는 상태 코드가 `NODE_FAIL`인 상태로 작업이 종료되었고 해당 노드는 스케줄러 대기열에서 제거되었습니다.

SGE

> **Note**
>
> 이는 AWS ParallelCluster 버전 2.11.4까지의 버전에만 적용됩니다. 버전 2.11.5부터는 AWS ParallelCluster 가 SGE 또는 Torque 스케줄러의 사용을 지원하지 않습니다.

작업이 종료됩니다. 작업에서 `qsub -r yes` 또는 `qalter -r yes` 중 하나를 사용하여 재실행 플래그를 활성화했거나 대기열에 `rerun` 구성이 `TRUE`로 설정된 경우 작업이 다시 예약됩니다. 컴퓨팅 인스턴스가 스케줄러 대기열에서 제거됩니다. 이 동작은 다음 SGE 구성 파라미터에서 발생합니다.

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

Torque

> **Note**
>
> 이는 AWS ParallelCluster 버전 2.11.4까지의 버전에만 적용됩니다. 버전 2.11.5부터는 AWS ParallelCluster 가 SGE 또는 Torque 스케줄러의 사용을 지원하지 않습니다.

작업이 시스템에서 제거되고 노드가 스케줄러에서 제거됩니다. 작업이 다시 실행되지 않습니다. 인스턴스가 중단되었을 때 인스턴스에서 여러 작업이 실행 중이면 노드 제거 중에 Torque가 시간 초과될 수 있습니다. 오류가 `sqswatcher` 로그 파일에 표시될 수 있습니다. 이는 규모 조정 로직에 영향을 주지 않으며 후속 재시도 시 적절하게 정리됩니다.

시나리오 3: 다중 노드 작업을 실행하는 스팟 인스턴스가 중단됨

이 중단 동작은 사용 중인 스케줄러에 따라 다릅니다.

Slurm

상태 코드 `NODE_FAIL`으로 작업이 실패하고, 작업이 다시 대기열에 들어갑니다(작업 제출 시 `--no-requeue`이 지정되지 않은 한). 노드가 정적 노드인 경우 해당 노드가 교체됩니다. 노드가 동적 노드인 경우 노드가 종료되고 재설정됩니다. 종료된 작업을 실행 중이었던 다른 노드는 구성된

scaledown_idletime 시간이 경과한 후에 다른 대기 중인 작업에 할당되거나 스케일 다운될 수 있습니다.

> **Note**
>
> 이 동작은 AWS ParallelCluster 버전 2.9.0에서 변경되었습니다. 이전 버전에서는 상태 코드가 NODE_FAIL인 상태로 작업이 종료되었고 해당 노드는 스케줄러 대기열에서 제거되었습니다. 종료된 작업을 실행 중이었던 다른 노드는 구성된 scaledown_idletime 시간이 경과한 후에 스케일 다운될 수 있습니다.

SGE

> **Note**
>
> 이는 AWS ParallelCluster 버전 2.11.4까지의 버전에만 적용됩니다. 버전 2.11.5부터는 AWS ParallelCluster 가 SGE 또는 Torque 스케줄러의 사용을 지원하지 않습니다.

작업이 종료되지 않고 나머지 노드에서 계속 실행됩니다. 컴퓨팅 노드는 스케줄러 대기열에서 제거되지만 호스트 목록에 분리되어 사용할 수 없는 노드로 표시됩니다.

이 경우 사용자는 작업을 삭제해야 합니다(qdel *<jobid>*). 노드는 AWS ParallelCluster에는 영향을 주지 않지만 호스트 목록(qhost)에 계속 표시됩니다. 목록에서 호스트를 제거하려면 인스턴스를 교체한 후 다음 명령을 실행하세요.

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

Torque

> **Note**
>
> 이는 AWS ParallelCluster 버전 2.11.4까지의 버전에만 적용됩니다. 버전 2.11.5부터는 AWS ParallelCluster 가 SGE 또는 Torque 스케줄러의 사용을 지원하지 않습니다.

작업이 시스템에서 제거되고 노드가 스케줄러에서 제거됩니다. 작업이 다시 실행되지 않습니다. 인스턴스가 중단되었을 때 인스턴스에서 여러 작업이 실행 중이면 노드 제거 중에 Torque가 시간 초

과될 수 있습니다. 오류가 `sqswatcher` 로그 파일에 표시될 수 있습니다. 이는 규모 조정 로직에 영향을 주지 않으며 후속 재시도 시 적절하게 정리됩니다.

스팟 인스턴스에 대한 자세한 내용은 Amazon EC2 사용 설명서의 스팟 인스턴스를 참조하세요.

AWS Identity and Access Management 의 역할 AWS ParallelCluster

AWS ParallelCluster 는 Amazon EC2에 (AWS Identity and Access Management IAM) 역할을 사용하여 인스턴스가 클러스터의 배포 및 작업을 위한 AWS 서비스에 액세스할 수 있도록 합니다. 기본적으로 Amazon EC2의 IAM 역할은 클러스터를 생성할 때 생성됩니다. 즉, 다음 섹션에서 설명하는 대로 클러스터를 생성하는 사용자에게 적절한 수준의 사용 권한이 있어야 합니다.

AWS ParallelCluster 는 여러 AWS 서비스를 사용하여 클러스터를 배포하고 운영합니다. 전체 목록은 AWS ParallelCluster에서 사용되는AWS 서비스 섹션을 참조하세요.

GitHub의AWS ParallelCluster 설명서에서 예제 정책의 변경 사항을 추적할 수 있습니다.

주제

- 클러스터 생성을 위한 기본 설정
- Amazon EC2의 기존 IAM 역할 사용
- AWS ParallelCluster 인스턴스 및 사용자 정책 예제

클러스터 생성을 위한 기본 설정

클러스터 생성에 대한 기본 설정을 사용하면 Amazon EC2의 기본 IAM 역할이 클러스터에 의해 생성됩니다. 클러스터를 생성하는 사용자는 클러스터를 시작하는 데 필요한 모든 리소스를 생성할 권한이 있어야 합니다. 여기에는 Amazon EC2를 위한 IAM 역할 생성도 포함됩니다. 일반적으로 사용자는 기본 설정을 사용할 때 AdministratorAccess 관리형 정책의 사용 권한이 있어야 합니다. 관리형 정책에 대한 자세한 정보는 IAM 사용 설명서에서 AWS 관리형 정책을 참조하세요.

Amazon EC2의 기존 IAM 역할 사용

클러스터를 생성할 때 기본 설정 대신 기존 `ec2_iam_role` 역할을 사용할 수 있지만, 먼저 클러스터를 시작하기 전에 IAM 정책과 역할을 정의해야 합니다. 일반적으로 Amazon EC2에 대한 기존 IAM

역할을 선택하여 사용자가 클러스터를 시작할 때 부여되는 사용 권한을 최소화합니다. 에는 AWS ParallelCluster 및 해당 기능에 필요한 최소 권한이 AWS ParallelCluster 인스턴스 및 사용자 정책 예제 포함됩니다. 정책과 역할을 IAM에서 개별 정책으로 작성한 다음, 역할과 정책을 적절한 리소스에 연결해야 합니다. 일부 역할 정책은 크기가 커져 할당량 오류가 발생할 수 있습니다. 자세한 내용은 IAM 정책 크기 문제 해결 단원을 참조하십시오. 정책에서 *<REGION>*, *<AWS ACCOUNT ID>* 및 유사한 문자열을 적절한 값으로 바꿉니다.

클러스터 노드의 기본 설정에 추가 정책을 추가하려는 경우, `ec2_iam_role` 설정을 사용하는 대신 `additional_iam_policies` 설정과 함께 추가 사용자 지정 IAM 정책을 전달하는 것이 좋습니다.

AWS ParallelCluster 인스턴스 및 사용자 정책 예제

다음 예제 정책에는 리소스의 Amazon 리소스 이름(ARN)이 포함되어 있습니다. AWS GovCloud (US) 또는 AWS 중국 파티션에서 작업하는 경우 ARNs을 변경해야 합니다. 특히 파티션의 경우 "arn:aws"에서 "arn:aws-us-gov"로, AWS 중국 AWS GovCloud (US) 파티션의 경우 "arn:aws-cn"으로 변경해야 합니다. 자세한 내용은 AWS GovCloud (US) 사용 설명서의 리전의 AWS GovCloud (US) Amazon 리소스 이름(ARNs) 및 중국에서 AWS 서비스 시작하기의 중국 내 서비스에 대한 ARNs을 참조하세요. AWS

이러한 정책에는 현재에 필요한 최소 권한, 기능 및 리소스 AWS ParallelCluster가 포함됩니다. 일부 역할 정책은 크기가 커져 할당량 오류가 발생할 수 있습니다. 자세한 내용은 IAM 정책 크기 문제 해결 단원을 참조하십시오.

주제

- SGE, Slurm, 또는 Torque를 사용하는 ParallelClusterInstancePolicy
- awsbatch을 사용하는 ParallelClusterInstancePolicy
- Slurm을 사용하는 ParallelClusterUserPolicy
- SGE 또는 Torque를 사용하는 ParallelClusterUserPolicy
- awsbatch을 사용하는 ParallelClusterUserPolicy
- SGE, Slurm, 또는 Torque를 사용하는 ParallelClusterLambdaPolicy
- awsbatch을 사용하는 ParallelClusterLambdaPolicy
- 사용자를 위한 ParallelClusterUserPolicy

SGE, Slurm, 또는 Torque를 사용하는 **ParallelClusterInstancePolicy**

> **Note**
>
> 버전 2.11.5부터 SGE 또는 Torque 스케줄러 사용을 지원하지 AWS ParallelCluster 않습니다. 2.11.4 이하의 버전에서는 계속 사용할 수 있지만 AWS 서비스 및 AWS 지원 팀의 향후 업데이트 또는 문제 해결 지원을 받을 수 없습니다.

주제

- Slurm을 사용하는 ParallelClusterInstancePolicy
- SGE 또는 Torque를 사용하는 ParallelClusterInstancePolicy

Slurm을 사용하는 **ParallelClusterInstancePolicy**

다음 예제에서는 Slurm를 스케줄러로 사용하여 `ParallelClusterInstancePolicy`를 설정합니다.

```
{
    "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 또는 Torque를 사용하는 **ParallelClusterInstancePolicy**

다음 예제에서는 SGE 또는 Torque를 스케줄러로 사용하여 `ParallelClusterInstancePolicy`를 설정합니다.

> **Note**
>
> 이 정책은 AWS ParallelCluster 버전 2.11.4까지의 버전에만 적용됩니다. 버전 2.11.5부터는 AWS ParallelCluster 가 SGE 또는 Torque 스케줄러의 사용을 지원하지 않습니다.

```
{
    "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`을 사용하는 `ParallelClusterInstancePolicy`

다음 예제에서는 `awsbatch`를 스케줄러로 사용하여 `ParallelClusterInstancePolicy`를 설정합니다. 중첩 스택에 `BatchUserRole` 정의된 AWS Batch AWS CloudFormation 에 할당된 것과 동일한 정책을 포함해야 합니다. `BatchUserRole` ARN은 스택 출력으로 제공됩니다. 이 예제에서 "*<RESOURCES S3 BUCKET>*"은 `cluster_resource_bucket` 설정의 값

이고, cluster_resource_bucket이 지정되지 않은 경우 "*<RESOURCES S3 BUCKET>*"은 "parallelcluster-*"입니다. 다음 예는 필요한 권한의 개요입니다.

```
{
    "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"
        }
    ]
}
```

Slurm을 사용하는 **ParallelClusterUserPolicy**

다음 예제에서는 Slurm를 스케줄러로 사용하여 ParallelClusterUserPolicy를 설정합니다. 이 예제에서 "*<RESOURCES S3 BUCKET>*"은 cluster_resource_bucket 설정의 값이고, cluster_resource_bucket이 지정되지 않은 경우 "*<RESOURCES S3 BUCKET>*"은 "parallelcluster-*"입니다.

> **Note**
>
> 사용자 지정 역할 ec2_iam_role = *<role_name>*을 사용하는 경우 해당 역할의 이름을 포함하도록 IAM 리소스를 다음과 같이 변경해야 합니다.
>
> "Resource": "arn:aws:iam::*<AWS ACCOUNT ID>*:role/parallelcluster-*"
>
> To:

```
    "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 또는 Torque를 사용하는 **ParallelClusterUserPolicy**

> **Note**
>
> 이 섹션은 AWS ParallelCluster 버전 2.11.4까지의 버전에만 적용됩니다. 버전 2.11.5부터 AWS ParallelCluster 는 SGE 또는 Torque 스케줄러 사용을 지원하지 않습니다.

다음 예제에서는 SGE 또는 Torque를 스케줄러로 사용하여 ParallelClusterUserPolicy를 설정합니다. 이 예제에서 "*<RESOURCES S3 BUCKET>*"은 cluster_resource_bucket 설정의 값이고, cluster_resource_bucket이 지정되지 않은 경우 "*<RESOURCES S3 BUCKET>*"은 "parallelcluster-*"입니다.

> **Note**
>
> 사용자 지정 역할 ec2_iam_role = *<role_name>*을 사용하는 경우 해당 역할의 이름을 포함하도록 IAM 리소스를 다음과 같이 변경해야 합니다.
>
> "Resource": "arn:aws:iam::*<AWS ACCOUNT ID>*:role/parallelcluster-*"
>
> To:
>
> "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": "*"
        }
    ]
}
```

awsbatch을 사용하는 ParallelClusterUserPolicy

다음 예제에서는 awsbatch를 스케줄러로 사용하여 ParallelClusterUserPolicy를 설정합니다. 이 예제에서 "*<RESOURCES S3 BUCKET>*"은 cluster_resource_bucket 설정의 값이고, cluster_resource_bucket이 지정되지 않은 경우 "*<RESOURCES S3 BUCKET>*"은 "parallelcluster-*"입니다.

```
{
    "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, 또는 Torque를 사용하는 **ParallelClusterLambdaPolicy**

다음 예제에서는 SGE, Slurm 또는 Torque를 스케줄러로 사용하여 `ParallelClusterLambdaPolicy`를 설정합니다.

> **Note**
>
> 버전 2.11.5부터 SGE 또는 Torque 스케줄러 사용을 지원하지 AWS ParallelCluster 않습니다.

```
{
  "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"
    }
  ]
}
```

awsbatch을 사용하는 ParallelClusterLambdaPolicy

다음 예제에서는 awsbatch를 스케줄러로 사용하여 ParallelClusterLambdaPolicy를 설정합니다.

```
{
  "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"
    }
  ]
}
```

사용자를 위한 **ParallelClusterUserPolicy**

다음 예제는 클러스터를 만들거나 업데이트할 필요가 없는 사용자를 위한 ParallelClusterUserPolicy를 설정합니다. 다음 명령이 지원됩니다.

- 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": "*"
        }
    ]
}
```

에서 지원하는 스케줄러 AWS ParallelCluster

AWS ParallelCluster 는 `scheduler` 설정을 사용하여 설정된 여러 스케줄러를 지원합니다.

> **Note**
>
> 버전 2.11.5부터 SGE 또는 Torque 스케줄러 사용을 지원하지 AWS ParallelCluster 않습니다. 2.11.4 이하의 버전에서는 계속 사용할 수 있지만 AWS 서비스 및 AWS 지원 팀의 향후 업데이트 또는 문제 해결 지원을 받을 수 없습니다.

주제

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

- AWS Batch (awsbatch)

Son of Grid Engine (`sge`)

> **Note**
>
> 버전 2.11.5부터 SGE 또는 Torque 스케줄러 사용을 지원하지 AWS ParallelCluster 않습니다. 2.11.4 이하의 버전에서는 계속 사용할 수 있지만 AWS 서비스 및 AWS 지원 팀의 향후 업데이트 또는 문제 해결 지원을 받을 수 없습니다.

AWS ParallelCluster 버전 2.11.4 이하에서는 Son of Grid Engine 8.1.9를 사용합니다.

Slurm Workload Manager (`slurm`)

AWS ParallelCluster 버전 2.11.9는 Slurm 20.11.9을 사용합니다. Slurm에 대한 자세한 내용은 https://slurm.schedmd.com/ 섹션을 참조하세요. 다운로드는 https://github.com/SchedMD/slurm/tags를 참조하세요. 소스 코드는 https://github.com/SchedMD/slurm 섹션을 참조하세요.

> **Important**
>
> AWS ParallelCluster 는 기본적으로 제공되는 Slurm 구성 파라미터로 테스트됩니다. 이러한 Slurm 구성 파라미터를 변경하는 데 따른 위험은 사용자 본인이 감수해야 합니다. 이 서비스는 최선의 작업을 기반으로 지원됩니다.

AWS ParallelCluster 버전(들)	지원되는 Slurm 버전
2.11.7, 2.11.8, 2.11.9	20.11.9
2.11.4에서 2.11.6까지	20.11.8
2.11.0에서 2.11.3까지	20.11.7
2.10.4	20.02.7
2.9.0에서 2.10.3까지	20.02.4

AWS ParallelCluster 버전(들)	지원되는 Slurm 버전
2.6에서 2.8.1까지	19.05.5
2.5.0, 2.5.1	19.05.3-2
2.3.1에서 2.4.1까지	18.08.6-2
2.3.1 이전	16.05.3-1

다중 대기열 모드

AWS ParallelCluster 버전 2.9.0에는 여러 대기열 모드가 도입되었습니다. `scheduler`를 `slurm`로 설정하고 `queue_settings` 설정을 정의하면 다중 대기열 모드가 지원됩니다. 이 모드를 사용하면 컴퓨팅 노드에서 다양한 인스턴스 유형이 공존할 수 있습니다. 다양한 인스턴스 유형을 포함하는 컴퓨팅 리소스는 필요에 따라 스케일 업 또는 스케일 다운할 수 있습니다. 대기열 모드에서는 최대 5개의 대기열이 지원되며 각 `[queue]` 섹션은 최대 3개의 `[compute_resource]` 섹션을 참조할 수 있습니다. 각 `[queue]` 섹션은 Slurm Workload Manager의 파티션입니다. 자세한 내용은 다중 대기열 모드를 위한 Slurm 가이드 및 다중 대기열 모드 자습서 섹션을 참조하세요.

대기열의 각 `[compute_resource]` 섹션은 서로 다른 인스턴스 유형을 가져야 하며, 각 `[compute_resource]`는 다시 정적 노드와 동적 노드로 구분됩니다. 각 `[compute_resource]`의 정적 노드는 1부터 `min_count`의 값까지 번호가 매겨집니다. 각 `[compute_resource]`의 동적 노드는 1부터 (`max_count`-`min_count`)까지 번호가 매겨집니다. 예를 들어, `min_count`가 2이고 `max_count`가 10인 경우 `[compute_resource]`의 동적 노드는 1에서 8까지 번호가 매겨집니다. 언제든지 `[compute_resource]`에는 0과 동적 노드의 최대 수 사이의 번호가 있을 수 있습니다.

컴퓨팅 플릿으로 시작되는 인스턴스는 동적으로 할당됩니다. 이를 관리하는 데 도움이 되도록 각 노드에 대해 호스트 이름이 생성됩니다. 호스트 이름 형식은 다음과 같습니다.

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

- `$QUEUE`은 대기열의 이름입니다. 예를 들어 섹션이 시작되면 `[queue` *`queue-name`*`]` "`$QUEUE`"는 "*`queue-name`*"입니다.
- `$STATDYN`은 정적 노드에는 `st` 또는 동적 노드에는 `dy`입니다.
- `$INSTANCE_TYPE`은 `instance_type` 설정에 있는 `[compute_resource]`의 인스턴스 유형입니다.

- $NODENUM은 노드의 번호입니다. $NODENUM은 정적 노드의 경우 1과 min_count의 값 사이, 동적 노드의 경우 1과 (max_count-min_count) 사이입니다.

호스트 이름과 FQDN(정규화된 도메인 이름)은 모두 Amazon Route 53 호스팅 영역을 사용하여 생성됩니다. FQDN은 $HOSTNAME.$CLUSTERNAME.pcluster입니다. 여기서 $CLUSTERNAME는 클러스터에 사용되는 [cluster] 섹션의 이름입니다.

구성을 대기열 모드로 변환하려면 pcluster-config convert 명령을 사용합니다. 이름이 [queue compute]인 단일 [queue] 섹션으로 업데이트된 구성을 작성합니다. 해당 대기열에는 이름이 [compute_resource default]인 단일 [compute_resource] 섹션이이 있습니다. [queue compute] 및 [compute_resource default]는 지정된 [cluster] 섹션에서 마이그레이션된 설정이 있습니다.

다중 대기열 모드를 위한 Slurm 가이드

AWS ParallelCluster 버전 2.9.0에는 여러 대기열 모드와 Slurm Workload Manager (Slurm)에 대한 새로운 조정 아키텍처가 도입되었습니다.

다음 섹션은 새로 도입된 확장 아키텍처가 적용된 Slurm 클러스터 사용에 대한 일반적인 개요를 제공합니다.

개요

새로운 규모 조정 아키텍처는 Slurm의 클라우드 스케줄링 가이드 및 절전 플러그인을 기반으로 합니다. 절전 플러그인에 대한 자세한 내용은 Slurm 절전 가이드를 참조하세요. 새로운 아키텍처에서 클러스터에 사용할 수 있는 리소스는 일반적으로 Slurm 구성에서 클라우드 노드로 미리 정의됩니다.

클라우드 노드 수명 주기

클라우드 노드는 수명 주기 내내 POWER_SAVING, POWER_UP(pow_up), ALLOCATED(alloc), POWER_DOWN(pow_dn) 상태 중 여러 개 또는 전부로 전환됩니다. 경우에 따라 클라우드 노드가 OFFLINE 상태로 전환될 수 있습니다. 다음 목록은 클라우드 노드 수명 주기에서 이러한 상태의 여러 측면을 자세히 설명합니다.

- ~ 상태의 노드는 sinfo에서 POWER_SAVING 접미사(예: idle~)와 함께 표시됩니다. 이 상태에서는 노드를 지원하는 EC2 인스턴스가 없습니다. 하지만 Slurm은 여전히 노드에 작업을 할당할 수 있습니다.
- POWER_UP 상태로 전환되는 노드는 sinfo에서 # 접미사(예: idle#)와 함께 표시됩니다.

- Slurm이 POWER_SAVING 상태의 노드에 작업을 할당하면 노드가 자동으로 POWER_UP 상태로 전환됩니다. 그렇지 않으면 `scontrol update nodename=`*`nodename`* `state=power_up` 명령을 사용하여 노드를 수동으로 POWER_UP 상태로 전환할 수 있습니다. 이 단계에서는 ResumeProgram가 호출되고 EC2 인스턴스가 시작되고 POWER_UP 노드를 지원하도록 구성됩니다.
- 현재 사용할 수 있는 노드는 `sinfo`에 접미사(예: `idle`)가 없는 상태로 표시됩니다. 노드가 설정되고 클러스터에 가입되면 작업을 실행할 수 있게 됩니다. 이 단계에서는 노드가 적절하게 구성되고 사용할 준비가 됩니다. 일반적으로 EC2의 인스턴스 수는 사용 가능한 노드의 수와 같게 하는 것이 좋습니다. 대부분의 경우 정적 노드는 클러스터가 생성된 후에 항상 사용할 수 있습니다.
- POWER_DOWN 상태로 전환되는 노드는 `sinfo`에서 `%` 접미사(예: `idle%`)와 함께 표시됩니다. 동적 노드는 scaledown_idletime 이후에 자동으로 POWER_DOWN 상태가 됩니다. 반면 정적 노드는 대부분의 경우 전원이 꺼지지 않습니다. 하지만 `scontrol update nodename=`*`nodename`* `state=powering_down` 명령을 사용하여 노드를 수동으로 POWER_DOWN 상태로 전환할 수 있습니다. 이 상태에서는 노드와 연결된 인스턴스가 종료되고 노드는 다시 해당 POWER_SAVING 상태로 재설정되며 scaledown_idletime 이후에 사용할 수 있습니다. `scaledown-idletime` 설정이 Slurm 구성에 SuspendTimeout 설정으로 저장됩니다.
- 오프라인 상태인 노드는 `sinfo`에서 `*` 접미사(예: `down*`)와 함께 나타납니다. Slurm 컨트롤러가 노드에 연결할 수 없거나 정적 노드가 비활성화되고 지원 인스턴스가 종료되면 노드는 오프라인 상태가 됩니다.

이제 다음 `sinfo` 예제에 표시된 노드 상태를 고려해 보세요.

```
$ 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]` 및 `efa-st-c5n18xlarge-1` 노드에는 이미 백업 인스턴스가 설정되어 있고 사용할 수 있습니다. `ondemand-dy-c52xlarge-[1-2]` 노드는 현재 POWER_UP 상태이며 몇 분 이내에 사용할 수 있습니다. `gpu-dy-g38xlarge-1` 노드는 현재 POWER_DOWN 상태이고 scaledown_idletime(기본값 120초) 이후 POWER_SAVING 상태로 전환됩니다.

다른 모든 노드는 이를 지원하는 EC2 인스턴스가 없는 POWER_SAVING 상태입니다.

사용 가능한 노드로 작업하기

사용 가능한 노드는 EC2 인스턴스에서 지원합니다. 기본적으로 노드 이름을 사용하여 인스턴스에 직접 SSH로 연결할 수 있습니다(예:`ssh efa-st-c5n18xlarge-1`). `scontrol show nodes` *`nodename`* 명령과 NodeAddr 필드 확인을 사용하여 인스턴스의 사설 IP 주소를 검색할 수 있습니다. 사용할 수 없는 노드의 경우 NodeAddr 필드는 실행 중인 EC2 인스턴스를 가리키면 안 됩니다. 그보다는 노드 이름과 동일해야 합니다.

작업 상태 및 제출

대부분의 경우 제출된 작업은 시스템의 노드에 즉시 할당되거나, 모든 노드가 할당되면 보류 상태로 전환됩니다.

작업에 할당된 노드에 특정 POWER_SAVING 상태의 노드가 포함된 경우 작업은 CF 또는 CONFIGURING 상태로 시작됩니다. 이때 작업은 POWER_SAVING 상태의 노드가 해당 POWER_UP 상태로 전환되어 사용 가능한 상태가 될 때까지 기다립니다.

작업에 할당된 모든 노드를 사용할 수 있게 되면 작업은 RUNNING(R) 상태가 됩니다.

기본적으로 모든 작업은 기본 대기열(Slurm에서 파티션이라고 함)에 제출됩니다. 이는 대기열 이름 뒤에 * 접미사가 붙는 것으로 표시됩니다. `-p` 작업 제출 옵션을 사용하여 대기열을 선택할 수 있습니다.

모든 노드는 작업 제출 명령에 사용할 수 있는 다음 기능으로 구성됩니다.

- 인스턴스 유형(예: `c5.xlarge`)
- 노드 유형(`dynamic` 또는 `static`)

`scontrol show nodes` *`nodename`* 명령을 사용하고 AvailableFeatures 목록을 확인하여 특정 노드에 사용할 수 있는 모든 특성을 볼 수 있습니다.

또 다른 고려 사항은 작업입니다. `sinfo` 명령을 실행하여 확인할 수 있는 클러스터의 초기 상태를 먼저 고려해 보세요.

```
$ 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]
```

spot이 기본 대기열이라는 점에 유의하세요. * 접미사로 표시됩니다.

작업을 하나의 정적 노드에 기본 대기열(spot)에 제출합니다.

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

EFA 대기열에 있는 동적 노드 하나에 작업을 제출합니다.

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

ondemand 대기열에 있는 8개의 c5.2xlarge 노드와 2개의 t2.xlarge 노드에 작업을 제출하세요.

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

gpu 대기열에 있는 GPU 노드 하나에 작업을 제출하세요.

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

이제 squeue 명령을 사용하여 작업 상태를 살펴보세요.

```
$ 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
```

작업 7 및 8(spot 및 efa 대기열에 있음)은 이미 실행 중입니다(R). 작업 12와 13은 여전히 구성 중이며(CF), 아마도 인스턴스를 사용할 수 있을 때까지 기다리고 있을 것입니다.

```
# 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
```

노드 상태 및 특성

대부분의 경우 노드 상태는이 주제의 앞부분에서 설명한 클라우드 노드 수명 주기의 특정 프로세스에 AWS ParallelCluster 따라에서 완전히 관리됩니다.

그러나는 DOWN 및 DRAINED 상태의 비정상 노드와 비정상 지원 인스턴스가 있는 노드 AWS ParallelCluster 도 교체하거나 종료합니다. 자세한 내용은 clustermgtd 단원을 참조하십시오.

파티션 상태

AWS ParallelCluster 는 다음과 같은 파티션 상태를 지원합니다. Slurm 파티션은 AWS ParallelCluster 의 대기열입니다.

- UP: 파티션이 활성 상태임을 나타냅니다. 이것은 파티션의 기본 상태입니다. 이 상태에서는 파티션의 모든 노드가 활성화되어 사용할 수 있습니다.
- INACTIVE: 파티션이 비활성 상태임을 나타냅니다. 이 상태에서는 비활성 파티션의 노드를 지원하는 모든 인스턴스가 종료됩니다. 비활성 파티션의 노드에 대해서는 새 인스턴스가 시작되지 않습니다.

pcluster 시작 및 중지

pcluster stop이 실행되면 모든 파티션이 INACTIVE 상태로 전환되고 AWS ParallelCluster 프로세스는 파티션을 INACTIVE 상태로 유지합니다.

pcluster start를 실행하면 모든 파티션이 초기에 UP 상태로 전환됩니다. 그러나 AWS ParallelCluster 프로세스는 파티션을 UP 상태로 유지하지 않습니다. 파티션 상태를 수동으로 변경해야

합니다. 몇 분 후 모든 고정 노드를 사용할 수 있습니다. 참고로 파티션을 UP으로 설정해도 동적 용량은 증가하지 않습니다. initial_count이 max_count보다 크면 initial_count은 파티션 상태가 UP 상태로 변경될 때 만족하지 않을 수 있습니다.

pcluster start 및 pcluster stop가 실행되고 있는 경우 pcluster status 명령을 실행하고 ComputeFleetStatus를 확인하여 클러스터의 상태를 확인할 수 있습니다. 다음과 같은 잠재적인 상태가 있습니다.

- STOP_REQUESTED: pcluster stop 요청이 클러스터로 전송됩니다.
- STOPPING: pcluster 프로세스가 현재 클러스터를 중지하는 중입니다.
- STOPPED: pcluster 프로세스가 중지 프로세스를 완료했고, 모든 파티션이 INACTIVE 상태에 있으며, 모든 컴퓨팅 인스턴스가 종료되었습니다.
- START_REQUESTED: pcluster start 요청이 클러스터로 전송됩니다.
- STARTING: pcluster 프로세스가 현재 클러스터를 시작하는 중입니다.
- RUNNING: pcluster 프로세스가 시작 프로세스를 마쳤고, 모든 파티션이 UP 상태에 있으며, 몇 분 후에 정적 노드를 사용할 수 있습니다.

대기열 수동 제어

클러스터의 노드 또는 대기열(Slurm 파티션이라고 함)을 수동으로 제어해야 하는 경우도 있습니다. 다음과 같은 일반적인 절차를 통해 클러스터의 노드를 관리할 수 있습니다.

- POWER_SAVING 상태의 동적 노드 전원 켜기: scontrol update nodename=*nodename* state=power_up 명령을 실행하거나 특정 수의 노드를 요청하는 플레이스홀더 sleep 1 작업을 제출하고 Slurm에 의존하여 필요한 수의 노드에 전원을 공급하세요.
- 이전의 동적 노드 전원 끄기scaledown_idletime: scontrol update nodename=*nodename* state=down 명령을 DOWN 사용하여 동적 노드를 로 설정합니다.는 중단된 동적 노드를 AWS ParallelCluster 자동으로 종료하고 재설정합니다. 일반적으로 scontrol update nodename=*nodename* state=power_down 명령을 사용하여 노드를 POWER_DOWN으로 직접 설정하지 않는 것이 좋습니다. AWS ParallelCluster 가 전원 차단 프로세스를 자동으로 처리하기 때문입니다. 수동 개입이 필요하지 않습니다. 따라서 가능하면 노드를 DOWN로 설정하는 것이 좋습니다.
- 대기열(파티션)을 비활성화하거나 특정 파티션의 모든 정적 노드를 중지하세요. scontrol update partition=*queue name* state=inactive 명령을 사용하여 특정 대기열을 INACTIVE로 설정하세요. 이렇게 하면 파티션의 노드를 지원하는 모든 인스턴스가 종료됩니다.
- 대기열(파티션) 활성화: scontrol update partition=*queue name* state=up 명령으로 특정 대기열을 INACTIVE로 설정합니다.

동작 및 조정 규모 조정

다음은 일반적인 규모 조정 워크플로의 예입니다.

- 스케줄러는 두 개의 노드가 필요한 작업을 수신합니다.
- 스케줄러는 두 노드를 POWER_UP 상태로 전환하고 노드 이름(예: queue1-dy-c5xlarge-[1-2])을 사용하여 ResumeProgram을 호출합니다.
- ResumeProgram은 EC2 인스턴스 2개를 시작하고 queue1-dy-c5xlarge-[1-2]의 프라이빗 IP 주소와 호스트 이름을 할당합니다. 이후 ResumeTimeout을 기다린 후[기본 60분(1시간)] 노드를 재설정합니다.
- 인스턴스가 구성되어 클러스터에 들어갑니다. 인스턴스에서 작업이 실행되기 시작합니다.
- 작업이 완료되었습니다.
- 구성된 SuspendTime이 경과한 후(scaledown_idletime로 설정), 스케줄러는 인스턴스를 POWER_SAVING 상태로 전환합니다. 스케줄러는 queue1-dy-c5xlarge-[1-2]을 POWER_DOWN 상태를 설정하고 노드 이름을 사용하여 SuspendProgram을 호출합니다.
- 두 노드에 대해 SuspendProgram이 호출됩니다. 예를 들어 노드는 SuspendTimeout[기본 120초(2분)]을 위해 idle%로 남아 있음으로써 POWER_DOWN 상태로 유지됩니다. 노드의 전원이 꺼지고 있음을 clustermgtd가 감지하면 지원 인스턴스가 종료됩니다. 그런 다음 queue1-dy-c5xlarge-[1-2]을 유휴 상태로 구성하고 사설 IP 주소와 호스트 이름을 재설정하여 향후 작업에 다시 사용할 수 있도록 합니다.

이제 문제가 발생하여 특정 노드의 인스턴스를 어떤 이유로 시작할 수 없는 경우 다음과 같은 상황이 발생합니다.

- 스케줄러는 두 개의 노드가 필요한 작업을 수신합니다.
- 스케줄러는 두 개의 클라우드 버스팅 노드를 POWER_UP 상태로 전환하고 노드 이름을 사용하여 ResumeProgram를 호출합니다(예: queue1-dy-c5xlarge-[1-2]).
- ResumeProgram은 EC2 인스턴스 1개만 시작하고 queue1-dy-c5xlarge-1를 구성하지만 queue1-dy-c5xlarge-2에 대한 인스턴스를 시작하지 못했습니다.
- queue1-dy-c5xlarge-1은 영향을 받지 않으며 POWER_UP 상태에 도달하면 온라인 상태가 됩니다.
- queue1-dy-c5xlarge-2은 POWER_DOWN 상태가 되고 Slurm이 노드 장애를 감지했으므로 작업이 자동으로 대기열에 추가됩니다.
- queue1-dy-c5xlarge-2은 SuspendTimeout 이후에 사용할 수 있습니다[기본 120초(2분)]. 그동안에는 작업이 대기되며 다른 노드에서 실행을 시작할 수 있습니다.

- 장애가 발생하지 않고 사용 가능한 노드에서 작업을 실행할 수 있을 때까지 위 프로세스가 반복됩니다.

필요한 경우 두 가지 타이밍 매개 변수를 조정할 수 있습니다.

- `ResumeTimeout`[기본 60분(1시간)]: `ResumeTimeout`은 노드를 작동 중지 상태로 전환하기 전에 Slurm이 대기하는 시간을 제어합니다.
  - 사전/사후 설치 프로세스가 그렇게 오래 걸린다면 이것을 연장하는 것이 유용할 수 있습니다.
  - 또한 문제가 있는 경우 노드를 교체하거나 재설정하기 전에가 AWS ParallelCluster 대기하는 최대 시간입니다. 시작 또는 설정 중에 오류가 발생하면 컴퓨팅 노드가 자동으로 종료됩니다. 그런 다음 인스턴스가 종료된 것을 확인하면 AWS ParallelCluster 프로세스가 노드를 대체하기도 합니다.
- `SuspendTimeout`[기본 120초(2분)]: `SuspendTimeout`은 노드를 시스템에 다시 배치하고 다시 사용할 준비가 되는 시간을 제어합니다.
  - `SuspendTimeout`이 짧을수록 노드가 더 빨리 재설정되고, Slurm는 인스턴스를 더 자주 시작하려고 할 수 있습니다.
  - `SuspendTimeout`이 길수록 장애가 발생한 노드의 재설정 속도가 느려집니다. 그러는 동안 Slurm은 다른 노드를 사용하려고 합니다. `SuspendTimeout`이 몇 분 이상이면 Slurm가 시스템의 모든 노드를 순환하려 합니다. 대규모 시스템(1,000개 이상의 노드)에서는 Slurm가 받는 스트레스를 줄이기 위해 장애가 발생한 작업을 다시 대기열에 추가하려고 할 때 더 긴 `SuspendTimeout`을 사용하는 것이 유용할 수 있습니다.
  - `SuspendTimeout`는 노드의 백업 인스턴스를 종료하기 위해 AWS ParallelCluster 대기한 시간을 참조하지 않습니다. `power down` 노드의 백업 인스턴스는 즉시 종료됩니다. 종료 프로세스는 일반적으로 몇 분 이내에 완료됩니다. 하지만 이 기간 동안에는 노드가 전원 차단 상태를 유지하며 스케줄러에서 사용될 수 없습니다.

새 아키텍처에 대한 로그

다음 목록에는 다중 대기열 아키텍처의 키 로그가 포함되어 있습니다. Amazon CloudWatch Logs에 사용되는 로그 스트림 이름은 *`{hostname}.{instance_id}.{logIdentifier}`* 형식을 사용합니다. 여기서 *`LogiDentifier`*는 로그 이름 뒤에 옵니다. 자세한 내용은 Amazon CloudWatch Logs와 통합 단원을 참조하십시오.

- `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)

일반적인 문제 및 디버그 방법:

시작, 전원 켜기 또는 클러스터 조인에 실패한 노드

- 동적 노드:
  - ResumeProgram 로그를 확인하여 노드와 함께 ResumeProgram이 호출된 적이 있는지 확인하세요. 그렇지 않은 경우 slurmctld 로그를 확인하여 Slurm가 노드로 ResumeProgram을 호출하려 한 적이 있는지 확인하세요. ResumeProgram 권한이 올바르지 않으면 로그가 자동으로 실패할 수 있다는 점에 유의하세요.
  - ResumeProgram가 호출되면 해당 노드에 대한 인스턴스가 시작되었는지 확인하세요. 인스턴스가 시작되지 않은 경우 인스턴스 시작 실패 이유에 대한 명확한 오류 메시지가 표시되어야 합니다.
  - 인스턴스가 시작된 경우 부트스트랩 프로세스 중에 문제가 발생했을 수 있습니다. ResumeProgram 로그에서 해당 프라이빗 IP 주소와 인스턴스 ID를 찾고 CloudWatch Logs에서 특정 인스턴스에 대한 해당 부트스트랩 로그를 확인합니다.
- 고정 노드:
  - clustermgtd 로그를 확인하여 해당 노드의 인스턴스가 시작되었는지 확인합니다. 그렇지 않다면 인스턴스 시작 실패 이유에 대한 명확한 오류가 있을 것입니다.
  - 인스턴스가 시작된 경우 부트스트랩 프로세스에 문제가 있는 것입니다. clustermgtd 로그에서 해당 프라이빗 IP와 인스턴스 ID를 찾고 CloudWatch Logs에서 특정 인스턴스에 대한 해당 부트스트랩 로그를 확인합니다.

노드가 예기치 않게 교체되거나 종료되었으며, 노드 장애가 발생했습니다.

- 노드가 예기치 않게 교체/종료됨:
  - 대부분의 경우 clustermgtd가 모든 노드 유지 관리 작업을 처리합니다. clustermgtd가 노드를 교체하거나 종료했는지를 확인하려면 clustermgtd 로그를 확인하세요.
  - clustermgtd가 노드를 교체하거나 종료한 경우 작업 이유를 나타내는 메시지가 표시되어야 합니다. 이유가 스케줄러와 관련된 경우(예: 노드가 DOWN이었음) slurmctld 로그에서 자세한 내용을 확인하세요. 이유가 EC2와 관련된 것이라면 는 도구를 사용하여 해당 인스턴스의 상태 또는 로그를 확인합니다. 예를 들어 인스턴스에 예약된 이벤트가 있는지 또는 EC2 상태 확인에 실패했는지 확인할 수 있습니다.
  - clustermgtd가 노드를 종료하지 않은 경우 computemgtd가 노드를 종료했는지 또는 EC2가 스팟 인스턴스를 회수하기 위해 인스턴스를 종료했는지 확인하세요.
- 노드 장애
  - 대부분의 경우 노드에 장애가 발생하면 작업이 자동으로 대기열에 추가됩니다. slurmctld 로그에서 작업이나 노드에 장애가 발생한 이유를 확인하고 거기에서 상황을 분석하세요.

인스턴스 교체 또는 종료 시 실패, 노드 전원 차단 시 실패

- 일반적으로 clustermgtd가 모든 예상 인스턴스 종료 작업을 처리합니다. clustermgtd 로그에서 노드 교체 또는 종료에 실패한 이유를 확인하세요.
- 동적 노드가 scaledown_idletime에 실패한 경우 SuspendProgram 로그에서 slurmctld 프로그램이 특정 노드를 인수로 사용하여 호출했는지 확인하세요. SuspendProgram는 실제로 특정 작업을 수행하지 않습니다. 그보다는 호출될 때만 로그를 기록합니다. 모든 인스턴스 종료 및 NodeAddr 재설정은 clustermgtd가 완료합니다. Slurm는 SuspendTimeout 후 노드를 IDLE로 만듭니다.

기타 문제

- AWS ParallelCluster 는 작업 할당 또는 조정 결정을 내리지 않습니다. Slurm의 지침에 따라 리소스를 시작, 종료 및 유지 관리하려고 시도할 뿐입니다.

  작업 할당, 노드 할당 및 규모 조정 결정과 관련된 문제는 slurmctld 로그에서 오류를 확인하세요.

Torque Resource Manager (`torque`)

> **Note**
>
> 버전 2.11.5부터 SGE 또는 Torque 스케줄러 사용을 지원하지 AWS ParallelCluster 않습니다. 2.11.4 이하의 버전에서는 계속 사용할 수 있지만 AWS 서비스 및 AWS 지원 팀의 향후 업데이트 또는 문제 해결 지원을 받을 수 없습니다.

AWS ParallelCluster 버전 2.11.4 이하에서는 Torque Resource Manager 6.1.2를 사용합니다. Torque Resource Manager 6.1.2에 대한 자세한 내용은 http://docs.adaptivecomputing.com/torque/6-1-2/releaseNotes/torquerelnote.htm 섹션을 참조하세요. 설명서는 http://docs.adaptivecomputing.com/torque/6-1-2/adminGuide/torque.htm 섹션을 참조하세요. 소스 코드는 https://github.com/adaptivecomputing/torque/tree/6.1.2 섹션을 참조하세요.

AWS ParallelCluster 버전 2.4.0 이하에서는 Torque Resource Manager 6.0.2를 사용합니다. 릴리스 정보는 http://docs.adaptivecomputing.com/torque/6-0-2/releaseNotes/torqueReleaseNotes6.0.2.pdf 섹션을 참조하세요. 설명서는 http://docs.adaptivecomputing.com/torque/6-0-2/adminGuide/help.htm 섹션을 참조하세요. 소스 코드는 https://github.com/adaptivecomputing/torque/tree/6.0.2 섹션을 참조하세요.

AWS Batch (`awsbatch`)

에 대한 자세한 내용은 단원을 AWS Batch참조하십시오AWS Batch. 설명서는 AWS Batch 사용 설명서를 참조하세요.

AWS ParallelCluster 에 대한 CLI 명령 AWS Batch

`awsbatch` 스케줄러를 사용하면에 대한 AWS ParallelCluster CLI 명령이 AWS ParallelCluster 헤드 노드에 자동으로 설치 AWS Batch 됩니다. CLI는 AWS Batch API 작업을 사용하며 다음 작업을 허용합니다.

- 작업 제출 및 관리
- 작업, 대기열 및 호스트 모니터링
- 기존 스케줄러 명령 미러링

> **Important**
>
> AWS ParallelCluster 는에 대한 GPU 작업을 지원하지 않습니다 AWS Batch. 자세한 내용은 GPU 작업을 참조하세요.

주제

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

awsbsub

작업을 클러스터의 작업 대기열에 제출합니다.

```
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 는에 대한 GPU 작업을 지원하지 않습니다 AWS Batch. 자세한 내용은 GPU 작업을 참조하세요.

위치 인수

command

작업을 제출(지정된 명령이 컴퓨팅 인스턴스에서 사용 가능해야 함)하거나 전송할 파일 이름을 지정합니다. 또한 `--command-file` 섹션도 참조하세요.

arguments

(선택 사항) 명령 또는 명령 파일의 인수를 지정합니다.

이름 지정된 인수

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

작업 이름을 지정합니다. 첫 번째 자리는 문자 또는 숫자여야 합니다. 작업 이름은 최대 128자까지 포함할 수 있으며, 대문자와 소문자, 숫자, 하이픈(-), 밑줄(_)을 포함할 수 있습니다.

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

사용할 클러스터를 지정합니다.

-cf, --command-file

명령이 컴퓨팅 인스턴스로 전송될 파일임을 나타냅니다.

기본값: False

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

작업의 작업 디렉터리로 사용할 폴더를 지정합니다. 작업 디렉터리가 지정되지 않으면 작업이 사용자의 홈 디렉터리에 있는 `job-<AWS_BATCH_JOB_ID>` 하위 폴더에서 실행됩니다. 이 파라미터 또는 `--parent-working-dir` 파라미터를 사용할 수 있습니다.

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

작업의 작업 디렉터리에서 상위 폴더를 지정합니다. 상위 작업 디렉터리가 지정되지 않은 경우, 사용자의 홈 디렉터리가 기본적으로 지정됩니다. 상위 작업 디렉터리에 `job-<AWS_BATCH_JOB_ID>`라는 하위 폴더가 만들어집니다. 이 파라미터 또는 `--working-dir` 파라미터를 사용할 수 있습니다.

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

작업의 작업 디렉터리에서 컴퓨팅 인스턴스로 전송할 파일을 지정합니다. 여러 입력 파일 파라미터를 지정할 수 있습니다.

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

컨테이너를 위해 예약할 vCPU 개수를 지정합니다. –nodes와 함께 사용할 경우 노드당 vCPU 수를 식별합니다.

기본값: 1

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

작업에 제공할 메모리의 하드 제한(MiB)을 지정합니다. 작업에서 여기서 지정된 메모리 제한을 초과하려고 하면 해당 작업이 종료됩니다.

기본값: 128

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

작업 환경으로 내보낼 환경 변수 이름의 목록을 쉼표로 구분하여 지정합니다. 모든 환경 변수를 내보내려면 'all'을 지정하세요. `-env-blacklist` 파라미터에 나열된 변수, 또는 `PCLUSTER_*`나 `AWS_*`로 시작하는 변수는 'all' 환경 변수 목록에 포함되지 않습니다.

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

작업 환경으로 내보내지 않을 환경 변수 이름의 목록을 쉼표로 구분하여 지정합니다. 기본적으로, `HOME`, `PWD`, `USER`, `PATH`, `LD_LIBRARY_PATH`, `TERM` 및 `TERMCAP`은 내보내지 않습니다.

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

작업을 `RUNNABLE` 상태로 전환하는 횟수를 지정합니다. 1부터 10까지 시도 횟수를 지정할 수 있습니다. 시도 횟수가 1보다 큰 경우 작업이 실패하면 `RUNNABLE` 상태로 전환될 때까지 지정된 횟수만큼 다시 시도됩니다.

기본값: 1

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

완료되지 않은 경우가 작업을 AWS Batch 종료하는 시간을 초 단위(작업 시도의 `startedAt` 타임스탬프에서 측정)로 지정합니다. 제한 시간 값은 60초 이상이어야 합니다.

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

작업을 위해 예약할 노드 수를 지정합니다. 다중 노드 병렬 제출을 사용하려면 이 파라미터의 값을 지정합니다.

> **Note**
>
> `cluster_type` 파라미터가 `spot`로 설정된 경우 다중 노드 병렬 작업은 지원되지 않습니다.

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

배열의 크기를 지정합니다. 2~10,000 범위의 값을 지정할 수 있습니다. 작업에 배열 속성을 지정하면 배열 작업이 됩니다.

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

작업에 대해 세미콜론으로 구분된 종속성 목록을 지정합니다. 작업은 최대 20개의 작업에 종속될 수 있습니다. 배열 작업의 작업 ID를 지정하지 않고 SEQUENTIAL 유형의 종속성을 지정할 수 있습니다. 순차 종속성을 사용하면 각 하위 배열 작업을 인덱스 0부터 순차적으로 완료할 수 있습니다. 배열 작업의 작업 ID로 N_TO_N 유형의 종속성을 지정할 수도 있습니다. N_TO_N 종속성이란 이 작업의 각 인덱스 하위 항목은 각 종속성의 해당 인덱스 하위 항목이 완료될 때까지 기다린 후에만 시작할 수 있다는 의미입니다. 이 파라미터의 구문은 "jobId=*<string>*,type=*<string>*;..."입니다.

awsbstat

클러스터의 작업 대기열에 제출된 작업을 표시합니다.

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

위치 인수

job_ids

출력에 표시할 작업 ID 목록을 공백으로 구분하여 지정합니다. 작업이 작업 배열이면 모든 하위 작업이 표시됩니다. 단일 작업이 요청되면 자세한 버전으로 표시됩니다.

이름 지정된 인수

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

사용할 클러스터를 지정합니다.

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

포함할 작업 상태 목록을 쉼표로 구분하여 지정합니다. 기본 작업 상태는 “활성”입니다. 허용되는 값: SUBMITTED, PENDING, RUNNABLE, STARTING, RUNNING, SUCCEEDED, FAILED 및 ALL

기본값: “SUBMITTED,PENDING,RUNNABLE,STARTING,RUNNING”

-e, --expand-children

하위 항목(배열 및 다중 노드 병렬 모두)을 사용하여 작업을 확장합니다.

기본값: False

-d, --details

작업 세부 정보를 표시합니다.

기본값: False

awsbout

지정된 작업의 출력을 표시합니다.

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

위치 인수

job_id

작업 ID를 지정합니다.

이름 지정된 인수

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

사용할 클러스터를 지정합니다.

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

작업 출력의 첫 번째 *HEAD* 행을 가져옵니다.

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

작업 출력의 마지막 <tail> 행을 가져옵니다.

-s, --stream

작업 출력을 가져온 다음 추가 출력이 생성될 때까지 대기합니다. 이 인수는 -tail과 함께 사용하여 작업 출력의 마지막 <tail> 행에서 시작할 수 있습니다.

기본값: False

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

스트리밍 기간을 설정합니다.

기본값: 5

awsbkill

클러스터에 제출된 작업을 취소하거나 종료합니다.

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

위치 인수

job_ids

작업을 취소하거나 종료할 작업 ID 목록을 공백으로 구분하여 지정합니다.

이름 지정된 인수

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

사용할 클러스터의 이름을 나타냅니다.

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

작업에 첨부할 메시지를 표시하고 취소 이유를 설명합니다.

기본값: “Terminated by the user”

awsbqueues

클러스터와 연관된 작업 대기열을 표시합니다.

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

위치 인수

job_queues

표시할 대기열 이름의 목록을 공백으로 구분하여 지정합니다. 단일 대기열이 요청되면 자세한 버전으로 표시됩니다.

이름 지정된 인수

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

사용할 클러스터의 이름을 지정합니다.

-d, --details

대기열의 세부 정보 표시 여부를 나타냅니다.

기본값: False

awsbhosts

클러스터의 컴퓨팅 환경에 속한 호스트를 표시합니다.

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

위치 인수

instance_ids

공백으로 구분된 인스턴스 ID 목록을 지정합니다. 단일 인스턴스가 요청되면 자세한 버전으로 표시됩니다.

이름 지정된 인수

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

사용할 클러스터의 이름을 지정합니다.

-d, --details

호스트의 세부 정보 표시 여부를 나타냅니다.

기본값: False

AWS ParallelCluster 리소스 및 태그 지정

를 AWS ParallelCluster 사용하면 태그를 생성하여 AWS ParallelCluster 리소스를 추적하고 관리할 수 있습니다. 클러스터 구성 파일의 tags 섹션에서 생성하여 모든 클러스터 리소스에 전파 AWS

CloudFormation 하려는 태그를 정의합니다. 에서 AWS ParallelCluster 자동으로 생성하는 태그를 사용하여 리소스를 추적하고 관리할 수도 있습니다.

클러스터를 생성하면 클러스터와 해당 리소스에이 섹션에 정의된 AWS ParallelCluster 및 AWS 시스템 태그가 지정됩니다.

AWS ParallelCluster 는 클러스터 인스턴스, 볼륨 및 리소스에 태그를 적용합니다. 클러스터 스택을 식별하기 위해는 클러스터 인스턴스에 AWS 시스템 태그를 AWS CloudFormation 적용합니다. 클러스터 EC2 시작 템플릿을 식별하기 위해 EC2는 인스턴스에 시스템 태그를 적용합니다. 이러한 태그를 사용하여 AWS ParallelCluster 리소스를 보고 관리할 수 있습니다.

AWS 시스템 태그는 수정할 수 없습니다. AWS ParallelCluster 기능에 미치는 영향을 방지하려면 AWS ParallelCluster 태그를 수정하지 마십시오.

다음은 AWS ParallelCluster 리소스에 대한 AWS 시스템 태그의 예입니다. 해당 태그는 수정할 수 없습니다.

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

다음은 리소스에 적용된 AWS ParallelCluster 태그의 예입니다. 수정하지 마세요.

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

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

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

AWS Management Console의 EC2 섹션에서 이러한 태그를 볼 수 있습니다.

태그 보기

1. https://console.aws.amazon.com/ec2/에서 EC2 콘솔을 탐색하세요.
2. 모든 클러스터 태그를 보려면 탐색 창에서 태그를 선택합니다.
3. 인스턴스별로 클러스터 태그를 보려면 탐색 창에서 인스턴스를 선택합니다.
4. 클러스터 인스턴스를 선택합니다.
5. 인스턴스 세부 정보에서 태그 관리 탭을 선택하고 태그를 확인합니다.

6. 인스턴스 세부 정보에서 스토리지 탭을 선택합니다.
7. 볼륨 ID를 선택합니다.
8. 볼륨에서 볼륨을 선택합니다.
9. 볼륨 세부 정보에서 태그 탭을 선택하고 태그를 확인합니다.

AWS ParallelCluster 헤드 노드 인스턴스 태그

키	태그 값
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*
Version	*2.11.9*

AWS ParallelCluster 헤드 노드 루트 볼륨 태그

태그 키	태그 값
ClusterName	*clustername*
Application	parallelcluster- *clustername*
aws-parallelcluster-node-type	Master

AWS ParallelCluster 컴퓨팅 노드 인스턴스 태그

키	태그 값
ClusterName	*clustername*
aws-parallelcluster-node-type	Compute
aws:ec2launchtemplate:id	*lt-1234567890abcdef0*
aws:ec2launchtemplate:version	*1*
QueueName	*queue-name*
Version	*2.11.9*

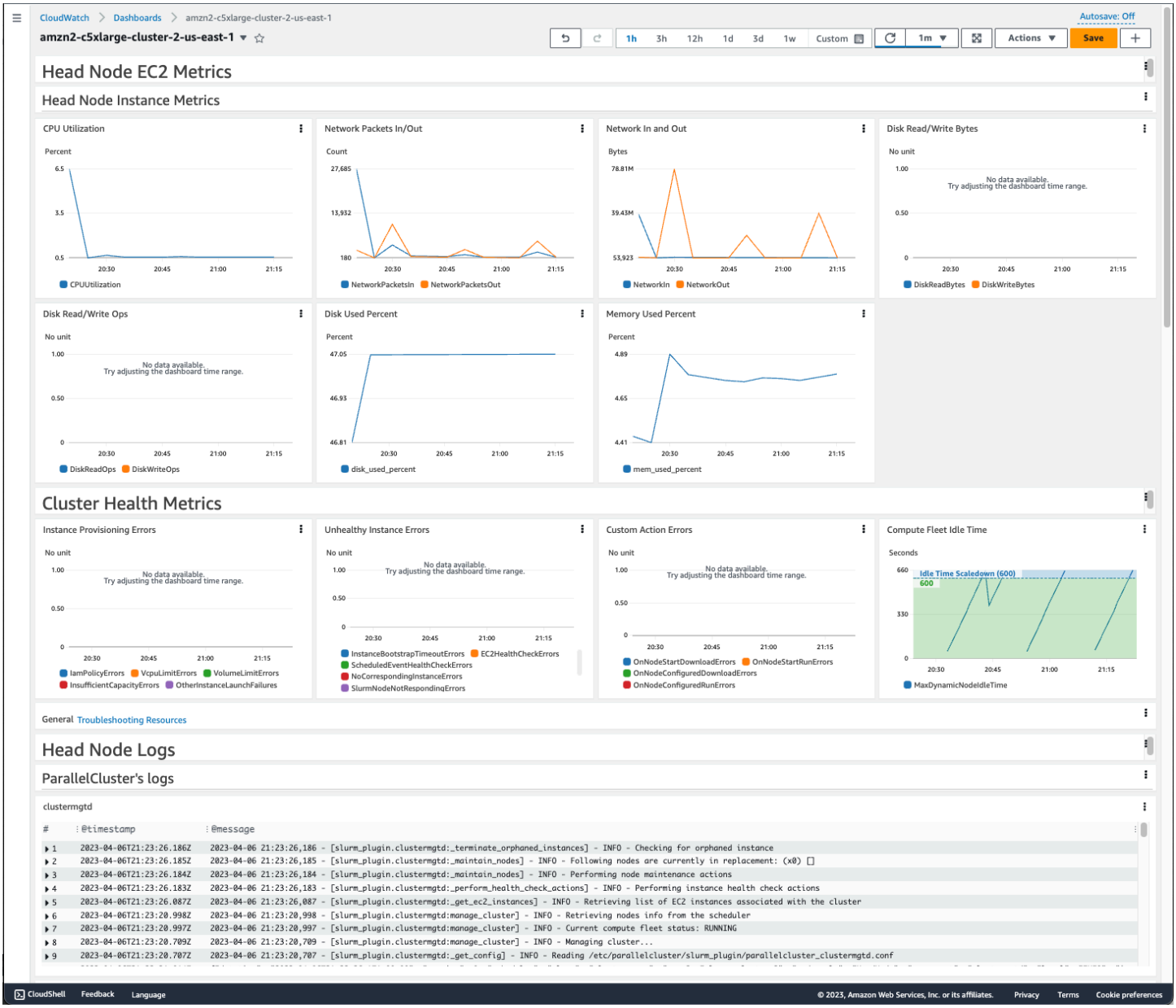
AWS ParallelCluster 컴퓨팅 노드 루트 볼륨 태그

태그 키	태그 값
ClusterName	*clustername*
Application	parallelcluster- *clustername*
aws-parallelcluster-node-type	Compute
QueueName	*queue-name*
Version	*2.11.9*

Amazon CloudWatch 대시보드

AWS ParallelCluster 버전 2.10.0부터 클러스터가 생성될 때 Amazon CloudWatch 대시보드가 생성됩니다. 이렇게 하면 클러스터의 노드를 더 쉽게 모니터링하고 Amazon CloudWatch Logs에 저장된 로그를 볼 수 있습니다. 대시보드의 이름은 `parallelcluster-`*`ClusterName`*`-`*`Region`*입니다. *`ClusterName`*은 클러스터의 이름이고 *`##`*은 클러스터의 AWS 리전 입니다. 콘솔에서 또는 `https://console.aws.amazon.com/cloudwatch/home?region=`*`Region`*`#dashboards:name=parallelcluster-`*`ClusterName`*를 열어서 대시보드에 액세스할 수 있습니다.

다음 이미지는 클러스터에 대한 CloudWatch 대시보드의 예를 보여 줍니다.



대시보드의 첫 번째 섹션에는 헤드 노드 EC2 지표의 그래프가 표시됩니다. 클러스터에 공유 스토리지가 있는 경우 다음 섹션에는 공유 스토리지 지표가 표시됩니다. 마지막 섹션은 ParallelCluster 로그, 스케줄러 로그, NICE DCV 통합 로그, 시스템 로그별로 그룹화된 헤드 노드 로그가 나열되어 있습니다.

Amazon CloudWatch 대시보드에 관한 자세한 내용은 Amazon CloudWatch 사용 설명서의 Amazon CloudWatch 대시보드 사용을 참조하세요.

Amazon CloudWatch 대시보드를 생성하지 않으려면 다음 단계를 완료해야 합니다. 먼저 구성 파일에 `[dashboard]` 섹션을 추가한 다음 해당 섹션의 이름을 `[cluster]`섹션의 `dashboard_settings` 설정 값으로 추가합니다. `[dashboard]` 섹션에서 `enable = false`을 설정합니다.

예를 들어 [dashboard] 섹션 이름이 myDashboard이고 [cluster] 섹션 이름이 myCluster이면 변경 내용은 다음과 같습니다.

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

[dashboard MyDashboard]
enable = false
```

Amazon CloudWatch Logs와 통합

AWS ParallelCluster 버전 2.6.0부터 공통 로그는 기본적으로 CloudWatch Logs에 저장됩니다. CloudWatch Logs에 대한 자세한 내용은 Amazon CloudWatch Logs 사용 설명서를 참조하세요. CloudWatch Logs 통합을 구성하려면 [cw_log] 섹션 및 cw_log_settings 설정을 참조하세요.

이름이 /aws/parallelcluster/*cluster-name*인 각 클러스터에 대해 로그 그룹이 생성됩니다 (예: /aws/parallelcluster/testCluster). 각 노드의 각 로그(또는 경로에 *가 포함된 경우 로그 집합)에는 *{hostname}*.*{instance_id}*.*{logIdentifier}*라는 로그 스트림이 있습니다. (예: ip-172-31-10-46.i-02587cf29cc3048f3.nodewatcher.) 로그 데이터는 모든 클러스터 인스턴스에서 root로 실행되는 CloudWatch 에이전트에 의해 CloudWatch로 전송됩니다.

AWS ParallelCluster 버전 2.10.0부터 클러스터가 생성될 때 Amazon CloudWatch 대시보드가 생성됩니다. 이 대시보드를 사용하면 CloudWatch Logs에 저장된 로그를 쉽게 검토할 수 있습니다. 자세한 내용은 Amazon CloudWatch 대시보드 단원을 참조하십시오.

이 목록에는 플랫폼, 스케줄러 및 노드에 사용할 수 있는 로그 스트림의 *logIdentifier* 및 경로가 포함되어 있습니다.

플랫폼, 스케줄러 및 노드에 사용할 수 있는 로그 스트림

플랫폼	스케줄러	노드	로그 스트림
amazon centos ubuntu	awsbat slurm	HeadNc	dcv-authenticator: /var/log/parallelcluster/pcluster_dcv_authenticator.log dcv-ext-authenticator: /var/log/parallelcluster/pcluster_dcv_connect.log

플랫폼	스케줄러	노드	로그 스트림
			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	awsbatc slurm	Comput eet HeadNc	cloud-init: /var/log/cloud-init.log supervisord: /var/log/supervisord.log
amazon centos ubuntu	slurm	Comput eet	cloud-init-output: /var/log/cloud-init-output.log computemgtd: /var/log/parallelcluster/computemgtd slurmd: /var/log/slurmd.log
amazon centos ubuntu	slurm	HeadNc	clustermgtd: /var/log/parallelcluster/clustermgtd slurm_resume: /var/log/parallelcluster/slurm_resume.log slurm_suspend: /var/log/parallelcluster/slurm_suspend.log slurmctld: /var/log/slurmctld.log

플랫폼	스케줄러	노드	로그 스트림
amazon centos	awsbatc slurm	Comput eet HeadNc	system-messages: `/var/log/messages`
ubuntu	awsbatc slurm	Comput eet HeadNc	syslog: `/var/log/syslog`

를 사용하는 클러스터의 작업은 CloudWatch Logs에 `RUNNING`, `SUCCEEDED`또는 `FAILED` 상태에 도달한 작업의 출력을 AWS Batch 저장합니다. 로그 그룹은 `/aws/batch/job`이며 로그 스트림 이름 형식은 *`jobDefinitionName`*`/default/`*`ecs_task_id`*입니다. 기본적으로 이러한 로그들은 만료되도록 설정하지 않지만 유지 기간을 수정할 수 있습니다. 자세한 내용은 Amazon CloudWatch Logs User Guide의 CloudWatch에서 로그 데이터 보존 기간을 변경을 참조하세요.

> **Note**
>
> `chef-client`버전 2.9.0에 , , , AWS ParallelCluster `slurm_resume`, `cloud-init-output` `clustermgtd` `computemgtd`및가 `slurm_suspend` 추가되었습니다. AWS ParallelCluster 버전 2.6.0의 경우 `/var/log/cfn-init-cmd.log`(`cfn-init-cmd`) 및 `/var/log/cfn-wire.log`(`cfn-wire`)도 CloudWatch Logs에 저장되었습니다.

Elastic Fabric Adapter

Elastic Fabric Adapter(EFA)는 동일한 서브넷에 있는 다른 인스턴스와의 대기 시간이 짧은 네트워크 통신을 위한 OS 바이패스 기능을 갖춘 네트워크 디바이스입니다. EFA는 Libfabric을 사용하여 노출되며 Messaging Passing Interface(MPI)를 사용하는 애플리케이션에서 사용할 수 있습니다.

에서 EFA를 사용하려면 `[queue]` 섹션에 `enable_efa = true` 선을 AWS ParallelCluster추가합니다.

EFA를 지원하는 EC2 인스턴스 목록을 보려면 Linux 인스턴스용 Amazon EC2 사용 설명서의 지원되는 인스턴스 유형을 참조하세요.

`enable_efa` 설정에 대한 자세한 내용은 [`[queue]` 섹션](#)의 [`enable_efa`](#)를 참조하세요.

클러스터 배치 그룹은 인스턴스 간의 대기 시간을 최소화하기 위해 사용해야 합니다. 자세한 내용은 [`placement`](#) 및 [`placement_group`](#) 섹션을 참조하세요.

자세한 내용은 Amazon EC2 사용 설명서의 Elastic Fabric Adapter와 AWS 오픈 소스 블로그의 Elastic Fabric Adapter 및 AWS ParallelCluster를 사용한 HPC 워크로드 규모 조정을 참조하세요.

> **Note**
>
> 기본적으로 Ubuntu 배포는 ptrace(프로세스 추적) 보호를 활성화합니다. AWS ParallelCluster 2,6.0부터는 ptrace 보호 기능이 비활성화되어 Libfabric이 제대로 작동합니다. 자세한 정보는 Amazon EC2 사용 설명서의 ptrace 보호 비활성화를 참조하세요.

> **Note**
>
> AWS ParallelCluster 버전 2.10.1에서 Arm 기반 Graviton2 인스턴스에 대한 지원 EFA가 추가되었습니다.

Intel 셀렉트 솔루션

AWS ParallelCluster 는 시뮬레이션 및 모델링을 위한 Intel Select 솔루션으로 사용할 수 있습니다. 구성은 Intel HPC 플랫폼 사양에서 설정한 표준을 충족하고, 특정 Intel 인스턴스 유형을 사용하며, Elastic Fabric Adapter (EFA) 네트워킹 인터페이스를 사용하도록 구성됩니다. AWS ParallelCluster 는 Intel Select Solutions 프로그램의 요구 사항을 충족하는 최초의 클라우드 솔루션입니다. 지원되는 인스턴스 유형은 `c5n.18xlarge`, `m5n.24xlarge`, `r5n.24xlarge`입니다. Intel 셀렉트 솔루션 표준과 호환되는 샘플 구성은 다음과 같습니다.

Example Intel 셀렉트 솔루션 구성

```
[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 및 Intel HPC 플랫폼 사양에 대한 자세한 내용은 섹션을 참조하세요Intel HPC 플랫폼 사양.

Intel MPI 활성화

Intel MPI는 AWS ParallelCluster AMIs. Intel MPI를 사용하려면 Intel 간소화된 소프트웨어 라이센스를 인정하고 이에 동의해야 합니다. 기본적으로 Open MPI는 경로에 배치됩니다. Open MPI 대신 Intel MPI를 활성화하려면 Intel MPI 모듈을 먼저 로드해야 합니다. 그런 다음 `module load intelmpi`를 사용하여 최신 버전을 설치해야 합니다. 모듈의 정확한 이름은 업데이트할 때마다 변경됩니다. 사용 가능한 모듈을 확인하려면 `module avail`을 실행합니다. 출력값은 다음과 같습니다.

```
$ 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
```

로드된 모듈을 확인하려면 `module list`를 실행합니다.

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

Intel MPI가 활성화되어 있는지 확인하려면 `mpirun --version`을 실행하세요.

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

```
Copyright 2003-2020, Intel Corporation.
```

Intel MPI 모듈이 로드되면 Intel MPI 도구를 사용하도록 여러 경로가 변경됩니다. Intel MPI 도구로 컴파일된 코드를 실행하려면 먼저 Intel MPI 모듈을 로드합니다.

> **Note**
>
> Intel MPI는 AWS Graviton 기반 인스턴스와 호환되지 않습니다.

> **Note**
>
> AWS ParallelCluster 버전 2.5.0 이전에는 중국(베이징) 및 중국(닝샤) 리전의 AWS ParallelCluster AMIs에서 Intel MPI를 사용할 수 없었습니다.

Intel HPC 플랫폼 사양

AWS ParallelCluster 는 Intel HPC 플랫폼 사양을 준수합니다. Intel HPC 플랫폼 사양은 높은 수준의 품질과 HPC 워크로드와의 호환성을 달성하기 위해 일련의 컴퓨팅, 패브릭, 메모리, 스토리지 및 소프트웨어 요구 사항을 제공합니다. 자세한 내용은 Intel HPC 플랫폼 사양 및 Intel HPC 플랫폼 사양과 호환이 검증된 애플리케이션을 참조하세요.

Intel HPC 플랫폼 사양을 준수하려면 다음 요구 사항을 충족해야 합니다.

- 운영 체제는 CentOS 7(`base_os = centos7`)이어야 합니다.
- 컴퓨팅 노드의 인스턴스 유형에는 Intel CPU와 64GB 이상의 메모리가 있어야 합니다. `c5` 패밀리의 인스턴스 유형인 경우 인스턴스 유형이 적어도 `c5.9xlarge`(`compute_instance_type = c5.9xlarge`)여야 합니다.
- 헤드 노드에는 200GB 이상의 스토리지가 있어야 합니다.
- Intel Parallel Studio에 대한 최종 사용자 라이선스 계약에 동의해야 합니다(`enable_intel_hpc_platform = true`).
- 각 컴퓨팅 노드에는 80GB 이상의 스토리지가 있어야 합니다(`compute_root_volume_size = 80`).

스토리지는 로컬 또는 네트워크(헤드 노드, Amazon EBS 또는 FSx for Lustre에서 공유된 NFS)에 있을 수 있으며 공유할 수 있습니다.

Arm 퍼포먼스 라이브러리

AWS ParallelCluster 버전 2.10.1부터 Arm 성능 라이브러리는 base_os 설정의 , alinux2, centos8 ubuntu1804및 ubuntu2004 값에 대한 AWS ParallelCluster AMIs에서 사용할 수 있습니다. Arm 퍼포먼스 라이브러리는 Arm 프로세서의 고성능 컴퓨팅 애플리케이션을 위해 최적화된 표준 코어 수학 라이브러리를 제공합니다. Arm 퍼포먼스 라이브러리를 사용하려면 Arm 퍼포먼스 라이브러리(무료 버전 - 최종 사용자 사용권 계약)의 약관을 확인하고 동의해야 합니다. Arm 퍼포먼스 라이브러리에 대한 자세한 내용은 무료 Arm 퍼포먼스 라이브러리를 참조하세요.

Arm 퍼포먼스 라이브러리를 활성화하려면 먼저 Arm 퍼포먼스 라이브러리 모듈을 로드해야 합니다. Armpl-21.0.0는 요구 사항으로 GCC-9.3이 필요합니다. armpl/21.0.0 모듈을 로드하면 gcc/9.3 모듈도 로드됩니다. 모듈의 정확한 이름은 업데이트할 때마다 변경됩니다. 사용 가능한 모듈을 확인하려면 module avail을 실행합니다. 그런 다음 module load armpl를 사용하여 최신 버전을 설치해야 합니다. 출력값은 다음과 같습니다.

```
$ 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
```

모듈을 로드하려면 module load *modulename*을 실행합니다. mpirun을 실행하는 데 사용된 스크립트에 이를 추가할 수 있습니다.

```
$ 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
```

로드된 모듈을 확인하려면 module list를 실행합니다.

```
$ 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 퍼포먼스 라이브러리가 활성화되었는지 확인하려면 예제 테스트를 실행하세요.

```
$ 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 퍼포먼스 라이브러리 모듈이 로드되면 Arm 퍼포먼스 라이브러리 도구를 사용하도록 여러 경로가 변경됩니다. Arm 퍼포먼스 라이브러리 도구로 컴파일된 코드를 실행하려면 먼저 Arm 퍼포먼스 라이브러리 모듈을 로드하세요.

> **Note**
>
> AWS ParallelCluster 2.10.1에서 2.10.4 사이의 버전은를 사용합니다`armpl/20.2.1`.

Amazon DCV를 통해 헤드 노드에 연결합니다.

Amazon DCV는 사용자가 원격 고성능 서버에 호스팅된 그래픽 집약형 3D 애플리케이션에 안전하게 연결할 수 있는 원격 시각화 기술입니다. 자세한 내용은 Amazon DCV를 참조하세요.

Amazon DCV 소프트웨어는 `base_os = alinux2`, `base_os = centos7`, `base_os = ubuntu1804` 또는 `base_os = ubuntu2004`를 사용할 때 헤드 노드에 자동으로 설치됩니다.

헤드 노드가 ARM 인스턴스인 경우 `base_os = alinux2`, `base_os = centos7`, 또는 `base_os = ubuntu1804`를 사용할 때 Amazon DCV 소프트웨어가 자동으로 설치됩니다.

헤드 노드에서 Amazon DCV를 활성화하려면 `dcv_settings`은 `enable = master`이 있는 `[dcv]` 섹션의 이름이 들어 있어야 하며 `base_os`는 `alinux2`, `centos7`, `ubuntu1804` 또는 `ubuntu2004`로 설정되어야 합니다. 헤드 노드가 ARM 인스턴스인 경우, `base_os`은 `alinux2`, `centos7` 또는 `ubuntu1804`로 설정되어야 합니다. 이렇게 하면 클러스터 구성 파라미터가 DCV 서버 스토리지 폴더`shared_dir`로 AWS ParallelCluster 설정됩니다.

```
[cluster custom-cluster]
```

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

Amazon DCV 구성 파라미터에 대한 자세한 내용은 dcv_settings 섹션을 참조하세요. Amazon DCV 세션에 연결하려면 pcluster dcv 명령을 사용합니다.

> **Note**
>
> AWS ParallelCluster 버전 2.10.4에서의 Amazon DCV에 대한 지원이 제거centos8되었습니다. AWS ParallelCluster 버전 2.10.0에서의 Amazon DCV에 대한 지원이 centos8 추가되었습니다. AWS Graviton 기반 인스턴스의 Amazon DCV에 대한 지원이 AWS ParallelCluster 버전 2.9.0에 추가되었습니다. alinux2 및의 Amazon DCV에 대한 지원이 AWS ParallelCluster 버전 2.6.0에 ubuntu1804 추가되었습니다. AWS ParallelCluster 버전 2.5.0에서의 Amazon DCV에 대한 지원이 centos7 추가되었습니다.

> **Note**
>
> Amazon DCV는 AWS ParallelCluster 버전 2.8.0 및 2.8.1의 AWS Graviton 기반 인스턴스에서는 지원되지 않습니다.

Amazon DCV HTTPS 인증서

Amazon DCV는 Amazon DCV 클라이언트와 Amazon DCV 서버 간의 트래픽을 보호하는 자체 서명된 인증서를 자동으로 생성합니다.

기본 자체 서명 Amazon DCV 인증서를 다른 인증서로 바꾸려면 먼저 헤드 노드에 연결합니다. 그런 다음 pcluster dcv 명령을 실행하기 전에 인증서와 키를 모두 /etc/dcv 폴더에 복사합니다.

자세한 내용은 Amazon DCV 관리자 안내서의 TLS 인증서 변경을 참조하세요.

Amazon DCV 라이선싱

Amazon DCV 서버는 Amazon EC2 인스턴스에서 실행될 때 라이선스 서버가 필요하지 않습니다. 그러나 Amazon DCV 서버는 정기적으로 Amazon S3 버킷에 연결하여 유효한 라이선스를 사용할 수 있는지 여부를 확인해야 합니다.

AWS ParallelCluster 는에 필요한 권한을 자동으로 추가합니다ParallelClusterInstancePolicy. 사용자 지정 IAM 인스턴스 정책을 사용할 때는 Amazon DCV 관리자 안내서의 Amazon EC2 기반 Amazon DCV에 설명된 권한을 사용하세요.

문제 해결 팁은 Amazon DCV의 문제 해결 섹션을 참조하세요.

pcluster update 사용하기

AWS ParallelCluster 버전 2.8.0부터는 현재 클러스터를 생성하는 데 사용된 설정과 구성 파일의 설정에 문제가 있는지 pcluster update 분석합니다. 문제가 발견되면 해당 문제가 보고되고 문제 해결을 위해 취해야 할 단계가 표시됩니다. 예를 들어 compute_instance_type 설정이 다른 인스턴스 유형으로 변경되면 업데이트를 진행하기 전에 컴퓨팅 플릿을 중지해야 합니다. 이 문제는 발견되면 보고됩니다. 차단 문제가 보고되지 않은 경우 변경 사항을 적용할지 여부를 묻는 메시지가 표시됩니다.

각 설정의 설명서에는 해당 설정에 대한 업데이트 정책이 정의되어 있습니다.

업데이트 정책: 업데이트 중에 이 설정들을 변경할 수 있습니다.,업데이트 정책: 업데이트 중에 이 설정을 변경할 수 있습니다.

> 이 설정들은 변경할 수 있으며 pcluster update를 사용하여 클러스터를 업데이트할 수 있습니다.

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

> 기존 클러스터를 삭제하지 않은 경우 이 설정들을 변경할 수 없습니다. 변경 내용을 되돌리거나 클러스터를 삭제한(pcluster delete 사용) 다음 이전 클러스터 대신 새 클러스터를 생성해야 (pcluster create 사용) 합니다.

업데이트 정책: 이 설정은 업데이트 중에 분석되지 않습니다.

> 이 설정들은 변경할 수 있으며 pcluster update를 사용하여 클러스터가 업데이트됩니다.

업데이트 정책: 업데이트를 위해 이 설정을 변경하려면 컴퓨팅 플릿을 중지해야 합니다.

> 컴퓨팅 플릿이 존재하는 동안에는 이 설정들을 변경할 수 없습니다. 변경 내용을 되돌리거나 컴퓨팅 플릿을 중지(pcluster stop 사용), 업데이트(pcluster update 사용)한 다음 새 컴퓨팅 플릿을 생성(pcluster start 사용)해야 합니다.

업데이트 정책: 업데이트 중에는 이 설정을 줄일 수 없습니다.

> 이 설정들은 변경할 수 있지만 줄일 수는 없습니다. 이러한 설정을 줄여야 하는 경우 클러스터를 삭제(pcluster delete 사용)하고 새 클러스터를 생성(pcluster create 사용)해야 합니다.

업데이트 정책: 대기열 크기를 현재 노드 수 미만으로 줄이려면 먼저 컴퓨팅 플릿을 중지해야 합니다.

이 설정들은 변경할 수 있지만 변경으로 인해 대기열 크기가 현재 크기보다 작아지면 컴퓨팅 플릿을 중지(pcluster stop 사용)하고 업데이트(pcluster update 사용)한 다음 새 컴퓨팅 플릿을 생성(pcluster start 사용)해야 합니다.

업데이트 정책: 대기열의 정적 노드 수를 줄이려면 먼저 컴퓨팅 플릿을 중지해야 합니다.

이 설정들은 변경할 수 있지만 변경으로 인해 대기열에 있는 정적 노드 수가 현재 크기 아래로 줄어드는 경우 컴퓨팅 플릿을 중지(pcluster stop 사용)하고 업데이트(pcluster update 사용)한 다음 새 컴퓨팅 플릿을 생성(pcluster start 사용)해야 합니다.

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다. 이 설정은 강제로 업데이트할 수 없습니다.

기존 클러스터를 삭제하지 않은 경우 이 설정들을 변경할 수 없습니다. 변경 내용을 되돌리거나 클러스터를 삭제한(pcluster delete 사용) 다음 이전 클러스터 대신 새 클러스터를 생성해야 (pcluster create 사용) 합니다.

업데이트 정책: 구성에 AWS ParallelCluster 관리형 Amazon FSx for Lustre 파일 시스템이 지정되지 않은 경우 업데이트 중에이 설정을 변경할 수 있습니다.

이 설정은 [cluster] fsx_settings가 지정되지 않았거나 fsx_settings와 [fsx fs]의 fsx-fs-id가 기존 외부 FSx for Lustre 파일 시스템을 탑재하기 위해 지정되어 있으면 변경할 수 있습니다.

이 예제는 업데이트를 차단하는 일부 변경 사항이 적용된 pcluster update를 보여줍니다.

```
$ 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 패치 및 EC2 인스턴스 교체

동적으로 시작된 모든 클러스터 컴퓨팅 노드가 일관된 방식으로 작동하도록 하기 위해 클러스터 인스턴스 자동 OS 업데이트를 AWS ParallelCluster 비활성화합니다. 또한 각 버전 AWS ParallelCluster 및 관련 CLI에 대해 특정 AWS ParallelCluster AMIs 집합이 빌드됩니다. 이 특정 AMIs 세트는 변경되지 않으며 빌드된 AWS ParallelCluster 버전에서만 지원됩니다. 릴리스된 버전의 AWS ParallelCluster AMIs는 업데이트되지 않습니다.

그러나 새로운 보안 문제로 인해 고객은 이러한 AMI에 패치를 추가한 다음 패치된 AMI로 클러스터를 업데이트하기를 원할 수 있습니다. 이는 AWS ParallelCluster 공동 책임 모델과 일치합니다.

현재 사용 중인 AWS ParallelCluster CLI 버전에서 지원하는 특정 AWS ParallelCluster AMIs 세트를 보려면 다음을 실행합니다.

```
$ pcluster version
```

그런 다음 GitHub 리포지토리에서 amis.txt AWS ParallelCluster를 봅니다.

AWS ParallelCluster 헤드 노드는 정적 인스턴스이며 수동으로 업데이트할 수 있습니다. 인스턴스 유형에 인스턴스 스토어가 없는 경우 AWS ParallelCluster 버전 2.11부터 헤드 노드의 재시작 및 재부팅이 완전히 지원됩니다. 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 인스턴스 스토어 볼륨이 있는 인스턴스 유형을 참조하세요. 기존 클러스터의 AMI는 업데이트할 수 없습니다.

클러스터 컴퓨팅 인스턴스의 AMI 업데이트를 통한 헤드 노드 재시작 및 재부팅은 AWS ParallelCluster 버전 3.0.0부터 완벽하게 지원됩니다. 이러한 특성을 사용하려면 최신 버전으로 업그레이드하는 것이 좋습니다.

헤드 노드 인스턴스 업데이트 또는 교체

경우에 따라 헤드 노드를 재시작하거나 재부팅해야 할 수 있습니다. 예를 들어, OS를 수동으로 업데이트하거나 헤드 노드 인스턴스를 다시 시작해야 하는 AWS 예약된 인스턴스 사용 중지가 있는 경우 이 설정이 필요합니다.

인스턴스에 휘발성 드라이브가 없는 경우 언제든지 이 인스턴스를 중지했다가 다시 시작할 수 있습니다. 사용 중지가 예정된 경우 중지된 인스턴스를 시작하면 새 하드웨어를 사용하도록 마이그레이션됩니다.

마찬가지로 인스턴스 스토어가 없는 인스턴스를 수동으로 중지하고 시작할 수 있습니다. 이 경우와 휘발성 볼륨이 없는 다른 인스턴스의 경우 클러스터의 헤드 노드 중지 및 시작로 가세요.

인스턴스에 휘발성 드라이브가 있고 중지된 경우 인스턴스 스토어의 데이터가 손실됩니다. 헤드 노드에 사용되는 인스턴스 유형에 인스턴스 스토어가 있는지 여부는 인스턴스 스토어 볼륨에 있는 표에서 확인할 수 있습니다.

다음 섹션은 인스턴스 스토어 볼륨과 함께 인스턴스를 사용할 때의 제한 사항을 설명합니다.

인스턴스 스토어 제한

인스턴스 스토어에서 AWS ParallelCluster 버전 2.11 및 인스턴스 유형을 사용할 때의 제한 사항은 다음과 같습니다.

- 임시 드라이브가 암호화되지 않은 경우(`encrypted_ephemeral` 파라미터가 로 설정`false`되거나 설정되지 않은 경우) AWS ParallelCluster 인스턴스가 중지된 후 인스턴스를 부팅할 수 없습니다. 이

는 존재하지 않는 오래된 휘발성 스토리지에 대한 정보가 fstab에 기록되고 OS가 존재하지 않는 스토리지를 탑재하려고 하기 때문입니다.

- 임시 드라이브가 암호화되면(encrypted_ephemeral 파라미터가 로 설정됨true) 중지 후 AWS ParallelCluster 인스턴스를 시작할 수 있지만 새 임시 드라이브는 설정, 탑재 또는 사용할 수 없습니다.
- 임시 드라이브가 암호화되면 AWS ParallelCluster 인스턴스를 재부팅할 수 있지만 재부팅 시 손실된 메모리에 암호화 키가 생성되므로 이전 임시 드라이브(인스턴스 재부팅 후에도 유지됨)에 액세스할 수 없습니다.

지원되는 유일한 경우는 휘발성 드라이브가 암호화되지 않는 인스턴스 재부팅입니다. 이는 드라이브가 재부팅 후에도 유지되고 fstab에 항목이 기록되어 다시 탑재되기 때문입니다.

인스턴스 스토어 제한 해결 방법

먼저 데이터를 저장합니다. 보존해야 할 데이터가 있는지 확인하려면 ephemeral_dir 폴더의 콘텐츠를 확인하세요(기본값 /scratch). 데이터를 루트 볼륨 또는 클러스터에 연결된 공유 스토리지 시스템(예: Amazon FSx, Amazon EFS 또는 Amazon EBS)으로 전송할 수 있습니다. 단, 원격 스토리지로 데이터를 전송할 때는 추가 비용이 발생할 수 있습니다.

제한 사항의 근본 원인은가 인스턴스 스토어 볼륨을 포맷하고 탑재하는 데 AWS ParallelCluster 사용하는 로직에 있습니다. 로직은 /etc/fstab 형식에 항목을 추가합니다.

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

${ephemeral_dir}은 pcluster 구성 파일의 ephemeral_dir 파리미터 값입니다(/scratch으로 디폴트).

이 행이 추가되어 노드가 재부팅되는 경우 또는 노드가 재부팅될 때 인스턴스 스토어 볼륨이 자동으로 다시 탑재됩니다. 이는 재부팅 후에도 휘발성 드라이브의 데이터가 유지되기 때문에 바람직합니다. 하지만 휘발성 드라이브의 데이터는 시작 또는 중지 주기 동안 유지되지 않습니다. 즉, 포맷되고 데이터 없이 탑재됩니다.

지원되는 유일한 경우는 휘발성 드라이브가 암호화되지 않는 인스턴스 재부팅입니다. 이는 드라이브가 재부팅 후에도 유지되고 fstab에 기록되어 다시 탑재되기 때문입니다.

다른 모든 경우에 데이터를 보존하려면 인스턴스를 중지하기 전에 논리적 볼륨 항목을 제거해야 합니다. 예를 들어, 인스턴스를 중지하기 전에 /etc/fstab에서 /dev/vg.01/lv_ephemeral를 제거하

세요. 이렇게 한 후에는 임시 볼륨을 탑재하지 않고 인스턴스를 시작합니다. 하지만 인스턴스 중지 또는 시작 후에는 인스턴스 스토어 탑재를 다시 사용할 수 없습니다.

데이터를 저장한 다음 `fstab` 항목을 제거한 후 다음 섹션으로 가세요.

클러스터의 헤드 노드 중지 및 시작

> **Note**
>
> AWS ParallelCluster 버전 2.11부터 헤드 노드 중지 및 시작은 인스턴스 유형에 인스턴스 스토어가 없는 경우에만 지원됩니다.

1. 클러스터에 실행 중인 작업이 없는지 확인하세요.

   Slurm 스케줄러를 사용하는 경우:

   - `sbatch --no-requeue` 옵션이 지정되어 있지 않으면 실행 중인 작업이 다시 대기됩니다.
   - `--no-requeue` 옵션이 지정되어 있으면 실행 중인 작업이 실패합니다.
2. 클러스터 컴퓨팅 플릿 중지 요청:

   ```
   $ 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. 컴퓨팅 플릿 상태가 `STOPPED`일 때까지 기다리세요.

   ```
   $ pcluster status cluster-name
   ...
   ComputeFleetStatus: STOP_REQUESTED
   $ pcluster status cluster-name
   ...
   ComputeFleetStatus: STOPPED
   ```
4. OS 재부팅 또는 인스턴스 재시작을 통한 수동 업데이트의 경우 AWS Management Console 또는 를 사용할 수 있습니다 AWS CLI. 다음은 AWS CLI를 사용한 예입니다.

   ```
   $ 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. 클러스터의 컴퓨팅 플릿을 시작합니다.

```
$ 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 명령

`pcluster` 및 `pcluster-config`는 AWS ParallelCluster CLI 명령입니다. `pcluster`를 사용하여에서 HPC 클러스터를 시작 AWS 클라우드 및 관리하고 `pcluster-config`를 사용하여 구성을 업데이트합니다.

`pcluster`를 사용하려면 실행에 필요한 권한이 있는 IAM 역할이 있어야 합니다.

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

주제

- pcluster
- pcluster-config

pcluster

`pcluster`는 기본 AWS ParallelCluster CLI 명령입니다. `pcluster`를 사용하여 AWS 클라우드에서 HPC 클러스터를 시작하고 관리할 수 있습니다.

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

인수

pcluster *command*

가능한 선택: configure, create, createami, dcv, delete, instances, list, ssh, start, status, stop, update, version

하위 명령:

주제

- 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 구성을 시작합니다. 자세한 내용은 구성 AWS ParallelCluster 단원을 참조하십시오.

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

이름 지정된 인수

-h, --help

pcluster configure에 대한 도움말 텍스트를 표시합니다.

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

사용할 대체 구성 파일의 전체 경로를 지정합니다.

기본값은 ~/.parallelcluster/config입니다.

자세한 내용은 구성 AWS ParallelCluster 단원을 참조하십시오.

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

사용할 AWS 리전 를 지정합니다. 이를 지정하면 구성이 AWS 리전 감지를 건너뜁니다.

VPC에서 네트워크 리소스를 삭제하려면 CloudFormation 네트워킹 스택을 삭제하면 됩니다. 스택 이름은 “parallelclusternetworking-”으로 시작하며 “YYYYMMDDHHMMSS” 형식의 생성 시간을 포함합니다. list-stacks 명령을 사용하여 스택을 나열할 수 있습니다.

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

스택은 delete-stack 명령을 사용하여 삭제할 수 있습니다.

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

pcluster configure가 사용자를 위해 생성하는 VPC는 CloudFormation 네트워킹 스택에서 생성되지 않습니다. 콘솔에서 또는 AWS CLI를 사용하여 해당 VPC를 수동으로 삭제할 수 있습니다.

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

pcluster create

새 클러스터를 생성합니다.

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

위치 인수

cluster_name

클러스터의 이름을 정의합니다. AWS CloudFormation 스택 이름은 입니다parallelcluster-*cluster_name*.

이름 지정된 인수

`-h, --help`

pcluster create에 대한 도움말 텍스트를 표시합니다.

`-c CONFIG_FILE, --config CONFIG_FILE`

사용할 대체 구성 파일을 지정합니다.

기본값은 ~/.parallelcluster/config입니다.

`-r REGION, --region REGION`

사용할 AWS 리전 를 지정합니다. 새 클러스터의 AWS 리전 를 선택하는 데 사용되는 우선 순위는 다음과 같습니다.

1. pcluster create의 -r 또는 --region 파라미터.
2. AWS_DEFAULT_REGION 환경 변수.
3. aws_region_name AWS ParallelCluster 구성 파일의 [aws] 섹션에서 설정(기본 위치는 ~/.parallelcluster/config) pcluster configure 명령으로 업데이트된 위치입니다.
4. region AWS CLI 구성 파일(~/.aws/config)의 [default] 섹션에서 설정

`-nw, --nowait`

스택 명령을 실행한 후 스택 이벤트를 기다리지 않습니다.

기본값은 False입니다.

`-nr, --norollback`

오류 시 스택 롤백을 비활성화합니다.

기본값은 False입니다.

`-u TEMPLATE_URL, --template-url TEMPLATE_URL`

사용자 지정 템플릿이 생성될 때 사용된 경우 사용자 지정 AWS CloudFormation 템플릿의 URL을 지정합니다.

`-t CLUSTER_TEMPLATE, --cluster-template CLUSTER_TEMPLATE`

사용할 클러스터 템플릿을 지정합니다.

`-p EXTRA_PARAMETERS, --extra-parameters EXTRA_PARAMETERS`

스택 생성에 파라미터를 추가합니다.

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

스택에 추가할 추가 태그를 지정합니다.

명령이 호출되고 해당 호출의 상태에 대한 폴링을 시작하면 안전하게 "Ctrl-C"를 사용하여 종료할 수 있습니다. `pcluster status mycluster`를 호출하여 현재 상태 보기로 돌아갈 수 있습니다.

AWS ParallelCluster 버전 2.11.7을 사용하는 예:

```
$ 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) 함께 사용할 사용자 지정 AMI를 생성합니다 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 ]
```

필수 종속 항목

AWS ParallelCluster CLI 외에도를 실행하려면 `pcluster createami`다음 종속성이 필요합니다.

- Packer: https://developer.hashicorp.com/packer/downloads에서 최신 버전을 다운로드합니다.

> **Note**
>
> AWS ParallelCluster 버전 2.8.0 이전에는를 사용하려면 Berkshelf(를 사용하여 설치됨`gem install berkshelf`)가 필요했습니다`pcluster createami`.

이름 지정된 인수

`-h, --help`

`pcluster createami`에 대한 도움말 텍스트를 표시합니다.

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

AMI를 빌드하는 데 사용할 기본 AWS ParallelCluster AMI를 지정합니다.

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

기본 AMI의 OS를 지정합니다. 유효한 옵션은 `alinux2`, `ubuntu1804`, `ubuntu2004` 및 `centos7`입니다.

> **Note**
>
> OS는 다양한 AWS ParallelCluster 버전의 변경 사항을 지원합니다.
>
> - AWS ParallelCluster 버전 2.10.4에서에 대한 지원이 제거`centos8`되었습니다.
> - AWS ParallelCluster 버전 2.10.0에서 `centos6`에 대한 지원은 제거되었으며 `centos8`에 대한 지원이 추가되었습니다.
> - alinux2에 대한 지원이 AWS ParallelCluster 버전 2.6.0에 추가되었습니다.
> - AWS ParallelCluster 버전 2.5.0에서 ubuntu1804에 대한 지원이 추가되었습니다.

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

AMI를 생성하는 데 사용할 인스턴스 유형을 지정합니다.

기본값은 `t2.xlarge`입니다.

> **Note**
>
> `--instance-type` 인수에 대한 지원이 AWS ParallelCluster 버전 2.4.1에 추가되었습니다.

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

결과 AWS ParallelCluster AMI의 접두사 이름을 지정합니다.

기본값은 custom-ami-입니다.

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

AWS ParallelCluster AMI를 빌드하는 데 사용할 쿡북을 지정합니다.

--post-install *POST_INSTALL_SCRIPT*

설치 후 스크립트의 경로를 지정합니다. 경로에는 s3://, https:// 또는 file:// URL 체계를 사용해야 합니다. 예는 다음과 같습니다.

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

> **Note**
>
> --post-install 인수에 대한 지원이 AWS ParallelCluster 버전 2.10.0에 추가되었습니다.

--no-public-ip

AMI를 생성하는 데 사용된 인스턴스에 퍼블릭 IP 주소를 연결하지 마세요. 기본적으로 인스턴스에는 퍼블릭 IP 주소가 연결됩니다.

> **Note**
>
> --no-public-ip 인수에 대한 지원이 AWS ParallelCluster 버전 2.5.0에 추가되었습니다.

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

사용할 대체 구성 파일을 지정합니다.

기본값은 ~/.parallelcluster/config입니다.

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

VPC 및 서브넷 설정을 검색하는 데 사용할 *CONFIG_FILE*# [cluster] 섹션을 지정합니다.

> **Note**
>
> `--cluster-template` 인수에 대한 지원이 AWS ParallelCluster 버전 2.4.0에 추가되었습니다.

`--vpc-id` *`VPC_ID`*

AWS ParallelCluster AMI를 빌드하는 데 사용할 VPC의 ID를 지정합니다.

> **Note**
>
> `--vpc-id` 인수에 대한 지원이 AWS ParallelCluster 버전 2.5.0에 추가되었습니다.

`--subnet-id` *`SUBNET_ID`*

AWS ParallelCluster AMI를 빌드하는 데 사용할 서브넷의 ID를 지정합니다.

> **Note**
>
> `--vpc-id` 인수에 대한 지원이 AWS ParallelCluster 버전 2.5.0에 추가되었습니다.

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

사용할 AWS 리전 를 지정합니다. 기본값은 `pcluster configure` 명령을 사용하여 AWS 리전 지정된 입니다.

pcluster dcv

헤드 노드에서 실행되는 Amazon DCV 서버와 상호 작용합니다.

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

`pcluster dcv` *`command`*

가능한 선택: `connect`

> **Note**
>
> OS는 다양한 AWS ParallelCluster 버전의 `pcluster dcv` 명령에 대한 변경 사항을 지원합니다.
>
> - `centos8`의 `pcluster dcv` 명령에 대한 지원이 AWS ParallelCluster 2.10.0에서 추가되었습니다.
> - AWS Graviton 기반 인스턴스의 `pcluster dcv` 명령에 대한 지원이 AWS ParallelCluster 버전 2.9.0에 추가되었습니다.
> - `ubuntu1804`의 `pcluster dcv` 명령에 대한 지원이 AWS ParallelCluster 2.6.0에서 추가되었습니다.
> - `centos7`의 `pcluster dcv` 명령에 대한 지원이 AWS ParallelCluster 2.5.0에서 추가되었습니다.

이름 지정된 인수

`-h, --help`

`pcluster dcv`에 대한 도움말 텍스트를 표시합니다.

하위 명령

`pcluster dcv connect`

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

> **Important**
>
> URL은 발행된 후 30초 지나면 만료됩니다. URL이 만료되기 전에 연결이 이루어지지 않으면 `pcluster dcv connect`를 다시 실행하여 새 URL을 생성합니다.

위치 인수

`cluster_name`

연결할 클러스터의 이름을 지정합니다.

이름 지정된 인수

-h, --help

pcluster dcv connect에 대한 도움말 텍스트를 표시합니다.

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

연결에 사용할 SSH 키의 키 경로입니다.

키는 key_name 구성 파라미터에서 클러스터 생성 시 지정되어야 합니다. 이 인수는 선택 사항이지만 지정하지 않으면 SSH 클라이언트에 대해 기본적으로 키를 사용할 수 있어야 합니다. 예를 들어 ssh-agent를 ssh-add와 함께 추가합니다.

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

사용할 AWS 리전 를 지정합니다. 기본값은 pcluster configure 명령을 사용하여 AWS 리전 지정된 입니다.

-s, --show-url

Amazon DCV 세션에 연결하기 위한 일회성 URL을 표시합니다. 이 옵션을 지정하면 기본 브라우저가 열리지 않습니다.

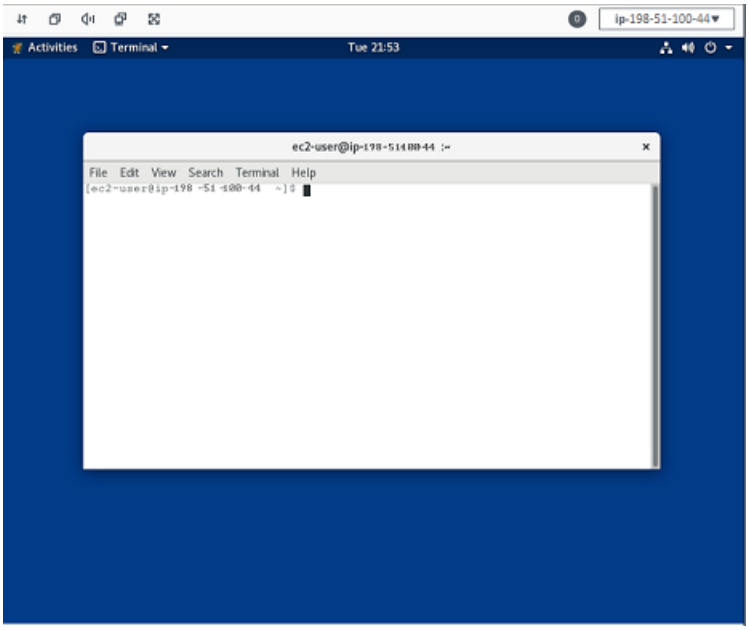
> **Note**
>
> --show-url 인수에 대한 지원이 AWS ParallelCluster 버전 2.5.1에 추가되었습니다.

AWS ParallelCluster 버전 2.11.7 사용 예제:

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

기본 브라우저를 열어 헤드 노드에서 실행 중인 Amazon DCV 세션에 연결합니다.

아직 시작되지 않은 경우 새 Amazon DCV 세션이 생성됩니다.



pcluster delete

클러스터를 삭제합니다.

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

위치 인수

cluster_name

삭제할 클러스터의 이름을 지정합니다.

이름 지정된 인수

-h, --help

pcluster delete에 대한 도움말 텍스트를 표시합니다.

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

사용할 대체 구성 파일을 지정합니다.

기본값은 ~/.parallelcluster/config입니다.

--keep-logs

클러스터를 삭제한 후 CloudWatch Logs 데이터를 보관합니다. 로그 그룹은 수동으로 삭제될 때까지 유지되지만 로그 이벤트는 retention_days 설정에 따라 만료됩니다. 설정 기본값은 14일입니다.

> **Note**
>
> **--keep-logs** 인수에 대한 지원이 AWS ParallelCluster 버전 2.6.0에서 추가되었습니다.

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

사용할 AWS 리전 를 지정합니다. 기본값은 pcluster configure 명령을 사용하여 AWS 리전 지정된 입니다.

명령이 호출되고 해당 호출의 상태에 대한 폴링을 시작하면 안전하게 "Ctrl-C"를 사용하여 종료할 수 있습니다. `pcluster status mycluster`를 호출하여 현재 상태 보기로 돌아갈 수 있습니다.

AWS ParallelCluster 버전 2.11.7 사용 예제:

```
$ pcluster delete -c path/to/config -r us-east-1 mycluster
Deleting: mycluster
Status: RootRole - DELETE_COMPLETE
Cluster deleted successfully.
```

VPC에서 네트워크 리소스를 삭제하려면 CloudFormation 네트워킹 스택을 삭제하면 됩니다. 스택 이름은 “parallelclusternetworking-”으로 시작하며 “YYYYMMDDHHMMSS” 형식의 생성 시간을 포함합니다. list-stacks 명령을 사용하여 스택을 나열할 수 있습니다.

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

스택은 delete-stack 명령을 사용하여 삭제할 수 있습니다.

```
$ aws --region us-east-1 cloudformation delete-stack \
```

```
    --stack-name parallelclusternetworking-pubpriv-20191029205804
```

pcluster configure가 사용자를 위해 생성하는 VPC는 CloudFormation 네트워킹 스택에서 생성되지 않습니다. 콘솔에서 또는 AWS CLI를 사용하여 해당 VPC를 수동으로 삭제할 수 있습니다.

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

pcluster instances

클러스터에 있는 모든 인스턴스의 목록을 표시합니다.

```
pcluster instances [ -h ] [ -c CONFIG_FILE ] [ -r REGION ] cluster_name
```

위치 인수

cluster_name

제공된 이름의 클러스터에 대한 인스턴스를 표시합니다.

이름 지정된 인수

-h, --help

pcluster instances에 대한 도움말 텍스트를 표시합니다.

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

사용할 대체 구성 파일을 지정합니다.

기본값은 ~/.parallelcluster/config입니다.

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

사용할 AWS 리전 를 지정합니다. 기본값은 pcluster configure 명령을 사용하여 AWS 리전 지정된 입니다.

AWS ParallelCluster 버전 2.11.7 사용 예제:

```
$ pcluster instances -c path/to/config -r us-east-1 mycluster
MasterServer         i-1234567890abcdef0
```

```
ComputeFleet         i-abcdef01234567890
```

pcluster list

연결된 스택 목록을 표시합니다 AWS ParallelCluster.

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

이름 지정된 인수

-h, --help

pcluster list에 대한 도움말 텍스트를 표시합니다.

--color

클러스터 상태를 색상으로 표시합니다.

기본값은 False입니다.

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

사용할 대체 구성 파일을 지정합니다.

기본값은 c입니다.

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

사용할 AWS 리전 를 지정합니다. 기본값은 pcluster configure 명령을 사용하여 AWS 리전 지정된 입니다.

이름이 인 AWS CloudFormation 스택의 이름을 나열합니다parallelcluster-*.

AWS ParallelCluster 버전 2.11.7 사용 예제:

```
$ 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

클러스터 사용자 이름과 IP 주소가 미리 채워진 상태로 ssh 명령을 실행합니다. 임의의 인수가 ssh 명령 끝에 추가됩니다. 이 명령은 구성 파일의 별칭 섹션에서 사용자 지정할 수 있습니다.

```
pcluster ssh [ -h ] [ -d ] [ -r REGION ] cluster_name
```

위치 인수

cluster_name

연결할 클러스터의 이름을 지정합니다.

이름 지정된 인수

-h, --help

pcluster ssh에 대한 도움말 텍스트를 표시합니다.

-d, --dryrun

실행할 명령을 인쇄하고 종료합니다.

기본값은 False입니다.

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

사용할 AWS 리전 를 지정합니다. pcluster configure 명령을 사용하여 지정된 리전이 기본값입니다.

AWS ParallelCluster 버전 2.11.7을 사용하는 예:

```
$ 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
```

클러스터의 IP 주소와 사용자 이름이 미리 채워진 상태로 ssh 명령을 실행합니다.

```
ssh ec2-user@1.1.1.1 -i ~/.ssh/id_rsa
```

ssh 명령은 [aliases] 섹션 아래의 전역 구성 파일에서 정의되며 다음과 같이 사용자 지정할 수 있습니다.

```
[ aliases ]
ssh = ssh {CFN_USER}@{MASTER_IP} {ARGS}
```

대체된 변수:

CFN_USER

선택된 base_os의 사용자 이름입니다.

MASTER_IP

헤드 노드의 IP 주소입니다.

ARGS

ssh 명령에 전달할 선택적 인수입니다.

pcluster start

중지된 클러스터에 대한 컴퓨팅 플릿을 시작합니다.

```
pcluster start [ -h ] [ -c CONFIG_FILE ] [ -r REGION ] cluster_name
```

위치 인수

cluster_name

제공된 클러스터 이름의 컴퓨팅 플릿을 시작합니다.

이름 지정된 인수

-h, --help

pcluster start에 대한 도움말 텍스트를 표시합니다.

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

사용할 대체 구성 파일을 지정합니다.

기본값은 ~/.parallelcluster/config입니다.

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

사용할 AWS 리전 를 지정합니다. 기본값은 pcluster configure 명령을 사용하여 AWS 리전 지정된 입니다.

AWS ParallelCluster 버전 2.11.7 사용 예제:

```
$ 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
```

이 명령은 Auto Scaling 그룹 파라미터를 다음 중 하나로 설정합니다.

- 클러스터를 생성하는 데 사용된 템플릿의 초기 구성 값(max_queue_size 및 initial_queue_size)
- 클러스터가 처음 생성된 이후에 클러스터를 업데이트하는 데 사용된 구성 값

pcluster status

클러스터의 현재 상태를 풀합니다.

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

위치 인수

cluster_name

제공된 이름이 있는 클러스터의 상태를 표시합니다.

이름 지정된 인수

-h, --help

pcluster status에 대한 도움말 텍스트를 표시합니다.

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

사용할 대체 구성 파일을 지정합니다.

기본값은 ~/.parallelcluster/config입니다.

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

사용할 AWS 리전 를 지정합니다. 기본값은 pcluster configure 명령을 사용하여 AWS 리전 지정된 입니다.

-nw, --nowait

스택 명령을 처리한 후 스택 이벤트를 기다리지 않습니다.

기본값은 False입니다.

AWS ParallelCluster 버전 2.11.7 사용 예제:

```
$ pcluster status -c path/to/config -r us-east-1 mycluster
Status: ComputeFleetHITSubstack - CREATE_IN_PROGRESS
```

pcluster stop

헤드 노드를 실행 중 상태로 두고 컴퓨팅 플릿을 중지합니다.

```
pcluster stop [ -h ] [ -c CONFIG_FILE ] [ -r REGION ] cluster_name
```

위치 인수

cluster_name

제공된 클러스터 이름의 컴퓨팅 플릿을 중지합니다.

AWS ParallelCluster 버전 2.11.7 사용 예제:

이름 지정된 인수

-h, --help

pcluster stop에 대한 도움말 텍스트를 표시합니다.

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

사용할 대체 구성 파일을 지정합니다.

기본값은 ~/.parallelcluster/config입니다.

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

사용할 AWS 리전 를 지정합니다. 기본값은 pcluster configure 명령을 사용하여 AWS 리전 지정된 입니다.

```
$ 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 그룹 파라미터를 최소/최대/원하는 값 = 0/0/0으로 설정하고 컴퓨팅 플릿을 종료합니다. 헤드는 계속 실행됩니다. 모든 EC2 리소스를 종료하고 EC2 요금을 방지하려면 클러스터를 삭제하는 것이 좋습니다.

pcluster update

구성 파일을 분석하여 클러스터를 안전하게 업데이트할 수 있는지 확인합니다. 분석 결과 클러스터를 업데이트할 수 있는 것으로 확인되면 변경 사항을 확인하라는 메시지가 표시됩니다. 분석 결과 클러스터를 업데이트할 수 없는 것으로 나타나는 경우 충돌의 원인인 구성 설정이 세부 정보와 함께 열거됩니다. 자세한 내용은 pcluster update 사용하기 단원을 참조하십시오.

```
pcluster update [ -h ] [ -c CONFIG_FILE ] [ --force ] [ -r REGION ] [ -nr ]
                [ -nw ] [ -t CLUSTER_TEMPLATE ] [ -p EXTRA_PARAMETERS ] [ -rd ]
                [ --yes ] cluster_name
```

위치 인수

cluster_name

업데이트할 클러스터의 이름을 지정합니다.

이름 지정된 인수

-h, --help

pcluster update에 대한 도움말 텍스트를 표시합니다.

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

사용할 대체 구성 파일을 지정합니다.

기본값은 ~/.parallelcluster/config입니다.

--force

하나 이상의 설정에 차단 변경이 있거나 업데이트를 진행하기 전에 특별한 조치(예: 컴퓨팅 플릿 중지)가 필요한 경우에도 업데이트를 활성화합니다. 이것을 --yes 인수와 함께 사용해서는 안 됩니다.

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

사용할 AWS 리전 를 지정합니다. 기본값은 pcluster configure 명령을 사용하여 AWS 리전 지정된 입니다.

-nr, --norollback

오류 발생 시 AWS CloudFormation 스택 롤백을 비활성화합니다.

기본값은 False입니다.

-nw, --nowait

스택 명령을 처리한 후 스택 이벤트를 기다리지 않습니다.

기본값은 False입니다.

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

사용할 클러스터 템플릿의 섹션을 지정합니다.

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

스택 업데이트에 파라미터를 추가합니다.

-rd, --reset-desired

Auto Scaling 그룹의 현재 용량을 초기 구성 값으로 재설정합니다.

기본값은 False입니다.

--yes

모든 질문 메시지에 대한 답은 자동적으로 예라고 가정합니다. 이것을 --force 인수와 함께 사용해서는 안 됩니다.

```
$ 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
```

명령이 호출되고 해당 호출의 상태에 대한 폴링을 시작하면 안전하게 "Ctrl-C"를 사용하여 종료할 수 있습니다. `pcluster status mycluster`를 호출하여 현재 상태 보기로 돌아갈 수 있습니다.

pcluster version

AWS ParallelCluster 버전을 표시합니다.

```
pcluster version [ -h ]
```

명령별 플래그를 보려면 `pcluster [command] --help`를 실행하세요.

이름 지정된 인수

-h, --help

`pcluster version`에 대한 도움말 텍스트를 표시합니다.

명령이 호출되고 해당 호출의 상태에 대한 폴링을 시작하면 안전하게 "Ctrl-C"를 사용하여 종료할 수 있습니다. `pcluster status mycluster`를 호출하여 현재 상태 보기로 돌아갈 수 있습니다.

```
$ pcluster version
2.11.7
```

pcluster-config

AWS ParallelCluster 구성 파일을 업데이트합니다.

```
pcluster-config [ -h ] [convert]
```

명령별 플래그를 보려면 `pcluster-config [command] -h`를 실행하세요.

이름 지정된 인수

`-h, --help`

`pcluster-config`에 대한 도움말 텍스트를 표시합니다.

> **Note**
>
> `pcluster-config` 명령이 AWS ParallelCluster 버전 2.9.0에 추가되었습니다.

하위 명령

`pcluster-config convert`

```
pcluster-config convert [ -h ] [ -c CONFIG_FILE ] [ -t CLUSTER_TEMPLATE ]
                        [ -o OUTPUT_FILE ]
```

이름 지정된 인수

`-h, --help`

`pcluster-config convert`에 대한 도움말 텍스트를 표시합니다.

`-c CONFIG_FILE, --config-file CONFIG_FILE`

읽을 구성 파일의 경로를 지정합니다.

기본값은 `~/.parallelcluster/config`입니다.

자세한 내용은 구성 AWS ParallelCluster 단원을 참조하십시오.

`-t CLUSTER_TEMPLATE, --cluster-template CLUSTER_TEMPLATE`

사용할 `[cluster]` 섹션 항목을 나타냅니다. 이 인수를 지정하지 않으면 `pcluster-config convert`는 `[global]` 섹션의 `cluster_template` 설정을 사용합니다. 지정되지 않은 경우 `[cluster default]` 섹션이 사용됩니다.

-o *OUTPUT_FILE*, --output *OUTPUT_FILE*

기록할 변환된 구성 파일의 경로를 지정합니다. 기본적으로 출력은 STDOUT에 기록됩니다.

예시

```
$ pcluster-config convert -t alpha -o ~/.parallelcluster/multiinstance
```

~/.parallelcluster/config의 [cluster alpha] 섹션에 지정된 클러스터 구성을 변환하면서, 변환된 구성 파일을 ~/.parallelcluster/multiinstance에 씁니다.

구성

기본적으로는 모든 구성 파라미터에 `~/.parallelcluster/config` 파일을 AWS ParallelCluster 사용합니다. 사용자 지정 구성 파일을 `-c` 또는 `--config` 명령줄 옵션이나 `AWS_PCLUSTER_CONFIG_FILE` 환경 변수를 사용하여 지정할 수 있습니다.

예제 구성 파일은의 Python 디렉터리 AWS ParallelCluster 에와 함께 설치됩니다`site-packages/aws-parallelcluster/examples/config`. 예제 구성 파일은 GitHub(https://github.com/aws/aws-parallelcluster/blob/v2.11.9/cli/src/pcluster/examples/config)에서도 사용할 수 있습니다.

현재 AWS ParallelCluster 2 버전: 2.11.9.

주제

- 레이아웃
- [global] 섹션
- [aws] 섹션
- [aliases] 섹션
- [cluster] 섹션
- [compute_resource] 섹션
- [cw_log] 섹션
- [dashboard] 섹션
- [dcv] 섹션
- [ebs] 섹션
- [efs] 섹션
- [fsx] 섹션
- [queue] 섹션
- [raid] 섹션
- [scaling] 섹션
- [vpc] 섹션
- 예시

레이아웃

AWS ParallelCluster 구성은 여러 섹션에 정의되어 있습니다.

다음 [global] 섹션 및 [aws] 섹션이 필요합니다.

최소 [cluster] 섹션 및 [vpc] 섹션이 하나씩 포함되어야 합니다.

한 섹션은 대괄호 안의 섹션 이름으로 시작하며 뒤에 파라미터와 구성이 옵니다.

```
[global]
cluster_template = default
update_check = true
sanity_check = true
```

[global] 섹션

pcluster와 관련된 전역 구성 옵션을 지정합니다.

```
[global]
```

주제

- cluster_template
- update_check
- sanity_check

cluster_template

클러스터에 기본적으로 사용되는 cluster 섹션의 이름을 정의합니다. cluster 섹션에 대한 자세한 내용은 [cluster] 섹션을 참조하세요. 클러스터 이름은 문자로 시작해야 하고, 60자 이하여야 하고, 문자, 숫자 및 하이픈(-)만 포함해야 합니다.

예를 들어, 다음 설정은 [cluster default]를 시작하는 섹션이 기본적으로 사용되도록 지정합니다.

```
cluster_template = default
```

업데이트 정책: 이 설정은 업데이트 중에 분석되지 않습니다.

update_check

(선택 사항) pcluster에 대한 업데이트가 있는지 확인합니다.

기본값은 true입니다.

```
update_check = true
```

업데이트 정책: 이 설정은 업데이트 중에 분석되지 않습니다.

sanity_check

(선택 사항) 클러스터 파라미터에 정의된 리소스의 구성을 확인하려고 시도합니다.

기본값은 true입니다.

> **Warning**
>
> sanity_check를 false로 설정하면 중요한 검사를 건너뛰게 됩니다. 이로 인해 기능이 의도한 대로 작동하지 않을 수 있습니다.

```
sanity_check = true
```

> **Note**
>
> AWS ParallelCluster 버전 2.5.0 이전, sanity_check 기본값은 입니다false.

업데이트 정책: 이 설정은 업데이트 중에 분석되지 않습니다.

[aws] 섹션

(선택 사항)를 선택하는 데 사용됩니다 AWS 리전.

클러스터 생성은이 우선 순위 순서를 사용하여 새 클러스터의 AWS 리전 를 선택합니다.

1. pcluster create의 -r 또는 --region 파라미터.
2. AWS_DEFAULT_REGION 환경 변수.
3. aws_region_name AWS ParallelCluster 구성 파일의 [aws] 섹션에서 설정(기본 위치는 ~/.parallelcluster/config) pcluster configure 명령으로 업데이트된 위치입니다.
4. region AWS CLI 구성 파일(~/.aws/config)의 [default] 섹션에서 설정

> **Note**
>
> AWS ParallelCluster 버전 2.10.0 이전에는 이러한 설정이 필요했으며 모든 클러스터에 적용되었습니다.

보안 인증을 저장하려면 AWS ParallelCluster 구성 파일에 보안 인증을 저장하는 대신 환경, Amazon EC2의 IAM 역할 또는 AWS CLI를 사용할 수 있습니다.

```
[aws]
aws_region_name = Region
```

업데이트 정책: 이 설정은 업데이트 중에 분석되지 않습니다.

[aliases] 섹션

별칭을 지정하고 ssh 명령을 사용자 지정할 수 있습니다.

다음 기본 설정을 참조하세요.

- CFN_USER는 OS의 기본 사용자 이름으로 설정됩니다.
- MASTER_IP는 헤드 노드의 IP 주소로 설정되어 있습니다.
- ARGS는 사용자가 pcluster ssh cluster_name 다음에 제공하는 인수로 설정됩니다.

```
[aliases]
# This is the aliases section, you can configure
# ssh alias here
ssh = ssh {CFN_USER}@{MASTER_IP} {ARGS}
```

업데이트 정책: 이 설정은 업데이트 중에 분석되지 않습니다.

[cluster] 섹션

클러스터를 생성하는 데 사용할 수 있는 클러스터 템플릿을 정의합니다. 구성 파일은 여러 [cluster] 섹션을 포함할 수 있습니다.

동일한 클러스터 템플릿을 사용하여 여러 클러스터를 만들 수 있습니다.

형식은 [cluster *cluster-template-name*]입니다. [global] 섹션의 cluster_template 설정으로 이름이 지정된 [cluster] 섹션이 기본적으로 사용되지만 pcluster 명령줄에서 재정의할 수 있습니다.

*cluster-template-name*은 문자로 시작해야 하고, 30자 이내로 문자, 숫자, 하이픈(-) 및 밑줄(_)만 포함되어야 합니다.

```
[cluster default]
```

주제

- 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

(선택 사항) 클러스터와 함께 시작할 추가 AWS CloudFormation 템플릿을 정의합니다. 이 추가 템플릿은 클러스터 외부에 있지만 클러스터 수명 주기의 일부인 리소스를 생성하는 데 사용됩니다.

해당 값은 모든 파라미터가 제공된 퍼블릭 템플릿의 HTTP URL이어야 합니다.

기본값이 없습니다.

```
additional_cfn_template = https://<bucket-name>.s3.amazonaws.com/my-cfn-template.yaml
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

additional_iam_policies

(선택 사항) Amazon EC2에 대한 IAM 정책의 Amazon 리소스 이름(ARN) 목록을 쉼표로 구분하여 지정합니다. 이 목록은 쉼표로 구분된 AWS ParallelCluster 에서 필요한 권한 외에 클러스터에서 사용되는 루트 역할에 연결됩니다. IAM 정책 이름과 해당 ARN은 서로 다릅니다. 이름은 additional_iam_policies의 인수로 사용할 수 없습니다.

클러스터 노드의 기본 설정에 추가 정책을 추가하려는 경우 ec2_iam_role 설정을 사용하여 특정 EC2 정책을 추가하는 대신 additional_iam_policies 설정과 함께 추가 사용자 지정 IAM 정책을 전달하는 것이 좋습니다. additional_iam_policies가 AWS ParallelCluster 필요한 기본 권한에 추가되기 때문입니다. 기존 ec2_iam_role에는 필요한 모든 권한이 포함되어야 합니다. 그러나 특성이 추가됨에 따라 필요한 권한이 릴리스마다 변경되는 경우가 많기 때문에 기존 ec2_iam_role는 더 이상 사용되지 않을 수 있습니다.

기본값이 없습니다.

```
additional_iam_policies = arn:aws:iam::123456789012:policy/CustomEC2Policy
```

> **Note**
>
> additional_iam_policies에 대한 지원이 AWS ParallelCluster 버전 2.5.0에서 추가되었습니다.

업데이트 정책: 업데이트 중에 이 설정을 변경할 수 있습니다.

base_os

(필수) 클러스터에서 사용되는 OS 유형을 지정합니다.

사용 가능한 옵션은 다음과 같습니다.

- alinux2
- centos7
- ubuntu1804
- ubuntu2004

> **Note**
>
> AWS Graviton 기반 인스턴스의 경우 , ubuntu1804또는 alinux2만 지원ubuntu2004됩니다.

> **Note**
>
> AWS ParallelCluster 버전 2.11.4에서에 대한 지원이 제거centos8되었습니다. AWS ParallelCluster 버전 2.11.0에서 alinux 및 ubuntu1604에 대한 지원이 제거되었으며 ubuntu2004에 대한 지원이 추가되었습니다. AWS ParallelCluster 버전 centos8 2.10.0에서에 대한 지원이 추가되고에 대한 지원이 제거centos6되었습니다. AWS ParallelCluster 버전 2.6.0에서 alinux2에 대한 지원이 추가되었습니다. AWS ParallelCluster 버전 2.5.0에서 ubuntu1404에 대한 지원은 제거되었으며 ubuntu1804에 대한 지원이 추가되었습니다.

를 지원하지 않는 다음 표에 AWS 리전 언급된 특정가 아닌 다른 centos7. 다른 모든 AWS 상용 리전은 다음 운영 체제를 모두 지원합니다.

파티션(AWS 리전)	**alinux2**	**centos7**	**ubuntu1804** 및 **ubuntu2004**
상용(모두 특별히 언급 AWS 리전 되지 않음)	True	True	True
AWS GovCloud(미국 동부)(us-gov-east-1)	True	False	True
AWS GovCloud(미국 서부)(us-gov-west-1)	True	False	True
중국(베이징)(cn-north-1)	True	False	True
중국(닝샤)(cn-northwest-1)	True	False	True

> **Note**
>
> base_os 파라미터는 클러스터에 로그인하는 데 사용되는 사용자 이름도 결정합니다.

- centos7: centos
- ubuntu1804 및 ubuntu2004: ubuntu
- alinux2: ec2-user

> **Note**
>
> AWS ParallelCluster 버전 2.7.0 이전에는 base_os 파라미터가 선택 사항이었고 기본값은 이었습니다alinux. AWS ParallelCluster 버전 2.7.0부터 base_os 파라미터가 필요합니다.

> **Note**
>
> scheduler 파라미터가 awsbatch인 경우 alinux2만 지원됩니다.

```
base_os = alinux2
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

cluster_resource_bucket

(선택 사항) 클러스터를 생성할 때 생성되는 리소스를 호스트하는 데 사용되는 Amazon S3 버킷의 이름을 지정합니다. 버킷에 버전 관리를 활성화해야 합니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 버전 관리 사용을 참조하세요. 이 버킷은 여러 클러스터에 사용할 수 있습니다. 버킷이 클러스터와 동일한 리전에 있어야 합니다.

이 파라미터를 지정하지 않으면 클러스터를 생성할 때 새 버킷이 생성됩니다. 새 버킷의 이름은 parallelcluster-*random_string*입니다. 이 이름에서 *random_string*은 임의의 영숫자 문자열입니다. 모든 클러스터 리소스는 형식의 경로에 있는이 버킷에 저장됩니다*bucket_name*/*resource_directory*. resource_directory에는 형식이 있습니다. *stack_name*-*random_string*여기서 *stack_name*은에서 사용하는 AWS CloudFormation 스택 중 하나의 이름입니다 AWS ParallelCluster. *bucket_name*의 값은 parallelcluster-*clustername* 스택 출력의 ResourcesS3Bucket 값에서 찾을 수 있습니다. *resource_directory* 값은 동일한 스택의 ArtifactS3RootDirectory 출력 값에서 찾을 수 있습니다.

기본값은 parallelcluster-*random_string*입니다.

```
cluster_resource_bucket = amzn-s3-demo-bucket
```

> **Note**
>
> AWS ParallelCluster 버전 2.10.0에서에 대한 지원이 cluster_resource_bucket 추가되었습니다.

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다. 이 설정은 강제로 업데이트할 수 없습니다.

cluster_type

(선택 사항) 시작할 클러스터의 유형을 정의합니다. queue_settings 설정이 정의된 경우 이 설정을 [queue] 섹션의 compute_type 설정으로 바꿔야 합니다.

유효한 옵션은 ondemand 및 spot입니다.

기본값은 ondemand입니다.

스팟 인스턴스에 대한 자세한 내용은 스팟 인스턴스 작업 섹션을 참조하세요.

> **Note**
>
> 스팟 인스턴스를 사용하려면 계정에 AWSServiceRoleForEC2Spot 서비스 연결 역할이 있어야 합니다. 를 사용하여 계정에서이 역할을 생성하려면 다음 명령을 AWS CLI실행합니다.
>
> ```
> aws iam create-service-linked-role --aws-service-name spot.amazonaws.com
> ```
>
> 자세한 내용은 Amazon EC2 사용 설명서에서 스팟 인스턴스 요청을 위한 서비스 연결 역할을 참조하세요.

```
cluster_type = ondemand
```

업데이트 정책: 업데이트를 위해 이 설정을 변경하려면 컴퓨팅 플릿을 중지해야 합니다.

compute_instance_type

(선택 사항) 클러스터 컴퓨팅 노드에 사용되는 Amazon EC2 인스턴스 유형을 정의합니다. 인스턴스 유형의 아키텍처는 master_instance_type 설정에 사용된 아키텍처와 동일해야 합니다. queue_settings 설정이 정의된 경우 이 설정을 [compute_resource] 섹션의 instance_type 설정으로 바꿔야 합니다.

awsbatch 스케줄러를 사용하는 경우 지원되는 인스턴스 유형 목록은 AWS Batch UI의 컴퓨팅 환경 생성을 참조하세요.

스케줄러가 awsbatch인 경우 기본적으로 t2.micro, optimal로 설정됩니다.

```
compute_instance_type = t2.micro
```

> **Note**
>
> AWS Graviton 기반 인스턴스(A1 및 C6g 인스턴스 포함)에 대한 지원이 AWS ParallelCluster 버전 2.8.0에 추가되었습니다.

업데이트 정책: 업데이트를 위해 이 설정을 변경하려면 컴퓨팅 플릿을 중지해야 합니다.

compute_root_volume_size

(선택 사항) ComputeFleet 루트 볼륨 크기를 기비바이트(GiB) 단위로 지정합니다. AMI는 `growroot`를 지원해야 합니다.

기본값은 35입니다.

> **Note**
>
> 2.5.0~2.10.4 AWS ParallelCluster 버전의 경우 기본값은 25였습니다. AWS ParallelCluster 버전 2.5.0 이전의 기본값은 20이었습니다.

```
compute_root_volume_size = 35
```

업데이트 정책: 업데이트를 위해 이 설정을 변경하려면 컴퓨팅 플릿을 중지해야 합니다.

custom_ami

(선택 사항) 기본 게시된 AMI 대신 헤드 및 컴퓨팅 노드에 사용할 사용자 지정 AMI의 ID를 지정합니다. 자세한 내용은 AMI 수정 또는 사용자 지정 AWS ParallelCluster AMI 빌드을 참조하세요.

기본값이 없습니다.

```
custom_ami = ami-00d4efc81188687a0
```

사용자 지정 AMI를 시작하기 위해 추가 권한이 필요한 경우 이러한 권한을 사용자 및 헤드 노드 정책 모두에 추가해야 합니다.

예를 들어 사용자 지정 AMI에 암호화된 스냅샷이 연결된 경우 사용자 및 헤드 노드 정책 모두에 다음과 같은 추가 정책이 필요합니다.

```
{
   "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>"
            ]
        }
    ]
}
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

cw_log_settings

(선택 사항) CloudWatch Logs 구성이 있는 [cw_log] 섹션을 식별합니다. 섹션 이름은 문자로 시작해야 하고, 30자 이내로 문자, 숫자, 하이픈(-) 및 밑줄(_)만 포함되어야 합니다.

자세한 내용은 [cw_log] 섹션, Amazon CloudWatch 대시보드 및 Amazon CloudWatch Logs와 통합을 참조하세요.

예를 들어, 다음 설정은 [cw_log custom-cw]을 시작하는 섹션이 CloudWatch Logs 구성에 사용되도록 지정합니다.

```
cw_log_settings = custom-cw
```

> **Note**
>
> AWS ParallelCluster 버전 2.6.0에서에 대한 지원이 cw_log_settings 추가되었습니다.

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

dashboard_settings

(선택 사항) CloudWatch 대시보드 구성을 사용하여 [dashboard] 섹션을 식별합니다. 섹션 이름은 문자로 시작해야 하고, 30자 이내로 문자, 숫자, 하이픈(-) 및 밑줄(_)만 포함되어야 합니다.

자세한 내용은 [dashboard] 섹션을 참조하십시오.

예를 들어, 다음 설정은 [dashboard custom-dashboard을 시작하는 섹션이 CloudWatch 대시보드 구성에 사용되도록 지정합니다.

```
dashboard_settings = custom-dashboard
```

> **Note**
>
> AWS ParallelCluster 버전 2.10.0에서에 대한 지원이 dashboard_settings 추가되었습니다.

업데이트 정책: 업데이트 중에 이 설정을 변경할 수 있습니다.

dcv_settings

(선택 사항) Amazon DCV 구성이 있는 [dcv] 섹션을 식별합니다. 섹션 이름은 문자로 시작해야 하고, 30자 이내로 문자, 숫자, 하이픈(-) 및 밑줄(_)만 포함되어야 합니다.

자세한 내용은 [dcv] 섹션을 참조하십시오.

예를 들어, 다음 설정은 [dcv custom-dcv]를 시작하는 섹션이 Amazon DCV 구성에 사용되도록 지정합니다.

```
dcv_settings = custom-dcv
```

> **Note**
>
> AWS Graviton 기반 인스턴스에서는 Amazon DCV가 에서만 지원됩니다alinux2.

> **Note**
>
> AWS ParallelCluster 버전 2.5.0에서에 대한 지원이 dcv_settings 추가되었습니다.

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

desired_vcpus

(선택 사항) 컴퓨팅 환경에 필요한 vCPU 수를 지정합니다. 스케줄러가 awsbatch인 경우에만 사용됩니다.

기본값은 4입니다.

```
desired_vcpus = 4
```

업데이트 정책: 이 설정은 업데이트 중에 분석되지 않습니다.

disable_cluster_dns

(선택 사항) 클러스터의 DNS 항목을 생성하지 않아야 하는지 여부를 지정합니다. 기본적으로 Route 53 호스팅 영역을 AWS ParallelCluster 생성합니다. disable_cluster_dns이 true로 설정되어 있으면 호스팅 영역이 생성되지 않습니다.

기본값은 false입니다.

```
disable_cluster_dns = true
```

> **Warning**
>
> 클러스터가 제대로 작동하려면 이름 확인 시스템이 필요합니다. disable_cluster_dns이 true로 설정된 경우 추가 이름 확인 시스템도 제공해야 합니다.

> **Important**
>
> disable_cluster_dns = true는 queue_settings 설정이 지정된 경우에만 지원됩니다.

> **Note**
>
> AWS ParallelCluster 버전 2.9.1에서에 대한 지원이 disable_cluster_dns 추가되었습니다.

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

disable_hyperthreading

(선택 사항) 헤드 및 컴퓨팅 노드에서 하이퍼 스레딩을 비활성화합니다. 모든 인스턴스 유형이 하이퍼 스레딩을 비활성화할 수 있는 것은 아닙니다. 하이퍼스레딩 비활성화를 지원하는 인스

턴스 유형 목록은 Amazon EC2 사용 설명서에서 인스턴스 유형별 각 CPU 코어의 CPU 코어 및 스레드를 참조하세요. queue_settings 설정이 정의되어 있으면 이 설정과 [queue] 섹션의 disable_hyperthreading 설정 중에 하나만 정의할 수 있습니다.

기본값은 false입니다.

```
disable_hyperthreading = true
```

> **Note**
>
> disable_hyperthreading은 scheduler = awsbatch일 때만 헤드 노드에 영향을 줍니다.

> **Note**
>
> AWS ParallelCluster 버전 2.5.0에서 disable_hyperthreading에 대한 지원이 추가되었습니다.

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

ebs_settings

(선택 사항) 헤드 노드에 탑재된 Amazon EBS 볼륨이 있는 [ebs] 섹션을 식별합니다. 여러 Amazon EBS 볼륨을 사용하는 경우 각 매개 변수를 쉼표로 구분하여 목록에 입력합니다. 섹션 이름은 문자로 시작해야 하고, 30자 이내로 문자, 숫자, 하이픈(-) 및 밑줄(_)만 포함되어야 합니다.

최대 5개의 추가 Amazon EBS 볼륨이 지원됩니다.

자세한 내용은 [ebs] 섹션을 참조하십시오.

예를 들어, 다음 설정은 [ebs custom1] 및 [ebs custom2]를 시작하는 섹션이 Amazon EBS 볼륨에 사용되도록 지정합니다.

```
ebs_settings = custom1, custom2
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

ec2_iam_role

(선택 사항) 클러스터에 있는 모든 인스턴스에 연결된 Amazon EC2에 대한 기존 IAM 역할의 이름을 정의합니다. IAM 역할 이름과 Amazon 리소스 이름(ARN)은 다릅니다. ARN은 ec2_iam_role의 인수로 사용할 수 없습니다.

이 옵션을 지정하면 additional_iam_policies 설정이 무시됩니다. 클러스터 노드의 기본 설정에 추가 정책을 추가하려는 경우, ec2_iam_role 설정을 사용하는 대신 additional_iam_policies 설정과 함께 추가 사용자 지정 IAM 정책을 전달하는 것이 좋습니다.

이 옵션을 지정하지 않으면 Amazon EC2의 기본 AWS ParallelCluster IAM 역할이 사용됩니다. 자세한 내용은 AWS Identity and Access Management 의 역할 AWS ParallelCluster 단원을 참조하십시오.

기본값이 없습니다.

```
ec2_iam_role = ParallelClusterInstanceRole
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

efs_settings

(선택 사항) Amazon EFS 파일 시스템과 관련된 설정을 지정합니다. 섹션 이름은 문자로 시작해야 하고, 30자 이내로 문자, 숫자, 하이픈(-) 및 밑줄(_)만 포함되어야 합니다.

자세한 내용은 [efs] 섹션을 참조하십시오.

예를 들어, 다음 설정은 [efs customfs]를 시작하는 섹션이 Amazon EFS 파일 시스템 구성에 사용되도록 지정합니다.

```
efs_settings = customfs
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

enable_efa

(선택 사항) 있는 경우 Elastic Fabric Adapter(EFA)가 컴퓨팅 노드에 대해 활성화되도록 지정합니다. EFA를 지원하는 EC2 인스턴스 목록을 보려면 Linux 인스턴스용 Amazon EC2 사용 설명서의 지원되는 인스턴스 유형을 참조하세요. 자세한 내용은 Elastic Fabric Adapter 단원을 참조하십시오. queue_settings 설정이 정의되어 있으면 이 설정과 [queue] 섹션의 enable_efa 설정 중에 하나

만 정의할 수 있습니다. 클러스터 배치 그룹은 인스턴스 간의 대기 시간을 최소화하기 위해 사용해야 합니다. 자세한 내용은 placement 및 placement_group 섹션을 참조하세요.

```
enable_efa = compute
```

> **Note**
>
> Arm 기반 Graviton2 인스턴스의 EFA에 대한 지원이 AWS ParallelCluster 버전 2.10.1에 추가되었습니다.

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

enable_efa_gdr

(선택 사항) AWS ParallelCluster 버전 2.11.3부터는이 설정이 적용되지 않습니다. GPUDirect RDMA(원격 직접 메모리 액세스)에 대한 Elastic Fabric Adapter(EFA) 지원은 인스턴스 유형과 운영 체제에서 모두 지원되는 경우 항상 활성화됩니다.

> **Note**
>
> AWS ParallelCluster 버전 2.10.0~2.11.2: compute인 경우 컴퓨팅 노드에 대해 GPUDirect RDMA(원격 직접 메모리 액세스)에 대한 EFA(Elastic Fabric Adapter) 지원이 활성화되도록 지정합니다. 이 설정을 compute로 설정하려면 enable_efa 설정을 compute로 설정해야 합니다. GPUDirect RDMA에 대한 EFA 지원은 특정 운영 체제(base_os가 alinux2, centos7, ubuntu1804, 또는 ubuntu2004)의 특정 인스턴스 유형(p4d.24xlarge)에서 지원됩니다. queue_settings 설정이 정의되어 있으면 이 설정과 [queue] 섹션의 enable_efa_gdr 설정 중에 하나만 정의할 수 있습니다. 클러스터 배치 그룹은 인스턴스 간의 대기 시간을 최소화하기 위해 사용해야 합니다. 자세한 내용은 placement 및 placement_group 섹션을 참조하세요.

```
enable_efa_gdr = compute
```

> **Note**
>
> AWS ParallelCluster 버전 2.10.0에서에 대한 지원이 enable_efa_gdr 추가되었습니다.

업데이트 정책: 업데이트를 위해 이 설정을 변경하려면 컴퓨팅 플릿을 중지해야 합니다.

enable_intel_hpc_platform

(선택 사항) 있는 경우 Intel Parallel Studio에 대한 최종 사용자 라이선스 계약에 동의했음을 나타냅니다. 이렇게 하면 Intel Parallel Studio가 헤드 노드에 설치되고 컴퓨팅 노드와 공유됩니다. 이로 인해 헤드 노드의 부트스트랩을 수행하는 데 걸리는 시간이 몇 분 더 추가됩니다. enable_intel_hpc_platform 설정은 CentOS 7(base_os = centos7)에서만 지원됩니다.

기본값은 false입니다.

```
enable_intel_hpc_platform = true
```

> **Note**
>
> enable_intel_hpc_platform 파라미터는 AWS Graviton 기반 인스턴스와 호환되지 않습니다.

> **Note**
>
> AWS ParallelCluster 버전 2.5.0에서 enable_intel_hpc_platform에 대한 지원이 추가되었습니다.

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

encrypted_ephemeral

(선택 사항) LUKS(Linux Unified Key Setup)를 사용하여 복구 불가능한 인 메모리 키로 임시 인스턴스 스토어 볼륨을 암호화합니다.

자세한 내용은 https://gitlab.com/cryptsetup/cryptsetup/blob/master/README.md 단원을 참조하십시오.

기본값은 false입니다.

```
encrypted_ephemeral = true
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

ephemeral_dir

(선택 사항) 인스턴스 스토어 볼륨이 사용되는 경우 탑재되는 경로를 정의합니다.

기본값은 /scratch입니다.

```
ephemeral_dir = /scratch
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

extra_json

(선택 사항) Chef dna.json에 병합되는 추가 JSON을 정의합니다. 자세한 내용은 사용자 지정 AWS ParallelCluster AMI 빌드 단원을 참조하십시오.

기본값은 {}입니다.

```
extra_json = {}
```

> **Note**
>
> AWS ParallelCluster 버전 2.6.1부터 노드를 시작할 때 시작 시간을 개선하기 위해 대부분의 설치 레시피를 기본적으로 건너뜁니다. 시작 시간을 희생하면서 이전 버전과의 호환성을 개선하기 위해 모든 설치 레시피를 실행하려면 extra_json 설정의 cluster 키에 "skip_install_recipes" : "no"를 추가하세요. 예시:
>
> ```
> extra_json = { "cluster" : { "skip_install_recipes" : "no" } }
> ```

업데이트 정책: 업데이트를 위해 이 설정을 변경하려면 컴퓨팅 플릿을 중지해야 합니다.

fsx_settings

(선택 사항) FSx for Lustre 구성을 정의하는 섹션을 지정합니다. 섹션 이름은 문자로 시작해야 하고, 30자 이내로 문자, 숫자, 하이픈(-) 및 밑줄(_)만 포함되어야 합니다.

자세한 내용은 [fsx] 섹션을 참조하십시오.

예를 들어, 다음 설정은 [fsx fs]를 시작하는 섹션이 FSx for Lustre 구성에 사용되도록 지정합니다.

```
fsx_settings = fs
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

iam_lambda_role

(선택 사항) 기존 AWS Lambda 실행 역할의 이름을 정의합니다. 이 역할은 클러스터의 모든 Lambda 함수에 연결됩니다. 자세한 내용은AWS Lambda 개발자 가이드의 AWS Lambda 실행 역할을 참조하세요.

> **Note**
>
> 버전 2.11.5부터 SGE 또는 Torque 스케줄러 사용을 지원하지 AWS ParallelCluster 않습니다.

IAM 역할 이름과 Amazon 리소스 이름(ARN)은 다릅니다. ARN은 `iam_lambda_role`의 인수로 사용할 수 없습니다. `ec2_iam_role`와 `iam_lambda_role` 가 모두 정의되고 `scheduler`이 `sge`, `slurm` 또는 `torque`인 경우 역할이 생성되지 않습니다. `scheduler`이 `awsbatch`인 경우 `pcluster start` 중에 역할이 생성됩니다. 예제 정책은 SGE, Slurm, 또는 Torque를 사용하는 `ParallelClusterLambdaPolicy` 및 `awsbatch`을 사용하는 `ParallelClusterLambdaPolicy` 항목을 참조하세요.

기본값이 없습니다.

```
iam_lambda_role = ParallelClusterLambdaRole
```

> **Note**
>
> AWS ParallelCluster 버전 2.10.1에서에 대한 지원이 `iam_lambda_role` 추가되었습니다.

업데이트 정책: 업데이트 중에 이 설정을 변경할 수 있습니다.

initial_queue_size

(선택 사항) 클러스터에서 컴퓨팅 노드로 시작할 초기 Amazon EC2 인스턴스 수를 설정합니다. `queue_settings` 설정이 정의된 경우 이 설정을 제거한 다음 `[compute_resource]` 섹션의 `initial_count` 설정으로 교체해야 합니다.

> **Note**
> 버전 2.11.5부터 SGE 또는 Torque 스케줄러 사용을 지원하지 AWS ParallelCluster 않습니다.

이 설정은 기존 스케줄러(SGE, Slurm 및 Torque)에만 적용할 수 있습니다.
`maintain_initial_size` 설정이 `true`인 경우 `initial_queue_size` 설정이 1 이상이어야 합니다.

스케줄러가 `awsbatch`인 경우 `min_vcpus`를 사용합니다.

기본값은 2입니다.

```
initial_queue_size = 2
```

업데이트 정책: 업데이트 중에 이 설정을 변경할 수 있습니다.

key_name

(선택 사항) 인스턴스에 대한 SSH 액세스를 활성화하는 기존 Amazon EC2 키 페어의 이름을 지정합니다.

```
key_name = mykey
```

> **Note**
> AWS ParallelCluster 버전 2.11.0 이전에는가 필수 설정`key_name`이었습니다.

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

maintain_initial_size

> **Note**
> 버전 2.11.5부터 SGE 또는 Torque 스케줄러 사용을 지원하지 AWS ParallelCluster 않습니다.

(선택 사항) 기존 스케줄러(SGE, Slurm 및 Torque)에 대한 오토 스케일링 그룹의 초기 크기를 유지합니다.

스케줄러가 awsbatch인 경우 desired_vcpus를 사용합니다.

이 설정은 부울 플래그입니다. true로 설정하면 오토 스케일링 그룹의 구성원 수는 initial_queue_size의 값보다 적지 않으며 initial_queue_size의 값은 1 이상이어야 합니다. 클러스터는 max_queue_size의 값으로 여전히 스케일 업할 수 있습니다. cluster_type = spot인 경우 오토 스케일링 그룹에서 인스턴스가 중단될 수 있으며 크기가 initial_queue_size 아래로 떨어질 수 있습니다.

false로 설정하면 필요 없는 리소스가 유휴 상태로 있지 않도록 오토 스케일링 그룹이 0개의 멤버로 스케일 다운될 수 있습니다.

queue_settings 설정이 정의된 경우 이 설정을 제거한 다음 [compute_resource] 섹션의 initial_count 및 min_count 설정으로 교체해야 합니다.

기본값은 false입니다.

```
maintain_initial_size = false
```

업데이트 정책: 업데이트 중에 이 설정을 변경할 수 있습니다.

master_instance_type

(선택 사항) 헤드 노드에 사용되는 Amazon EC2 인스턴스 유형을 정의합니다. 인스턴스 유형의 아키텍처는 compute_instance_type 설정에 사용된 아키텍처와 동일해야 합니다.

프리 티어 AWS 리전 가 있는에서는 기본적으로 프리 티어 인스턴스 유형(t2.micro 또는)으로 설정됩니다t3.micro. 프리 티어 AWS 리전 가 없는의 기본값은 입니다t3.micro. AWS 프리 티어에 대한 자세한 내용은 AWS 프리 티어 FAQs.

```
master_instance_type = t2.micro
```

> **Note**
>
> AWS ParallelCluster 버전 2.10.1 이전에는 기본적으로 모두 로 설정되어 t2.micro 있습니다 AWS 리전. AWS ParallelCluster 버전 2.10.0에서는 헤드 노드에 p4d.24xlarge가 지원되지 않았습니다. AWS Graviton 기반 인스턴스(예: A1 및 C6g)에 대한 지원이 AWS ParallelCluster 버전 2.8.0에 추가되었습니다.

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

master_root_volume_size

(선택 사항) 헤드 노드 루트 볼륨 크기를 기비바이트(GiB) 단위로 지정합니다. AMI는 growroot를 지원해야 합니다.

기본값은 35입니다.

> **Note**
>
> 2.5.0~2.10.4 AWS ParallelCluster 버전의 경우 기본값은 25였습니다. AWS ParallelCluster 버전 2.5.0 이전의 기본값은 20이었습니다.

```
master_root_volume_size = 35
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

max_queue_size

(선택 사항) 클러스터에서 시작할 수 있는 최대 Amazon EC2 인스턴스 수를 설정합니다. queue_settings 설정이 정의된 경우 이 설정을 제거한 다음 [compute_resource] 섹션의 max_count 설정으로 교체해야 합니다.

> **Note**
>
> 버전 2.11.5부터 SGE 또는 Torque 스케줄러 사용을 지원하지 AWS ParallelCluster 않습니다.

이 설정은 기존 스케줄러(SGE, Slurm 및 Torque)에만 적용할 수 있습니다.

스케줄러가 awsbatch인 경우 max_vcpus를 사용합니다.

기본값은 10입니다.

```
max_queue_size = 10
```

업데이트 정책: 이 설정은 업데이트 중에 변경할 수 있지만 값이 감소하면 컴퓨팅 플릿을 중지해야 합니다. 그렇지 않으면 기존 노드가 종료될 수 있습니다.

max_vcpus

(선택 사항) 컴퓨팅 환경의 최대 vCPU 수를 지정합니다. 스케줄러가 awsbatch인 경우에만 사용됩니다.

기본값은 20입니다.

```
max_vcpus = 20
```

업데이트 정책: 업데이트 중에는 이 설정을 줄일 수 없습니다.

min_vcpus

(선택 사항) awsbatch 스케줄러에 대한 오토 스케일링 그룹의 초기 크기를 유지합니다.

> **Note**
>
> 버전 2.11.5부터 SGE 또는 Torque 스케줄러 사용을 지원하지 AWS ParallelCluster 않습니다.

스케줄러가 SGE, Slurm 또는 Torque인 경우 maintain_initial_size를 사용합니다.

컴퓨팅 환경의 구성원은 min_vcpus 값보다 적을 수 없습니다.

기본값은 0입니다.

```
min_vcpus = 0
```

업데이트 정책: 업데이트 중에 이 설정을 변경할 수 있습니다.

placement

(선택 사항) 클러스터 배치 그룹 로직을 정의합니다. 이렇게 하면 전체 클러스터 또는 클러스터 배치 그룹을 사용할 컴퓨팅 인스턴스만 활성화됩니다.

queue_settings 설정이 정의된 경우 이 설정을 제거한 다음 각 [queue] 섹션에 대해 placement_group 설정으로 교체해야 합니다. 다른 인스턴스 유형에 동일한 배치 그룹을 사용하는 경우 용량 부족 오류로 인해 요청이 실패할 가능성이 높습니다. 자세한 내용은 Amazon EC2 사용 설명서의 부족한 인스턴스 용량을 참조하세요. 배치 그룹을 미리 생성하여 각 대기열의

placement_group 설정에 구성한 경우에만 여러 대기열이 공유할 수 있습니다. 각 [queue] 섹션이 placement_group 설정을 정의하는 경우 헤드 노드는 대기열의 배치 그룹에 속할 수 없습니다.

유효한 옵션은 cluster 또는 compute입니다.

스케줄러가 awsbatch인 경우에는 이 파라미터가 사용되지 않습니다.

기본값은 compute입니다.

```
placement = compute
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

placement_group

(선택 사항) 클러스터 배치 그룹을 정의합니다. queue_settings 설정이 정의된 경우 이 설정을 제거한 다음 [queue] 섹션의 placement_group 설정으로 교체해야 합니다.

유효한 옵션은 다음과 같습니다.

- DYNAMIC
- 기존의 Amazon EC2 클러스터 배치 그룹 이름입니다.

DYNAMIC으로 설정하면 고유의 배치 그룹이 클러스터 스택의 일부로 생성되고 삭제됩니다.

스케줄러가 awsbatch인 경우에는 이 파라미터가 사용되지 않습니다.

배치 그룹에 대한 자세한 내용은 Amazon EC2 사용 설명서의 배치 그룹을 참조하세요. 다른 인스턴스 유형에 동일한 배치 그룹을 사용하는 경우 용량 부족 오류로 인해 요청이 실패할 가능성이 높습니다. 자세한 내용은 Amazon EC2 사용 설명서의 부족한 인스턴스 용량을 참조하세요.

기본값이 없습니다.

모든 인스턴스 유형이 클러스터 배치 그룹을 지원하는 것은 아닙니다. 예를 들어, t3.micro의 기본 인스턴스 유형은 클러스터 배치 그룹을 지원하지 않습니다. 클러스터 배치 그룹을 지원하는 인스턴스 유형에 대한 자세한 내용은 Amazon EC2 사용 설명서의 클러스터 배치 그룹의 규칙 및 제한 사항을 참조하세요. 배치 그룹 작업 시 팁은 배치 그룹 및 인스턴스 시작 문제 섹션을 참고하세요.

```
placement_group = DYNAMIC
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

post_install

(선택 사항) 모든 노드 부트스트랩 작업이 완료된 후 실행되는 설치 후 스크립트의 URL을 지정합니다. 자세한 내용은 사용자 지정 부트스트랩 작업 단원을 참조하십시오.

awsbatch를 스케줄러로 사용할 경우 사후 설치 스크립트는 헤드 노드에서만 실행됩니다.

파라미터 형식을 http://*hostname/path/to/script.sh* 또는 s3://*bucket-name/path/to/script.sh*로 지정할 수 있습니다.

기본값이 없습니다.

```
post_install = s3://<bucket-name>/my-post-install-script.sh
```

업데이트 정책: 업데이트를 위해 이 설정을 변경하려면 컴퓨팅 플릿을 중지해야 합니다.

post_install_args

(선택 사항) 사후 설치 스크립트에 전달할 인수의 목록을 따옴표로 묶어 지정합니다.

기본값이 없습니다.

```
post_install_args = "argument-1 argument-2"
```

업데이트 정책: 업데이트를 위해 이 설정을 변경하려면 컴퓨팅 플릿을 중지해야 합니다.

pre_install

(선택 사항) 노드 배포 부트스트랩 작업이 시작되기 전에 실행되는 사전 설치 스크립트의 URL을 지정합니다. 자세한 내용은 사용자 지정 부트스트랩 작업 단원을 참조하십시오.

awsbatch를 스케줄러로 사용할 경우 사전 설치 스크립트는 헤드 노드에서만 실행됩니다.

파라미터 형식을 http://*hostname/path/to/script.sh* 또는 s3://*bucket-name/path/to/script.sh*로 지정할 수 있습니다.

기본값이 없습니다.

```
pre_install = s3://bucket-name/my-pre-install-script.sh
```

업데이트 정책: 업데이트를 위해 이 설정을 변경하려면 컴퓨팅 플릿을 중지해야 합니다.

pre_install_args

(선택 사항) 사전 설치 스크립트에 전달할 인수의 목록을 따옴표로 묶어 지정합니다.

기본값이 없습니다.

```
pre_install_args = "argument-3 argument-4"
```

업데이트 정책: 업데이트를 위해 이 설정을 변경하려면 컴퓨팅 플릿을 중지해야 합니다.

proxy_server

(선택 사항) HTTP 또는 HTTPS 프록시 서버, 일반적으로 `http://x.x.x.x:8080`을 정의합니다.

기본값이 없습니다.

```
proxy_server = http://10.11.12.13:8080
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

queue_settings

(선택 사항) 클러스터가 동종 컴퓨팅 플릿 대신 대기열을 사용하도록 지정하고 사용되는 [queue] 섹션을 지정합니다. 나열된 첫 번째 [queue] 섹션은 기본 스케줄러 대기열입니다. queue 섹션 이름은 소문자로 시작해야 하고 30자 이내로 소문자, 숫자 및 하이픈(-)만 포함되어야 합니다.

> **Important**
>
> queue_settings은 scheduler가 slurm로 설정된 경우에만 지원됩니다. cluster_type, compute_instance_type, initial_queue_size, maintain_initial_size, max_queue_size, placement, placement_group 및 spot_price 설정은 지정하지 않아야 합니다. disable_hyperthreading 및 enable_efa 설정은[cluster] 섹션 또는 [queue] 섹션에서 지정할 수 있지만 둘 다에서 지정할 수는 없습니다.

[queue] 섹션은 최대 5개까지 지원됩니다.

자세한 내용은 [queue] 섹션을 참조하십시오.

예를 들어, 다음 설정은 [queue q1] 및 [queue q2]을 시작하는 섹션을 지정합니다.

```
queue_settings = q1, q2
```

> **Note**
>
> AWS ParallelCluster 버전 2.9.0에서에 대한 지원이 queue_settings 추가되었습니다.

업데이트 정책: 업데이트를 위해 이 설정을 변경하려면 컴퓨팅 플릿을 중지해야 합니다.

raid_settings

(선택 사항) Amazon EBS 볼륨 RAID 구성이 있는 [raid] 섹션을 식별합니다. 섹션 이름은 문자로 시작해야 하고, 30자 이내로 문자, 숫자, 하이픈(-) 및 밑줄(_)만 포함되어야 합니다.

자세한 내용은 [raid] 섹션을 참조하십시오.

예를 들어, 다음 설정은 [raid rs]를 시작하는 섹션이 Auto Scaling 구성에 사용되도록 지정합니다.

```
raid_settings = rs
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

s3_read_resource

(선택 사항) AWS ParallelCluster 노드에 읽기 전용 액세스 권한이 부여된 Amazon S3 리소스를 지정합니다.

예를 들어, arn:aws:s3:::*my_corporate_bucket**은 *my_corporate_bucket* 버킷과 그 안에 있는 모든 객체에 대한 읽기 전용 액세스를 제공합니다.

형식에 대한 자세한 내용은 Amazon S3 작업을 참조하세요.

기본값이 없습니다.

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

업데이트 정책: 업데이트 중에 이 설정을 변경할 수 있습니다.

s3_read_write_resource

(선택 사항) AWS ParallelCluster 노드에 읽기/쓰기 액세스를 부여할 Amazon S3 리소스를 지정합니다.

예를 들어, `arn:aws:s3:::my_corporate_bucket/Development/*`는 *my_corporate_bucket* 버킷의 `Development` 폴더에 있는 모든 객체에 대한 읽기/쓰기 액세스를 제공합니다.

형식에 대한 자세한 내용은 Amazon S3 작업을 참조하세요.

기본값이 없습니다.

```
s3_read_write_resource = arn:aws:s3:::my_corporate_bucket/*
```

업데이트 정책: 업데이트 중에 이 설정을 변경할 수 있습니다.

scaling_settings

Auto Scaling 구성이 있는 `[scaling]` 섹션을 식별합니다. 섹션 이름은 문자로 시작해야 하고, 30자 이내로 문자, 숫자, 하이픈(-) 및 밑줄(_)만 포함되어야 합니다.

자세한 내용은 `[scaling]` 섹션을 참조하십시오.

예를 들어, 다음 설정은 `[scaling custom]`를 시작하는 섹션이 Auto Scaling 구성에 사용되도록 지정합니다.

```
scaling_settings = custom
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

scheduler

(필수) 클러스터 스케줄러를 정의합니다.

유효한 옵션은 다음과 같습니다.

`awsbatch`

AWS Batch

`awsbatch` 스케줄러에 대한 자세한 내용은 네트워킹 설정 및 AWS Batch (awsbatch)을 참조하세요.

`sge`

> **Note**
>
> 버전 2.11.5부터 SGE 또는 Torque 스케줄러 사용을 지원하지 AWS ParallelCluster 않습니다.

Son of Grid Engine (SGE)

`slurm`

Slurm Workload Manager (Slurm)

`torque`

> **Note**
>
> 버전 2.11.5부터 SGE 또는 Torque 스케줄러 사용을 지원하지 AWS ParallelCluster 않습니다.

Torque Resource Manager (Torque)

> **Note**
>
> AWS ParallelCluster 버전 2.7.0 이전에는 `scheduler` 파라미터가 선택 사항이었고 기본값은 이었습니다`sge`. AWS ParallelCluster 버전 2.7.0부터는 `scheduler` 파라미터가 필요합니다.

```
scheduler = slurm
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

shared_dir

(선택 사항) 공유 Amazon EBS 볼륨이 탑재될 경로를 정의합니다.

다중 Amazon EBS 볼륨에는 이 옵션을 사용하지 마세요. 대신 각 [ebs] 섹션에 shared_dir 값을 제공합니다.

여러 Amazon EBS 볼륨을 사용하는 방법에 대한 자세한 내용은 [ebs] 섹션을 참조하세요.

기본값은 /shared입니다.

다음 예제에서는 /myshared에 탑재된 공유 Amazon EBS 볼륨을 보여줍니다.

```
shared_dir = myshared
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

spot_bid_percentage

(선택 사항) awsbatch가 스케줄러인 경우 ComputeFleet의 최고 스팟 가격을 계산하는 데 사용되는 온디맨드 비율을 설정합니다.

값을 지정하지 않으면 온디맨드 가격 이하의 현재 스팟 시장 가격이 선택됩니다.

```
spot_bid_percentage = 85
```

업데이트 정책: 업데이트 중에 이 설정을 변경할 수 있습니다.

spot_price

> **Note**
>
> 버전 2.11.5부터 SGE 또는 Torque 스케줄러 사용을 지원하지 AWS ParallelCluster 않습니다.

(선택 사항) 기존 스케줄러(SGE, Slurm 및 Torque)에서 ComputeFleet에 대한 최고 스팟 가격을 설정할 수 있습니다. cluster_type 설정이 spot으로 설정된 경우에만 사용됩니다. 값을 지정하지 않으면 온디맨드 가격 이하의 스팟 가격이 부과됩니다. queue_settings 설정이 정의된 경우 이 설정을 제거한 다음 [compute_resource] 섹션의 spot_price 설정으로 교체해야 합니다.

스케줄러가 awsbatch인 경우 spot_bid_percentage를 사용합니다.

필요에 맞는 입찰 스팟 인스턴스를 찾는 데 도움이 필요한 경우 스팟 인스턴스 어드바이저를 참조하세요.

```
spot_price = 1.50
```

> **Note**
>
> AWS ParallelCluster 버전 2.5.0에서는 spot_price를 지정cluster_type = spot하지 않으면 ComputeFleet의 인스턴스 시작이 실패합니다. 이는 AWS ParallelCluster 버전 2.5.1에서 수정되었습니다.

업데이트 정책: 업데이트 중에 이 설정을 변경할 수 있습니다.

tags

(선택 사항) 사용할 태그를 정의합니다 AWS CloudFormation.

--tags를 통해 명령줄 태그를 지정하면 이러한 태그는 구성 태그와 병합됩니다.

명령줄 태그는 동일한 키가 있는 구성 태그를 덮어씁니다.

태그는 JSON 형식입니다. 중괄호 밖에 따옴표를 사용하지 마세요.

자세한 내용을 알아보려면AWS CloudFormation 사용 설명서의 AWS CloudFormation 리소스 태그 유형을 참조하세요.

```
tags = {"key" : "value", "key2" : "value2"}
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

> **Note**
>
> 업데이트 정책에서는 AWS ParallelCluster 버전 2.8.0에서 버전 2.9.1까지의 tags 설정 변경을 지원하지 않았습니다.
> 버전 2.10.0에서 버전 2.11.7의 경우 tags 설정 변경을 지원하는 나열된 업데이트 정책이 정확하지 않습니다. 이 설정을 수정할 때는 클러스터 업데이트가 지원되지 않습니다.

template_url

(선택 사항) 클러스터를 생성하는 데 사용되는 AWS CloudFormation 템플릿의 경로를 정의합니다.

업데이트는 원래 스택을 생성하는 데 사용된 템플릿을 사용합니다.

기본값은 https://*aws_region_name*-aws-parallelcluster.s3.amazonaws.com/templates/aws-parallelcluster-*version*.cfn.json입니다.

> **Warning**
> 이것은 고급 파라미터입니다. 이 설정을 변경할 때는 사용자의 주의가 필요합니다.

```
template_url = https://us-east-1-aws-parallelcluster.s3.amazonaws.com/templates/aws-parallelcluster-2.11.9.cfn.json
```

업데이트 정책: 이 설정은 업데이트 중에 분석되지 않습니다.

vpc_settings

(필수) 클러스터가 배포되는 Amazon VPC 구성이 있는 [vpc] 섹션을 식별합니다. 섹션 이름은 문자로 시작해야 하고, 30자 이내로 문자, 숫자, 하이픈(-) 및 밑줄(_)만 포함되어야 합니다.

자세한 내용은 [vpc] 섹션을 참조하십시오.

예를 들어, 다음 설정은 [vpc public]을 시작하는 섹션이 Amazon VPC 구성에 사용되도록 지정합니다.

```
vpc_settings = public
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

[compute_resource] 섹션

컴퓨팅 리소스의 구성 설정을 정의합니다. [queue] 섹션의 compute_resource_settings 설정은 [compute_resource] 섹션을 참조합니다. [compute_resource] 섹션은 scheduler이 slurm로 설정된 경우에만 지원됩니다.

형식은 [compute_resource *<compute-resource-name>*]입니다. *compute-resource-name*은 문자로 시작해야 하고, 30자 이하여야 하며, 문자, 숫자, 하이픈(-) 및 밑줄(_)만 포함되어야 합니다.

```
[compute_resource cr1]
instance_type = c5.xlarge
min_count = 0
```

```
initial_count = 2
max_count = 10
spot_price = 0.5
```

> **Note**
>
> [compute_resource] 섹션 지원이 AWS ParallelCluster 버전 2.9.0에 추가되었습니다.

주제

- initial_count
- instance_type
- max_count
- min_count
- spot_price

initial_count

(선택 사항) 이 컴퓨팅 리소스에 대해 시작할 Amazon EC2 인스턴스의 초기 수를 설정합니다. 클러스터 생성은 최소한 이만큼의 노드가 컴퓨팅 리소스로 실행되기 전까지는 완료되지 않습니다. 대기열의 compute_type 설정이 spot이고 사용 가능한 스팟 인스턴스가 충분하지 않은 경우 클러스터 생성 시간이 초과되어 실패할 수 있습니다. min_count 설정보다 큰 개수는 동적 용량이 scaledown_idletime 설정의 영향을 받습니다. 이 설정은 initial_queue_size 설정을 대체합니다.

기본값은 0입니다.

```
initial_count = 2
```

업데이트 정책: 업데이트를 위해 이 설정을 변경하려면 컴퓨팅 플릿을 중지해야 합니다.

instance_type

(필수) 이 컴퓨팅 리소스에 사용되는 Amazon EC2 인스턴스 유형을 정의합니다. 인스턴스 유형의 아키텍처는 master_instance_type 설정에 사용된 아키텍처와 동일해야 합니다. instance_type 설정은 [queue] 섹션에서 참조하는 각 [compute_resource] 섹션마다 고유해야 합니다. 이 설정은 compute_instance_type 설정을 대체합니다.

```
instance_type = t2.micro
```

업데이트 정책: 업데이트를 위해 이 설정을 변경하려면 컴퓨팅 플릿을 중지해야 합니다.

max_count

(선택 사항) 이 컴퓨팅 리소스에서 시작할 수 있는 Amazon EC2 인스턴스의 최대 개수를 설정합니다. initial_count 설정보다 큰 개수는 전원 차단 모드에서 시작됩니다. 이 설정은 max_queue_size 설정을 대체합니다.

기본값은 10입니다.

```
max_count = 10
```

업데이트 정책: 대기열 크기를 현재 노드 수 미만으로 줄이려면 먼저 컴퓨팅 플릿을 중지해야 합니다.

> **Note**
>
> 업데이트 정책은 버전 2.0.0부터 AWS ParallelCluster 버전 2.9.1까지 컴퓨팅 플릿이 중지될 때까지 max_count 설정 변경을 지원하지 않았습니다.

min_count

(선택 사항) 이 컴퓨팅 리소스에서 시작할 수 있는 Amazon EC2 인스턴스의 최소 수를 설정합니다. 이러한 노드는 모두 정적 용량입니다. 클러스터 생성은 최소한 이 수만큼의 노드가 컴퓨팅 리소스로 실행되기 전까지는 완료되지 않습니다.

기본값은 0입니다.

```
min_count = 1
```

업데이트 정책: 대기열의 정적 노드 수를 줄이려면 먼저 컴퓨팅 플릿을 중지해야 합니다.

> **Note**
>
> 업데이트 정책은 버전 2.0.0부터 AWS ParallelCluster 버전 2.9.1까지 컴퓨팅 플릿이 중지될 때까지 min_count 설정 변경을 지원하지 않았습니다.

spot_price

(선택 사항) 이 컴퓨팅 리소스의 최대 스팟 가격을 설정합니다. 이 컴퓨팅 리소스가 포함된 대기열의 compute_type 설정이 spot로 설정된 경우에만 사용됩니다. 이 설정은 spot_price 설정을 대체합니다.

값을 지정하지 않으면 온디맨드 가격 이하의 스팟 가격이 부과됩니다.

필요에 맞는 입찰 스팟 인스턴스를 찾는 데 도움이 필요한 경우 스팟 인스턴스 어드바이저를 참조하세요.

```
spot_price = 1.50
```

업데이트 정책: 업데이트를 위해 이 설정을 변경하려면 컴퓨팅 플릿을 중지해야 합니다.

[cw_log] 섹션

CloudWatch Logs의 구성 설정을 정의합니다.

형식은 [cw_log *cw-log-name*]입니다. *cw-Log-name*은 문자로 시작해야 하고 30자 이하여야 하며 문자, 숫자, 하이픈(-) 및 밑줄(_)만 포함할 수 있습니다.

```
[cw_log custom-cw-log]
enable = true
retention_days = 14
```

자세한 내용은 Amazon CloudWatch Logs와 통합, Amazon CloudWatch 대시보드, Amazon CloudWatch Logs와 통합 섹션을 참조하세요.

> **Note**
>
> AWS ParallelCluster 버전 2.6.0에서에 대한 지원이 cw_log 추가되었습니다.

enable

(선택 사항) CloudWatch Logs가 활성화되었는지 여부를 나타냅니다.

기본값은 true입니다. false는 CloudWatch Logs를 비활성화하는 데 사용합니다.

다음 예는 CloudWatch Logs를 활성화합니다.

```
enable = true
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

retention_days

(선택 사항) CloudWatch Logs가 개별 로그 이벤트를 유지하는 일 수를 나타냅니다.

기본값은 14입니다. 지원되는 값은 1, 3, 5, 7, 14, 30, 60, 90, 120, 150, 180, 365, 400, 545, 731, 1827, 3653입니다.

다음 예에서는 30일 동안 로그 이벤트를 유지하도록 CloudWatch Logs를 구성합니다.

```
retention_days = 30
```

업데이트 정책: 업데이트 중에 이 설정을 변경할 수 있습니다.

[dashboard] 섹션

CloudWatch 대시보드의 구성 설정을 정의합니다.

형식은 [dashboard *dashboard-name*]입니다. *dashboard-name*은 문자로 시작해야 하고, 30자 이하여야 하며 문자, 숫자, 하이픈(-) 및 밑줄(_)만 포함할 수 있습니다.

```
[dashboard custom-dashboard]
enable = true
```

> **Note**
>
> AWS ParallelCluster 버전 2.10.0에서에 대한 지원이 dashboard 추가되었습니다.

enable

(선택 사항) CloudWatch 대시보드가 활성화되었는지 여부를 나타냅니다.

기본값은 true입니다. false는 CloudWatch 대시보드를 비활성화합니다.

다음 예는 CloudWatch 대시보드를 활성화합니다.

```
enable = true
```

업데이트 정책: 업데이트 중에 이 설정을 변경할 수 있습니다.

[dcv] 섹션

헤드 노드에서 실행 중인 Amazon DCV 서버에 대한 구성 설정을 정의합니다.

Amazon DCV 서버를 생성하고 구성하려면 dcv 섹션에서 정의한 이름으로 dcv_settings 클러스터를 지정하고, enable은 master로, base_os는 alinux2, centos7, ubuntu1804 또는 ubuntu2004로 설정합니다. 헤드 노드가 ARM 인스턴스인 경우 base_os는 alinux2, centos7 또는 ubuntu1804로 설정합니다.

형식은 [dcv *dcv-name*]입니다. *dcv-name*은 문자로 시작해야 하고, 30자 이하여야 하며 문자, 숫자, 하이픈(-) 및 밑줄(_)만 포함할 수 있습니다.

```
[dcv custom-dcv]
enable = master
port = 8443
access_from = 0.0.0.0/0
```

자세한 내용은 Amazon DCV를 통해 헤드 노드에 연결합니다. 섹션을 참조하세요.

Important

기본적으로의 Amazon DCV 포트 설정은 모든 IPv4 주소에 AWS ParallelCluster 열려 있습니다. 그러나 사용자는 Amazon DCV 세션에 대한 URL이 있는 경우에만 Amazon DCV 포트에 연결하고 pcluster dcv connect에서 URL이 반환된 후 30초 이내에 Amazon DCV 세션에 연결할 수 있습니다. access_from 설정을 사용해 CIDR 형식 IP 범위의 Amazon DCV 포트로의 액세스를 추가 제한하고, port 설정을 사용하여 비표준 포트를 설정합니다.

Note

centos8의 [dcv] 섹션에 대한 지원은 AWS ParallelCluster 버전 2.10.4에서 제거되었습니다. 의 [dcv] 섹션에 대한 지원이 AWS ParallelCluster 버전 2.10.0에 centos8 추가되었습니다. AWS Graviton 기반 인스턴스의 [dcv] 섹션에 대한 지원이 AWS ParallelCluster 버전

2.9.0에 추가되었습니다. `alinux2` 및의 [dcv] 섹션에 대한 지원이 AWS ParallelCluster 버전 2.6.0에 `ubuntu1804` 추가되었습니다. 의 [dcv] 섹션에 대한 지원이 AWS ParallelCluster 버전 2.5.0에 `centos7` 추가되었습니다.

access_from

(선택 및 권장 사항) Amazon DCV에 연결할 CIDR 형식의 IP 범위를 지정합니다. 이 설정은가 보안 그룹을 AWS ParallelCluster 생성하는 경우에만 사용됩니다.

기본값은 어떤 인터넷 주소에서도 액세스할 수 있는 `0.0.0.0/0`입니다.

```
access_from = 0.0.0.0/0
```

업데이트 정책: 업데이트 중에 이 설정을 변경할 수 있습니다.

enable

(필수) Amazon DCV가 헤드 노드에서 활성화되었는지 여부를 나타냅니다. 헤드 노드에서 Amazon DCV를 활성화하고 필요한 보안 그룹 규칙을 구성하려면 `enable` 설정을 `master`로 설정합니다.

다음 예에서는 헤드 노드에서 Amazon DCV를 활성화합니다.

```
enable = master
```

> **Note**
>
> Amazon DCV는 Amazon DCV 클라이언트와 헤드 노드에서 실행 중인 Amazon DCV 서버 간의 트래픽을 보호하는 데 사용되는 자체 서명된 인증서를 자동으로 생성합니다. 자체 인증서를 구성하려면 Amazon DCV HTTPS 인증서 섹션을 참조하세요.

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

port

(선택 사항) Amazon DCV에 대한 포트를 지정합니다.

기본값은 `8443`입니다.

```
port = 8443
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

[ebs] 섹션

헤드 노드에 탑재되고 NFS를 통해 컴퓨팅 노드에 공유되는 볼륨에 대한 Amazon EBS 볼륨 구성 설정을 정의합니다.

클러스터 정의에 Amazon EBS 볼륨을 포함하는 방법을 알아보려면 [cluster] # #/ebs_settings를 참조하세요.

클러스터 수명 주기에 구애받지 않는 장기 영구 스토리지에 기존 Amazon EBS 볼륨을 사용하려면 ebs_volume_id을 지정하세요.

를 지정하지 않으면는 클러스터를 ebs_volume_id AWS ParallelCluster 생성할 때 [ebs] 설정에서 EBS 볼륨을 생성하고 클러스터가 삭제될 때 볼륨과 데이터를 삭제합니다.

자세한 내용은 모범 사례: 클러스터를 새 AWS ParallelCluster 마이너 또는 패치 버전으로 이동 단원을 참조하십시오.

형식은 [ebs *ebs-name*]입니다. *ebs-name*은 문자로 시작해야 하고, 30자 이하여야 하며, 문자, 숫자, 하이픈(-) 및 밑줄(_)만 포함되어야 합니다.

```
[ebs custom1]
shared_dir = vol1
ebs_snapshot_id = snap-xxxxx
volume_type = io1
volume_iops = 200
...

[ebs custom2]
shared_dir = vol2
...

...
```

주제

- shared_dir
- ebs_kms_key_id

- ebs_snapshot_id
- ebs_volume_id
- encrypted
- volume_iops
- volume_size
- volume_throughput
- volume_type

shared_dir

(필수) 공유 Amazon EBS 볼륨이 탑재될 경로를 지정합니다.

이 파라미터는 여러 Amazon EBS 볼륨을 사용할 때 필요합니다.

Amazon EBS 볼륨 하나를 사용하는 경우 이 옵션은 [cluster] 섹션에 지정된 shared_dir을 덮어씁니다. 다음 예에서는 볼륨이 /vol1에 탑재됩니다.

```
shared_dir = vol1
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

ebs_kms_key_id

(선택 사항) 암호화에 사용할 사용자 지정 AWS KMS 키를 지정합니다.

이 파라미터는 encrypted = true와 함께 사용해야 합니다. 또한 사용자 지정 ec2_iam_role이 있어야 합니다.

자세한 내용은 사용자 지정 KMS 키를 사용한 디스크 암호화 항목을 참조하세요.

```
ebs_kms_key_id = xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

ebs_snapshot_id

(선택 사항) 스냅샷을 볼륨의 소스로 사용하는 경우 Amazon EBS 스냅샷 ID를 정의합니다.

기본값이 없습니다.

```
ebs_snapshot_id = snap-xxxxx
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

ebs_volume_id

(선택 사항) 헤드 노드에 연결할 기존 Amazon EBS 볼륨의 볼륨 ID를 정의합니다.

기본값이 없습니다.

```
ebs_volume_id = vol-xxxxxx
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

encrypted

(선택 사항) Amazon EBS 볼륨을 암호화할지 여부를 지정합니다. 참고: 스냅샷과 함께 사용하지 마세요.

기본값은 `false`입니다.

```
encrypted = false
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

volume_iops

(선택 사항) `io1`, `io2` 및 `gp3` 유형 볼륨의 IOPS 수를 정의합니다.

기본값, 지원되는 값, `volume_iops`와 `volume_size`의 비율 등은 `volume_type` 및 `volume_size`에 따라 달라집니다.

`volume_type` = `io1`

기본 `volume_iops` = 100

지원되는 값 `volume_iops` = 100-64000 †

최대 volume_iops:volume_size 비율은 GiB당 50 IOPS입니다. 5000 IOPS에는 최소 100GiB의 volume_size 필요합니다.

volume_type = io2

기본 volume_iops = 100

지원되는 값 volume_iops = 100-64000 (io2 블록 익스프레스 볼륨의 경우 256000) †

최대 volume_iops:volume_size 비율은 GiB당 500 IOPS입니다. 5000 IOPS에는 최소 10GiB의 volume_size가 필요합니다.

volume_type = gp3

기본 volume_iops = 3000

지원되는 값 volume_iops = 3000-16000 †

최대 volume_iops:volume_size 비율은 GiB당 500 IOPS입니다. 5000 IOPS에는 최소 10GiB의 volume_size 필요합니다.

```
volume_iops = 200
```

업데이트 정책: 업데이트 중에 이 설정을 변경할 수 있습니다.

† 최대 IOPS는 32,000 IOPS 이상으로 프로비저닝된 Nitro 시스템 기반 인스턴스에서만 보장됩니다. 다른 인스턴스는 최대 32,000 IOPS를 보장합니다. 볼륨을 수정하지 않는 한 이전 io1 볼륨은 전체 성능에 도달할 수 없습니다. io2 블록 익스프레스 볼륨은 최대 256,000의 volume_iops 값을 지원합니다. 자세한 내용을 알아보려면 Amazon EC2 사용 설명서의 io2 Block Express 볼륨(평가판)를 참조하세요.

volume_size

(선택 사항) 생성할 볼륨의 크기(GiB)를 지정합니다(스냅샷을 사용하지 않는 경우).

기본 값과 지원되는 값은 volume_type에 따라 달라집니다.

volume_type = standard

기본 volume_size = 20GiB

지원되는 값 volume_size = 1~1024GiB

volume_type = gp2, io1, io2 및 gp3

기본 volume_size = 20GiB

지원되는 값 volume_size = 1~16384GiB

volume_type = sc1 및 st1

기본 volume_size = 500GiB

지원되는 값 volume_size = 500~16384GiB

```
volume_size = 20
```

> **Note**
>
> AWS ParallelCluster 버전 2.10.1 이전에는 모든 볼륨 유형의 기본값이 20GiB였습니다.

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

volume_throughput

(선택 사항) gp3 볼륨 유형의 처리량을 MiB/s 단위로 정의합니다.

기본값은 125입니다.

지원되는 값 volume_throughput = 125~1000MiB/s

volume_throughput:volume_iops의 비율은 0.25를 초과할 수 없습니다. 1000MiB/s의 최대 처리량을 위해서는 volume_iops 설정이 최소 4000이어야 합니다.

```
volume_throughput = 1000
```

> **Note**
>
> AWS ParallelCluster 버전 2.10.1에서에 대한 지원이 volume_throughput 추가되었습니다.

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

volume_type

(선택 사항) 시작할 볼륨의 Amazon EBS 볼륨 유형을 지정합니다.

유효한 옵션은 다음 볼륨 유형입니다.

`gp2`, `gp3`

General Purpose SSD

`io1`, `io2`

Provisioned IOPS SSD

`st1`

처리량 최적화 HDD

`sc1`

콜드 HDD

`standard`

이전 세대 마그네틱

자세한 내용을 알아보려면 Amazon EC2 사용 설명서의 Amazon EBS 볼륨 유형을 참조하세요.

기본값은 `gp2`입니다.

```
volume_type = io2
```

> **Note**
>
> AWS ParallelCluster 버전 2.10.1에서 `gp3` 및에 대한 지원이 `io2` 추가되었습니다.

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

[efs] 섹션

헤드 및 컴퓨팅 노드에 탑재된 Amazon EFS에 대한 구성 설정을 정의합니다. 자세한 내용은 Amazon EFS API Reference의 CreateFileSystem을 참조하세요.

클러스터 정의에 Amazon EFS 파일 시스템을 포함하는 방법을 알아보려면 [cluster] # #/efs_settings를 참조하세요.

클러스터 수명 주기에 구애받지 않는 장기 영구 스토리지에 기존 Amazon EFS 파일 시스템을 사용하려면 efs_fs_id을 지정하세요.

를 지정하지 않으면는 클러스터를 efs_fs_id AWS ParallelCluster 생성할 때 [efs] 설정에서 Amazon EFS 파일 시스템을 생성하고 클러스터가 삭제될 때 파일 시스템 및 데이터를 삭제합니다.

자세한 내용은 모범 사례: 클러스터를 새 AWS ParallelCluster 마이너 또는 패치 버전으로 이동 단원을 참조하십시오.

형식은 [efs *efs-name*]입니다. *efs-name*은 문자로 시작해야 하고, 30자 이하여야 하며, 문자, 숫자, 하이픈(-) 및 밑줄(_)만 포함되어야 합니다.

```
[efs customfs]
shared_dir = efs
encrypted = false
performance_mode = generalPurpose
```

주제

- efs_fs_id
- efs_kms_key_id
- encrypted
- performance_mode
- provisioned_throughput
- shared_dir
- throughput_mode

efs_fs_id

(선택 사항) 기존 파일 시스템의 Amazon EFS 파일 시스템 ID를 정의합니다.

이 옵션을 지정하면 shared_dir을 제외한 기타 모든 Amazon EFS 옵션이 무효화됩니다.

이 옵션을 설정하면 다음과 같은 파일 시스템 유형만 지원됩니다.

- 스택의 가용 영역에 탑재 대상이 없는 파일 시스템

- 스택의 가용 영역에 기존 탑재 대상이 있으며 `0.0.0.0/0`에서 허용된 인바운드 및 아웃바운드 NFS 트래픽이 있는 파일 시스템

efs_fs_id를 확인하기 위한 안전성 검사를 수행하려면 IAM 역할에 다음 권한이 있어야 합니다.

- `elasticfilesystem:DescribeMountTargets`
- `elasticfilesystem:DescribeMountTargetSecurityGroups`
- `ec2:DescribeSubnets`
- `ec2:DescribeSecurityGroups`
- `ec2:DescribeNetworkInterfaceAttribute`

오류를 방지하려면 이러한 권한을 IAM 역할에 추가하거나 `sanity_check = false`를 설정해야 합니다.

> **Important**
>
> 에서 허용되는 인바운드 및 아웃바운드 NFS 트래픽으로 탑재 대상을 설정하면 탑재 대상의 가용 영역에 있는 모든 위치에서 NFS 탑재 요청에 파일 시스템이 `0.0.0.0/0`노출됩니다. 스택의 가용 영역에 탑재 대상을 생성하는 AWS 것은 권장하지 않습니다. 대신이 단계를 AWS 처리해 보겠습니다. 스택의 가용 영역에 탑재 대상이 있어야 하는 경우 [vpc] 섹션에서 vpc_security_group_id 옵션을 제공하여 사용자 지정 보안 그룹을 사용하는 것을 고려합니다. 그런 다음 해당 보안 그룹을 탑재 대상에 추가하고 `sanity_check`을 해제하여 클러스터를 생성합니다.

기본값이 없습니다.

```
efs_fs_id = fs-12345
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

efs_kms_key_id

(선택 사항) 암호화된 파일 시스템을 보호하는 데 사용할 AWS Key Management Service (AWS KMS) 고객 관리형 키를 식별합니다. 이 옵션이 설정된 경우 encrypted 설정을 `true`로 지정해야 합니다. 이는 Amazon EFS API 참조의 KmsKeyId 파라미터에 대응합니다.

기본값이 없습니다.

```
efs_kms_key_id = 1234abcd-12ab-34cd-56ef-1234567890ab
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

encrypted

(선택 사항) 파일 시스템이 암호화되는지 여부를 나타냅니다. 이는 Amazon EFS API 참조의 Encrypted 파라미터에 대응합니다.

기본값은 `false`입니다.

```
encrypted = true
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

performance_mode

(선택 사항) 파일 시스템의 성능 모드를 정의합니다. 이는 Amazon EFS API 참조의 PerformanceMode 파라미터에 대응합니다.

유효한 옵션은 다음과 같습니다.

- `generalPurpose`
- `maxIO`

두 값 모두 대소문자를 구분합니다.

대부분의 파일 시스템에 `generalPurpose` 성능 모드를 사용하는 것이 좋습니다.

`maxIO` 성능 모드를 사용하는 파일 시스템은 더 높은 수준의 집계 처리량 및 초당 작업으로 확장할 수 있습니다. 그러나 대부분의 파일 작업에서 대기 시간이 조금 더 길다는 단점이 있습니다.

파일 시스템을 생성한 후에는 이 파라미터를 변경할 수 없습니다.

기본값은 `generalPurpose`입니다.

```
performance_mode = generalPurpose
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

provisioned_throughput

(선택 사항) MiB/s로 측정되는 프로비저닝된 파일 시스템 처리량을 정의합니다. 이는 Amazon EFS API 참조의 ProvisionedThroughputInMibps 파라미터에 대응합니다.

이 파라미터를 사용할 경우 throughput_mode를 provisioned로 설정해야 합니다.

처리량 할당량은 1024MiB/s입니다. 할당량 증가를 요청하려면 지원에 문의하십시오.

최소값은 0.0MiB/s입니다.

```
provisioned_throughput = 1024
```

업데이트 정책: 업데이트 중에 이 설정을 변경할 수 있습니다.

shared_dir

(필수) 헤드 및 컴퓨팅 노드의 Amazon EFS 탑재 지점을 정의합니다.

이 파라미터는 필수 사항입니다. Amazon EFS 섹션은 shared_dir이 지정되는 경우에만 사용됩니다.

NONE 또는 /NONE을 공유 디렉터리로 사용하지 마세요.

다음 예에서는 Amazon EFS를 /efs에 탑재합니다.

```
shared_dir = efs
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

throughput_mode

(선택 사항) 파일 시스템의 처리량 모드를 정의합니다. 이는 Amazon EFS API 참조의 ThroughputMode 파라미터에 대응합니다.

유효한 옵션은 다음과 같습니다.

- bursting

- `provisioned`

기본값은 `bursting`입니다.

```
throughput_mode = provisioned
```

업데이트 정책: 업데이트 중에 이 설정을 변경할 수 있습니다.

`[fsx]` 섹션

연결된 FSx for Lustre 파일 시스템에 대한 구성 설정을 정의합니다. 자세한 내용을 알아보려면 Amazon FSx API 참조의 Amazon FSx CreateFileSystem을 참조하세요.

`base_os`이 `alinux2`, `centos7`, `ubuntu1804` 또는 `ubuntu2004`인 경우, FSx for Lustre가 지원됩니다.

Amazon Linux를 사용할 때는 커널이 `4.14.104-78.84.amzn1.x86_64` 또는 그 이후 버전이어야 합니다. 지침은 Amazon FSx for Lustre용 사용 설명서의 lustre 클라이언트 설치를 참조하세요.

> **Note**
>
> FSx for Lustre는 현재 `awsbatch`를 스케줄러로 사용할 경우에는 지원되지 않습니다.

> **Note**
>
> AWS ParallelCluster 버전 2.10.4에서의 FSx for Lustre에 대한 지원이 제거`centos8`되었습니다. AWS ParallelCluster 버전 2.11.0에서의 FSx for Lustre에 대한 지원이 `ubuntu2004` 추가되었습니다. `centos8`에서의 FSx for Lustre에 대한 지원은 AWS ParallelCluster 버전 2.10.0에서 추가되었습니다. 버전 2.6.0에서 `ubuntu1604`, 및 `alinux2`에 대한 AWS ParallelCluster FSx for Lustre 지원이 `ubuntu1804` 추가되었습니다. `centos7`에서의 FSx for Lustre에 대한 지원은 AWS ParallelCluster 버전 2.4.0에서 추가되었습니다.

기존 파일 시스템을 사용하는 경우 988 포트까지 인바운드 및 아웃바운드 TCP 트래픽을 허용하는 보안 그룹에 연결해야 합니다. 보안 그룹 규칙에서 소스를 `0.0.0.0/0`으로 설정하면 클라이언트가 해당 규칙의 프로토콜 및 포트 범위에 대해 VPC 보안 그룹 내의 모든 IP 범위에서 액세스할 수 있습니다.

파일 시스템에 대한 액세스를 추가로 제한하려면 보안 그룹 규칙에 보다 제한적인 소스를 사용하는 것이 좋습니다. 예를 들어 보다 구체적인 CIDR 범위, IP 주소 또는 보안 그룹 ID를 사용할 수 있습니다. vpc_security_group_id를 사용하지 않을 경우 이 작업이 자동으로 수행됩니다.

클러스터 수명 주기에 구애받지 않는 장기 영구 스토리지에 기존 Amazon FSx 파일 시스템을 사용하려면 fsx_fs_id을 지정하세요.

를 지정하지 않으면는 클러스터를 fsx_fs_id AWS ParallelCluster 생성할 때 [fsx] 설정에서 FSx for Lustre 파일 시스템을 생성하고 클러스터가 삭제될 때 파일 시스템 및 데이터를 삭제합니다.

자세한 내용은 모범 사례: 클러스터를 새 AWS ParallelCluster 마이너 또는 패치 버전으로 이동 단원을 참조하십시오.

형식은 [fsx *fsx-name*]입니다. *fsx-name*은 문자로 시작해야 하고, 30자 이하여야 하며, 문자, 숫자, 하이픈(-) 및 밑줄(_)만 포함되어야 합니다.

```
[fsx fs]
shared_dir = /fsx
fsx_fs_id = fs-073c3803dca3e28a6
```

새 파일 시스템을 생성하고 구성하려면 다음 파라미터를 사용합니다.

```
[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
```

주제

- 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

(선택 사항) FSx for Lustre 파일 시스템을 생성하는 데 사용된 S3 버킷의 변경 사항을 반영하기 위한 자동 가져오기 정책을 지정합니다. 가능한 값은 다음과 같습니다.

`NEW`

FSx for Lustre가 현재 FSx for Lustre 파일 시스템에 존재하지 않는 연결된 S3 버킷에 추가된 새 객체의 디렉터리 목록을 자동으로 가져옵니다.

`NEW_CHANGED`

FSx for Lustre가 S3 버킷에 추가된 새 객체 및 S3 버킷에서 변경된 기존 객체의 파일 및 디렉터리 목록을 자동으로 가져옵니다.

이는 AutoImportPolicy 속성에 해당합니다. 자세한 내용은 Amazon FSx for Lustre 사용 설명서의 S3 버킷에서 업데이트 자동 가져오기를 참조하세요. `auto_import_policy` 파라미터를 지정할 때는, `automatic_backup_retention_days`, `copy_tags_to_backups`, `daily_automatic_backup_start_time` 및 `fsx_backup_id` 파라미터를 지정하지 않아야 합니다.

`auto_import_policy` 설정을 지정하지 않으면 자동 가져오기가 비활성화됩니다. FSx for Lustre는 파일 시스템이 생성될 때 링크된 S3 버킷의 파일 및 디렉토리 목록만 업데이트합니다.

```
auto_import_policy = NEW_CHANGED
```

> **Note**
>
> AWS ParallelCluster 버전 2.10.0에서에 대한 지원이 auto_import_policy 추가되었습니다.

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

automatic_backup_retention_days

(선택 사항) 자동 백업을 보존할 일수를 지정합니다. 이것은 PERSISTENT_1 배포 유형에서만 사용할 수 있습니다. automatic_backup_retention_days 파라미터를 지정할 때는, auto_import_policy, export_path, import_path 및 imported_file_chunk_size 파라미터를 지정하지 않아야 합니다. 이는 AutomaticBackupRetentionDays 속성에 해당합니다.

기본값은 0입니다. 이 설정은 자동 백업을 비활성화합니다. 가능한 값은 0에서 35 사이의 정수(포함)입니다.

```
automatic_backup_retention_days = 35
```

> **Note**
>
> AWS ParallelCluster 버전 2.8.0에서 automatic_backup_retention_days에 대한 지원이 추가되었습니다.

업데이트 정책: 업데이트 중에 이 설정을 변경할 수 있습니다.

copy_tags_to_backups

(선택 사항) 파일 시스템의 태그를 백업에 복사할지 여부를 지정합니다. 이것은 PERSISTENT_1 배포 유형에서만 사용할 수 있습니다. copy_tags_to_backups 파라미터를 지정할 때 automatic_backup_retention_days를 0보다 큰 값으로 지정해야 하며, auto_import_policy, export_path, import_path 및 imported_file_chunk_size 파라미터는 지정하지 않아야 합니다. 이는 CopyTagsToBackups 속성에 해당합니다.

기본값은 false입니다.

```
copy_tags_to_backups = true
```

> **Note**
>
> AWS ParallelCluster 버전 2.8.0에서에 대한 지원이 copy_tags_to_backups 추가되었습니다.

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

daily_automatic_backup_start_time

(선택 사항) 자동 백업을 시작할 시간(UTC)을 지정합니다. 이것은 PERSISTENT_1 배포 유형에서만 사용할 수 있습니다. daily_automatic_backup_start_time 파라미터를 지정할 때 automatic_backup_retention_days를 0보다 큰 값으로 지정해야 하며, auto_import_policy, export_path, import_path 및 imported_file_chunk_size 파라미터는 지정하지 않아야 합니다. 이는 DailyAutomaticBackupStartTime 속성에 해당합니다.

형식은 HH:MM입니다. HH는 하루 중 제로 패딩된 시간(0-23)이고, MM은 시간의 제로 패딩된 분입니다. 예를 들어, UTC 기준 오전 1:03 시간은 다음과 같습니다.

```
daily_automatic_backup_start_time = 01:03
```

기본값은 00:00 및 23:59 사이의 무작위 시간입니다.

> **Note**
>
> AWS ParallelCluster 버전 2.8.0에서 daily_automatic_backup_start_time에 대한 지원이 추가되었습니다.

업데이트 정책: 업데이트 중에 이 설정을 변경할 수 있습니다.

data_compression_type

(선택 사항)FSx for Lustre 데이터 압축 유형을 지정합니다. 이는 DataCompressionType 속성에 해당합니다. 자세한 내용을 알아보려면 Amazon FSx for Lustre 사용 설명서의 Lustre 데이터 압축을 참조하세요.

유일한 유효 값은 LZ4입니다. 데이터 압축을 비활성화하려면 data_compression_type 파라미터를 제거하세요.

```
data_compression_type = LZ4
```

> **Note**
>
> AWS ParallelCluster 버전 2.11.0에서에 대한 지원이 data_compression_type 추가되었습니다.

업데이트 정책: 업데이트 중에 이 설정을 변경할 수 있습니다.

deployment_type

(선택 사항) FSx for Lustre 배포 유형을 지정합니다. 이는 DeploymentType 속성에 해당합니다. 자세한 내용은 Amazon FSx for Lustre 사용 설명서의 FSx for Lustre 배포 옵션을 참조하세요. 데이터의 임시 보관 및 단기 처리를 위한 스크래치 배포 유형을 선택합니다. SCRATCH_2는 최신 스크래치 파일 시스템입니다. 기본 처리량보다 높은 버스트 처리량과 전송 중 데이터 암호화를 제공합니다.

유효한 값은 SCRATCH_1, SCRATCH_2 및 PERSISTENT_1입니다.

SCRATCH_1

FSx for Lustre의 기본 배포 유형입니다. 이 배포 유형에서는 storage_capacity 설정 값에 1200, 2400 및 3600의 배수가 가능합니다. AWS ParallelCluster 버전 2.4.0에서에 대한 지원이 SCRATCH_1 추가되었습니다.

SCRATCH_2

최신 스크래치 파일 시스템입니다. 워크로드가 급증하는 경우 기준 처리량의 최대 6배를 지원합니다. 또한 지원되는 AWS 리전에서 지원되는 인스턴스 유형에 대해 전송 중 데이터 암호화를 지원합니다. 자세한 내용을 알아보려면 Amazon FSx for Lustre 사용 설명서의 전송 중 데이터 암호화를 참조하세요. 이 배포 유형에서는 storage_capacity 설정 값에 1200 및 2400의 배수가 가능합니다. SCRATCH_2에 대한 지원이 AWS ParallelCluster 버전 2.6.0에서 추가되었습니다.

PERSISTENT_1

장기 보관을 위해 설계되었습니다. 파일 서버는 가용성이 높으며 파일 시스템이 위치한 AWS 가용 영역 내에 데이터가 자동으로 복제됩니다. 지원되는 인스턴스 유형에 대해 전송 중 데이터 암호화

를 지원합니다. 이 배포 유형에서는 storage_capacity 설정 값에 1200 및 2400의 배수가 가능합니다. AWS ParallelCluster 버전 2.6.0에서에 대한 지원이 PERSISTENT_1 추가되었습니다.

기본값은 SCRATCH_1입니다.

```
deployment_type = SCRATCH_2
```

> **Note**
>
> AWS ParallelCluster 버전 2.6.0에서에 대한 지원이 deployment_type 추가되었습니다.

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

drive_cache_type

(선택 사항)파일 시스템에 SSD 드라이브 캐시가 있도록 지정합니다. storage_type 설정을 HDD로 지정한 경우 이 옵션만 설정할 수 있습니다. 이는 DriveCacheType 속성에 해당합니다. 자세한 내용은 Amazon FSx for Lustre 사용 설명서의 FSx for Lustre 배포 옵션을 참조하세요.

유일한 유효 값은 READ입니다. SSD 드라이브 캐시를 비활성화하려면 drive_cache_type 설정을 지정하지 마세요.

```
drive_cache_type = READ
```

> **Note**
>
> AWS ParallelCluster 버전 2.10.0에서에 대한 지원이 drive_cache_type 추가되었습니다.

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

export_path

(선택 사항) 파일 시스템의 루트를 내보내는 Amazon S3 경로를 지정합니다. export_path 파라미터를 지정할 때는, automatic_backup_retention_days, copy_tags_to_backups, daily_automatic_backup_start_time 및 fsx_backup_id 파라미터를 지정하지 않아야 합니다. 이는 ExportPath 속성에 해당합니다. 파일 데이터와 메타데이터는 export_path로 자동으로 내

보내지지 않습니다. 데이터 및 메타데이터를 내보내는 방법에 대한 자세한 내용은 Amazon FSx for Lustre 사용 설명서의 데이터 리포지토리에 대한 변경 내용 내보내기를 참조하세요.

기본값은 s3://*import-bucket*/FSxLustre*[creation-timestamp]*입니다. 여기서 *import-bucket*은 import_path 파라미터에서 제공되는 버킷입니다.

```
export_path = s3://bucket/folder
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

fsx_backup_id

(선택 사항) 기존 백업에서 파일 시스템을 복원하는 데 사용할 백업 ID를 지정합니다. fsx_backup_id 파라미터를 지정할 때는, auto_import_policy, deployment_type, export_path, fsx_kms_key_id, import_path, imported_file_chunk_size, storage_capacity 및 per_unit_storage_throughput 파라미터를 지정하지 않아야 합니다. 이러한 파라미터는 백업에서 읽혀집니다. 또한, auto_import_policy, export_path, import_path 및 imported_file_chunk_size 파라미터를 지정하지 않아야 합니다.

이는 BackupId 속성에 해당합니다.

```
fsx_backup_id = backup-fedcba98
```

> **Note**
>
> AWS ParallelCluster 버전 2.8.0에서에 대한 지원이 fsx_backup_id 추가되었습니다.

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

fsx_fs_id

(선택 사항) 기존 FSx for Lustre 파일 시스템을 연결합니다.

이 옵션을 지정하면 [fsx] 섹션의 shared_dir 및 fsx_fs_id 설정만 사용되고 [fsx] 섹션의 다른 설정은 무시됩니다.

```
fsx_fs_id = fs-073c3803dca3e28a6
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

fsx_kms_key_id

(선택 사항) AWS Key Management Service (AWS KMS) 고객 관리형 키의 키 ID를 지정합니다.

이 키는 유휴 파일 시스템에서 데이터를 암호화하는 데 사용됩니다.

이 항목은 사용자 지정 ec2_iam_role과 함께 사용해야 합니다. 자세한 내용은 사용자 지정 KMS 키를 사용한 디스크 암호화 단원을 참조하십시오. 이는 Amazon FSx API 참조의 KmsKeyId 파라미터에 해당합니다.

```
fsx_kms_key_id = xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

> **Note**
>
> AWS ParallelCluster 버전 2.6.0에서에 대한 지원이 fsx_kms_key_id 추가되었습니다.

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

import_path

(선택 사항) 파일 시스템으로 데이터를 로드하고 내보내기 버킷으로 작동할 S3 버킷을 지정합니다. 자세한 내용은 export_path 단원을 참조하십시오. import_path 파라미터를 지정하는 경우, automatic_backup_retention_days, copy_tags_to_backups, daily_automatic_backup_start_time 및 fsx_backup_id 파라미터를 지정해서는 안 됩니다. 이는 Amazon FSx API 참조의 ImportPath 파라미터에 해당합니다.

클러스터 생성 시 가져오기가 수행됩니다. 자세한 내용은 Amazon FSx for Lustre 사용 설명서의 데이터 리포지토리에서 데이터 가져오기를 참조하세요. 가져올 때는 파일 메타데이터(이름, 소유권, 타임스탬프, 권한)만 가져옵니다. 파일 데이터는 파일에 처음 액세스할 때까지 S3 버킷에서 가져오지 않습니다. 파일 콘텐츠 사전 로드에 대한 자세한 내용은 Amazon FSx for Lustre 사용 설명서의 파일 시스템에 파일 사전 로드를 참조하세요.

값을 제공하지 않으면 파일 시스템이 비어 있습니다.

```
import_path =  s3://bucket
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

imported_file_chunk_size

(선택 사항) 데이터 리포지토리에서 가져온 파일의 경우(import_path 사용) 이 값은 단일 물리적 디스크에 저장된 파일당 스트라이프 수 및 최대 데이터 양(MiB)을 결정합니다. 단일 파일을 스트라이프할 수 있는 최대 디스크 수는 파일 시스템을 구성하는 총 디스크 수에 따라 제한됩니다. imported_file_chunk_size 파라미터를 지정할 때는, automatic_backup_retention_days, copy_tags_to_backups, daily_automatic_backup_start_time 및 fsx_backup_id 파라미터를 지정하지 않아야 합니다. 이 속성은 ImportedFileChunkSize 속성에 해당합니다.

청크 크기 기본값은 1024(1GiB)이며 최대 512,000MiB(500GiB)까지 가능합니다. Amazon S3 객체의 크기는 최대 5TB입니다.

```
imported_file_chunk_size = 1024
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

per_unit_storage_throughput

(**PERSISTENT_1** 배포 유형에 필수) deployment_type = PERSISTENT_1 배포 유형의 경우 스토리지의 1테비바이트(TiB)당 읽기 및 쓰기 처리량(MB/s/TiB)을 설명합니다. 파일 시스템 처리량 용량은 파일 시스템 스토리지 용량(TiB)에 per_unit_storage_throughput(MB/s/TiB)를 곱하여 계산됩니다. 2.4TiB 파일 시스템의 경우 50MB/s/TiB의 per_unit_storage_throughput를 프로비저닝하여 120MB/s의 파일 시스템 처리량을 얻을 수 있습니다. 프로비저닝한 처리량에 대해 비용을 지불합니다. 이는 PerUnitStorageThroughput 속성에 해당합니다.

가능한 값은 storage_type 설정의 값에 따라 달라집니다.

storage_type = SSD

가능한 값은 50, 100, 200입니다.

storage_type = HDD

가능한 값은 12, 40입니다.

```
per_unit_storage_throughput = 200
```

> **Note**
>
> per_unit_storage_throughput에 대한 지원이 AWS ParallelCluster 버전 2.6.0에서 추가되었습니다.

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

shared_dir

(필수) 헤드 및 컴퓨팅 노드에서 FSx for Lustre 파일 시스템의 탑재 지점을 정의합니다.

NONE 또는 /NONE은 공유 디렉터리로 사용하지 마세요.

다음 예에서는 /fsx에 파일 시스템을 탑재합니다.

```
shared_dir = /fsx
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

storage_capacity

(필수 사항) 파일 시스템의 스토리지 용량(GiB)을 지정합니다. 이는 StorageCapacity 속성에 해당합니다.

가능한 스토리지 용량 값은 deployment_type 설정에 따라 다릅니다.

SCRATCH_1

가능한 값은 1200, 2400 및 3600의 배수입니다.

SCRATCH_2

가능한 값은 1200 및 2400의 배수입니다.

PERSISTENT_1

가능한 값은 다른 설정의 값에 따라 다릅니다.

storage_type = SSD

가능한 값은 1200 및 2400의 배수입니다.

storage_type = HDD

가능한 값은 per_unit_storage_throughput 설정에 따라 다릅니다.

per_unit_storage_throughput = 12

가능한 값은 6000의 배수입니다.

per_unit_storage_throughput = 40

가능한 값은 1800의 배수입니다.

```
storage_capacity = 7200
```

> **Note**
>
> AWS ParallelCluster 버전 2.5.0 및 2.5.1의 경우는 가능한 값 1200, 2400 및 3600의 배수를 storage_capacity 지원했습니다. 버전 2.5.0 이전 AWS ParallelCluster 버전의 경우의 최소 크기는 3600storage_capacity입니다.

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

storage_type

(선택 사항) 파일 시스템의 스토리지 유형을 지정합니다. 이는 StorageType 속성에 해당합니다. 가능한 값은 SSD와 HDD입니다. 기본값은 SSD입니다.

스토리지 유형에 따라 다른 설정의 가능한 값이 변경됩니다.

storage_type = SSD

솔리드 스테이트 드라이브(SSD) 스토리지 유형을 지정합니다.

storage_type = SSD는 다른 여러 설정의 가능한 값을 변경합니다.

drive_cache_type

이 설정은 지정할 수 없습니다.

deployment_type

이 설정은 SCRATCH_1, SCRATCH_2 또는 PERSISTENT_1로 설정할 수 있습니다.

per_unit_storage_throughput

deployment_type가 PERSISTENT_1로 설정된 경우 이 설정을 지정해야 합니다. 가능한 값은 50, 100 또는 200입니다.

storage_capacity

이 설정은 지정되어야 합니다. 가능한 값은 deployment_type에 따라 다릅니다.

deployment_type = SCRATCH_1

storage_capacity는 1200, 2400 또는 3600의 배수가 될 수 있습니다.

deployment_type = SCRATCH_2 또는 deployment_type = PERSISTENT_1

storage_capacity는 1200 또는 2400의 임의의 배수일 수 있습니다.

storage_type = HDD

하드 디스크 드라이브(HDD) 스토리지 유형을 지정합니다.

storage_type = HDD는 다른 설정의 가능한 값을 변경합니다.

drive_cache_type

이 설정은 지정할 수 있습니다.

deployment_type

이 설정은 PERSISTENT_1로 설정되어야 합니다.

per_unit_storage_throughput

이 설정은 지정되어야 합니다. 가능한 값은 12 또는 40입니다.

storage_capacity

이 설정은 지정되어야 합니다. 가능한 값은 per_unit_storage_throughput 설정에 따라 다릅니다.

storage_capacity = 12

storage_capacity는 6000의 배수일 수 있습니다.

storage_capacity = 40

storage_capacity는 1800의 배수일 수 있습니다.

```
storage_type = SSD
```

> **Note**
>
> storage_type 설정에 대한 지원이 AWS ParallelCluster 버전 2.10.0에 추가되었습니다.

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

weekly_maintenance_start_time

(선택 사항) 주별 유지 관리를 수행할 기본 시간을 UTC 시간대로 지정합니다. 이는 WeeklyMaintenanceStartTime 속성에 해당합니다.

형식은 [요일]:[시간]:[분]입니다. 예를 들어 월요일 자정은 다음과 같습니다.

```
weekly_maintenance_start_time = 1:00:00
```

업데이트 정책: 업데이트 중에 이 설정을 변경할 수 있습니다.

[queue] 섹션

단일 대기열에 대한 구성 설정을 정의합니다. [queue] 섹션은 scheduler가 slurm로 설정된 경우에만 지원됩니다.

형식은 [queue *<queue-name>*]입니다. *### ##*은 소문자로 시작해야 하고, 30자를 이내로 소문자, 숫자 및 하이픈(-)만 포함되어야 합니다.

```
[queue q1]
compute_resource_settings = i1,i2
placement_group = DYNAMIC
enable_efa = true
disable_hyperthreading = false
compute_type = spot
```

> **Note**
>
> [queue] 섹션 지원이 AWS ParallelCluster 버전 2.9.0에 추가되었습니다.

주제

- compute_resource_settings
- compute_type
- disable_hyperthreading
- enable_efa
- enable_efa_gdr
- placement_group

compute_resource_settings

(필수) 이 대기열의 컴퓨팅 리소스 구성이 포함된 [compute_resource] 섹션을 식별합니다. 섹션 이름은 문자로 시작해야 하고, 30자를 이내로, 문자, 숫자, 하이픈(-) 및 밑줄(_)만 포함되어야 합니다.

최대 세(3) 개의 [compute_resource] 섹션이 각 [queue] 섹션에 지원됩니다.

예를 들어, 다음 설정은 [compute_resource cr1] 및 [compute_resource cr2]을 시작하는 섹션을 지정합니다.

```
compute_resource_settings = cr1, cr2
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

compute_type

(선택 사항) 이 대기열에 대해 시작할 인스턴스 유형을 정의합니다. 이 설정은 cluster_type 설정을 대체합니다.

유효한 옵션은 ondemand 및 spot입니다.

기본값은 ondemand입니다.

스팟 인스턴스에 대한 자세한 내용은 스팟 인스턴스 작업 섹션을 참조하세요.

> **Note**
>
> 스팟 인스턴스를 사용하려면 계정에 AWSServiceRoleForEC2Spot 서비스 연결 역할이 있어야 합니다. 를 사용하여 계정에서이 역할을 생성하려면 다음 명령을 AWS CLI실행합니다.

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

자세한 내용은 Amazon EC2 사용 설명서에서 스팟 인스턴스 요청을 위한 서비스 연결 역할을 참조하세요.

다음 예제에서는 이 대기열의 컴퓨팅 노드에 SpotInstances를 사용합니다.

```
compute_type = spot
```

업데이트 정책: 업데이트를 위해 이 설정을 변경하려면 컴퓨팅 플릿을 중지해야 합니다.

disable_hyperthreading

(선택 사항) 이 대기열의 노드에서 하이퍼스레딩을 비활성화합니다. 모든 인스턴스 유형이 하이퍼 스레딩을 비활성화할 수 있는 것은 아닙니다. 하이퍼스레딩 비활성화를 지원하는 인스턴스 유형 목록은 Amazon EC2 사용 설명서에서 인스턴스 유형별 각 CPU 코어의 CPU 코어 및 스레드를 참조하세요. [cluster] 섹션의 disable_hyperthreading 설정이 정의된 경우 이 설정을 정의할 수 없습니다.

기본값은 false입니다.

```
disable_hyperthreading = true
```

업데이트 정책: 업데이트를 위해 이 설정을 변경하려면 컴퓨팅 플릿을 중지해야 합니다.

enable_efa

(선택 사항) true로 설정하면 이 대기열의 노드에 대해 Elastic Fabric Adapter(EFA)가 활성화되도록 지정합니다. EFA를 지원하는 EC2 인스턴스 목록을 보려면 Linux 인스턴스용 Amazon EC2 사용 설명서의 지원되는 인스턴스 유형을 참조하세요. [cluster] 섹션의 enable_efa 설정이 정의된 경우 이 설정을 정의할 수 없습니다. 클러스터 배치 그룹은 인스턴스 간의 대기 시간을 최소화하기 위해 사용해야 합니다. 자세한 내용은 placement 및 placement_group 섹션을 참조하세요.

```
enable_efa = true
```

업데이트 정책: 업데이트를 위해 이 설정을 변경하려면 컴퓨팅 플릿을 중지해야 합니다.

enable_efa_gdr

(선택 사항) AWS ParallelCluster 버전 2.11.3부터는이 설정이 적용되지 않습니다. GPUDirect RDMA(원격 다이렉트 메모리 액세스)에 대한 Elastic Fabric Adapter(EFA) 지원은 컴퓨팅 노드에 대해 활성화되며, 인스턴스 유형에서 지원하는 경우 컴퓨팅 노드는 항상 활성화됩니다.

> **Note**
>
> AWS ParallelCluster 버전 2.10.0~2.11.2: `true`인 경우이 대기열의 노드에 대해 EFA(Elastic Fabric Adapter) GPUDirect RDMA(원격 직접 메모리 액세스)가 활성화되도록 지정합니다. 이를 `true`으로 설정하려면 `enable_efa` 설정을 `true`으로 설정해야 합니다. EFA GPUDirect RDMA는 이러한 운영 체제(`alinux2`, `centos7`, `ubuntu1804`, `ubuntu2004`)의 다음 인스턴스 유형(`p4d.24xlarge`)에서 지원됩니다. `[cluster]` 섹션의 `enable_efa_gdr` 설정이 정의된 경우 이 설정을 정의할 수 없습니다. 클러스터 배치 그룹은 인스턴스 간의 대기 시간을 최소화하기 위해 사용해야 합니다. 자세한 내용은 `placement` 및 `placement_group` 섹션을 참조하세요.

기본값은 `false`입니다.

```
enable_efa_gdr = true
```

> **Note**
>
> AWS ParallelCluster 버전 2.10.0에서에 대한 지원이 `enable_efa_gdr` 추가되었습니다.

업데이트 정책: 업데이트를 위해 이 설정을 변경하려면 컴퓨팅 플릿을 중지해야 합니다.

placement_group

(선택 사항) 존재하는 경우 이 대기열의 배치 그룹을 정의합니다. 이 설정은 `placement_group` 설정을 대체합니다.

유효한 옵션은 다음과 같습니다.

- `DYNAMIC`
- 기존의 Amazon EC2 클러스터 배치 그룹 이름입니다.

DYNAMIC으로 설정하면 이 대기열에 대한 고유의 배치 그룹이 클러스터 스택의 일부로 생성되고 삭제됩니다.

배치 그룹에 대한 자세한 내용은 Amazon EC2 사용 설명서의 배치 그룹을 참조하세요. 다른 인스턴스 유형에 동일한 배치 그룹을 사용하는 경우 용량 부족 오류로 인해 요청이 실패할 가능성이 높습니다. 자세한 내용은 Amazon EC2 사용 설명서의 부족한 인스턴스 용량을 참조하세요.

기본값이 없습니다.

모든 인스턴스 유형이 클러스터 배치 그룹을 지원하는 것은 아닙니다. 예를 들어,t2.micro는 클러스터 배치 그룹을 지원하지 않습니다. 클러스터 배치 그룹을 지원하는 인스턴스 유형에 대한 자세한 내용은 Amazon EC2 사용 설명서의 클러스터 배치 그룹의 규칙 및 제한 사항을 참조하세요. 배치 그룹 작업 시 팁은 배치 그룹 및 인스턴스 시작 문제 단원을 참고하세요.

```
placement_group = DYNAMIC
```

업데이트 정책: 업데이트를 위해 이 설정을 변경하려면 컴퓨팅 플릿을 중지해야 합니다.

[raid] 섹션

수많은 동일 Amazon EBS 볼륨에서 빌드된 RAID 배열에 대한 구성 설정을 정의합니다. RAID 드라이브는 헤드 노드에 탑재되고 NFS를 통해 컴퓨팅 노드로 내보내집니다.

형식은 [raid *raid-name*]입니다. *raid-name#* 문자로 시작해야 하고 30자 이하여야 하며 문자, 숫자, 하이픈(-) 및 밑줄(_)만 포함할 수 있습니다.

```
[raid rs]
shared_dir = raid
raid_type = 1
num_of_raid_volumes = 2
encrypted = true
```

주제

- shared_dir
- ebs_kms_key_id
- encrypted
- num_of_raid_volumes
- raid_type

- volume_iops
- volume_size
- volume_throughput
- volume_type

shared_dir

(필수) 헤드 및 컴퓨팅 노드에서 RAID 배열의 탑재 지점을 정의합니다.

이 파라미터가 지정된 경우에만 RAID 드라이브가 생성됩니다.

NONE 또는 /NONE은 공유 디렉터리로 사용하지 마세요.

다음 예에서는 배열을 /raid에 탑재합니다.

```
shared_dir = raid
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

ebs_kms_key_id

(선택 사항) 암호화에 사용할 사용자 지정 AWS KMS 키를 지정합니다.

이 파라미터는 encrypted = true와 함께 사용해야 하며 사용자 지정 ec2_iam_role이 있어야 합니다.

자세한 내용은 사용자 지정 KMS 키를 사용한 디스크 암호화 단원을 참조하십시오.

```
ebs_kms_key_id = xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

encrypted

(선택 사항) 파일 시스템이 암호화되는지 여부를 지정합니다.

기본값은 false입니다.

```
encrypted = false
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

num_of_raid_volumes

(선택 사항) RAID 배열을 어셈블할 Amazon EBS 볼륨 수를 정의합니다.

최소 볼륨 수는 2입니다.

최대 볼륨 수는 5입니다.

기본값은 2입니다.

```
num_of_raid_volumes = 2
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

raid_type

(필수) RAID 배열의 RAID 유형을 정의합니다.

이 파라미터가 지정된 경우에만 RAID 드라이브가 생성됩니다.

유효한 옵션은 다음과 같습니다.

- `0`
- `1`

RAID 유형에 대한 자세한 내용은 Amazon EC2 사용 설명서의 RIAD 정보를 참조하세요.

다음은 RAID `0` 배열을 생성하는 예입니다.

```
raid_type = 0
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

volume_iops

(선택 사항) `io1`, `io2` 및 `gp3` 유형 볼륨의 IOPS 수를 정의합니다.

기본값, 지원되는 값, `volume_iops`와 `volume_size`의 비율 등은 `volume_type` 및 `volume_size`에 따라 달라집니다.

`volume_type = io1`

기본 `volume_iops` = 100

지원되는 값 `volume_iops` = 100~64000†

최대 `volume_iops`와 `volume_size`의 비율 = 50IOPS/GiB. 5000 IOPS는 최소 100GiB의 `volume_size`가 필요합니다.

`volume_type = io2`

기본 `volume_iops` = 100

지원되는 값 `volume_iops` = 100~64000(`io2` Block Express 볼륨의 경우 256000)†

최대 `volume_iops`와 `volume_size` 비율 = 500IOPS/GiB. 5000 IOPS에는 최소 10GiB의 `volume_size`가 필요합니다.

`volume_type = gp3`

기본 `volume_iops` = 3000

지원되는 값 `volume_iops` = 3000~16000

최대 `volume_iops`와 `volume_size` 비율 = 500IOPSIOPS/GiB. 5000 IOPS에는 최소 10GiB`volume_size`가 필요합니다.

```
volume_iops = 3000
```

업데이트 정책: 업데이트 중에 이 설정을 변경할 수 있습니다.

† 최대 IOPS는 32,000 IOPS 이상으로 프로비저닝된 Nitro 시스템 기반 인스턴스에서만 보장됩니다. 다른 인스턴스는 최대 32,000 IOPS를 보장합니다. 이전 `io1` 볼륨은 `io2` 볼륨을 수정하지 않는 한 전체 성능에 도달할 수 없습니다. Block Express 볼륨은 최대 256000의 `volume_iops` 값을 지원합니다. 자세한 내용을 알아보려면 Amazon EC2 사용 설명서의 `io2` Block Express 볼륨(평가판)를 참조하세요.

`volume_size`

(선택 사항) 생성할 볼륨의 크기를 GiB 단위로 정의합니다.

기본값 및 지원되는 값은 volume_type에 따라 다릅니다.

volume_type = standard

기본 volume_size = 20GiB

지원되는 값 volume_size = 1~1024GiB

volume_type = gp2, io1, io2 및 gp3

기본 volume_size = 20GiB

지원되는 값 volume_size = 1~16384GiB

volume_type = sc1 및 st1

기본 volume_size = 500GiB

지원되는 값 volume_size = 500~16384GiB

```
volume_size = 20
```

> **Note**
>
> AWS ParallelCluster 버전 2.10.1 이전에는 모든 볼륨 유형의 기본값이 20GiB였습니다.

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

volume_throughput

(선택 사항) gp3 볼륨 유형의 처리량을 MiB/s 단위로 정의합니다.

기본값은 125입니다.

지원되는 값 volume_throughput = 125~1000MiB/s

volume_throughput:volume_iops의 비율은 0.25를 초과할 수 없습니다. 1000MiB/s의 최대 처리량을 위해서는 volume_iops 설정이 최소 4000이어야 합니다.

```
volume_throughput = 1000
```

> **Note**
>
> AWS ParallelCluster 버전 2.10.1에서에 대한 지원이 `volume_throughput` 추가되었습니다.

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

`volume_type`

(선택 사항) 빌드할 볼륨의 유형을 정의합니다.

유효한 옵션은 다음과 같습니다.

`gp2`, `gp3`

General Purpose SSD

`io1`, `io2`

Provisioned IOPS SSD

`st1`

처리량 최적화 HDD

`sc1`

콜드 HDD

`standard`

이전 세대 마그네틱

자세한 내용을 알아보려면 Amazon EC2 사용 설명서의 Amazon EBS 볼륨 유형을 참조하세요.

기본값은 `gp2`입니다.

```
volume_type = io2
```

> **Note**
>
> AWS ParallelCluster 버전 2.10.1에서 `gp3` 및에 대한 지원이 `io2` 추가되었습니다.

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

[scaling] 섹션

주제

- scaledown_idletime

컴퓨팅 노드가 규모 조정되는 방식을 정의하는 설정을 지정합니다.

형식은 [scaling *scaling-name*]과 같습니다. *scaling-name*은 문자로 시작해야 하고, 30자 이하여야 하며, 문자, 숫자, 하이픈(-) 및 밑줄(_)만 포함할 수 있습니다.

```
[scaling custom]
scaledown_idletime = 10
```

scaledown_idletime

(선택 사항) 컴퓨팅 노드가 종료될 때까지 작업이 없는 시간(분)을 지정합니다.

스케줄러가 awsbatch인 경우에는 이 파라미터가 사용되지 않습니다.

기본값은 10입니다.

```
scaledown_idletime = 10
```

업데이트 정책: 업데이트를 위해 이 설정을 변경하려면 컴퓨팅 플릿을 중지해야 합니다.

[vpc] 섹션

Amazon VPC 구성 설정을 지정합니다. VPC에 대한 자세한 내용은 Amazon VPC란 무엇인가? 및 Amazon VPC 사용 설명서의 VPC에 대한 보안 모범 사례를 참조하세요.

형식은 [vpc *vpc-name*]입니다. *vpc-name*은 문자로 시작해야 하고, 30자 이하여야 하고, 문자, 숫자, 하이픈(-) 및 밑줄(_)만 포함되어야 합니다.

```
[vpc public]
vpc_id = vpc-xxxxxx
```

```
master_subnet_id = subnet-xxxxxx
```

주제

- additional_sg
- compute_subnet_cidr
- compute_subnet_id
- master_subnet_id
- ssh_from
- use_public_ips
- vpc_id
- vpc_security_group_id

additional_sg

(선택 사항) 모든 인스턴스에 대한 추가 Amazon VPC 보안 그룹 ID를 제공합니다.

기본값이 없습니다.

```
additional_sg = sg-xxxxxx
```

compute_subnet_cidr

(선택 사항) Classless Inter-Domain Routing(CIDR) 블록을 지정합니다. 컴퓨팅 서브넷 AWS ParallelCluster 을 생성하려면이 파라미터를 사용합니다.

```
compute_subnet_cidr = 10.0.100.0/24
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

compute_subnet_id

(선택 사항) 컴퓨팅 노드를 프로비저닝할 기존 서브넷의 ID를 지정합니다.

지정하지 않으면 compute_subnet_id는 master_subnet_id 값을 사용합니다.

서브넷이 비공개인 경우 웹 액세스를 위한 NAT를 설정해야 합니다.

```
compute_subnet_id = subnet-xxxxxx
```

업데이트 정책: 업데이트를 위해 이 설정을 변경하려면 컴퓨팅 플릿을 중지해야 합니다.

master_subnet_id

(필수) 헤드 노드를 프로비저닝할 기존 서브넷의 ID를 지정합니다.

```
master_subnet_id = subnet-xxxxxx
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

ssh_from

(선택 사항) SSH 액세스를 허용할 CIDR 형식의 IP 범위를 지정합니다.

이 파라미터는가 보안 그룹을 AWS ParallelCluster 생성하는 경우에만 사용됩니다.

기본값은 `0.0.0.0/0`입니다.

```
ssh_from = 0.0.0.0/0
```

업데이트 정책: 업데이트 중에 이 설정을 변경할 수 있습니다.

use_public_ips

(선택 사항) 컴퓨팅 인스턴스에 공개 IP 주소를 할당할지 여부를 정의합니다.

`true`로 설정하면 탄력적 IP 주소가 헤드 노드에 연결됩니다.

`false`로 설정하면 "공개 IP 자동 할당" 서브넷 구성 파라미터의 값에 따라 헤드 노드에 공개 IP가 있거나 없게 됩니다.

예는 네트워킹 구성 섹션을 참조하세요.

기본값은 `true`입니다.

```
use_public_ips = true
```

> **Important**
>
> 기본적으로 모든 AWS 계정 는 각각에 대해 다섯(5) 개의 탄력적 IP 주소로 제한됩니다 AWS 리전. 자세한 내용은 Amazon EC2 사용 설명서에서 탄력적 IP 주소 제한을 참조하세요.

업데이트 정책: 업데이트를 위해 이 설정을 변경하려면 컴퓨팅 플릿을 중지해야 합니다.

vpc_id

(필수) 클러스터를 프로비저닝할 Amazon VPC의 ID를 지정합니다.

```
vpc_id = vpc-xxxxxx
```

업데이트 정책: 이 설정을 변경하면 업데이트가 허용되지 않습니다.

vpc_security_group_id

(선택 사항) 모든 인스턴스에 기존 보안 그룹을 사용하도록 지정합니다.

기본값이 없습니다.

```
vpc_security_group_id = sg-xxxxxx
```

에서 생성한 보안 그룹은 ssh_from 설정에 지정된 주소 또는 ssh_from 설정이 지정되지 않은 경우 모든 IPv4 주소(0.0.0.0/0)에서 포트 22를 사용하여 SSH 액세스를 AWS ParallelCluster 허용합니다. Amazon DCV가 활성화된 경우 보안 그룹은 access_from 설정에 지정된 주소 또는 access_from 설정이 지정되지 않은 모든 IPv4 주소(0.0.0.0/0)에서 8443 포트(또는 port 설정에 지정된 포트)를 사용하여 Amazon DCV에 액세스할 수 있습니다.

> **Warning**
>
> [cluster] fsx_settings이 지정되지 않거나 fsx_settings과 기존 외부 FSx for Lustre 파일 시스템 모두 [fsx fs]에서 fsx-fs-id가 지정된 경우 이 파라미터의 값을 변경하고 클러스터를 업데이트할 수 있습니다.
> AWS ParallelCluster 관리형 FSx for Lustre 파일 시스템이 fsx_settings 및에 지정된 경우 이 파라미터의 값을 변경할 수 없습니다[fsx fs].

업데이트 정책: 구성에 AWS ParallelCluster 관리형 Amazon FSx for Lustre 파일 시스템이 지정되지 않은 경우 업데이트 중에이 설정을 변경할 수 있습니다.

예시

다음 예제 구성은 Slurm, Torque및 AWS Batch 스케줄러를 사용한 AWS ParallelCluster 구성을 보여줍니다.

> **Note**
>
> 버전 2.11.5부터 SGE 또는 Torque 스케줄러 사용을 지원하지 AWS ParallelCluster 않습니다.

목차

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

Slurm Workload Manager (`slurm`)

다음 예제는 `slurm` 스케줄러를 사용하는 클러스터를 시작합니다. 예제 구성에서는 2개의 작업 대기열이 있는 클러스터 1개를 시작합니다. 첫 번째 대기열에는 처음에 `spot` 사용할 수 있는 `t3.micro` 스팟 인스턴스 2개가 있습니다. 최대 10개의 인스턴스까지 스케일 업할 수 있으며, 10분 동안 작업이 실행되지 않은 경우 최소 1개의 인스턴스로 스케일 다운할 수 있습니다(`scaledown_idletime` 설정을 사용하여 조정 가능). 두 번째 대기열인 ondemand는 인스턴스 없이 시작하여 최대 5개의 `t3.micro` 온디맨드 인스턴스까지 스케일 업할 수 있습니다.

```
[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**) 및 Torque Resource Manager(**torque**)

> **Note**
>
> 이 예제는 AWS ParallelCluster 버전 2.11.4까지의 버전에만 적용됩니다. 버전 2.11.5부터는 AWS ParallelCluster 가 SGE 또는 Torque 스케줄러의 사용을 지원하지 않습니다.

다음 예제는 `torque` 또는 `sge` 스케줄러를 사용하는 클러스터를 시작합니다. SGE을 사용하려면 `scheduler = torque`를 `scheduler = sge`로 변경하세요. 예제 구성은 최대 5개의 동시 실행 노드를 허용하고 10분 동안 작업이 실행되지 않으면 2개로 스케일 다운됩니다.

```
[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**
>
> 버전 2.11.5부터 SGE 또는 Torque 스케줄러 사용을 지원하지 AWS ParallelCluster 않습니다. 이러한 버전을 사용하는 경우 해당 버전을 계속 사용하거나 AWS 서비스 및 지원 팀의 AWS 지원 문제를 해결할 수 있습니다.

AWS Batch (**awsbatch**)

다음 예제는 awsbatch 스케줄러를 사용하는 클러스터를 시작합니다. 작업 리소스 요구 사항에 따라 더 나은 인스턴스 유형을 선택하도록 설정되어 있습니다.

예제 구성은 최대 40개의 동시 실행 vCPU를 허용하고 10분 동안 작업이 실행되지 않으면 0으로 스케일 다운됩니다(scaledown_idletime 설정을 사용하여 조정 가능).

```
[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
```

AWS ParallelCluster 작동 방식

AWS ParallelCluster 는 클러스터를 관리하는 방법뿐만 아니라 AWS 서비스를 사용하여 HPC 환경을 구축하는 방법에 대한 참조로 구축되었습니다.

주제

- AWS ParallelCluster 프로세스
- AWS 에서 사용하는 서비스 AWS ParallelCluster
- AWS ParallelCluster Auto Scaling

AWS ParallelCluster 프로세스

이 섹션은 지원되는 기존 작업 스케줄러 중 하나(SGE, Slurm 또는 Torque)를 사용하여 배포된 HPC 클러스터에만 적용됩니다. 이러한 스케줄러와 함께 사용할 경우는 Auto Scaling 그룹 및 기본 작업 스케줄러와 상호 작용하여 컴퓨팅 노드 프로비저닝 및 제거를 AWS ParallelCluster 관리합니다.

기반 HPC 클러스터의 경우 AWS Batch컴퓨팅 노드 관리를 AWS Batch 위해에서 제공하는 기능을 AWS ParallelCluster 사용합니다.

> **Note**
>
> 버전 2.11.5부터 SGE 또는 Torque 스케줄러 사용을 지원하지 AWS ParallelCluster 않습니다. 2.11.4 이하의 버전에서는 계속 사용할 수 있지만 AWS 서비스 및 AWS 지원 팀의 향후 업데이트 또는 문제 해결 지원을 받을 수 없습니다.

주제

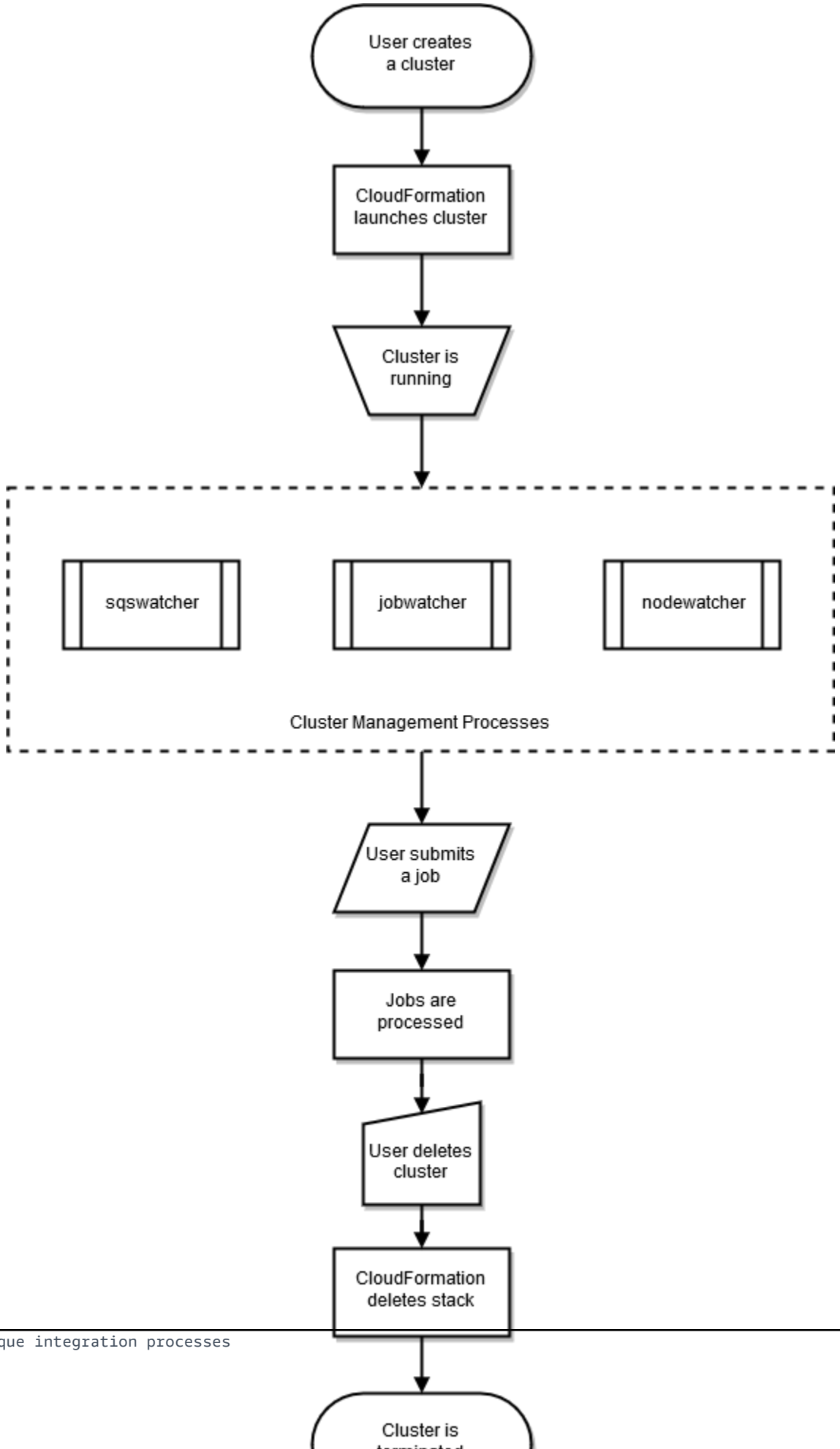
- SGE and Torque integration processes
- Slurm integration processes

SGE and Torque integration processes

> **Note**
>
> 이 섹션은 AWS ParallelCluster 버전 2.11.4까지의 버전에만 적용됩니다. 버전 2.11.5부터 AWS ParallelCluster 은 Amazon SNS SGE 및 Torque 스케줄러 Amazon SQS의 사용을 지원하지 않습니다.

일반 개요

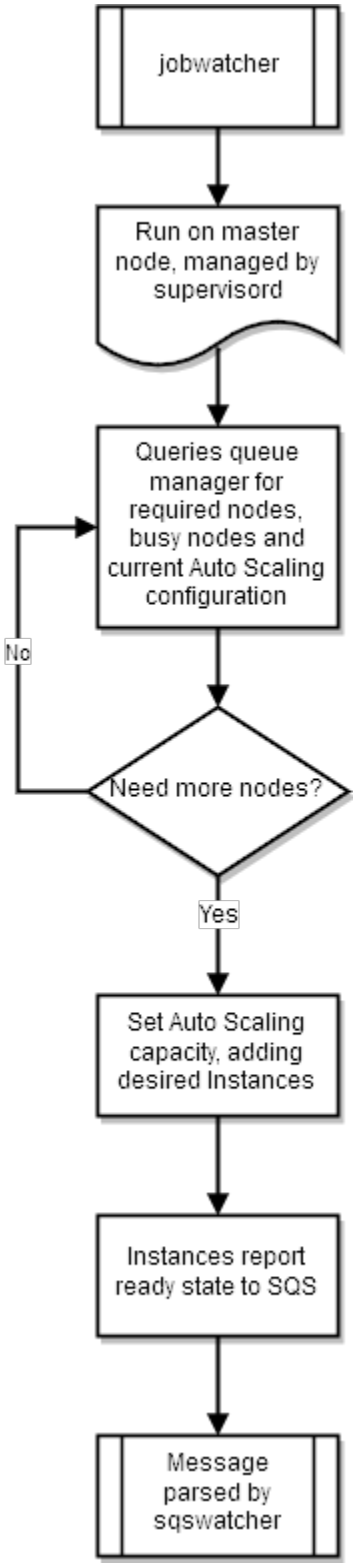
클러스터의 수명 주기는 사용자가 클러스터를 생성한 후 시작됩니다. 일반적으로 클러스터는 명령줄 인터페이스(CLI)에서 생성됩니다. 생성된 클러스터는 삭제될 때까지 존재합니다. AWS ParallelCluster 데몬은 주로 HPC 클러스터 탄력성을 관리하기 위해 클러스터 노드에서 실행됩니다. 다음 다이어그램은 사용자 워크플로우와 클러스터 수명 주기를 보여줍니다. 다음 섹션에서는 클러스터를 관리하는 데 사용되는 AWS ParallelCluster 데몬을 설명합니다.


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 및 Torque 스케줄러를 사용하면 , nodewatcher jobwatcher및 sqswatcher 프로세스를 AWS ParallelCluster 사용합니다.

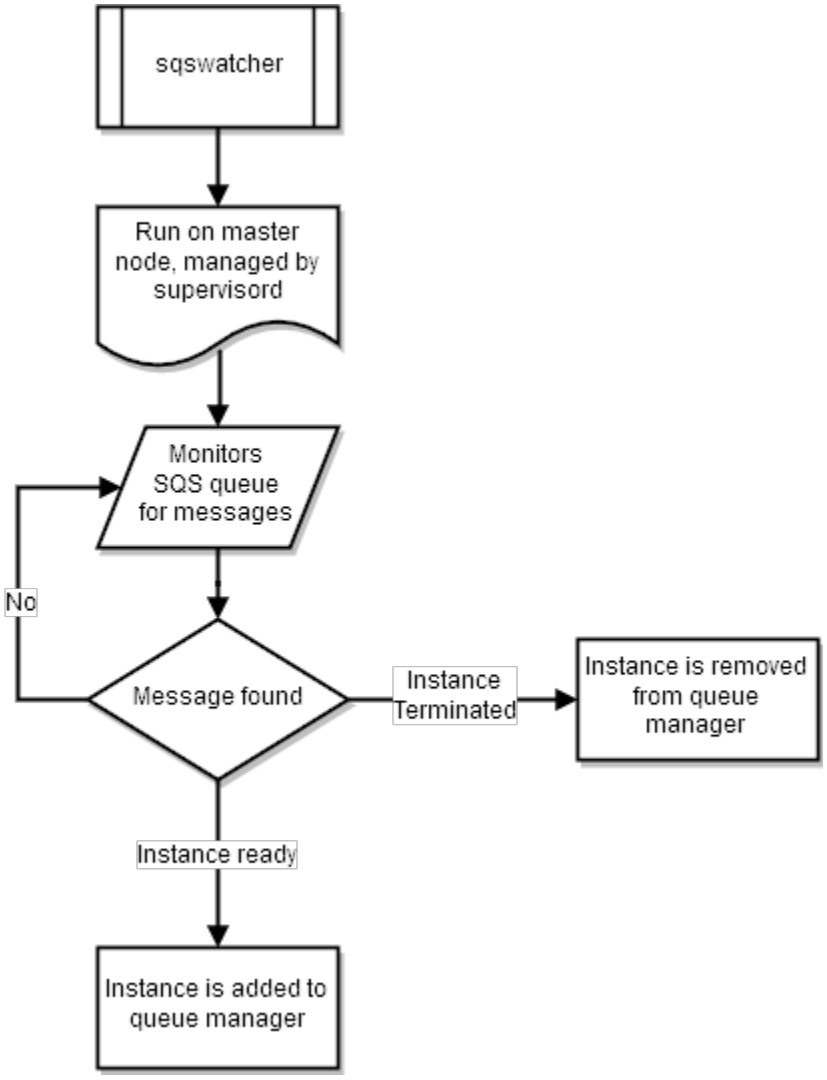
jobwatcher

클러스터가 실행 중인 경우 루트 사용자가 소유한 프로세스가 구성된 스케줄러(SGE 또는 Torque)를 모니터링합니다. 1분마다 대기열을 평가하여 규모 스케일 업 시기를 결정합니다.


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

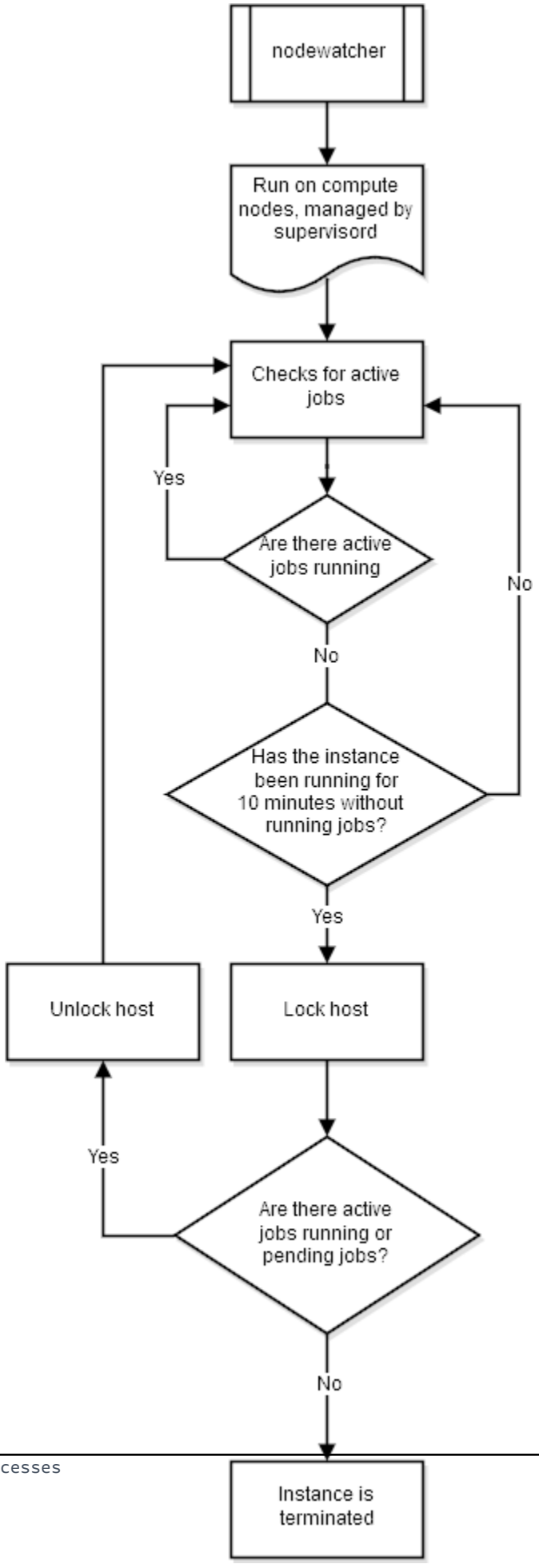
sqswatcher

sqswatcher 프로세스는 Auto Scaling에서 발송된 Amazon SQS 메시지를 모니터링합니다. 이 메시지는 클러스터 내의 상태 변경을 알립니다. 인스턴스가 온라인 상태가 되면 “인스턴스 준비” 메시지를 Amazon SQS에 제출합니다. 이 메시지는 헤드 노드에서 실행 중인 sqs_watcher에 의해 선택됩니다. 이러한 메시지는 새 인스턴스가 온라인으로 전환되거나 종료될 때 대기열 관리자에게 알리기 위해 사용되므로 대기열에 추가하거나 삭제할 수 있습니다.



nodewatcher

nodewatcher 프로세스는 컴퓨팅 플릿의 각 노드에서 실행됩니다. 사용자가 정의한 scaledown_idletime 기간 후에는 인스턴스가 종료됩니다.


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

Slurm 스케줄러, AWS ParallelCluster 및 clustermgtd computemgt 프로세스를 사용합니다.

clustermgtd

이기종 모드(queue_settings 값 지정으로 표시)에서 실행되는 클러스터에는 헤드 노드에서 실행되는 클러스터 관리 대몬(daemon)(clustermgtd) 프로세스가 있습니다. 이러한 작업은 클러스터 관리 대몬(daemon)이 수행합니다.

- 비활성 파티션 정리
- 정적 용량 관리: 정적 용량이 항상 정상 상태인지 확인하세요.
- 스케줄러를 Amazon EC2와 동기화합니다.
- 분리된 인스턴스 정리
- 일시 중지 워크플로 외부에서 발생하는 Amazon EC2 종료 시 스케줄러 노드 상태 복원
- 비정상 Amazon EC2 인스턴스 관리(Amazon EC2 상태 확인 실패)
- 정기 유지 관리 이벤트 관리
- 비정상 스케줄러 노드 관리(스케줄러 상태 점검 실패)

computemgtd

이기종 모드(queue_settings 값 지정으로 표시)에서 실행되는 클러스터에는 각 컴퓨팅 노드에서 실행되는 컴퓨팅 관리 대몬(daemon)(computemgtd) 프로세스가 있습니다. 컴퓨팅 관리 대몬(daemon)은 5분마다 헤드 노드에 연결할 수 있고 정상 상태인지 확인합니다. 헤드 노드에 도달할 수 없거나 정상이 아닌 상태로 5분이 경과하면 컴퓨팅 노드가 종료됩니다.

AWS 에서 사용하는 서비스 AWS ParallelCluster

다음 Amazon Web Services(AWS) 서비스가에서 사용됩니다 AWS ParallelCluster.

주제

- 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 컨테이너 레지스트리
- 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(S3)
- Amazon VPC

AWS Auto Scaling

> **Note**
>
> 이 섹션은 AWS ParallelCluster 버전 2.11.4까지의 버전에만 적용됩니다. 버전 2.11.5부터 AWS ParallelCluster 는 사용을 지원하지 않습니다 AWS Auto Scaling.

AWS Auto Scaling 는 애플리케이션을 모니터링하고 변경되는 특정 서비스 요구 사항에 따라 용량을 자동으로 조정하는 서비스입니다. 이 서비스는 ComputeFleet 인스턴스를 Auto Scaling 그룹으로 관리합니다. 그룹은 변화하는 워크로드에 따라 탄력적으로 구동되거나 초기 인스턴스 구성에 따라 정적으로 고정될 수 있습니다.

AWS Auto Scaling 는 ComputeFleet 인스턴스에서 사용되지만 AWS Batch 클러스터에서는 사용되지 않습니다.

에 대한 자세한 내용은 https://aws.amazon.com/autoscaling/ 및 https://docs.aws.amazon.com/autoscaling/ AWS Auto Scaling참조하세요.

AWS Batch

AWS Batch 는 AWS 관리형 작업 스케줄러 서비스입니다. AWS Batch 클러스터에서 컴퓨팅 리소스(예: CPU 또는 메모리 최적화 인스턴스)의 최적의 수량과 유형을 동적으로 프로비저닝합니다. 이러한 리소스는 볼륨 요구 사항을 포함하여 일괄 작업의 특정 요구 사항을 기반으로 프로비저닝됩니다. 를 AWS Batch사용하면 작업을 효과적으로 실행하기 위해 추가 배치 컴퓨팅 소프트웨어 또는 서버 클러스터를 설치하거나 관리할 필요가 없습니다.

AWS Batch 는 AWS Batch 클러스터에만 사용됩니다.

에 대한 자세한 내용은 https://aws.amazon.com/batch/ 및 https://docs.aws.amazon.com/batch/ AWS Batch참조하세요.

AWS CloudFormation

AWS CloudFormation 는 클라우드 환경에서 AWS 및 타사 애플리케이션 리소스를 모델링하고 프로비저닝하기 위한 공통 언어를 제공하는 infrastructure-as-code 서비스입니다. 에서 사용하는 기본 서비스입니다 AWS ParallelCluster. 의 각 클러스터 AWS ParallelCluster 는 스택으로 표시되며 각 클러스터에 필요한 모든 리소스는 AWS ParallelCluster AWS CloudFormation 템플릿 내에 정의됩니다. 대부분의 경우 AWS ParallelCluster CLI 명령은 생성, 업데이트 및 삭제 명령과 같은 스택 명령에 직접 해당합니다 AWS CloudFormation . 클러스터 내에서 시작된 인스턴스는 클러스터가 시작된 AWS 리전 의 AWS CloudFormation 엔드포인트에 HTTPS를 호출합니다.

에 대한 자세한 내용은 https://aws.amazon.com/cloudformation/ 및 https://docs.aws.amazon.com/cloudformation/ AWS CloudFormation참조하세요.

Amazon CloudWatch

Amazon CloudWatch(CloudWatch)는 데이터 및 실행 가능한 인사이트를 제공하는 모니터링 및 관찰 가능 서비스입니다. 이러한 인사이트를 사용하여 애플리케이션을 모니터링하고, 성능 변화 및 서비스 예외에 대응하고, 리소스 활용도를 최적화할 수 있습니다. 에서 AWS ParallelCluster CloudWatch는 대시보드에 사용되어 Docker 이미지 빌드 단계와 AWS Batch 작업 출력을 모니터링하고 로깅합니다.

AWS ParallelCluster 버전 2.10.0 이전에는 CloudWatch가 AWS Batch 클러스터에서만 사용되었습니다.

CloudWatch에 대한 자세한 내용은 https://aws.amazon.com/cloudwatch/ 및 https://docs.aws.amazon.com/cloudwatch/을 참조하세요.

Amazon CloudWatch Logs

Amazon CloudWatch Logs(이하 CloudWatch Logs)는 Amazon CloudWatch의 핵심 기능 중 하나입니다. 이것은 AWS ParallelCluster에서 사용하는 많은 구성 요소에 대한 로그 파일을 모니터링, 저장, 확인 및 검색하는 데 사용됩니다.

AWS ParallelCluster 버전 2.6.0 이전에는 CloudWatch Logs가 AWS Batch 클러스터에서만 사용되었습니다.

자세한 내용은 Amazon CloudWatch Logs와 통합 단원을 참조하십시오.

AWS CodeBuild

AWS CodeBuild (CodeBuild)는 소스 코드를 준수하고, 테스트를 실행하고, 배포할 준비가 된 소프트웨어 패키지를 생성하는 AWS 관리형 지속적 통합 서비스입니다. 에서 AWS ParallelCluster CodeBuild는 클러스터가 생성될 때 도커 이미지를 자동으로 투명하게 빌드하는 데 사용됩니다.

CodeBuild는 AWS Batch 클러스터에만 사용됩니다.

CodeBuild에 대한 자세한 내용은 https://aws.amazon.com/codebuild/ 및 https://docs.aws.amazon.com/codebuild/을 참조하세요.

Amazon DynamoDB

Amazon DynamoDB(이하 DynamoDB)는 빠르고 유연한 NoSQL 데이터베이스 서비스입니다. 이것은는 클러스터의 최소 상태 정보를 저장하는 데 사용됩니다. 헤드 노드는 DynamoDB 테이블의 프로비저닝된 인스턴스를 추적합니다.

DynamoDB는 AWS Batch 클러스터와 함께 사용되지 않습니다.

DynamoDB에 대한 자세한 내용은 https://aws.amazon.com/dynamodb/ 및 https://docs.aws.amazon.com/dynamodb/을 참조하세요.

Amazon Elastic Block Store

Amazon Elastic Block Store(Amazon EBS)는 공유 볼륨을 위한 영구 스토리지를 제공하는 고성능 블록 스토리지 서비스입니다. 모든 Amazon EBS 설정을 구성을 통해 전달할 수 있습니다. Amazon EBS 볼륨은 빈 상태로 초기화하거나 기존 Amazon EBS 스냅샷에서 초기화할 수 있습니다.

Amazon EBS에 대한 자세한 내용은 https://aws.amazon.com/ebs/ 및 https://docs.aws.amazon.com/ebs/을 참조하세요.

Amazon Elastic Compute Cloud

Amazon Elastic Compute Cloud(Amazon EC2)는 컴퓨팅 용량을 제공합니다 AWS ParallelCluster. 헤드 및 컴퓨팅 노드는 Amazon EC2 인스턴스입니다. HVM을 지원하는 모든 인스턴스 유형을 선택할 수 있습니다. 헤드 노드와 컴퓨팅 노드는 서로 다른 인스턴스 유형일 수 있습니다. 또한 여러 대기열을 사용하는 경우 컴퓨팅 노드 일부 또는 전체를 스팟 인스턴스로 시작할 수도 있습니다. 인스턴스에 있는 인스턴스 스토어 볼륨은 스트라이프된 LVM 볼륨으로 탑재됩니다.

Amazon EC2에 대한 자세한 내용은 https://aws.amazon.com/ec2/ 및 https://docs.aws.amazon.com/ec2/을 참조하세요.

Amazon Elastic 컨테이너 레지스트리

Amazon Elastic Container Registry(Amazon ECR)는 개발자가 Docker 컨테이너 이미지를 간편하게 저장, 관리 및 배포할 수 있게 해주는 완전관리형 Docker 컨테이너 레지스트리입니다. 에서 AWS ParallelCluster Amazon ECR은 클러스터가 생성될 때 빌드된 Docker 이미지를 저장합니다. 그런 다음 Docker 이미지는에서 제출된 작업의 컨테이너를 실행하는 AWS Batch 데 사용됩니다.

Amazon ECR은 AWS Batch 클러스터에만 사용됩니다.

자세한 내용은 https://aws.amazon.com/ecr/ 및 https://docs.aws.amazon.com/ecr/을 참조하세요.

Amazon EFS

Amazon Elastic File System(Amazon EFS)는 AWS 클라우드 서비스 및 온프레미스 리소스와 함께 사용할 수 있는 간단하고 확장 가능한 완전 관리형 탄력적 NFS 파일 시스템을 제공합니다. Amazon EFS는 `efs_settings` 설정이 지정되고 `[efs]` 섹션을 참조할 때 사용됩니다. Amazon EFS에 대한 지원이 AWS ParallelCluster 버전 2.1.0에 추가되었습니다.

Amazon EFS에 대한 자세한 내용은 https://aws.amazon.com/efs/ 및 https://docs.aws.amazon.com/efs/를 참조하세요.

Amazon FSx for Lustre

FSx for Lustre는 오픈 소스 Lustre 파일 시스템을 사용하는 고성능 파일 시스템을 제공합니다. FSx for Lustre는 `fsx_settings` 설정이 지정되고 `[fsx]` 섹션을 참조할 때 사용됩니다. FSx for Lustre에 대한 지원이 AWS ParallelCluster 버전 2.2.1에서 추가되었습니다.

FSx for Lustre에 대한 자세한 내용은 https://aws.amazon.com/fsx/lustre/ 및 https://docs.aws.amazon.com/fsx/을 참조하세요.

AWS Identity and Access Management

AWS Identity and Access Management (IAM)는 내부에서 각 개별 클러스터에 고유한 인스턴스에 대해 Amazon EC2에 대한 최소 권한의 IAM 역할을 제공하는 AWS ParallelCluster 데 사용됩니다. AWS ParallelCluster 인스턴스에는 클러스터를 배포하고 관리하는 데 필요한 특정 API 호출에 대해서만 액세스 권한이 부여됩니다.

AWS Batch 클러스터를 사용하면 클러스터가 생성될 때 Docker 이미지 구축 프로세스와 관련된 구성 요소에 대한 IAM 역할도 생성됩니다. 이러한 구성 요소에는 Amazon ECR 리포지토리에, 또는 그 리포지토리로부터 도커 이미지를 추가 및 삭제할 수 있는 Lambda 함수가 포함됩니다. 또한 클러스터 및 CodeBuild 프로젝트용으로 생성된 Amazon S3 버킷을 삭제할 수 있는 함수도 포함됩니다. AWS Batch 리소스, 인스턴스 및 작업에 대한 역할도 있습니다.

IAM에 대한 자세한 내용은 https://aws.amazon.com/iam/ 및 https://docs.aws.amazon.com/iam/을 참조하세요.

AWS Lambda

AWS Lambda (Lambda)는 Docker 이미지 생성을 오케스트레이션하는 함수를 실행합니다. 또한 Lambda는 Amazon ECR 리포지토리 및 Amazon S3에 저장된 도커 이미지와 같은 사용자 지정 클러스터 리소스의 정리를 관리합니다.

Lambda에 대한 자세한 내용은 https://aws.amazon.com/lambda/ 및 https://docs.aws.amazon.com/lambda/를 참조하세요.

Amazon DCV

Amazon DCV는 다양한 네트워크 조건에서 모든 장치에 원격 데스크톱 및 애플리케이션 스트리밍을 제공하는 안전한 방법을 제공하는 고성능 원격 디스플레이 프로토콜입니다. Amazon DCV는 `dcv_settings` 설정이 지정되고 `[dcv]` 섹션을 참조할 때 사용됩니다. Amazon DCV에 대한 지원이 AWS ParallelCluster 버전 2.5.0에 추가되었습니다.

Amazon DCV에 대한 자세한 내용은 https://aws.amazon.com/hpc/dcv/ 및 https://docs.aws.amazon.com/dcv/을 참조하세요.

Amazon Route 53

Amazon Route 53(Route 53)은 각 컴퓨팅 노드에 대해 호스트 이름과 정규화된 도메인 이름을 포함하는 호스팅 영역을 생성하는 데 사용됩니다.

Route 53에 대한 자세한 내용은 https://aws.amazon.com/route53/ 및 https://docs.aws.amazon.com/route53/을 참조하세요.

Amazon Simple Notification Service

> **Note**
>
> 이 섹션은 AWS ParallelCluster 버전 2.11.4까지의 버전에만 적용됩니다. 버전 2.11.5부터는 AWS ParallelCluster 가 Amazon Simple Notification Service 사용을 지원하지 않습니다.

Amazon Simple Notification Service(SNS) 는 Auto Scaling으로부터 알림을 받습니다. 이러한 이벤트는 수명 주기 이벤트라고 하며 Auto Scaling 그룹에서 인스턴스가 시작되거나 종료될 때 생성됩니다. 내에서 Auto Scaling 그룹에 대한 AWS ParallelCluster Amazon SNS 주제는 Amazon SQS 대기열에 구독됩니다.

Amazon SNS는 AWS Batch 클러스터와 함께 사용되지 않습니다.

Amazon SNS에 대한 자세한 내용은 https://aws.amazon.com/sns/ 및 https://docs.aws.amazon.com/sns/를 참조하세요.

Amazon Simple Queue Service

> **Note**
>
> 이 섹션은 AWS ParallelCluster 버전 2.11.4까지의 버전에만 적용됩니다. 버전 2.11.5부터는 AWS ParallelCluster 가 Amazon Simple Queue Service 사용을 지원하지 않습니다.

Amazon Simple Queue Service(Amazon SQS)는 Auto Scaling에서 전송된 알림, Amazon SNS를 통해 전송된 알림, 컴퓨팅 노드에서 전송된 알림을 보관합니다. Amazon SQS는 알림 전송과 알림 수신을 분리합니다. 이렇게 하면 헤드 노드가 폴링 프로세스를 통해 알림을 처리할 수 있습니다. 이 프로세스에서 헤드 노드는 Amazon SQSwatcher를 실행하고 대기열을 폴링합니다. Auto Scaling 및 컴퓨팅 인스턴스는 메시지를 대기열에 게시합니다.

Amazon SQS는 AWS Batch 클러스터와 함께 사용되지 않습니다.

Amazon SQS에 대한 자세한 내용은 https://aws.amazon.com/sqs/ 및 https://docs.aws.amazon.com/sqs/을 참조하세요.

Amazon Simple Storage Service(S3)

Amazon Simple Storage Service(Amazon S3)는 각에 위치한 AWS ParallelCluster 템플릿을 저장합니다 AWS 리전. CLI/SDK 도구가 Amazon S3를 사용할 수 있도록 구성할 수 AWS ParallelCluster 있습니다.

AWS Batch 클러스터를 사용하면 계정의 Amazon S3 버킷이 관련 데이터를 저장하는 데 사용됩니다. 예를 들어, 이 버킷은 도커 이미지와 스크립트가 제출된 작업에서 생성될 때 생성된 아티팩트를 저장합니다.

자세한 내용은 https://aws.amazon.com/s3/ 및 https://docs.aws.amazon.com/s3/을 참조하세요.

Amazon VPC

Amazon VPC는 클러스터의 노드가 사용하는 네트워크를 정의합니다. 클러스터의 VPC 설정은 [vpc] 섹션에 정의되어 있습니다.

아마존 VPC에 대한 자세한 내용은 https://aws.amazon.com/vpc/ 및 https://docs.aws.amazon.com/vpc/을 참조하세요.

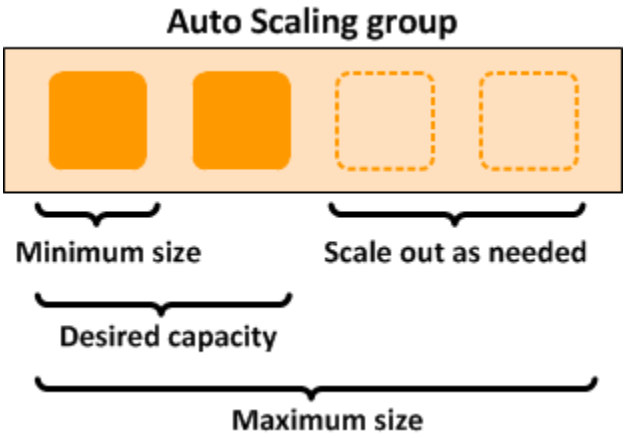
AWS ParallelCluster Auto Scaling

> **Note**
>
> 이 섹션은 AWS ParallelCluster 버전 2.11.4까지의 버전에만 적용됩니다. 버전 2.11.5부터는 AWS ParallelCluster 가 SGE 또는 Torque 스케줄러의 사용을 지원하지 않습니다. 2.11.4 이하의 버전에서는 계속 사용할 수 있지만 서비스 및 AWS 지원 팀의 향후 업데이트 또는 문제 해결 지원을 AWS 받을 수 없습니다.
> AWS ParallelCluster 버전 2.9.0부터 Auto Scaling은 Slurm Workload Manager (Slurm)에서 사용할 수 없습니다. Slurm 및 다중 대기열 스케일링에 대한 자세한 내용은 다중 대기열 모드 자습서를 참조하세요.

이 항목에 설명된 Auto Scaling 전략은 Son of Grid Engine(SGE) 또는 Torque Resource Manager(Torque) 중 하나를 사용하여 배포되는 HPC 클러스터에 적용됩니다. 이러한 스케줄러 중 하나를 사용하여 배포한 경우 AWS ParallelCluster 는 컴퓨팅 노드의 Auto Scaling 그룹을 관리하고 필요에 따라 스케줄러 구성을 변경하여 조정 기능을 구현합니다. 기반 HPC 클러스터의 경우 AWS Batch는

AWS 관리형 작업 스케줄러에서 제공하는 탄력적 조정 기능을 AWS ParallelCluster 사용합니다. 자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 Amazon EC2 Auto Scaling이란?을 참조하세요.

와 함께 배포된 클러스터 AWS ParallelCluster 는 여러 가지 방식으로 탄력적입니다. `initial_queue_size`는 ComputeFleet Auto Scaling 그룹의 최소 크기 값을 설정하고 원하는 용량 값도 설정합니다. `max_queue_size`는 ComputeFleet Auto Scaling 그룹의 최대 크기 값을 설정합니다.



확장

`jobwatcher`라는 프로세스가 1분마다 헤드 노드에서 실행됩니다. 이 프로세스는 대기열에서 보류 중인 작업에 필요한 현재 인스턴스 수를 평가합니다. 사용 중인 노드와 요청된 노드의 총 수가 Auto Scaling 그룹에서 현재 원하는 값보다 크면 인스턴스가 추가됩니다. 추가 작업을 제출하면 대기열이 다시 평가되고 Auto Scaling 그룹이 지정된 `max_queue_size`까지 업데이트됩니다.

SGE 스케줄러에서는 각 작업에 실행할 슬롯이 여러 개 필요합니다(슬롯 하나가 vCPU와 같은 처리 장치 하나에 해당). 현재 대기 중인 작업을 제공하는 데 필요한 인스턴스 수를 평가할 때 `jobwatcher`는 요청된 슬롯의 총 수를 단일 컴퓨팅 노드의 용량으로 나눕니다. 사용 가능한 vCPU 수인 컴퓨팅 노드 용량은 클러스터 구성에서 지정된 Amazon EC2 인스턴스 유형에 따라 다릅니다.

Slurm (AWS ParallelCluster 버전 2.9.0 이전) 및 Torque 스케줄러를 사용하면 상황에 따라 각 작업에 노드 수와 모든 노드에 대한 슬롯 수가 모두 필요할 수 있습니다. 각 요청에 대해 `jobwatcher`는 새로운 계산 요구 사항을 충족시키는 데 필요한 컴퓨팅 노드 수를 결정합니다. 예를 들어, 컴퓨팅 인스턴스 유형이 `c5.2xlarge`(8 vCPU)이고, 다음과 같은 요구 사항을 가진 3개의 작업이 대기열에 보류 중이라고 가정합니다.

- job1: 노드 2개/각각 슬롯 4개
- job2: 노드 3개/각각 슬롯 2개

- job3: 노드 1개/각각 슬롯 4개

이 예제에서 `jobwatcher`는 3개의 작업에 대해 사용할 3개의 새로운 컴퓨팅 인스턴스가Auto Scaling 그룹에 필요합니다.

현재 제한: 자동 스케일 업 로직에서는 부분적으로 로드된 사용 중인 노드를 고려하지 않습니다. 예를 들어, 작업을 실행 중인 노드는 빈 슬롯이 있더라도 사용 중인 노드로 간주됩니다.

축소

각 컴퓨팅 노드에서 `nodewatcher` 프로세스가 실행되고 노드의 유휴 시간을 평가합니다. 다음 조건이 모두 충족되면 인스턴스가 종료됩니다.

- 인스턴스에 `scaledown_idletime`(기본 설정: 10분)보다 기간이 긴 작업이 없는 경우
- 클러스터에 보류 중인 작업이 없는 경우

인스턴스를 종료하기 위해 `nodewatcher`는 TerminateInstanceInAutoScalingGroup API 작업을 호출합니다. 이 작업은 Auto Scaling 그룹 크기가 최소 Auto Scaling 그룹 크기인 경우에 인스턴스를 제거합니다. 이 프로세스는 실행 중인 작업에 영향을 미치지 않고 클러스터를 축소합니다. 또한 기본 인스턴스 수가 고정된 탄력적인 클러스터를 활성화할 수 있습니다.

정적 클러스터

자동 확장 값은 HPC의 경우 다른 워크로드와 동일합니다. 유일한 차이점은 AWS ParallelCluster에 더 지능적으로 상호 작용하는 코드가 있다는 것입니다. 예를 들어, 정적 클러스터가 필요한 경우 `initial_queue_size` 및 `max_queue_size` 파라미터를 필요한 클러스터의 정확한 크기로 설정한 다음 `maintain_initial_size` 파라미터를 true로 설정합니다. 이렇게 하면 ComputeFleet Auto Scaling 그룹은 최소, 최대 및 원하는 용량에 대해 동일한 값을 갖게 됩니다.

자습서

다음 자습서에서는를 시작하는 방법을 보여주고 몇 가지 일반적인 작업에 대한 모범 사례 지침을 AWS ParallelCluster제공합니다.

주제

- 에서 첫 번째 작업 실행 AWS ParallelCluster
- 사용자 지정 AWS ParallelCluster AMI 빌드
- AWS ParallelCluster 및 awsbatch 스케줄러를 사용하여 MPI 작업 실행
- 사용자 지정 KMS 키를 사용한 디스크 암호화
- 다중 대기열 모드 자습서

에서 첫 번째 작업 실행 AWS ParallelCluster

이 자습서에서는 첫 번째 Hello World 작업을 실행하는 방법을 안내합니다 AWS ParallelCluster.

사전 조건

- AWS ParallelCluster 가 설치됩니다.
- 설치 AWS CLI 및 구성됩니다.
- EC2 키 페어가 있습니다.
- `pcluster` CLI를 실행하는 데 필요한 권한을 가진 IAM 역할이 있습니다.

설치 확인

먼저 AWS ParallelCluster 가 올바르게 설치 및 구성되었는지 확인합니다.

```
$ pcluster version
```

그러면의 실행 버전이 반환됩니다 AWS ParallelCluster. 구성에 대한 메시지가 출력되는 경우 다음을 실행하여 AWS ParallelCluster를 구성해야 합니다.

```
$ pcluster configure
```

첫 번째 클러스터 생성

이제 첫 번째 클러스터를 생성할 시간입니다. 이 자습서의 워크로드는 성능 집약적이 아니기 때문에 `t2.micro`의 기본 인스턴스 크기를 사용합니다. (프로덕션 워크로드의 경우 더 적합한 인스턴스 크기를 선택해야 합니다.)

클러스터 이름을 hello-world라고 하겠습니다.

```
$ pcluster create hello-world
```

클러스터가 생성되면 다음과 비슷한 출력이 표시됩니다.

```
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/
```

`CREATE_COMPLETE` 메시지는 클러스터가 성공적으로 생성되었음을 보여 줍니다. 헤드 노드의 공개 및 비공개 IP 주소도 알려줍니다. 로그인하려면 이 IP가 필요합니다.

헤드 노드에 로그인

OpenSSH pem 파일을 사용하여 헤드 노드에 로그인합니다.

```
pcluster ssh hello-world -i /path/to/keyfile.pem
```

로그인되면 `qhost` 명령을 실행하여 컴퓨팅 노드가 설정되고 구성되어 있는지 확인합니다.

```
$ 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
```

출력을 통해 클러스터에 두 개의 컴퓨팅 노드가 있고 두 개 모두 2개의 스레드를 사용할 수 있다는 것을 알 수 있습니다.

SGE를 사용하여 첫 번째 작업 실행

> **Note**
>
> 이 예제는 AWS ParallelCluster 버전 2.11.4까지의 버전에만 적용됩니다. 버전 2.11.5부터는 AWS ParallelCluster 가 SGE 또는 Torque 스케줄러의 사용을 지원하지 않습니다.

이제 잠시 동안 대기한 다음 고유의 호스트 이름을 출력하는 작업을 생성합니다.

다음 콘텐츠를 통해 `hellojob.sh`라는 파일을 생성합니다.

```
#!/bin/bash
sleep 30
echo "Hello World from $(hostname)"
```

그런 다음 `qsub`를 사용하여 작업을 제출하고 작업이 실행되는지 확인합니다.

```
$ qsub hellojob.sh
Your job 1 ("hellojob.sh") has been submitted
```

이제 대기열을 보고 작업 상태를 확인할 수 있습니다.

```
$ 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
```

출력을 통해 작업이 현재 실행 중 상태인 것을 알 수 있습니다. 작업이 완료될 때까지 30초 동안 기다린 후 `qstat`을 다시 실행합니다.

```
$ qstat
$
```

이제 대기열에 작업이 없으므로 현재 디렉터리에서 출력을 확인할 수 있습니다.

```
$ 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
```

출력에서 작업 스크립트에 "e1" 및 "o1" 파일이 있다는 것을 알 수 있습니다. e1 파일이 비어 있으므로 stderr에 대한 출력이 없습니다. o1 파일을 보면 작업의 출력을 볼 수 있습니다.

```
$ cat hellojob.sh.o1
Hello World from ip-192-168-1-125
```

출력에서는 작업이 ip-192-168-1-125 인스턴스에서 성공적으로 실행되었음을 보여줍니다.

클러스터 생성 및 사용에 대해 자세히 알아보려면 모범 사례 섹션을 참조하세요.

사용자 지정 AWS ParallelCluster AMI 빌드

> **Important**
>
> 사용자 지정 AMI 빌드는 AWS ParallelCluster를 사용자 지정하기 위해 권장되는 접근 방식이 아닙니다.
> 자체 AMI를 빌드한 후에는 향후 릴리스에서 업데이트 또는 버그 수정이 더 이상 수신되지 않기 때문입니다 AWS ParallelCluster. 또한 사용자 지정 AMI를 빌드하는 경우 새 AWS ParallelCluster 릴리스마다 사용자 지정 AMI를 생성하는 데 사용한 단계를 반복해야 합니다.

더 읽기 전에 먼저 사용자 지정 부트스트랩 작업 섹션을 확인하여 수정하려는 내용을 향후 AWS ParallelCluster 릴리스에서 스크립팅하고 지원할 수 있는지 확인하는 것이 좋습니다.

사용자 지정 AMI를 구축하는 것이 이상적이지 않더라도(앞에서 언급한 이유로 인해)에 대한 사용자 지정 AMI를 빌드해야 하는 시나리오 AWS ParallelCluster 는 여전히 있습니다. 이 자습서는 이러한 시나리오에 맞게 사용자 지정 AMI를 구축하는 프로세스를 안내합니다.

> **Note**
>
> AWS ParallelCluster 버전 2.6.1부터는 노드를 시작할 때 대부분의 설치 레시피를 기본적으로 건너뜁니다. 이는 시작 시간을 개선하기 위한 것입니다. 시작 시간을 희생하면서 이전 버전과의 호환성을 개선하기 위해 모든 설치 레시피를 실행하려면 `extra_json` 설정의 `cluster` 키에 `"skip_install_recipes" : "no"`를 추가하세요. 예시:
>
> ```
> extra_json = { "cluster" : { "skip_install_recipes" : "no" } }
> ```

사전 조건

- AWS ParallelCluster 가 설치됩니다.
- 설치 AWS CLI 및 구성됩니다.
- EC2 키 페어가 있습니다.
- `pcluster` CLI를 실행하는 데 필요한 권한을 가진 IAM 역할이 있습니다.

AWS ParallelCluster AMI를 사용자 지정하는 방법

다음 단원에서 설명하는 사용자 지정 AWS ParallelCluster AMI를 사용하는 세 가지 방법이 있습니다. 그 세 가지 중 두 가지 방법을 사용하려면 AWS 계정에서 사용할 수 있는 새로운 AMI를 빌드해야 합니다. 세 번째 방법(런타임 시 사용자 지정 AMI 사용)은 미리 빌드할 필요는 없지만 배포에 위험을 가중시킵니다. 필요에 맞는 방법을 선택하세요.

AMI 수정

이 방법이 가장 안전하고 권장되는 방법입니다. 기본 AWS ParallelCluster AMI는 종종 새 릴리스로 업데이트되기 때문에이 AMI에는 설치 및 구성 시가 작동하는 AWS ParallelCluster 데 필요한 모든 구성 요소가 있습니다. 이 AMI를 기본으로 사용하여 시작할 수 있습니다.

New EC2 console

1. AWS ParallelCluster AMI 목록에서 사용하는 특정에 해당하는 AMI AWS 리전 를 찾습니다. 선택한 AMI 목록은 AWS ParallelCluster 사용하는 버전과 일치해야 합니다. `pcluster version`을 실행하여 버전을 확인합니다. AWS ParallelCluster 버전 2.11.9의 경우 https://github.com/aws/aws-parallelcluster/blob/v2.11.9/amis.txt 이동합니다. 다른 버전을 선택하려면 동일한 링크를 사용하고 태그: 2.11.9 버튼을 선택하고 태그 탭을 선택한 다음 적절한 버전을 선택합니다.

2. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/ec2/ Amazon EC2 콘솔을 엽니다.
3. Amazon EC2 대시보드에서 인스턴스 시작을 선택하세요.
4. 애플리케이션 및 OS 이미지에서 더 많은 AMIs 선택하고 커뮤니티 AMIs로 이동한 다음 검색 상자에의 AWS ParallelCluster AMI ID AWS 리전 를 입력합니다.
5. AMI를 선택하고, 인스턴스 유형과 속성을 선택한 다음 키 페어를 선택하고 인스턴스를 시작합니다.
6. OS 사용자와 SSH 키를 사용하여 인스턴스에 로그인합니다. 자세한 내용은 인스턴스로 이동하여 새 인스턴스를 선택하고 연결하세요.
7. 필요에 따라 인스턴스를 사용자 지정합니다.
8. 다음 명령을 실행하여 AMI 생성을 위해 인스턴스를 준비합니다.

   ```
   sudo /usr/local/sbin/ami_cleanup.sh
   ```

9. 인스턴스를 중지하려면 인스턴스를 선택하고 인스턴스 상태, 인스턴스 중지를 차례로 선택합니다.
10. EC2 콘솔 또는 AWS CLI create-image를 사용하여 인스턴스에서 새 AMI를 생성합니다.

    EC2 콘솔에서

    a. 탐색 창에서 인스턴스를 선택합니다.

    b. 생성하고 수정한 인스턴스를 선택합니다.

    c. 작업, 이미지 및 템플릿, 이미지 생성을 차례로 선택합니다.

    d. 이미지 생성을 선택합니다.
11. 클러스터 구성 내의 custom_ami 필드에 새 AMI ID를 입력합니다.

Old EC2 console

1. AWS ParallelCluster AMI 목록에서 사용하는 특정에 해당하는 AMI AWS 리전 를 찾습니다. 선택한 AMI 목록은 AWS ParallelCluster 사용하는 버전과 일치해야 합니다. `pcluster version`을 실행하여 버전을 확인합니다. AWS ParallelCluster 버전 2.11.9의 경우 https://github.com/aws/aws-parallelcluster/blob/v2.11.9/amis.txt 이동합니다. 다른 버전을 선택하려면 동일한 링크를 사용하고 태그: 2.11.9 버튼을 선택하고 태그 탭을 선택한 다음 적절한 버전을 선택합니다.
2. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/ec2/ Amazon EC2 콘솔을 엽니다.

3. Amazon EC2 대시보드에서 인스턴스 시작을 선택하세요.
4. 커뮤니티 AMIs 선택하고 AWS ParallelCluster AMI ID를 검색한 다음 선택합니다.
5. 인스턴스 유형을 선택하고 다음: 인스턴스 세부 정보 구성 또는 검토 및 시작을 선택하여 인스턴스를 시작합니다.
6. 시작을 선택하고 키 페어를 선택한 다음 인스턴스를 시작합니다.
7. OS 사용자와 SSH 키를 사용하여 인스턴스에 로그인합니다. 자세한 내용은 인스턴스로 이동하여 새 인스턴스를 선택하고 연결하세요.
8. 필요에 따라 인스턴스를 사용자 지정합니다.
9. 다음 명령을 실행하여 AMI 생성을 위해 인스턴스를 준비합니다.

```
sudo /usr/local/sbin/ami_cleanup.sh
```

10. 인스턴스로 이동한 후, 새 인스턴스를 선택하고 인스턴스 상태, 중지를 차례로 선택합니다.
11. EC2 콘솔 또는 AWS CLI create-image를 사용하여 인스턴스에서 새 AMI를 생성합니다.

    EC2 콘솔에서

    a. 탐색 창에서 인스턴스를 선택합니다.

    b. 생성하고 수정한 인스턴스를 선택합니다.

    c. 작업, 이미지, 이미지 생성을 차례로 선택합니다.

    d. 이미지 생성을 선택합니다.
12. 클러스터 구성 내의 custom_ami 필드에 새 AMI ID를 입력합니다.

사용자 지정 AWS ParallelCluster AMI 빌드

사용자 지정 AMI와 소프트웨어가 이미 배치된 경우 이를 기반으로 AWS ParallelCluster 에 필요한 변경 사항을 적용할 수 있습니다.

1. AWS ParallelCluster CLI와 함께 로컬 시스템에 다음을 설치합니다.

- Packer: Packer 웹 사이트에서 최신 OS 버전을 검색하여 설치합니다. 버전은 1.4.0 이상이어야 하지만 최신 버전을 사용하는 것이 좋습니다. PATH에서 `packer` 명령을 사용할 수 있는지 확인하세요.

> **Note**
>
> AWS ParallelCluster 버전 2.8.0 이전에는 Berkshelf(를 사용하여 설치됨`gem install berkshelf`)가를 사용해야 했습니다`pcluster createami`.

2. Packer가 사용자를 대신하여 AWS API 작업을 호출할 수 있도록 AWS 계정 자격 증명을 구성합니다. Packer가 작동하는 데 필요한 최소 권한 세트는 Packer 설명서의 Amazon AMI Builder 항목의 IAM 태스크 또는 인스턴스 역할 섹션에 설명되어 있습니다.
3. AWS ParallelCluster CLI`createami`에서 명령을 사용하여 기본으로 제공하는 AMI부터 시작하여 AWS ParallelCluster AMI를 빌드합니다.

```
pcluster createami --ami-id <BASE_AMI> --os <BASE_AMI_OS>
```

> **Important**
>
> 실행 중인 클러스터의 AWS ParallelCluster AMI를 `createami` 명령에 *<BASE_AMI>*로 사용해서는 안 됩니다. 그렇지 않으면 명령이 실패합니다.

   다른 파라미터에 대한 내용은 `pcluster createami`를 참조하세요.
4. 이 명령은 4단계의 Packer를 실행하여 다음 단계를 수행합니다.
   a. 제공된 기본 AMI를 사용하여 인스턴스를 시작합니다.
   b. 인스턴스에 AWS ParallelCluster 쿡북을 적용하여 관련 소프트웨어를 설치하고 기타 필요한 구성 작업을 수행합니다.
   c. 인스턴스를 중지합니다.
   d. 인스턴스에서 새 AMI를 생성합니다.
   e. AMI가 생성된 후 인스턴스를 종료합니다.
   f. 클러스터를 생성하는 데 사용할 새 AMI ID를 출력합니다.
5. 클러스터를 생성하려면 클러스터 구성 내의 custom_ami 필드에 AMI ID를 입력합니다.

> **Note**
>
> 사용자 지정 AWS ParallelCluster AMI를 빌드하는 데 사용되는 인스턴스 유형은 입니다 `t2.xlarge`. 이 인스턴스 유형은 AWS 프리 티어에 적합하지 않으므로이 AMI를 빌드할 때 생성된 모든 인스턴스에 대해 요금이 부과됩니다.

런타임 시 사용자 지정 AMI 사용

> **Warning**
>
> 호환되지 않는 AMI를 사용할 위험을 방지하려면이 방법을 사용하지 않는 것이 AWS ParallelCluster좋습니다.
> 런타임 시 잠재적으로 테스트되지 않은 AMIs로 컴퓨팅 노드 AWS ParallelCluster를 시작하는 경우의 필수 소프트웨어의 런타임 설치와 호환되지 않으면 AWS ParallelCluster 가 작동하지 않을 수 있습니다.

어떤 것도 미리 생성하지 않으려면 AMI를 사용하고 해당 AMI AWS ParallelCluster 에서를 생성할 수 있습니다.

이 방법을 사용하면 클러스터를 생성할 AWS ParallelCluster 때에 필요한 모든 소프트웨어를 설치해야 하므로를 생성하는 AWS ParallelCluster 데 시간이 더 오래 걸립니다. 게다가 스케일 업에도 시간이 더 오래 걸립니다.

- 클러스터 구성 내의 custom_ami 필드에 AMI ID를 입력합니다.

AWS ParallelCluster 및 **`awsbatch`** 스케줄러를 사용하여 MPI 작업 실행

이 자습서는 `awsbatch`를 스케줄러로 사용하여 MPI 작업을 실행하는 과정을 단계별로 안내합니다.

사전 조건

- AWS ParallelCluster 가 설치됩니다.
- 설치 AWS CLI 및 구성됩니다.
- EC2 키 페어가 있습니다.

- pcluster CLI를 실행하는 데 필요한 권한을 가진 IAM 역할이 있습니다.

클러스터 생성

먼저, awsbatch를 스케줄러로 사용하는 클러스터에 대한 구성을 생성합니다. vpc 섹션과 key_name 필드에서 누락된 데이터를 구성 시 생성한 리소스로 삽입해야 합니다.

```
[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-#######
```

이제 클러스터 생성을 시작할 수 있습니다. *awsbatch-tutorial* 클러스터를 직접 호출하려고 합니다.

```
$ pcluster create -c /path/to/the/created/config/aws_batch.config -t awsbatch awsbatch-tutorial
```

클러스터가 생성되면 다음과 비슷한 출력이 표시됩니다.

```
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
```

헤드 노드에 로그인

AWS ParallelCluster 배치 CLI 명령은 모두가 설치된 클라이언트 시스템에서 사용할 AWS ParallelCluster 수 있습니다. 그러나 헤드 노드에 대해 SSH로 접속하고 작업을 제출하겠습니다. 이를 통해 헤드와 AWS Batch 작업을 실행하는 모든 Docker 인스턴스 간에 공유되는 NFS 볼륨을 활용할 수 있습니다.

SSH pem 파일을 사용하여 헤드 노드에 로그인합니다.

```
$ pcluster ssh awsbatch-tutorial -i /path/to/keyfile.pem
```

로그인하면 awsbqueues 및 명령을 실행awsbhosts하여 구성된 AWS Batch 대기열과 실행 중인 Amazon ECS 인스턴스를 표시합니다.

```
[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
```

출력에 나타난 대로 하나의 단일 실행 호스트가 있습니다. 그 이유는 구성에서 min_vcpus에 대해 선택한 값 때문입니다. AWS Batch 대기열 및 호스트에 대한 추가 세부 정보를 표시하려면 명령에 -d 플래그를 추가합니다.

를 사용하여 첫 번째 작업 실행 AWS Batch

MPI로 이동하기 전에 잠시 동안 대기한 다음 고유의 호스트 이름을 출력하는 더미 작업을 생성하여 파라미터로 전달된 이름에게 인사합니다.

다음 콘텐츠가 포함된 "hellojob.sh"라는 파일을 생성합니다.

```
#!/bin/bash

sleep 30
echo "Hello $1 from $HOSTNAME"
echo "Hello $1 from $HOSTNAME" > "/shared/secret_message_for_${1}_by_
${AWS_BATCH_JOB_ID}"
```

그런 다음 awsbsub를 사용하여 작업을 제출하고 작업이 실행되는지 확인합니다.

```
$ awsbsub -jn hello -cf hellojob.sh Luca
Job 6efe6c7c-4943-4c1a-baf5-edbfeccab5d2 (hello) has been submitted.
```

대기열을 보고 작업 상태를 확인합니다.

```
$ awsbstat
jobId                                    jobName      status    startedAt
 stoppedAt    exitCode
-------------------------------------  -----------  --------  -------------------
 -----------  ----------
6efe6c7c-4943-4c1a-baf5-edbfeccab5d2  hello        RUNNING   2018-11-12 09:41:29  -
        -
```

출력은 작업에 대한 세부 정보도 제공합니다.

```
$ 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
-------------------------
```

작업은 현재 RUNNING 상태입니다. 작업이 완료될 때까지 30초 동안 기다린 후 awsbstat을 다시 실행합니다.

```
$ awsbstat
jobId                                    jobName      status    startedAt
 stoppedAt    exitCode
-------------------------------------  -----------  --------  -------------------
 -----------  ----------
```

작업이 SUCCEEDED 상태임을 확인할 수 있습니다.

```
$ 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
```

이제 대기열에 작업이 없으므로 awsbout 명령을 통해 출력을 확인할 수 있습니다.

```
$ 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
```

작업이 "ip-172-31-4-234" 인스턴스에서 성공적으로 실행되었음을 확인할 수 있습니다.

또한 `/shared` 디렉터리를 살펴보면 사용자를 위한 비밀 메시지도 찾을 수 있습니다.

이 자습서에 포함되지 않은 사용 가능한 기능을 모두 살펴 보려면 AWS ParallelCluster 배치 CLI 설명서를 참조하세요. 준비가 되었으면 계속해서 MPI 작업을 제출하는 방법을 살펴보겠습니다.

다중 노드 병렬 환경에서 MPI 작업 실행

헤드 노드에 로그인한 상태에서 `/shared` 디렉터리에 `mpi_hello_world.c`라는 파일을 만듭니다. 다음 MPI 프로그램을 파일에 추가합니다.

```
// 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();
}
```

이제 다음 코드를 submit_mpi.sh로 저장합니다.

```
#!/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
```

이제 첫 번째 MPI 작업을 제출하고 세 노드에서 동시에 실행할 준비가 되었습니다.

```
$ awsbsub -n 3 -cf submit_mpi.sh
```

이제 작업 상태를 모니터링하고 이 작업이 RUNNING 상태로 전환될 때까지 기다립니다.

```
$ watch awsbstat -d
```

작업이 RUNNING 상태로 전환되면 출력을 살펴볼 수 있습니다. 기본 노드의 출력을 표시하려면 #0을 작업 ID에 추가합니다. 컴퓨팅 노드의 출력을 표시하려면 #1 및 #2를 사용합니다.

```
[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!
```

이제 작업이 성공적으로 완료되었는지 확인할 수 있습니다.

```
[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  -
```

참고: 작업이 끝나기 전에 작업을 종료하려는 경우 `awsbkill` 명령을 사용할 수 있습니다.

사용자 지정 KMS 키를 사용한 디스크 암호화

AWS ParallelCluster 는 구성 옵션 `ebs_kms_key_id` 및 `fsx_kms_key_id`를 지원합니다. 이러한 옵션을 사용하면 Amazon EBS 디스크 암호화 또는 FSx for Lustre에 대한 사용자 지정 AWS KMS 키를 제공할 수 있습니다. 사용하려면 `ec2_iam_role`를 지정해야 합니다.

클러스터를 생성하려면 AWS KMS 키가 클러스터 역할의 이름을 알아야 합니다. 이렇게 하면 클러스터 생성 시 생성된 역할을 사용할 수 없으므로 사용자 지정 `ec2_iam_role`가 필요합니다.

사전 조건

- AWS ParallelCluster 가 설치됩니다.
- 설치 AWS CLI 및 구성됩니다.
- EC2 키 페어가 있습니다.
- `pcluster` CLI를 실행하는 데 필요한 권한을 가진 IAM 역할이 있습니다.

역할 만들기

먼저 정책을 생성합니다.

1. IAM 콘솔(https://console.aws.amazon.com/iam/home)로 이동합니다.
2. 정책, 정책 생성에서 JSON 탭을 클릭합니다.
3. 정책 본문으로 인스턴스 정책에 붙여 넣습니다. *<AWS ACCOUNT ID>*과 *<REGION>*가 나오는 부분을 모두 바꿔야 합니다.
4. `ParallelClusterInstancePolicy` 정책 이름을 지정한 다음 정책 생성을 클릭합니다.

그런 다음 역할을 생성합니다.

1. 역할에서 역할을 생성합니다.
2. 신뢰할 수 있는 엔티티로 `EC2`를 클릭합니다.
3. 권한에서 방금 생성한 `ParallelClusterInstancePolicy` 역할을 검색하여 연결합니다.
4. `ParallelClusterInstanceRole` 역할 이름을 지정한 다음 역할 생성을 클릭합니다.

키 권한을 부여하세요.

AWS KMS 콘솔 > 고객 관리형 키 >에서 키의 별칭 또는 키 ID를 클릭합니다.

키 정책 탭 아래에 있는 키 사용자 상자에서 추가 버튼을 클릭하고 방금 생성한 ParallelClusterInstanceRole을 검색합니다. 연결합니다.

클러스터 생성

이제 클러스터를 생성합니다. 다음은 암호화된 `Raid 0` 드라이브가 있는 클러스터의 예입니다.

```
[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
```

FSx for Lustre 파일 시스템의 예는 다음과 같습니다.

```
[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
```

Amazon EBS 및 Amazon FSx 기반 파일 시스템에도 비슷한 구성이 적용됩니다.

다중 대기열 모드 자습서

여러 대기열 모드로 AWS ParallelCluster 에서 작업 실행

이 자습서에서는를 AWS ParallelCluster 사용하여에서 첫 번째 Hello World 작업을 실행하는 방법을 안내합니다다중 대기열 모드.

사전 조건

- AWS ParallelCluster 가 설치됩니다.
- 설치 AWS CLI 및 구성됩니다.
- EC2 키 페어가 있습니다.
- `pcluster` CLI를 실행하는 데 필요한 권한을 가진 IAM 역할이 있습니다.

> **Note**
>
> 다중 대기열 모드는 AWS ParallelCluster 버전 2.9.0 이상에서만 지원됩니다.

클러스터 구성

먼저 다음 명령을 실행하여 AWS ParallelCluster 가 올바르게 설치되었는지 확인합니다.

```
$ pcluster version
```

`pcluster version`에 대한 자세한 정보는 `pcluster version`을 참조하세요.

이 명령은의 실행 버전을 반환합니다 AWS ParallelCluster.

다음으로 `pcluster configure`을 실행하여 기본 구성 파일을 생성합니다. 이 명령 다음에 나오는 모든 메시지를 따릅니다.

```
$ pcluster configure
```

`pcluster configure` 명령에 대한 자세한 내용은 `pcluster configure` 단원을 참조하세요.

이 단계를 완료하면 ~/.parallelcluster/config에 기본 구성 파일이 있어야 합니다. 이 파일에는 기본 클러스터 구성과 VPC 섹션이 포함되어 있어야 합니다.

자습서의 다음 부분에서는 새로 만든 구성을 수정하고 다중 대기열을 포함하는 클러스터를 시작하는 방법을 간략하게 설명합니다.

> **Note**
>
> 이 자습서에서 사용된 일부 인스턴스는 프리 티어에 사용할 수 없습니다.

이 자습서에서는 다음 구성을 사용합니다.

```
[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
```

클러스터 생성

이 섹션에서는 다중 대기열 모드 클러스터 생성 방법을 자세히 설명합니다.

먼저 클러스터의 이름을 `multi-queue-hello-world`로 지정하고 이전 섹션에서 정의한 `multi-queue` 클러스터 섹션에 따라 클러스터를 생성합니다.

```
$ pcluster create multi-queue-hello-world -t multi-queue
```

`pcluster create`에 대한 자세한 정보는 pcluster create을 참조하세요.

클러스터가 생성되면 다음과 같은 출력이 표시됩니다.

```
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
```

`CREATE_COMPLETE` 메시지는 클러스터가 성공적으로 생성되었음을 나타냅니다. 헤드 노드의 공개 및 비공개 IP 주소도 알려줍니다.

헤드 노드에 로그인

비공개 SSH 키 파일을 사용하여 헤드 노드에 로그인합니다.

```
$ pcluster ssh multi-queue-hello-world -i ~/path/to/keyfile.pem
```

`pcluster ssh`에 대한 자세한 정보는 `pcluster ssh`을 참조하세요.

로그인되면 `sinfo` 명령을 실행하여 스케줄러 대기열이 설정 및 구성되어 있는지 확인합니다.

`sinfo`에 대한 자세한 내용은 Slurm 설명서에서 sinfo 섹션을 참조하세요.

```
$ 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
```

출력을 통해 클러스터에서 사용할 수 있는 `idle` 상태의 `t2.micro` 컴퓨팅 노드가 두 개 있는 것을 알 수 있습니다.

> **Note**
>
> - `spot-st-t2micro-1`은 이름에 `st`가 포함된 정적 노드입니다. 이 노드는 항상 사용 가능하며 클러스터 구성의 `min_count = 1`에 해당합니다.
> - `spot-dy-t2micro-1`은 이름에 `dy`가 포함된 동적 노드입니다. 이 노드는 클러스터 구성에 따라 `initial_count - min_count = 1`에 해당하므로 현재 사용 가능합니다. 이 노드는 사용자 지정 `scaledown_idletime` 시간 5분이 지나면 스케일 다운됩니다.

다른 노드는 모두 이를 지원하는 EC2 인스턴스가 없는 절전 상태입니다. 노드 상태에 ~ 접미사로 표시되어 있습니다. 기본 대기열은 대기열 이름 뒤에 * 접미사로 지정되며 `spot`이 기본 작업 대기열입니다.

다중 대기열 모드에서 작업 실행

다음으로 작업을 실행하여 잠시 동안 대기합니다. 작업은 잠시 후에 자체 호스트 이름을 출력합니다. 반드시 현재 사용자가 이 스크립트를 실행해야 합니다.

```
$ 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 명령을 사용하여 작업을 제출합니다. -N 2 옵션으로 이 작업의 노드 두 개를 요청하고 작업이 성공적으로 제출되는지 확인합니다. sbatch에 대한 자세한 내용은 Slurm 설명서에서 sbatch 섹션을 참조하세요.

```
$ sbatch -N 2 --wrap "srun hellojob.sh"
Submitted batch job 2
```

squeue 명령으로 대기열을 보고 작업 상태를 확인할 수 있습니다. 단, 특정 대기열을 지정하지 않았으므로 기본 대기열(spot)이 사용됩니다. squeue에 대한 자세한 내용은 Slurm 설명서에서 squeue 섹션을 참조하세요.

```
$ 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
```

출력을 통해 작업이 현재 실행 중 상태인 것을 알 수 있습니다. 작업이 완료될 때까지 30초 동안 기다린 후 squeue을 다시 실행합니다.

```
$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
```

이제 대기열의 작업이 모두 완료되었으니 현재 디렉터리에서 slurm-2.out 출력 파일을 찾아보세요.

```
$ cat slurm-2.out
Hello World from spot-dy-t2micro-1
Hello World from spot-st-t2micro-1
```

출력을 통해 spot-st-t2micro-1 및 spot-st-t2micro-2 노드에서 작업이 성공적으로 실행되었음도 알 수 있습니다.

이제 다음 명령으로 특정 인스턴스에 대한 제약 조건을 지정하여 동일한 작업을 제출하세요.

```
$ sbatch -N 3 -p spot -C "[c5.xlarge*1&t2.micro*2]" --wrap "srun hellojob.sh"
Submitted batch job 3
```

이 매개 변수를 sbatch에 사용했습니다.

- -N 3– 노드 세 개를 요청합니다.
- -p spot– 작업을 spot 대기열에 제출합니다. -p ondemand를 지정하여 작업을 ondemand 대기열에 제출할 수도 있습니다.
- -C "[c5.xlarge*1&t2.micro*2]"– 이 작업에 대한 특정 노드 제약 조건을 지정합니다. 이 작업에 사용할 c5.xlarge 노드 1개와 t2.micro 노드 2개를 요청합니다.

sinfo 명령을 실행하여 노드와 대기열을 확인합니다. (의 대기열 AWS ParallelCluster 을의 파티션이라고 합니다Slurm.)

```
$ 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
```

노드에 전원이 공급되고 있습니다. 이는 노드 상태에 # 접미사가 붙는 것으로 표시됩니다. squeue 명령을 실행하여 클러스터에서 작업에 대한 정보를 봅니다.

```
$ 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
```

작업은 CF(CONFIGURING) 상태이며, 인스턴스가 스케일 업되어 클러스터에 합류하기를 기다리고 있습니다.

약 3분 후에 노드를 사용할 수 있고 작업이 R(RUNNING) 상태로 전환됩니다.

```
$ 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
```

작업이 완료되어 세 노드 모두 `idle` 상태입니다.

```
$ 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
```

대기열에 남아 있는 작업이 없으면 로컬 디렉토리에서 `slurm-3.out`을 확인할 수 있습니다.

```
$ 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
```

출력을 통해 해당 노드에서 작업이 성공적으로 실행되었음을 알 수 있습니다.

스케일 다운 프로세스를 관찰할 수 있습니다. 클러스터 구성에서 사용자 지정 scaledown_idletime 시간을 5분으로 지정했습니다. 유휴 상태에서 5분이 지나면 `spot-dy-c5xlarge-1` 및 `spot-dy-t2micro-1` 동적 노드가 자동으로 스케일 다운되어 `POWER_DOWN` 모드로 전환됩니다. 참고로 `spot-st-t2micro-1` 정적 노드는 스케일 다운되지 않습니다.

```
$ 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
```

위 코드를 보면 spot-dy-c5xlarge-1 및 spot-dy-t2micro-1가 POWER_DOWN 모드에 있는 것을 확인할 수 있습니다. 이는 % 접미사로 표시됩니다. 해당 인스턴스는 즉시 종료되지만 노드는 POWER_DOWN 상태를 유지하고 120초(2분) 동안 사용할 수 없습니다. 이 시간이 지나면 노드는 절전 상태로 돌아와 다시 사용할 수 있게 됩니다. 자세한 내용은 다중 대기열 모드를 위한 Slurm 가이드 단원을 참조하십시오.

클러스터의 최종 상태는 다음과 같아야 합니다.

```
$ 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
```

클러스터를 로그오프한 후 pcluster delete를 실행하여 정리할 수 있습니다. pcluster list 및 pcluster delete에 대한 자세한 내용은 pcluster list 및 pcluster delete 섹션을 참조하세요.

```
$ pcluster list
multi-queue CREATE_COMPLETE 2.11.9
$ pcluster delete multi-queue
Deleting: multi-queue
...
```

EFA 및 GPU 인스턴스가 있는 클러스터에서 작업 실행

자습서의 이 부분에서는 구성을 수정하고 EFA 네트워킹 및 GPU 리소스가 있는 인스턴스가 포함된 다중 대기열로 클러스터를 시작하는 방법을 자세히 설명합니다. 참고로 이 자습서에서 사용되는 인스턴스는 가격이 더 높은 인스턴스입니다.

이 자습서에 간략히 설명된 단계를 진행하기 전에 계정 한도를 확인하여 이러한 인스턴스를 사용할 권한이 있는지 확인하세요.

다음과 같이 시스템 구성 파일을 수정합니다.

```
[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
```

클러스터를 생성합니다.

```
$ pcluster create multi-queue-special -t multi-queue-special
```

클러스터를 생성한 후에는 비공개 SSH 키 파일을 사용하여 헤드 노드에 로그인합니다.

```
$ pcluster ssh multi-queue-special -i ~/path/to/keyfile.pem
```

클러스터의 초기 상태는 다음과 같아야 합니다.

```
$ 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]
```

이 섹션에서는 노드에 EFA 또는 GPU 리소스가 있는지 확인하는 일부 작업을 제출하는 방법을 설명합니다.

먼저 작업 스크립트를 작성합니다. `efa_job.sh`는 30초 동안 대기 상태가 됩니다. 그런 다음 `lspci` 명령 출력에서 EFA를 찾아보세요. `gpu_job.sh`는 30초 동안 대기 상태가 됩니다. 그런 다음 `nvidia-smi`을 실행하여 노드에 대한 GPU 정보를 표시합니다.

```
$ 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`를 사용하여 작업을 제출하세요.

```
$ 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]
```

5분 후 노드가 온라인 상태이고 작업이 실행 중인 것을 확인할 수 있습니다.

```
[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
```

작업이 완료되면 출력을 확인합니다. slurm-2.out 파일의 출력에서 efa-dy-c5n18xlarge-1 노드에 EFA가 있음을 확인할 수 있습니다. slurm-3.out 파일의 출력에서 gpu-dy-g38xlarge-1 노드의 GPU 정보가 포함된 nvidia-smi 출력을 확인할 수 있습니다.

```
$ 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                                                 |
+-----------------------------------------------------------------------------+
```

스케일 다운 프로세스를 관찰할 수 있습니다. 클러스터 구성에서 이전에 사용자 지정 scaledown_idletime 시간을 5분으로 지정했습니다. 따라서 유휴 상태에서 5분 지나면 spot-dy-c5xlarge-1 및 spot-dy-t2micro-1 동적 노드가 자동으로 스케일 다운되어 POWER_DOWN 모드로 전환됩니다. 결국 노드는 절전 모드로 전환되어 다시 사용할 수 있게 됩니다.

```
$ 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]
```

클러스터를 로그오프한 후 pcluster delete *<cluster name>*를 실행하여 정리할 수 있습니다.

```
$ pcluster list
multi-queue-special CREATE_COMPLETE 2.11.9
$ pcluster delete multi-queue-special
Deleting: multi-queue-special
...
```

자세한 내용은 다중 대기열 모드를 위한 Slurm 가이드 단원을 참조하십시오.

개발

다음 섹션을 사용하여 개발을 시작할 수 있습니다 AWS ParallelCluster.

> **Important**
>
> 다음 섹션에는 쿡북 레시피의 사용자 지정 버전 또는 사용자 지정 AWS ParallelCluster 노드 패키지를 사용하기 위한 지침이 포함되어 있습니다. 이 정보에서는 디버깅하기 어려울 수 있는 잠재적 문제가 AWS ParallelCluster있는 고급 사용자 지정 방법을 다룹니다. 설치 후 후크는 일반적으로 디버깅이 더 쉽고 릴리스 간에 더 이동하기 쉬우므로 AWS ParallelCluster 팀은 사용자 지정 부트스트랩 작업의 스크립트를 사용자 지정에 사용할 것을 적극 권장합니다 AWS ParallelCluster.

주제

- 사용자 지정 AWS ParallelCluster 쿡북 설정
- 사용자 지정 AWS ParallelCluster 노드 패키지 설정

사용자 지정 AWS ParallelCluster 쿡북 설정

> **Important**
>
> 다음은 AWS ParallelCluster 쿡북 레시피의 사용자 지정 버전을 사용하기 위한 지침입니다. 이는 디버깅하기 어려울 수 있는 잠재적 문제가 AWS ParallelCluster있는 고급 사용자 지정 방법입니다. 이 AWS ParallelCluster 팀은 사용자 지정 부트스트랩 작업의 스크립트를 사용자 지정에 사용할 것을 적극 권장합니다. 설치 후 후크는 일반적으로의 릴리스에서 디버깅하기 쉽고 이동하기 쉽기 때문입니다 AWS ParallelCluster.

단계

1. AWS ParallelCluster 쿡북 코드를 복제한 AWS ParallelCluster 쿡북 작업 디렉터리를 식별합니다.

```
_cookbookDir=<path to cookbook>
```

2. AWS ParallelCluster 쿡북의 현재 버전을 감지합니다.

```
_version=$(grep version ${_cookbookDir}/metadata.rb|awk '{print $2}'| tr -d \')
```

3. AWS ParallelCluster 쿡북의 아카이브를 생성하고 md5를 계산합니다.

```
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 버킷을 생성하고 아카이브, md5 및 마지막 수정 날짜를 버킷에 업로드합니다. public-read ACL을 통해 퍼블릭 읽기 가능 권한을 부여합니다.

```
_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] 섹션 아래의 AWS ParallelCluster 구성 파일에 다음 변수를 추가합니다.

```
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 버전 2.6.1부터 노드를 시작할 때 시작 시간을 개선하기 위해 대부분의 설치 레시피를 기본적으로 건너뜁니다. 이전 버전과의 호환성을 희생하면서 시작 시간을 개선하기 위해 대부분의 설치 레시피를 건너뛰려면 extra_json 설정의 cluster 키에서 "skip_install_recipes" : "no"를 제거하세요.

사용자 지정 AWS ParallelCluster 노드 패키지 설정

> **Warning**
>
> 다음은 AWS ParallelCluster 노드 패키지의 사용자 지정 버전을 사용하기 위한 지침입니다. 이는 디버깅하기 어려울 수 AWS ParallelCluster있는 잠재적 문제가 있는 고급 사용자 지정 방법입니다. 이 AWS ParallelCluster 팀은 사용자 지정 부트스트랩 작업의 스크립트를 사용자 지정에 사용할 것을 적극 권장합니다. 설치 후 후크는 일반적으로의 릴리스에서 디버깅하기 쉽고 이동하기 쉽기 때문입니다 AWS ParallelCluster.

단계

1. AWS ParallelCluster 노드 코드를 복제한 AWS ParallelCluster 노드 작업 디렉터리를 식별합니다.

```
_nodeDir=<path to node package>
```

2. AWS ParallelCluster 노드의 현재 버전을 감지합니다.

```
_version=$(grep "version = \"" ${_nodeDir}/setup.py |awk '{print $3}' | tr -d \")
```

3. AWS ParallelCluster 노드의 아카이브를 생성합니다.

```
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 버킷을 생성하고 버킷에 아카이브를 업로드합니다. public-read ACL을 통해 퍼블릭 읽기 가능 권한을 부여합니다.

```
_bucket=<the bucket name>
aws s3 cp --acl public-read aws-parallelcluster-node-${_version}.tgz s3://${_bucket}/
node/aws-parallelcluster-node-${_version}.tgz
```

5. [cluster] 섹션 아래의 AWS ParallelCluster 구성 파일에 다음 변수를 추가합니다.

```
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 버전 2.6.1부터 노드를 시작할 때 시작 시간을 개선하기 위해 대부분의 설치 레시피를 기본적으로 건너뜁니다. 이전 버전과의 호환성을 희생하면서 시작 시간을 개선하기 위해 대부분의 설치 레시피를 건너뛰려면 extra_json 설정의 cluster 키에서 "skip_install_recipes" : "no"를 제거하세요.

AWS ParallelCluster 문제 해결

AWS ParallelCluster 커뮤니티는 AWS ParallelCluster GitHub Wiki에서 다양한 문제 해결 팁을 제공하는 Wiki 페이지를 유지 관리합니다. 알려진 문제 목록을 알아보려면 알려진 문제를 참조하세요.

주제

- 로그 검색 및 보존
- 스택 배포 문제 해결
- 다중 대기열 모드 클러스터의 문제 해결
- 단일 대기열 모드 클러스터의 문제 해결
- 배치 그룹 및 인스턴스 시작 문제
- 교체할 수 없는 디렉터리
- Amazon DCV의 문제 해결
- AWS Batch 통합을 통한 클러스터의 문제 해결
- 리소스 생성 실패 시 문제 해결
- IAM 정책 크기 문제 해결
- 추가 지원

로그 검색 및 보존

로그는 문제를 해결하는 데 유용한 리소스입니다. 로그를 사용하여 AWS ParallelCluster 리소스 문제를 해결하려면 먼저 클러스터 로그 아카이브를 만들어야 합니다. AWS ParallelCluster GitHub Wiki의 클러스터 로그 아카이브 생성 항목에 설명된 단계에 따라 이 프로세스를 시작하세요.

실행 중인 클러스터 중 하나에 문제가 발생하는 경우 문제 해결을 시작하기 전에 `pcluster stop <cluster_name>` 명령을 실행하여 클러스터를 STOPPED 상태로 만들어야 합니다. 이렇게 하면 예상치 못한 비용이 발생하는 것을 방지할 수 있습니다.

`pcluster`의 작동이 중지되거나 로그를 보존하면서 클러스터를 삭제하려면 `pcluster delete –keep-logs <cluster_name>` 명령을 실행하세요. 이 명령을 실행하면 클러스터는 삭제되지만 Amazon CloudWatch에 저장된 로그 그룹은 보존됩니다. 이 명령에 대한 자세한 내용은 `pcluster delete` 설명서를 참조하세요.

스택 배포 문제 해결

클러스터 생성에 실패하고 스택 생성을 롤백하는 경우 다음 로그 파일을 살펴보고 문제를 진단할 수 있습니다. 이 로그에서 ROLLBACK_IN_PROGRESS의 출력을 찾아보겠습니다. 장애 메시지는 다음과 같아야 합니다.

```
$ 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
```

문제를 진단하려면 `--norollback` 플래그를 포함하여 `pcluster create`를 사용하여 클러스터를 다시 생성하세요. 그런 다음 SSH로 클러스터에 연결합니다.

```
$ pcluster create mycluster --norollback
...
$ pcluster ssh mycluster
```

헤드 노드에 로그인한 후에는 오류를 정확히 찾는 데 사용할 수 있는 세 개의 기본 로그 파일을 찾을 수 있습니다.

- `/var/log/cfn-init.log`는 `cfn-init` 스크립트의 로그입니다. 먼저 이 로그를 확인하세요. 이 로그에 `Command chef failed`와 같은 오류가 표시될 수 있습니다. 오류 메시지와 관련된 자세한 내용은 이 줄 바로 앞에 있는 줄을 참조하세요. 자세한 내용은 cfn-init을 참조하세요.
- `/var/log/cloud-init.log`은 cloud-init에 대한 로그입니다. `cfn-init.log`에 아무것도 표시되지 않으면 다음으로 이 로그를 확인해 보세요.
- `/var/log/cloud-init-output.log`은 cloud-init이 실행한 명령의 출력입니다. 여기에는 `cfn-init`의 출력이 포함됩니다. 대부분의 경우 이러한 유형의 문제를 해결하기 위해 이 로그를 볼 필요가 없습니다.

다중 대기열 모드 클러스터의 문제 해결

이 섹션은 Slurm 작업 스케줄러와 함께 AWS ParallelCluster 버전 2.9.0 이상을 사용하여 설치된 클러스터와 관련이 있습니다. 다중 대기열 모드에 대한 자세한 내용은 다중 대기열 모드 섹션을 참조하세요.

주제

- 키 로그
- 노드 초기화 문제 해결
- 예상치 못한 노드 교체 및 종료 문제 해결
- 문제가 있는 인스턴스 및 노드 교체, 종료 또는 전원 끄기
- 기타 알려진 노드 및 작업 문제 해결

키 로그

다음 표에서는 헤드 노드의 키 로그 개요를 제공합니다.

`/var/log/cfn-init.log`

이는 AWS CloudFormation init 로그입니다. 여기에는 인스턴스가 설정될 때 실행된 모든 명령이 들어 있습니다. 초기화 문제를 해결하는 데 유용합니다.

`/var/log/chef-client.log`

이것은 Chef 클라이언트 로그입니다. 여기에는 Chef/CINC를 통해 실행된 모든 명령이 포함됩니다. 초기화 문제를 해결하는 데 유용합니다.

`/var/log/parallelcluster/slurm_resume.log`

이것은 `ResumeProgram` 로그입니다. 동적 노드용 인스턴스를 시작하며 동적 노드 시작 문제를 해결하는 데 유용합니다.

`/var/log/parallelcluster/slurm_suspend.log`

이것은 `SuspendProgram` 로그입니다. 동적 노드의 인스턴스가 종료될 때 직접적으로 호출되며 동적 노드 종료 문제를 해결하는 데 유용합니다. 이 로그를 확인할 때는 `clustermgtd` 로그도 확인해야 합니다.

`/var/log/parallelcluster/clustermgtd`

이것은 `clustermgtd` 로그입니다. 이 대몬은 대부분의 클러스터 작업을 관리하는 중앙 대몬(daemon)으로 실행됩니다. 시작, 종료 또는 클러스터 작업 문제를 해결하는 데 유용합니다.

`/var/log/slurmctld.log`

제어 Slurm 데몬 로그입니다. 조정 결정을 내 AWS ParallelCluster 리지 않습니다. 오히려 Slurm 요구 사항을 충족하는 리소스를 시작하려고 시도할 뿐입니다. 규모 조정 및 할당 문제, 작업 관련 문제, 스케줄러 관련 시작 및 종료 문제에 유용합니다.

컴퓨팅 노드의 키 노트는 다음과 같습니다.

`/var/log/cloud-init-output.log`

이것은 cloud-init 로그입니다. 여기에는 인스턴스가 설정될 때 실행된 모든 명령이 들어 있습니다. 초기화 문제를 해결하는 데 유용합니다.

`/var/log/parallelcluster/computemgtd`

이것은 `computemgtd` 로그입니다. 이것은 헤드 노드의 `clustermgtd` 대몬(daemon)이 오프라인 상태인 드문 상황에서 각 컴퓨팅 노드에서 실행되어 노드를 모니터링합니다. 예상치 못한 종료 문제를 해결하는 데 유용합니다.

`/var/log/slurmd.log`

이것은 Slurm 컴퓨팅 대몬(daemon) 로그입니다. 초기화 및 컴퓨팅 장애 관련 문제를 해결하는 데 유용합니다.

노드 초기화 문제 해결

이 섹션에서는 노드 초기화 문제를 해결하는 방법을 다룹니다. 여기에는 노드가 시작, 전원 공급 또는 클러스터 조인에 실패하는 문제가 포함됩니다.

헤드 노드:

해당 로그:

- `/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`및 `/var/log/chef-client.log` 로그를 확인하세요. 이 로그에는 헤드 노드가 설정될 때 실행된 모든 작업이 포함되어야 합니다. 설정 중에 발생하는 대부분의 오류에는 `/var/log/chef-client.log` 로그에 오류 메시지가 있을 것입니다. 클러스터 구성에 사전 설치 또는 설치 후 스크립트가 지정된 경우 로그 메시지를 통해 스크립트가 성공적으로 실행되는지 다시 확인하세요.

클러스터를 생성할 때 헤드 노드는 컴퓨팅 노드가 클러스터에 조인할 때까지 기다려야 클러스터에 조인할 수 있습니다. 따라서 컴퓨팅 노드가 클러스터에 조인하지 못하면 헤드 노드도 조인하지 못합니다. 사용하는 컴퓨팅 노드 유형에 따라 다음 일련의 절차 중 하나를 수행하여 이러한 유형의 문제를 해결할 수 있습니다.

동적 컴퓨팅 노드:

- ResumeProgram 로그(/var/log/parallelcluster/slurm_resume.log)에서 컴퓨팅 노드 이름을 검색하여 해당 노드와 함께 ResumeProgram이 직접 호출된 적이 있는지 확인합니다. (ResumeProgram가 호출되지 않은 경우 slurmctld 로그(/var/log/slurmctld.log)를 확인하여가 노드ResumeProgram로 호출을 시도Slurm했는지 확인할 수 있습니다.)
- 권한이 올바르지 않으면 ResumeProgram가 ResumeProgram를 자동으로 실패하게 할 수 있습니다. ResumeProgram 설정을 수정하여 사용자 지정 AMI를 사용하는 경우 slurm 사용자가 ResumeProgram를 소유하고 있으며 744(rwxr--r--) 권한이 있는지 확인하세요.
- ResumeProgram이 직접 호출되면 해당 노드에 대한 인스턴스가 시작되었는지 확인하세요. 시작된 인스턴스가 없는 경우 시작 실패를 설명하는 오류 메시지를 볼 수 있을 것입니다.
- 인스턴스가 시작된 경우 설정 프로세스 중에 문제가 있을 수 있습니다. ResumeProgram 로그에서 해당 프라이빗 IP 주소와 인스턴스 ID를 확인할 수 있습니다. 또한 특정 인스턴스의 대응하는 설정 로그를 볼 수 있습니다. 컴퓨팅 노드 설정 오류의 문제를 해결하는 방법에 대한 자세한 내용은 다음 섹션을 참조하세요.

정적 컴퓨팅 노드:

- clustermgtd(/var/log/parallelcluster/clustermgtd) 로그를 확인하여 해당 노드의 인스턴스가 시작되었는지 확인합니다. 시작되지 않은 경우 시작 실패를 자세히 설명하는 명확한 오류 메시지가 있어야 합니다.
- 인스턴스가 시작되면 설정 프로세스 중에 몇 가지 문제가 있습니다. ResumeProgram 로그에서 해당 프라이빗 IP 주소와 인스턴스 ID를 확인할 수 있습니다. 또한 특정 인스턴스의 대응하는 설정 로그를 볼 수 있습니다.

- 컴퓨팅 노드:
  - 적용 가능한 로그:
    - /var/log/cloud-init-output.log
    - /var/log/slurmd.log
  - 컴퓨팅 노드가 시작된 경우 먼저 /var/log/cloud-init-output.log를 확인하세요. 헤드 노드의 /var/log/chef-client.log와 비슷한 설정 로그가 들어 있을 것입니다. 설정 중에 발생

하는 대부분의 오류에는 /var/log/cloud-init-output.log 로그에 오류 메시지가 있을 것입니다. 클러스터 구성에 사전 설치 또는 설치 후 스크립트가 지정된 경우 해당 스크립트가 성공적으로 실행되었는지 확인하세요.
- Slurm 구성을 수정하여 사용자 지정 AMI를 사용하는 경우 컴퓨팅 노드가 클러스터에 조인하지 못하게 하는 Slurm 관련 오류가 있을 수 있습니다. 스케줄러 관련 오류의 경우 /var/log/slurmd.log 로그를 확인하세요.

예상치 못한 노드 교체 및 종료 문제 해결

이 섹션에서는 특히 노드가 예기치 않게 교체되거나 종료되는 경우 노드 관련 문제를 해결하는 방법을 계속 살펴봅니다.

- 적용 가능한 로그:
  - /var/log/parallelcluster/clustermgtd (헤드 노드)
  - /var/log/slurmctld.log (헤드 노드)
  - /var/log/parallelcluster/computemgtd (컴퓨팅 노드)
- 노드가 예기치 않게 교체되거나 종료됨
  - clustermgtd로그(/var/log/parallelcluster/clustermgtd)를 확인하여 clustermgtd가 노드를 교체 또는 종료했는지 확인합니다. clustermgtd가 모든 일반적인 노드 유지 관리 작업을 처리한다는 점에 유의하세요.
  - clustermgtd가 노드를 교체하거나 종료한 경우 해당 노드를 그렇게 처리한 이유를 설명하는 메시지가 있을 것입니다. 이유가 스케줄러와 관련된 경우(예: 노드가 DOWN에 있기 때문) slurmctld 로그에서 자세한 내용을 확인하세요. 이유가 Amazon EC2와 관련된 것이라면 교체가 필요한 Amazon EC2 관련 문제를 자세히 설명하는 정보 메시지가 있을 것입니다.
  - 가 노드를 종료하지 clustermgtd 않은 경우 먼저 이것이 Amazon EC2에 의한 예상 종료인지, 특히 스팟 종료인지 확인합니다. 컴퓨팅 노드에서 computemgtd실행되는는 clustermgtd가 비정상으로 확인되면 노드를 종료하는 작업을 수행할 수도 있습니다. computemgtd로그(/var/log/parallelcluster/computemgtd)를 확인하여 computemgtd이 노드를 종료했는지 확인하세요.
- 노드에 장애가 발생한 경우
  - slurmctld로그(/var/log/slurmctld.log)를 확인하여 작업이나 노드가 실패한 이유를 확인하세요. 단, 노드에 장애가 발생하면 작업이 자동으로 다시 대기열에 추가된다는 점에 유의하세요.
  - slurm_resume이 해당 노드가 시작되었다고 보고하고 clustermgtd가 몇 분 후에 Amazon EC2에 해당 노드에 대응하는 인스턴스가 없다고 보고하면 설정 중에 노드가 실패할 수 있습니다. 컴퓨팅(/var/log/cloud-init-output.log)에서 로그를 검색하려면 다음 단계를 따르세요.
    - Slurm가 새 노드를 가동할 수 있도록 작업을 제출하세요.

- 노드가 시작된 후 이 명령을 사용하여 종료 보호를 활성화합니다.

```
aws ec2 modify-instance-attribute --instance-id i-xyz --disable-api-termination
```

- 이 명령을 사용하여 노드에서 콘솔 출력을 검색합니다.

```
aws ec2 get-console-output --instance-id i-xyz --output text
```

문제가 있는 인스턴스 및 노드 교체, 종료 또는 전원 끄기

- 적용 가능한 로그:
  - `/var/log/parallelcluster/clustermgtd` (헤드 노드)
  - `/var/log/parallelcluster/slurm_suspend.log` (헤드 노드)
- 대부분의 경우 `clustermgtd`가 모든 예상 인스턴스 종료 작업을 처리합니다. `clustermgtd` 로그에서 노드 교체 또는 종료에 실패한 이유를 확인하세요.
- 동적 노드에 `scaledown_idletime` 장애가 발생한 경우 `SuspendProgram` 로그를 확인하여 특정 노드를 인수로 사용하여 `SuspendProgram`이 `slurmctld`에 의해 직접 호출되었는지 확인하세요. `SuspendProgram`는 실제로 어떤 작업도 수행하지 않습니다. 그보다는 직접 호출될 때만 로그를 기록합니다. 모든 인스턴스 종료 및 `NodeAddr` 재설정은 `clustermgtd`에 의해 수행됩니다. Slurm은 `SuspendTimeout` 이후에 노드를 자동으로 `POWER_SAVING` 상태로 되돌립니다.

기타 알려진 노드 및 작업 문제 해결

알려진 문제의 또 다른 유형은가 작업을 할당하지 못하거나 조정 결정을 내리지 못할 AWS ParallelCluster 수 있다는 것입니다. 이러한 유형의 문제에서는가 Slurm 지침에 따라 리소스 AWS ParallelCluster 만 시작, 종료 또는 유지 관리합니다. 이러한 문제의 경우 `slurmctld` 로그를 확인하여 문제를 해결하세요.

단일 대기열 모드 클러스터의 문제 해결

> **Note**
>
> 버전 2.11.5부터 SGE 또는 Torque 스케줄러 사용을 지원하지 AWS ParallelCluster 않습니다.

이 섹션은 다음 두 구성 중 하나를 사용하는 다중 대기열 모드가 없는 클러스터에 적용됩니다.

- 2.9.0 이전 AWS ParallelCluster 버전과 SGE, Torque또는 Slurm 작업 스케줄러를 사용하여 시작되었습니다.
- AWS ParallelCluster 버전 2.9.0 이상 및 SGE 또는 Torque 작업 스케줄러를 사용하여 시작되었습니다.

주제

- 키 로그
- 시작 및 조인 작업 실패 문제 해결
- 규모 조정 문제 해결
- 기타 클러스터 관련 문제 해결

키 로그

다음 로그 파일은 헤드 노드의 키 로그입니다.

AWS ParallelCluster 버전 2.9.0 이상의 경우:

`/var/log/chef-client.log`

이것은 CINC(chef) 클라이언트 로그입니다. 여기에는 CINC를 통해 실행된 모든 명령이 포함됩니다. 초기화 문제를 해결하는 데 유용합니다.

모든 AWS ParallelCluster 버전의 경우:

`/var/log/cfn-init.log`

이것은 `cfn-init` 로그입니다. 여기에는 인스턴스 설정 시 실행된 모든 명령이 포함되므로 초기화 문제를 해결하는 데 유용합니다. 자세한 내용은 cfn-init을 참조하세요.

`/var/log/clustermgtd.log`

이것은 Slurm 스케줄러용 `clustermgtd` 로그입니다. `clustermgtd`는 대부분의 클러스터 작업을 관리하는 중앙 대몬(daemon)으로 실행됩니다. 시작, 종료 또는 클러스터 작업 문제를 해결하는 데 유용합니다.

`/var/log/jobwatcher`

이것은 SGE 및 Torque 스케줄러의 `jobwatcher` 로그입니다. `jobwatcher`는 스케줄러 대기열을 모니터링하고 Auto Scaling 그룹을 업데이트합니다. 노드 스케일 업과 관련된 문제를 해결하는 데 유용합니다.

`/var/log/sqswatcher`

이 로그는 SGE 및 Torque 스케줄러에 대한 `sqswatcher` 로그입니다. `sqswatcher`는 초기화 성공 후 컴퓨팅 인스턴스가 보낸 인스턴스 준비 이벤트를 처리합니다. 또한 스케줄러 구성에 컴퓨팅 노드를 추가합니다. 이 로그는 노드 또는 노드가 클러스터에 조인하지 못한 이유를 해결하는 데 유용합니다.

컴퓨팅 노드의 키 로그는 다음과 같습니다.

AWS ParallelCluster 버전 2.9.0 이상

`/var/log/cloud-init-output.log`

이것은 Cloud init 로그입니다. 여기에는 인스턴스가 설정될 때 실행된 모든 명령이 들어 있습니다. 초기화 문제를 해결하는 데 유용합니다.

AWS ParallelCluster 2.9.0 이전 버전

`/var/log/cfn-init.log`

이것은 CloudFormation init 로그입니다. 여기에는 인스턴스가 설정될 때 실행된 모든 명령이 들어 있습니다. 초기화 문제를 해결하는 데 유용합니다.

모든 버전

`/var/log/nodewatcher`

이것은 `nodewatcher` 로그입니다. `nodewatcher`는 SGE 및 Torque스케줄러를 사용할 때 각 컴퓨팅 노드에서 실행되는 대몬(daemon)입니다. 유휴 상태인 경우 노드를 스케일 다운합니다. 이 로그는 리소스 스케일 다운과 관련된 모든 문제에 유용합니다.

시작 및 조인 작업 실패 문제 해결

- 적용 가능한 로그:

  - /var/log/cfn-init-cmd.log(헤드 노드 및 컴퓨팅 노드)
  - /var/log/sqswatcher(헤드 노드)
- 노드 시작에 실패한 경우 /var/log/cfn-init-cmd.log 로그를 확인하여 특정 오류 메시지를 확인하세요. 대부분의 경우 노드 시작 실패는 설정 실패로 인해 발생합니다.
- 설치에 성공했는데도 컴퓨팅 노드가 스케줄러 구성에 조인하지 못한 경우 /var/log/sqswatcher 로그를 확인하여 sqswatcher의 이벤트 처리 여부를 확인하세요. 대부분의 경우 이러한 문제는 sqswatcher가 이벤트를 처리하지 않았기 때문입니다.

규모 조정 문제 해결

- 적용 가능한 로그:
  - /var/log/jobwatcher(헤드 노드)
  - /var/log/nodewatcher(컴퓨팅 노드)
- 스케일 업 문제: 헤드 노드의 경우 /var/log/jobwatcher 로그를 확인하여 jobwatcher 대몬(daemon)이 필요한 노드 수를 적절하게 계산하고 Auto Scaling 그룹을 업데이트했는지 확인하세요. 참고로 jobwatcher는 스케줄러 대기열을 모니터링하고 Auto Scaling 그룹을 업데이트합니다.
- 스케일 다운 문제: 컴퓨팅 노드의 경우 문제가 있는 노드의 /var/log/nodewatcher 로그를 확인하여 노드가 스케일 다운된 이유를 확인하세요. 참고로, 컴퓨팅 노드가 유휴 상태인 경우 nodewatcher 대몬(daemon)은 컴퓨팅 노드를 스케일 다운합니다.

기타 클러스터 관련 문제 해결

알려진 문제 중 하나는 대규모 클러스터, 특히 컴퓨팅 노드가 500개 이상인 클러스터에서 무작위 컴퓨팅 노트가 실패한다는 것입니다. 이 문제는 단일 대기열 클러스터의 확장 아키텍처 제한과 관련이 있습니다. 대규모 클러스터를 사용하고, AWS ParallelCluster 버전 v2.9.0 이상을 사용하고,를 사용하고Slurm,이 문제를 방지하려면 여러 대기열 모드 지원 클러스터로 업그레이드하고 전환해야 합니다. pcluster-config convert를 실행하여 그렇게 할 수 있습니다.

ultra-large-scale 클러스터의 경우 시스템에 대한 추가 튜닝이 필요할 수 있습니다. 자세한 내용은에 문의하십시오 지원.

배치 그룹 및 인스턴스 시작 문제

노드 간 지연 시간을 최소화하려면 배치 그룹을 사용하세요. 배치 그룹은 인스턴스가 동일한 네트워킹 백본에 위치하도록 보장합니다. 요청이 이루어질 때 사용 가능한 인스턴스가 충분하지 않으면 InsufficientInstanceCapacity 오류가 반환됩니다. 클러스터 배치 그룹을 사용할 때 이 오류가

발생할 가능성을 줄이려면 placement_group 파라미터를 DYNAMIC으로 설정하고 placement 파라미터를 compute로 설정합니다.

고성능 공유 파일 시스템이 필요한 경우 FSx for Lustre를 사용하는 것이 좋습니다.

헤드 노드가 배치 그룹에 있어야 하는 경우 헤드 및 컴퓨팅 노드 모두에 대해 동일한 인스턴스 유형과 서브넷을 사용합니다. 이렇게 하면 compute_instance_type 파라미터는 master_instance_type 파라미터와 동일한 값을 가지며 placement 파라미터는 cluster로 설정되고 compute_subnet_id 파라미터는 지정되지 않습니다. 이 구성에서는, master_subnet_id 파라미터 값이 컴퓨팅 노드에 사용됩니다.

자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 시작 문제 해결 및 배치 그룹 규칙 및 제한을 참조하세요.

교체할 수 없는 디렉터리

다음 디렉터리는 노드 간에 공유되므로 교체할 수 없습니다.

/home

여기에는 기본 사용자 홈 폴더(Amazon Linux의 /home/ec2_user, CentOS의 /home/centos, /home/ubuntu의 Ubuntu)가 포함됩니다.

/opt/intel

여기에는 Intel MPI, Intel Parallel Studio 및 관련 파일이 포함됩니다.

/opt/sge

> **Note**
>
> 버전 2.11.5부터 SGE 또는 Torque 스케줄러 사용을 지원하지 AWS ParallelCluster 않습니다.

여기에는 Son of Grid Engine 및 관련 파일이 포함됩니다. (조건부, scheduler = sge의 경우에만 해당.)

/opt/slurm

여기에는 Slurm Workload Manager 및 관련 파일이 포함됩니다. (조건부, scheduler = slurm의 경우에만 해당.)

`/opt/torque`

> **Note**
>
> 버전 2.11.5부터 SGE 또는 Torque 스케줄러 사용을 지원하지 AWS ParallelCluster 않습니다.

여기에는 Torque Resource Manager 및 관련 파일이 포함됩니다. (조건부, `scheduler = torque`의 경우에만 해당.)

Amazon DCV의 문제 해결

주제

- Amazon DCV용 로그
- Amazon DCV 인스턴스 유형 메모리
- Ubuntu Amazon DCV 문제

Amazon DCV용 로그

Amazon DCV에 대한 로그는 `/var/log/dcv/` 디렉터리의 파일에 기록됩니다. 이러한 로그를 검토하면 문제를 해결하는 데 도움이 될 수 있습니다.

Amazon DCV 인스턴스 유형 메모리

Amazon DCV를 실행하려면 인스턴스 유형에 1.7GiB 이상의 RAM이 있어야 합니다. Nano 및 micro 인스턴스 유형에 Amazon DCV를 실행할 메모리가 부족합니다.

Ubuntu Amazon DCV 문제

Ubuntu의 DCV 세션을 통해 Gnome 터미널을 실행하는 경우 로그인 셸을 통해를 AWS ParallelCluster 사용할 수 있는 사용자 환경에 자동으로 액세스하지 못할 수 있습니다. 사용자 환경은 openmpi 또는 intelmpi 같은 환경 모듈과 기타 사용자 설정을 제공합니다.

Gnome 터미널의 기본 설정으로 인해 쉘이 로그인 쉘로 시작되지 않습니다. 즉, 쉘 프로파일이 자동으로 소싱되지 않고 AWS ParallelCluster 사용자 환경이 로드되지 않습니다.

쉘 프로파일을 올바르게 소싱하고 AWS ParallelCluster 사용자 환경에 액세스하려면 다음 중 하나를 수행합니다.

- 기본 터미널 설정 변경
  1. Gnome 터미널에서 편집 메뉴를 선택합니다.
  2. 환경설정을 선택한 다음 프로필을 선택합니다.
  3. 명령을 선택하고 로그인 쉘로 명령 실행을 선택합니다.
  4. 새 터미널을 엽니다.
- 명령줄을 사용하여 사용 가능한 프로필을 가져올 수 있습니다.

```
$ source /etc/profile && source $HOME/.bashrc
```

AWS Batch 통합을 통한 클러스터의 문제 해결

이 섹션은 AWS Batch 스케줄러 통합이 있는 클러스터와 관련이 있습니다.

헤드 노드 문제

헤드 노드 관련 설정 문제는 단일 대기열 클러스터와 동일한 방식으로 해결할 수 있습니다. 이러한 문제에 대한 자세한 내용은 단일 대기열 모드 클러스터의 문제 해결 섹션을 참조하세요.

AWS Batch 다중 노드 병렬 작업 제출 문제

를 작업 스케줄러 AWS Batch 로 사용할 때 다중 노드 병렬 작업을 제출하는 데 문제가 있는 경우 AWS ParallelCluster 버전 2.5.0으로 업그레이드해야 합니다. 이것이 가능하지 않은 경우 다중 AWS Batch를 통해 노드 병렬 작업을 제출하는 데 사용되는 클러스터를 자체 패치하기 항목에 자세히 설명된 해결 방법을 사용할 수 있습니다.

컴퓨팅 문제

AWS Batch 는 서비스의 규모 조정 및 컴퓨팅 측면을 관리합니다. 컴퓨팅 관련 문제가 발생하는 경우 AWS Batch 문제 해결 설명서에서 도움말을 참조하세요.

작업 실패

작업이 실패할 경우 `awsbout` 명령을 실행하여 작업 출력을 검색할 수 있습니다. `awsbstat -d` 명령을 실행하여 Amazon CloudWatch에 저장된 작업 로그로 연결되는 링크를 얻을 수도 있습니다.

리소스 생성 실패 시 문제 해결

이 섹션은 클러스터 리소스를 생성하지 못한 경우와 관련이 있습니다.

리소스 생성에 실패하면 ParallelCluster는 다음과 같은 오류 메시지를 반환합니다.

```
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)
}
```

예를 들어, 이전 명령 응답에서 표시된 상태 메시지가 표시되면 현재 vCPU 한도를 초과하지 않는 인스턴스 유형을 사용하거나 vCPU 용량을 더 요청해야 합니다.

CloudFormation 콘솔을 사용하여 "`Cluster creation failed`" 상태에 대한 정보를 확인할 수도 있습니다.

콘솔에서 CloudFormation 오류 메시지를 확인합니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/cloudformation 이동합니다.
2. 이름이 parallelcluster-*cluster_name*인 스택을 선택합니다.
3. 이벤트 탭을 선택합니다.
4. 논리적 ID별로 리소스 이벤트 목록을 스크롤하여 생성에 실패한 리소스의 상태를 확인합니다. 하위 작업을 만들지 못한 경우 역방향으로 진행하여 실패한 리소스 이벤트를 찾아보세요.
5. AWS CloudFormation 오류 메시지의 예:

```
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 정책 크기 문제 해결

IAM 및 할당 AWS STS 량, 이름 요구 사항 및 문자 제한을 참조하여 역할에 연결된 관리형 정책에 대한 할당량을 확인합니다. 관리형 정책 크기가 할당량을 초과하는 경우 정책을 둘 이상의 정책으로 분할하세요. IAM 역할에 연결된 정책 수의 할당량을 초과하는 경우, 추가 역할을 생성하고 할당량을 충족하도록 역할을 역할 간에 정책을 분배하세요.

추가 지원

알려진 문제 목록은 기본 GitHub Wiki 페이지 또는 문제 페이지를 참조하세요. 더 긴급한 문제는에 문의 지원 하거나 새 GitHub 문제를 여세요.

AWS ParallelCluster 지원 정책

AWS ParallelCluster 는 동시에 여러 릴리스를 지원합니다. 모든 AWS ParallelCluster 릴리스에는 지원 종료(EOSL) 날짜가 예약되어 있습니다. EOSL 날짜 이후에는 해당 릴리스에 대한 추가 지원이나 유지 관리가 제공되지 않습니다.

AWS ParallelCluster 는 `major.minor.patch` 버전 체계를 사용합니다. 최신 메이저 버전 릴리스의 새 마이너 버전 릴리스에는 새로운 기능, 성능 개선, 보안 업데이트 및 버그 수정이 포함됩니다. 마이너 버전은 메이저 버전 내에서 이전 버전과 호환됩니다. 심각한 문제의 경우 AWS 는 패치 릴리스를 통해 수정 사항을 제공하지만 EOSL에 도달하지 않은 최신 마이너 버전의 릴리스에만 적용됩니다. 새 버전 릴리스의 업데이트를 사용하려면 새 마이너 버전 또는 패치 버전으로 업그레이드해야 합니다.

AWS ParallelCluster 버전	지원 기간 종료(EOSL) 날짜
2.10.4 이하	12/31/2021
2.11.*x*	12/31/2022

의 보안 AWS ParallelCluster

의 클라우드 보안 AWS 이 최우선 순위입니다. AWS 고객은 가장 보안에 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 이점을 누릴 수 있습니다.

보안은 AWS 와 사용자 간의 공동 책임입니다. 공동 책임 모델 에서는 이를 클라우드 자체의 보안과 클라우드 내부의 보안으로 설명합니다.

- 클라우드 보안 - AWS 는 AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호할 책임이 있습니다. AWS 또한는 안전하게 사용할 수 있는 서비스를 제공합니다. 타사 감사자는 AWS 규정 준수 프로그램 일환으로 보안의 효과를 정기적으로 테스트하고 확인합니다. 에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 준수 프로그램 AWS 제공 범위 내 서비스규정 준수 프로그램 제공 범위 내 서비스를 AWS ParallelCluster참조하세요.
- 클라우드의 보안 - 사용자의 책임은 사용하는 특정 AWS 서비스 또는 서비스에 따라 결정됩니다. 또한 귀하는 데이터의 민감도, 회사 요구 사항, 관련 법률 및 규정을 비롯한 기타 여러 관련된 요소에 대해서도 책임이 있습니다.

이 설명서에서는를 사용할 때 공동 책임 모델을 적용하는 방법을 설명합니다 AWS ParallelCluster. 다음 주제에서는 보안 및 규정 준수 목표를 충족하도록 AWS ParallelCluster 를 구성하는 방법을 보여줍니다. 또한 AWS 리소스를 모니터링하고 보호하는 데 도움이 되는 방식으로 AWS ParallelCluster 를 사용하는 방법을 알아봅니다.

주제

- 에서 사용하는 서비스에 대한 보안 정보 AWS ParallelCluster
- 의 데이터 보호 AWS ParallelCluster
- 에 대한 자격 증명 및 액세스 관리 AWS ParallelCluster
- 에 대한 규정 준수 검증 AWS ParallelCluster
- TLS 1.2의 최소 버전 적용

에서 사용하는 서비스에 대한 보안 정보 AWS ParallelCluster

- Amazon EC2의 보안
- Amazon API Gateway의 보안
- 의 보안 AWS Batch

- AWS CloudFormation의 보안
- Amazon CloudWatch의 보안
- AWS CodeBuild의 보안
- Amazon DynamoDB의 보안
- Amazon ECR의 보안
- Amazon ECS의 보안
- Amazon EFS의 보안
- FSx for Lustre의 보안
- AWS Identity and Access Management (IAM)의 보안
- EC2 Image Builder의 보안
- 의 보안 AWS Lambda
- Amazon Route 53의 보안
- Amazon SNS의 보안
- Amazon SQS의 보안(AWS ParallelCluster 버전 2.x의 경우)
- Amazon S3의 보안
- Amazon VPC의 보안

의 데이터 보호 AWS ParallelCluster

AWS 공동 책임 모델의 데이터 보호에 적용됩니다 AWS ParallelCluster. 이 모델에 설명된 대로 AWS 는 모든를 실행하는 글로벌 인프라를 보호할 책임이 있습니다 AWS 클라우드. 사용자는 인프라에서 호스팅되는 콘텐츠를 관리해야 합니다. 사용하는 AWS 서비스 의 보안 구성과 관리 태스크에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요.

데이터 보호를 위해 자격 증명을 보호하고 AWS 계정 AWS IAM Identity Center 또는 AWS Identity and Access Management (IAM)를 사용하여 개별 사용자를 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용하세요.
- SSL/TLS를 사용하여 AWS 리소스와 통신합니다. TLS 1.2는 필수이며 TLS 1.3을 권장합니다.

- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다 AWS CloudTrail. CloudTrail 추적을 사용하여 AWS 활동을 캡처하는 방법에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail 추적 작업을 참조하세요.
- AWS 암호화 솔루션과 내부의 모든 기본 보안 제어를 사용합니다 AWS 서비스.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용하세요.
- 명령줄 인터페이스 또는 API를 AWS 통해에 액세스할 때 FIPS 140-3 검증 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 Federal Information Processing Standard(FIPS) 140-3을 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 형식 텍스트 필드에 입력하지 않는 것이 좋습니다. 여기에는 콘솔, API AWS CLI또는 AWS SDKs를 사용하여 AWS ParallelCluster 또는 다른 AWS 서비스 로 작업하는 경우가 포함됩니다. 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 자격 증명을 URL에 포함해서는 안 됩니다.

데이터 암호화

보안 서비스의 주요 특징은 정보가 활발히 사용되지 않을 때 암호화된다는 것입니다.

저장 중 암호화

AWS ParallelCluster 는 사용자를 대신하여 AWS 서비스와 상호 작용하는 데 필요한 자격 증명 이외의 고객 데이터를 저장하지 않습니다.

클러스터의 노드에 있는 데이터의 경우 저장된 데이터가 암호화될 수 있습니다.

Amazon EBS 볼륨의 경우 암호화는 AWS ParallelCluster 버전 2.x `[ebs]` 섹션의 `ebs_kms_key_id` 설정을 사용하여 구성됩니다.) 자세한 내용은 Amazon EC2 사용 설명서의 Amazon EBS 암호화를 참조하세요.

Amazon EFS 볼륨의 경우 암호화는 AWS ParallelCluster 버전 2.x의 `[efs]` 섹션에 있는 `encrypted` 및 `efs_kms_key_id` 설정을 사용하여 구성됩니다). 자세한 내용은 Amazon Elastic File System 사용 설명서의 저장 데이터 암호화 작동 방식을 참조하세요.

FSx for Lustre 파일 시스템의 경우 Amazon FSx 파일 시스템을 생성할 때 저장 데이터의 암호화가 자동으로 활성화됩니다. 자세한 내용을 알아보려면 Amazon FSx for Lustre 사용 설명서의 저장 데이터 암호화를 참조하세요.

NVMe 볼륨이 있는 인스턴스 유형의 경우 NVMe 인스턴스 스토어 볼륨의 데이터는 인스턴스의 하드웨어 모듈에서 구현된 XTS-AES-256 암호를 사용하여 암호화합니다. 하드웨어 모듈을 사용하여 암호화 키를 생성하며, 암호화 키는 각 NVMe 인스턴스 스토리지 디바이스마다 고유합니다. 인스턴스가 중지되거나 종료되면 모든 암호화 키가 손상되어 복구가 불가능해집니다. 이 암호화를 비활성화할 수 없으며, 사용자 자신의 암호화 키를 제공할 수 없습니다. 자세한 내용은 Amazon EC2 사용 설명서의 저장된 암호화를 참조하세요.

AWS ParallelCluster 를 사용하여 로컬 컴퓨터에 고객 데이터를 전송하여 저장하는 AWS 서비스를 호출하는 경우 해당 데이터의 저장, 보호 및 암호화 방법에 대한 자세한 내용은 해당 서비스 사용 설명서의 보안 및 규정 준수 장을 참조하세요.

전송 중 암호화

기본적으로 AWS ParallelCluster 및 AWS 서비스 엔드포인트를 실행하는 클라이언트 컴퓨터에서 전송되는 모든 데이터는 HTTPS/TLS 연결을 통해 모든 데이터를 전송하여 암호화됩니다. 클러스터의 노드 간 트래픽은 선택한 인스턴스 유형에 따라 자동으로 암호화될 수 있습니다. 자세한 내용은 Amazon EC2 사용 설명서의 전송 중 암호화를 참조하세요.

다음 사항도 참조하세요.

- Amazon EC2의 데이터 보호
- EC2 Image Builder의 데이터 보호
- 의 데이터 보호 AWS CloudFormation
- Amazon EFS의 데이터 보호
- Amazon S3의 데이터 보호
- FSx for Lustre의 데이터 보호

에 대한 자격 증명 및 액세스 관리 AWS ParallelCluster

AWS ParallelCluster 는 역할을 사용하여 AWS 리소스와 해당 서비스에 액세스합니다. 가 권한을 부여하는 데 AWS ParallelCluster 사용하는 인스턴스 및 사용자 정책은에 설명되어 있습니다AWS Identity and Access Management 의 역할 AWS ParallelCluster.

유일한 주요 차이점은 표준 사용자와 장기 보안 인증을 사용할 때 인증하는 방법입니다. 사용자가 AWS 서비스의 콘솔에 액세스하려면 암호가 필요하지만 동일한 사용자가를 사용하여 동일한 작업을 수행하려면 액세스 키 페어가 필요합니다 AWS ParallelCluster. 다른 모든 단기 보안 인증은 콘솔에서 사용되는 것과 동일한 방식으로 사용됩니다.

에서 사용하는 자격 증명은 일반 텍스트 파일에 AWS ParallelCluster 저장되며 암호화되지 않습니다.

- `$HOME/.aws/credentials` 파일에는 AWS 리소스에 액세스하는 데 필요한 장기 보안 인증이 저장됩니다. 여기에는 계정 액세스 키 ID와 비밀 액세스 키가 포함됩니다.
- 수임하는 역할 또는 AWS IAM Identity Center 서비스에 대한 자격 증명과 같은 단기 자격 증명도 각각 `$HOME/.aws/cli/cache` 및 $HOME/.aws/sso/cache 폴더에 저장됩니다.

위험 완화

- `$HOME/.aws` 폴더와 해당 하위 폴더 및 파일에 대한 파일 시스템 권한을 구성하여 권한 있는 사용자만 액세스할 수 있도록 제한하는 것이 좋습니다.
- 보안 인증이 손상된 경우, 손상 가능성을 줄이려면 가능한 한 임시 보안 인증이 있는 역할을 사용합니다. 단기 역할 보안 인증을 요청하고 새로 고치는 경우에만 장기 보안 인증을 사용합니다.

에 대한 규정 준수 검증 AWS ParallelCluster

타사 감사자는 여러 규정 준수 프로그램의 일환으로 AWS 서비스의 보안 및 AWS 규정 준수를 평가합니다. 를 사용하여 서비스에 액세스 AWS ParallelCluster 해도 해당 서비스의 규정 준수는 변경되지 않습니다.

특정 규정 준수 프로그램 범위의 AWS 서비스 목록은 규정 준수 프로그램 AWS 범위의 서비스규정 준수 프로그램 . 일반 정보는 AWS 규정 준수 프로그램 참조하세요.

를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다 AWS Artifact. 자세한 정보는 에서 AWS Artifact에서 보고서 다운로드를 참조하세요.

사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률과 규정에 따라 AWS ParallelCluster 결정됩니다.는 규정 준수를 지원하기 위해 다음 리소스를 AWS 제공합니다.

- 보안 및 규정 준수 빠른 시작 가이드 -이 배포 가이드에서는 아키텍처 고려 사항에 대해 설명하고 보안 및 규정 준수 중심 기준 환경을 배포하기 위한 단계를 제공합니다 AWS.
- Amazon Web Services의 HIPAA 보안 및 규정 준수를 위한 설계 AWS 백서 -이 백서에서는 기업이 AWS 를 사용하여 HIPAA 준수 애플리케이션을 생성하는 방법을 설명합니다.
- AWS 규정 준수 리소스 -이 워크북 및 가이드 모음은 업계 및 위치에 적용될 수 있습니다.
- AWS Config 개발자 안내서의 규칙을 사용하여 리소스 평가 -이 AWS Config 서비스는 리소스 구성이 내부 관행, 업계 지침 및 규정을 얼마나 잘 준수하는지 평가합니다.

- AWS Security Hub -이 AWS 서비스는 보안 업계 표준 및 모범 사례 준수를 확인하는 데 도움이 AWS 되는 내 보안 상태에 대한 포괄적인 보기를 제공합니다.

TLS 1.2의 최소 버전 적용

AWS 서비스와 통신할 때 보안을 강화하려면 TLS 1.2 이상을 사용하도록를 AWS ParallelCluster 구성해야 합니다. 를 사용하면 AWS ParallelCluster Python이 TLS 버전을 설정하는 데 사용됩니다.

AWS ParallelCluster 가 TLS 1.2 이전 버전의 TLS를 사용하지 않도록 하려면 OpenSSL을 다시 컴파일하여이 최소값을 적용한 다음 Python을 다시 컴파일하여 새로 빌드된 OpenSSL을 사용해야 할 수 있습니다.

현재 지원되는 프로토콜 확인

먼저 OpenSSL을 사용하여 테스트 서버 및 Python SDK에 사용할 자체 서명된 인증서를 만듭니다.

```
$ openssl req -subj '/CN=localhost' -x509 -newkey rsa:4096 -nodes -keyout key.pem -out
 cert.pem -days 365
```

그런 다음 OpenSSL을 사용하여 테스트 서버를 가동합니다.

```
$ openssl s_server -key key.pem -cert cert.pem -www
```

새 터미널 창에서 가상 환경을 만들고 Python SDK를 설치합니다.

```
$ python3 -m venv test-env
source test-env/bin/activate
pip install botocore
```

SDK의 기본 HTTP 라이브러리를 사용하는 `check.py`라는 새로운 Python 스크립트를 만듭니다.

```
$ 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'))
```

새 스크립트를 실행합니다.

```
$ python check.py
```

그러면 연결에 대한 세부 정보가 표시됩니다. 출력에서 "프로토콜 : "을 검색합니다. 출력이 “TLSv1.2" 이상이면 SDK는 기본적으로 TLS v1.2 이상으로 설정됩니다. 이전 버전인 경우 OpenSSL을 다시 컴파일하고 Python을 다시 컴파일해야 합니다.

그러나 Python이 기본적으로 TLS v1.2 이상으로 설치되더라도 서버가 TLS v1.2 이상을 지원하지 않으면 Python이 TLS v1.2 이전 버전으로 다시 협상할 수 있습니다. Python이 이전 버전으로 자동으로 다시 협상하지 않는지 확인하려면 다음과 같이 테스트 서버를 다시 시작하세요.

```
$ openssl s_server -key key.pem -cert cert.pem -no_tls1_3 -no_tls1_2 -www
```

이전 버전의 OpenSSL을 사용하는 경우 `-no_tls_3` 플래그를 사용할 수 없을 수 있습니다. 이 경우 사용 중인 OpenSSL 버전이 TLS v1.3을 지원하지 않으므로 플래그를 제거합니다. 그런 다음 Python 스크립트를 다시 실행합니다.

```
$ python check.py
```

Python 설치가 TLS 1.2 이전 버전에서 올바르게 다시 협상되지 않으면 SSL 오류가 발생합니다.

```
$ 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)')))
```

연결할 수 있는 경우 TLS v1.2 이전의 프로토콜 협상을 비활성화하기 위해 OpenSSL과 Python을 다시 컴파일해야 합니다.

OpenSSL 및 Python 컴파일

AWS ParallelCluster 가 TLS 1.2 이전 버전에 대해 협상하지 않도록 하려면 OpenSSL 및 Python을 다시 컴파일해야 합니다. 이렇게 하려면 다음 내용을 복사하여 스크립트를 만들고 실행합니다.

```
#!/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
```

이것은 TLS 1.2 이전 버전을 자동으로 협상하지 않는 정적으로 연결된 OpenSSL을 가진 Python 버전을 컴파일합니다. 또한 `/opt/openssl-with-min-tls1_2` 디렉터리에 OpenSSL을 설치하고 `/opt/python-with-min-tls1_2` 디렉터리에 Python을 설치합니다. 이 스크립트를 실행한 후 새 버전의 Python 설치를 확인하세요.

```
$ /opt/python-with-min-tls1_2/bin/python3 --version
```

다음 사항이 인쇄되어야 합니다.

```
Python 3.8.1
```

이 새 버전의 Python이 TLS 1.2 이전 버전을 협상하지 않는지 확인하려면 새로 설치된 Python 버전(즉, 현재 지원되는 프로토콜 확인)을 사용하는 `/opt/python-with-min-tls1_2/bin/python3`의 단계를 다시 실행합니다.

릴리스 정보 및 문서 기록

다음 표에서AWS ParallelCluster 사용 설명서의 중요한 업데이트 및 새 기능이 나와 있습니다. 사용자로부터 받은 의견을 수렴하기 위해 설명서가 자주 업데이트됩니다.

변경 사항	설명	날짜
설명서 전용 릴리스	AWS ParallelCluster 버전 2별 사용 설명서가 게시되었습니다. 설명서 전용 릴리스 • AWS ParallelCluster 버전 2에는 별도의 사용 설명서가 있습니다.	2023년 7월 17일
AWS ParallelCluster 버전 2.11.9 릴리스	AWS ParallelCluster 버전 2.11.9 릴리스. 버그 수정: • 관리형 FSx for Lustre 파일 시스템의 교체 및 `vpc_security_group_id` 에 대한 변경 내용이 포함된 클러스터 업데이트 시 데이터 손실을 방지합니다. 변경 사항에 대한 자세한 내용은 GitHub의 aws-parallelcluster 패키지 `CHANGELOG` 파일을 참조하세요.	2022년 12월 2일

AWS ParallelCluster 버전 2.11.8 릴리스	AWS ParallelCluster 버전 2.11.8 릴리스.	2022년 11월 14일

변경 사항:

- Intel MPI Library를 버전 2021 Update 4에서 버전 2021 Update 6으로 업그레이드했습니다. 자세한 내용은 Intel® MPI Library 2021 Update 6을 참조하세요.
- EFA 설치 프로그램을 1.19.0으로 업그레이드
  - Efa-driver: `efa-1.16.0-1`
  - Efa-config: `efa-config-1.9-1` 에서 `efa-config-1.11-1` 로
  - Efa-profile: `efa-profile-1.5-1` (변경 없음)
  - Libfabric-aws: `libfabric-1.13.2` 에서 `libfabric-aws-1.16.0-1` 로
  - Rdma-core: `rdma-core-37.0` 에서 `rdma-core-41.0-2` 로
  - Open MPI: `openmpi40-aws-4.1.1-2` 에서 `openmpi40-aws-4.1.4-3` 로
- AWS Batch 통합의 Lambda 함수에서 사용하는 Python

	런타임을 python3.9로 업그레이드합니다. 버그 수정: • 지원되지 않는다는 이유로 클러스터 태그가 업데이트 중에 클러스터 변경되는 것을 방지 변경 사항에 대한 자세한 내용은 GitHub의 aws-parallelcluster 패키지 CHANGELOG 파일을 참조하세요.	
AWS ParallelCluster 버전 2.11.7 릴리스	AWS ParallelCluster 버전 2.11.7 릴리스. 변경 사항: • Slurm을 버전 20.11.9로 업그레이드 변경 사항에 대한 자세한 내용은 GitHub의 aws-parallelcluster 패키지 CHANGELOG 파일을 참조하세요.	2022년 5월 13일

AWS ParallelCluster 버전 2.11.6 릴리스	AWS ParallelCluster 버전 2.11.6 릴리스. 개선 사항: • 네트워킹이 누락된 경우 예외 관리 개선 변경 사항: • OS 패키지 업데이트 및 보안 수정 변경 사항에 대한 자세한 내용은 GitHub의 aws-parallelcluster 패키지 CHANGELOG 파일을 참조하세요.	2022년 4월 19일

AWS ParallelCluster 버전 2.11.5 릴리스	AWS ParallelCluster 버전 2.11.5 릴리스.	2022년 3월 1일

개선 사항:

- FSx for Lustre `AutoImportPolicy` 옵션의 값으로 `NEW_CHANGED_DELETED` 에 대한 지원 추가
- SGE 및 토크 스케줄러에 대한 지원 제거
- 잠재적인 성능 저하가 발생하지 않도록 Amazon Linux에서 `log4j-cve-2021-44228-hotpatch` 서비스 비활성화

변경 사항:

- NVIDIA 드라이버를 버전 `470.82.01` 에서 `470.103.01` 로 업그레이드
- NVIDIA 패브릭 매니저를 버전 `470.82.01` 에서 `470.103.01` 로 업그레이드
- CUDA 라이브러리를 버전 `11.4.3`에서 `11.4.4`로 업그레이드
- Intel MPI가 버전 2019 업데이트 8에서 버전 2021 Update 4 버전으로 업데이트되었습니다. 자세한 내용

은 Intel® MPI Library 2021 Update 4를 참조하세요.

- 헤드 노드 생성 타임아웃을 1시간으로 연장

버그 수정:

- 브라우저를 통한 DCV 연결 수정
- 사용자 지정 태그가 숫자로 파싱되지 않도록 YAML 인용 수정

변경 사항에 대한 자세한 내용은 GitHub의 aws-parallelcluster 패키지 `CHANGELOG` 파일을 참조하세요.

AWS ParallelCluster 버전 2.11.4 릴리스	AWS ParallelCluster 버전 2.11.4 릴리스. 변경 사항: • CentOS 8 지원이 제거되었습니다. CentOS 8은 2021년 12월 31일에 수명 종료에 도달합니다. • Slurm Workload Manager가 버전 20.11.8로 업그레이드됨 • Cinc 클라이언트를 `17.2.29`로 업그레이드 • Amazon DCV가 Amazon DCV 2021.2-11190으로 업데이트되었습니다. 자세한 내용은 Amazon DCV 관리자 가이드의 DCV 2021.2-11190 - 2021년 10월 11일을 참조하세요. • NVIDIA 드라이버를 버전 `460.73.01` 에서 `470.82.01` 로 업그레이드 • CUDA 라이브러리를 버전 `11.3.0`에서 `11.4.3`로 업그레이드 • NVIDIA 패브릭 매니저를 `470.82.01` 로 업그레이드 • Amazon Linux 2에서 인스턴스 시작 시 패키지 업데이트 비활성화 • Amazon Linux 2에서 Ubuntu 무인 패키지 업데이트 비활성화	2021년 12월 20일

- Python 3 버전의 AWS CloudFormation 헬퍼 스크립트를 CentOS 7과 Ubuntu 18.04에 설치 (이들은 아마존 리눅스 2와 Ubuntu 20.04에서 이미 사용되었습니다.)

수정 사항:

- `ec2_iam_role` 파라미터 업데이트 비활성화
- 시작 템플릿에서 T2 인스턴스의 `CpuOptions` 구성 수정

변경 사항에 대한 자세한 내용은 GitHub의 aws-parallelcluster, aws-parallelcluster-cookbook 및 aws-parallelcluster-node 패키지의 `CHANGELOG` 파일을 참조하세요.

AWS ParallelCluster 버전 2.11.3 릴리스	AWS ParallelCluster 버전 2.11.3 릴리스.	2021년 11월 3일
	• Son of Grid Engine 소스가 `arc.liv.ac.uk` 에서 사용될 수 없어 발생하는 `pcluster createami` 오류 수정	
	Elastic Fabric Adapter 설치 프로그램을 1.13.0에서 1.14.1로 업그레이드 • EFA 구성: `efa-config-1.9` 에서 `efa-config-1.9-1` 로 • EFA 프로필: `efa-profile-1.5-1` (변경 없음) • EFA 커널 모듈: `efa-1.13.0` 에서 `efa-1.14.2` 로 • RDMA 코어: `rdma-core-35.0amzn` 에서 `rdma-core-37.0` 로 • Libfabric: `libfabric-1.13.0amzn1.0` 에서 `libfabric-1.13.2` 로 • Open MPI: `openmpi40-aws-4.1.1-2` (변경 없음) 인스턴스 유형에서 지원하는 경우 GPUDirect RDMA는 항상 활성화됩니다.	

- `enable_efa_gdr` 및 `enable_efa_gdr` 구성 옵션은 영향을 주지 않습니다.

변경 사항에 대한 자세한 내용은 GitHub의 aws-parallelcluster, aws-parallelcluster-cookbook 및 aws-parallelcluster-node 패키지의 `CHANGELOG` 파일을 참조하세요.

AWS ParallelCluster 버전 2.11.2 릴리스	AWS ParallelCluster 버전 2.11.2 릴리스.
	
	변경 사항:
	
	• 기본 AMI에 EFA가 설치된 경우 부트스트랩 시 GPUDirect RDMA(GDR)가 활성화된 상태에서 EFA를 설치하지 마세요.
	• 에서 설치한 NVIDIA 드라이버 버전과 동기화된 상태로 유지되도록 `nvidia-fabricmanager` 패키지 버전을 잠급니다 AWS ParallelCluster.
	• Slurm: 노드 전원이 켜지는 동안 클러스터가 중지되었다가 다시 시작될 때 발생하는 문제를 수정했습니다.
	• Elastic Fabric Adapter 설치 관리자가 1.13.0으로 업데이트되었습니다.
	  • EFA 구성: `efa-config-1.9` (변경 없음)
	  • EFA 프로필: `efa-profile-1.5-1` (변경 없음)
	  • EFA 커널 모듈: `efa-1.13.0` (변경 없음)
	  • RDMA 코어: `rdma-core-32.1amzn` 에서 `rdma-core-35.0amzn` 로
	  • Libfabric: `libfabric-1.11.2amzn1.1` 에서
		2021년 8월 27일

`libfabric-1.13.0amzn1.0` 로

- Open MPI: `openmpi40-aws-4.1.1-2` (변경 없음)

- 사전 설치된 EFA 패키지와 함께 사용자 지정 AMI를 사용하는 경우 노드 부트스트랩 시간에 EFA가 변경되지 않습니다. 원본 EFA 패키지 배포는 보존됩니다.

변경 사항에 대한 자세한 내용은 GitHub의 aws-parallelcluster 및 aws-parallelcluster-cookbook 패키지의 변경 로그 파일을 참조하세요.

AWS ParallelCluster 버전 2.11.1 릴리스	AWS ParallelCluster 버전 2.11.1 릴리스.
	
	변경 사항:
	
	• noatime 탑재 옵션을 사용하여 파일 시스템을 탑재하면 파일을 읽을 때 마지막 액세스 시간 기록을 중지할 수 있습니다. 이렇게 하면 원격 파일 시스템의 성능이 향상됩니다.
	• Elastic Fabric Adapter 설치 관리자가 1.12.3으로 업데이트되었습니다.
	  • EFA 구성: efa-config-1.8-1 에서 efa-config-1.9 로
	  • EFA 프로필: efa-profile-1.5-1 (변경 없음)
	  • EFA 커널 모듈: efa-1.12.3 에서 efa-1.13.0 로
	  • RDMA 코어: rdma-core-32.1amzn (변경 없음)
	  • Libfabric: libfabric-1.11.2amzn1.1 (변경 없음)
	  • Open MPI: openmpi40-aws-4.1.1-2 (변경 없음)
	• 를 스케줄러 AWS Batch 로 사용할 때 헤드 노드에 aws-parallelcluster 패키지 설치를 다시 시도합니다.
		2021년 7월 23일

- 31개 이상의 vCPU가 있는 인스턴스 유형을 SGE 기반으로 구축할 때 실패 방지
- 버전 1.247348.0에서 나타나는 문제를 방지하기 위해 Amazon CloudWatch 에이전트 버전 1.247347.6에 고정

변경 사항에 대한 자세한 내용은 GitHub의 aws-parallelcluster 및 aws-parallelcluster-cookbook 패키지의 변경 로그 파일을 참조하세요.

AWS ParallelCluster 버전 2.11.0 릴리스	AWS ParallelCluster 버전 2.11.0 릴리스.
	
	변경 사항:
	
	• Ubuntu 20.04(ubuntu2004)에 대한 지원이 추가되고 Ubuntu 16.04(ubuntu1604) 및 Amazon Linux(alinux) 에 대한 지원이 제거되었습니다. Amazon Linux 2(alinux2)는 계속 완벽하게 지원됩니다. 자세한 내용은 base_os 항목을 참조하세요.
	• 3.6 미만의 Python 버전에 대한 지원이 제거되었습니다.
	• 기본 루트 볼륨 크기가 35 기비바이트(GiB)로 증가했습니다. 자세한 정보는 compute_root_volume_size 및 master_root_volume_size 항목을 참조하세요.
	• Elastic Fabric Adapter 설치 관리자가 1.12.2로 업데이트되었습니다.
	  • EFA 구성: efa-config-1.7 에서 efa-config-1.8-1 로
	  • EFA 프로필: efa-profile-1.4 에서 efa-profile-1.5-1 로
		2021년 7월 1일

  - EFA 커널 모듈: `efa-1.10.2` 에서 `efa-1.12.3` 로
  - RDMA 코어: `rdma-core-31.2amzn` 에서 `rdma-core-32.1amzn` 로
  - Libfabric: `libfabric-1.11.1amzn1.0` 에서 `libfabric-1.11.2amzn1.1` 로
  - Open MPI: `openmpi40-aws-4.1.0` 에서 `openmpi40-aws-4.1.1-2` 로
- Slurm을 버전 `20.02.7`에서 버전 `20.11.7`로 업그레이드
- `centos7` 및 `centos8`에 SSM 에이전트 설치 (SSM 에이전트는 `alinux2`, `ubuntu1804` 및에 사전 설치되어 있습니다`ubuntu2004` .)
- SGE: 항상 단축 이름을 `qstat`와 함께 호스트 이름 필터로 사용하세요.
- 인스턴스 메타데이터를 검색하는 데에 인스턴스 메타데이터 서비스 버전 1(IMDSv1) 대신 인스턴스 메타데이터 서비스 버전 2(IMDSv2)를 사용 자세한 내용은 Amazon EC2 사용 설명서의 인스턴

스 메타데이터 및 사용자 데이터를 참조하세요.

- NVIDIA 드라이버를 버전 `450.80.02` 에서 `460.73.01` 로 업그레이드
- CUDA 라이브러리를 버전 `11.0`에서 `11.3.0`로 업그레이드
- NVIDIA 패브릭 매니저를 `nvidia-fabricmanager-460` 로 업그레이드
- AWS ParallelCluster virtualenvs에 사용되는 Python을 `3.7.10` (에서)로 업그레이드합니다`3.6.13`.
- Cinc 클라이언트를 `16.13.16`로 업그레이드
- aws-parallelcluster-cookbook의 서드 파티 종속성 업그레이드:
  - `apt-7.3.0` 에서 `apt-7.4.0` 로
  - `iptables-7.1.0` 에서 `iptables-8.0.0` 로
  - `line-2.9.0` 에서 `line-4.0.1` 로
  - `openssh-2.8.1` 에서 `openssh-2.9.1` 로
  - `pyenv-3.1.1` 에서 `pyenv-3.4.2` 로
  - `selinux-2.1.1` 에서 `selinux-3.1.1` 로
  - `ulimit-1.0.0` 에서 `ulimit-1.1.1` 로

	• `yum-5.1.0` 에서 `yum-6.1.1` 로 • `yum-epel-3.3.0` 에서 `yum-epel-4.1.2` 로 변경 사항에 대한 자세한 내용은 GitHub의 aws-parallelcluster, aws-parallelcluster-cookbook 및 aws-parallelcluster-node 패키지의 변경 로그 파일을 참조하세요.	
AWS ParallelCluster 버전 2.10.4 릴리스	AWS ParallelCluster 버전 2.10.4 릴리스. 변경 사항: • Slurm을 버전 `20.02.4`에서 버전 `20.02.7`로 업그레이드 변경 사항에 대한 자세한 내용은 GitHub의 aws-parallelcluster 패키지에 대한 변경 로그 파일을 참조하세요.	2021년 5월 15일

AWS ParallelCluster 버전 2.10.3 릴리스	AWS ParallelCluster 버전 2.10.3 릴리스.
	
	변경 사항:
	
	• AWS 중국 및 Ubuntu의 Arm 기반 AWS Graviton 인스턴스에서 18.04 및 Amazon Linux 2에 대한 지원이 추가되었습니다 AWS GovCloud (US) AWS 리전.
	• Elastic Fabric Adapter 설치 관리자가 1.11.2로 업데이트되었습니다.
	  • EFA 구성: `efa-config-1.7` (변경 없음)
	  • EFA 프로필: `efa-profile-1.3` 에서 `efa-profile-1.4` 로
	  • EFA 커널 모듈: `efa-1.10.2` (변경 없음)
	  • RDMA 코어: `rdma-core-31.2amzn` (변경 없음)
	  • Libfabric: `libfabric-1.11.1amzn1.0` (변경 없음)
	  • Open MPI: `openmpi40-aws-4.1.0` (변경 없음)
	
	변경 사항에 대한 자세한 내용은 GitHub의 aws-parallelcluster 패키지에 대한 변경 로그 파일을 참조하세요.
		2021년 3월 18일

AWS ParallelCluster 버전 2.10.2 릴리스	AWS ParallelCluster 버전 2.10.2 릴리스. 변경 사항: • `--dry-run` 모드에서 Amazon EC2 RunInstances API 작업을 호출할 때 클러스터 대상 AMI를 사용하도록 클러스터 구성 검증을 개선 • AWS ParallelCluster 가상 환경에서 사용되는 Python 버전을 3.6.13로 업데이트합니다. • Arm 인스턴스 유형을 위한 `sanity_check` 수정 • Slurm 스케줄러 또는 Arm 인스턴스 유형과 함께 `centos8`를 사용할 때 `enable_efa` 를 수정했습니다. • 비대화형 모드(`-y`)에서 `apt update` 실행 • `alinux2`와 `centos8`에서 `encrypted_ephemeral` = true를 수정 변경 사항에 대한 자세한 내용은 GitHub의 aws-parallelcluster 패키지에 대한 변경 로그 파일을 참조하세요.	2021년 3월 2일

AWS ParallelCluster 버전 2.10.1 릴리스	AWS ParallelCluster 버전 2.10.1 릴리스.
	
	변경 사항:
	
	• 아프리카(케이프타운)(af-south-1), 유럽(밀라노)(me-south-1) 및 중동(바레인)(me-south-1)에 대한 지원이 추가되었습니다 AWS 리전. 출시 시 지원은 다음과 같은 방식으로 제한됩니다.
	  • FSx for Lustre 및 ARM 기반 Graviton 인스턴스는 이 중 어느 AWS 리전에도 지원되지 않습니다.
	  • AWS Batch 는 아프리카(케이프타운)에서 지원되지 않습니다.
	  • Amazon EBS io2 및 gp3 볼륨 유형은 아프리카(케이프타운) 및 유럽(밀라노)에서 지원되지 않습니다 AWS 리전.
	• Amazon EBS io2 및 gp3 볼륨 유형에 대한 지원이 추가되었습니다. 자세한 내용은 [ebs] 섹션 및 [raid] 섹션을 참조하세요.
	• alinux2, ubuntu1804 또는 ubuntu2004 에서 실행되는 ARM 기반 Graviton2 인스턴스에 대한 Elastic Fabric Adapter 지원이 추가되었습니다. 자세한 내용은 Elastic
		2020년 12월 22일

Fabric Adapter 항목을 참조하세요.

- Arm AMI에 Arm 퍼포먼스 라이브러리 20.2.1을 설치 (`alinux2`, `centos8` 및 `ubuntu1804`) 자세한 내용은 Arm 퍼포먼스 라이브러리 항목을 참조하세요.
- Intel MPI는 버전 2019 Update 7에서 버전 2019 Update 8로 업데이트되었습니다. 자세한 내용은 Intel® MPI Library 2019 Update 8을 참조하세요.
- 제한으로 인한 작업 실패를 AWS CloudFormation `DescribeStacks` 종료하기 위해 AWS Batch Docker 진입점에서 API 작업 호출을 제거했습니다 AWS CloudFormation.
- 클러스터 구성을 검증할 때 Amazon EC2 `DescribeInstanceTypes` API 작업 호출에 대한 호출을 개선했습니다.
- `awsbatch` 스케줄러용 도커 이미지를 빌드할 때 Amazon Linux 2 도커 이미지를 Amazon ECR Public에서 가져옵니다.
- 기본 인스턴스 유형이 하드 코딩된 `t2.micro` 인스턴스 유형에서 AWS 리전 (`t2.micro`에 `t3.micro`따

라 또는 AWS 리전) AWS 리전 의 프리 티어 인스턴스 유형으로 변경되었습니다. 에는 프리 티어 기본값이 `t3.micro` 인스턴스 유형으로 지정되지 않습니다.

- Elastic Fabric Adapter 설치 관리자가 1.11.1로 업데이트되었습니다.
  - EFA 구성: `efa-config-1.5` 에서 `efa-config-1.7` 로
  - EFA 프로필: `efa-profile-1.1` 에서 `efa-profile-1.3` 로
  - EFA 커널 모듈: `efa-1.10.2` (변경 없음)
  - RDMA 코어: `rdma-core-31.amzn0` 에서 `rdma-core-31.2amzn` 로
  - Libfabric: `libfabric-1.10.1amzn1.1` 에서 `libfabric-1.11.1amzn1.0` 로
  - Open MPI: `openmpi40-aws-4.0.5` 에서 `openmpi40-aws-4.1.0` 로
- 이제 `vpc_settings` , `vpc_id` 및 `master_subnet_id` 파라미터는 필수입니다.

- 헤드 노드의 `nfsd` 대몬(daemon)은 이제 최소 8개 스레드를 사용하도록 설정되었습니다. 코어가 8개 이상인 경우 코어 수만큼의 스레드를 사용하게 됩니다. `ubuntu1604` 가 사용되면 노드가 재부팅된 후에만 설정이 변경됩니다.
- Amazon DCV가 Amazon DCV 2020.2-9662로 업데이트되었습니다. 자세한 내용은 Amazon DCV 관리자 가이드의 DCV 2020.2-9662— 2020년 12월 4일을 참조하세요.
- 에 대한 Intel MPI 및 HPC 패키지 AWS ParallelCluster 는 Amazon S3에서 가져옵니다. 더 이상 인텔 yum 리포지토리에서 가져오지 않습니다.
- 공식 AMI를 생성하는 동안 모든 OS`multi-user.target` 에서 기본 systemd 실행 수준을 로 변경했습니다. AWS ParallelCluster AMIs DCV가 활성화된 경우에만 헤드 노드에서 실행 수준이 `graphical.target` 로 설정됩니다. 이렇게 하면 그래픽 서비스(예:`x/gdm`)가 필요하지 않을 때 실행되지 않습니다.
- 헤드 노드에 `p4d.24xlarge` 인스턴스 지원을 활성화했습니다.

- Amazon Route 53에 Slurm 노드를 등록할 때 최대 재시도 횟수 증가

변경 사항에 대한 자세한 내용은 GitHub의 aws-parallelcluster, aws-parallelcluster-cookbook 및 aws-parallelcluster-node 패키지의 변경 로그 파일을 참조하세요.

AWS ParallelCluster 버전 2.10.0 릴리스	AWS ParallelCluster 버전 2.10.0 릴리스.
	
	변경 사항:
	
	• 모든 AWS 리전 (AWS 중국 및 AWS GovCloud(미국) 리전 외부)에서 CentOS 8에 대한 지원이 추가되었습니다. CentOS 6에 대한 지원이 제거되었습니다.
	• 컴퓨팅 노드용 `p4d.24xlarge` 인스턴스에 대한 지원이 추가되었습니다.
	• 새 `enable_efa_gdr` 설정을 사용하여 EFA에서 NVIDIA GPUDirect RDMA 지원을 추가했습니다.
	• Amazon FSx for Lustre 특능에 대한 지원이 추가되었습니다.
	  • `auto_import_policy` 설정을 사용하여 Amazon FSx for Lustre 파일 시스템이 기본 설정을 가져오도록 구성합니다.
	  • `storage_type` 및 `drive_cache_type` 설정을 사용하여 HDD 기반 Amazon FSx for Lustre 파일 시스템에 대한 지원이 추가되었습니다.
	• 헤드 노드 지표와 클러스터 로그에 대한 간편한 액세스를 포함하는 Amazon CloudWatch 대시보드를 추
		2020년 11월 18일

가했습니다. 자세한 내용은 Amazon CloudWatch 대시보드 항목을 참조하세요.

- `cluster_resource_bucket` 설정을 사용하여 기존 Amazon S3 버킷을 사용하여 클러스터 구성 정보를 저장하는 지원이 추가되었습니다.
- `pcluster createami` 명령을 개선했습니다.
  - AMI를 구축할 때 사후 설치 스크립트를 사용하기 위한 `--post-install` 파라미터가 추가되었습니다.
  - 다른 버전의에서 생성한 기본 AMI를 사용할 때 실패하는 검증 단계를 추가했습니다 AWS ParallelCluster.
  - 선택한 OS가 기본 AMI의 OS와 다를 경우 실패하는 검증 단계를 추가했습니다.
  - AWS ParallelCluster 기본 AMI 사용에 대한 지원이 추가되었습니다.
- `pcluster update` 명령을 개선했습니다.
  - 이제 업데이트 중에 `tags` 설정을 변경할 수 있습니다.
  - 이제 컴퓨팅 플릿을 중단하지 않고도 업데이트 중

에 대기열 크기를 조정할 수 있습니다.

- `all_or_nothing_batch` 스크립트용 `slurm_resume` 구성 파라미터가 추가되었습니다. `True`일 시, `slurm_resume` 는 Slurm에 있는 대기 중인 모든 작업에 필요한 모든 인스턴스를 사용할 수 있는 경우에만 성공합니다. 자세한 내용은 GitHub의 AWS ParallelCluster Wiki에서 [`all_or_nothing_batch` 출시 소개]를 참조하세요.
- Elastic Fabric Adapter 설치 관리자가 1.10.1로 업데이트되었습니다.
  - EFA 구성: `efa-config-1.4` 에서 `efa-config-1.5` 로
  - EFA 프로필: `efa-profile-1.0.0` 에서 `efa-profile-1.1` 로
  - EFA 커널 모듈: `efa-1.6.0` 에서 `efa-1.10.2` 로
  - RDMA 코어: `rdma-core-28.amzn0` 에서 `rdma-core-31.amzn0` 로
  - Libfabric: `libfabric-1.10.1amzn1.1` 에서

`libfabric-1.11.1amzn1.0` 로

- Open MPI: `openmpi40-aws-4.0.3` 에서 `openmpi40-aws-4.0.5` 로

- AWS GovCloud (US) 리전에서 Amazon DCV 및에 대한 지원을 활성화합니다 AWS Batch.
- AWS 중국 리전에서는 Amazon FSx for Lustre에 대한 지원을 활성화합니다.
- NVIDIA 드라이버를 버전 450.51.05에서 버전 450.80.02로 업그레이드
- NVIDIA Fabric Manager를 설치하여 지원되는 플랫폼에서 NVIDIA NVSwitch를 활성화하세요.
- AWS 리전 의 기본값을 제거했습니다`us-east-1` . 기본값은 이 조회 순서를 사용합니다.
  - AWS 리전 `-r` 또는 `--region` 인수에 지정됩니다.
  - `AWS_DEFAULT_REGION` 환경 변수.
  - `aws_region_name` AWS ParallelCluster 구성 파일의 [`[aws]` 섹션에서](#) 설정(기본값은 `~/.parallelcluster/config`).

- `region` AWS CLI 구성 파일의 `[default]` 섹션에서 설정(기본값은 `~/aws/config`).

변경 사항에 대한 자세한 내용은 GitHub의 aws-parallelcluster, aws-parallelcluster-cookbook 및 aws-parallelcluster-node 패키지의 변경 로그 파일을 참조하세요.

AWS ParallelCluster 버전 2.9.0 릴리스	AWS ParallelCluster 버전 2.9.0 릴리스. 변경 사항: • Slurm Workload Manager와 함께 사용할 경우 컴퓨팅 플릿의 여러 대기열 및 여러 인스턴스 유형에 대한 지원이 추가되었습니다. 대기열을 사용하는 경우 Auto Scaling 그룹은 Slurm에서 더 이상 사용되지 않습니다. Amazon Route 53 호스팅 영역은 이제 클러스터와 함께 생성되며 Slurm 스케줄러 사용 시 컴퓨팅 노드의 DNS 확인에 사용됩니다. 자세한 내용은 다중 대기열 모드 단원을 참조하십시오. • Arm 기반 AWS Graviton 기반 인스턴스의 Amazon DCV에 대한 지원이 추가되었습니다. • 시작 템플릿에서 CPU 옵션을 지원하지 않는 인스턴스 유형(예: `*.metal` 인스턴스 유형)에서 하이퍼스레딩을 비활성화하는 지원이 추가되었습니다. • 헤드 노드에서 공유하는 파일 시스템을 위한 NFS 4에 대한 지원이 추가되었습니다. • 많은 수의 노드가 클러스터에 조인할 때 제한이 발	2020년 9월 11일

생하지 않도록 컴퓨팅 노드를 부트스트래핑 AWS CloudFormation 할 때 cfn-init에 대한 종속성을 제거했습니다.

- Elastic Fabric Adapter 설치 관리자가 1.9.5로 업데이트되었습니다.
  - EFA 구성: `efa-config-1.3` 에서 `efa-config-1.4` 로
  - EFA 프로필: `efa-profile-1.0.0` (신규)
  - 커널 모듈: `efa-1.6.0` (변경 없음)
  - RDMA 코어: `rdma-core-28.amzn0` (변경 없음)
  - Libfabric: `libfabric-1.10.1amzn1.1` (변경 없음)
  - Open MPI: `openmpi40-aws-4.0.3` (변경 없음)
- Slurm을 버전 `19.05.5`에서 버전 `20.02.4`로 업그레이드
- Amazon DCV가 Amazon DCV 2020.1-9012로 업데이트되었습니다. 자세한 내용은 Amazon DCV 관리자 가이드의 DCV 2020.1-9012 - 2020년 8월 24일 릴리스 노트를 참조하세요.
- 공유 NFS 드라이브를 탑재할 때는 호스트 이름 대신 헤

드 노드 사설 IP 주소를 사용하세요.

- CloudWatch Logs에 새 로그 스트림 `chef-client` , `clustermgtd` , `computemgtd` , `slurm_resume` , `slurm_suspend` 가 추가되었습니다.
- 설치 전 및 설치 후 스크립트에 대기열 이름에 대한 지원이 추가되었습니다.
- 에서 Amazon DynamoDB 온디맨드 결제 옵션을 AWS GovCloud (US) AWS 리전 사용합니다. 자세한 내용은 Amazon DynamoDB 개발자 안내서의 온디맨드 모드를 참조하세요.

변경 사항에 대한 자세한 내용은 GitHub의 aws-parallelcluster, aws-parallelcluster-cookbook 및 aws-parallelcluster-node 패키지의 변경 로그 파일을 참조하세요.

AWS ParallelCluster 버전 2.8.1 릴리스	AWS ParallelCluster 버전 2.8.1 릴리스. 변경 사항: • 사용자가 차단되는 것을 방지하기 위해 Amazon DCV 세션의 화면 잠금을 비활성화합니다. • ARM 기반 AWS Graviton 기반 인스턴스 유형을 포함할 때 `pcluster configure` 수정 변경 사항에 대한 자세한 내용은 GitHub의 aws-parallelcluster, aws-parallelcluster-cookbook 및 aws-parallelcluster-node 패키지의 변경 로그 파일을 참조하세요.	2020년 8월 4일

AWS ParallelCluster 버전 2.8.0 릴리스	AWS ParallelCluster 버전 2.8.0 릴리스. 변경 사항: • Arm 기반 AWS Graviton 기반 인스턴스(예: A1 및)에 대한 지원이 추가되었습니다C6g. • Amazon FSx for Lustre의 자동 일일 백업 특성에 대한 지원이 추가되었습니다. 자세한 내용은 automatic_backup_retention_days , copy_tags_to_backups , daily_automatic_backup_start_time 및 fsx_backup_id 부분을 참조하세요. • pcluster createami 에서 Berkshelf에 대한 종속성을 제거했습니다. • pcluster update의 견고성과 사용자 경험을 개선했습니다. 자세한 내용은 pcluster update 사용하기 항목을 참조하세요. • Elastic Fabric Adapter 설치 관리자가 1.9.4로 업데이트되었습니다. • 커널 모듈: efa-1.5.1 에서 efa-1.6.0 로 업데이트됨	2020년 7월 23일

  - RDMA 코어: `rdma-core-25.0` 에서 `rdma-core-28.amzn0` 로 업데이트됨
  - 라이브러리: `libfabric-aws-1.9.0amzn1.1` 에서 `libfabric-1.10.1amzn1.1` 로 업데이트됨
  - Open MPI: `openmpi40-aws-4.0.3` (변경 없음)
- NVIDIA 드라이버를 CentOS 6에서는 Tesla 버전 440.95.01 버전으로, 다른 모든 배포판에서는 버전 450.51.05로 업그레이드
- CentOS 6을 제외한 모든 배포판에서 CUDA 라이브러리를 버전 11.0으로 업그레이드

변경 사항에 대한 자세한 내용은 GitHub의 aws-parallelcluster, aws-parallelcluster-cookbook 및 aws-parallelcluster-node 패키지의 변경 로그 파일을 참조하세요.

AWS ParallelCluster 버전 2.7.0 릴리스	AWS ParallelCluster 버전 2.7.0 릴리스. 변경 사항: • base_os는 이제 필수 파라미터입니다. • scheduler 는 이제 필수 파라미터입니다. • Amazon DCV가 Amazon DCV 2020.0으로 업데이트되었습니다. 자세한 내용은 서라운드 사운드 7.1 및 스타일러스를 지원하는 Amazon DCV 릴리스 버전 2020.0을 참조하세요. Intel MPI는 2019 Update 7 버전(2019 Update 6 버전에서 업데이트됨)으로 업데이트되었습니다. 자세한 내용은 Intel® MPI Library 2019 Update 7을 참조하세요. Elastic Fabric Adapter 설치 관리자는 1.8.4로 업데이트되었습니다. • 커널 모듈: efa-1.5.1 (변경 없음) • RDMA 코어: rdma-core-25.0 (변경 없음) • Libfabric: libfabric-aws-1.9.0amzn1.1 (변경 없음) • Open MPI: openmpi40-aws-4.0.3 (openmpi40	2020년 5월 19일

	`-aws-4.0.2` 에서 업데이트됨) • CentOS 7 AMI를 7.8-2003 버전으로 업그레이드합니다(7.7-1908에서 업데이트됨). 자세한 내용은 CentOS-7(2003) 릴리스 정보를 참조하세요.	
AWS ParallelCluster 버전 2.6.1 릴리스	AWS ParallelCluster 버전 2.6.1 릴리스. 변경 사항: • Amazon CloudWatch Logs에 저장된 `cfn-wire` 로그에서 `cfn-init-cmd` 를 제거했습니다. 자세한 내용은 Amazon CloudWatch Logs와 통합 항목을 참조하세요.	2020년 4월 17일

AWS ParallelCluster 버전 2.6.0 릴리스	AWS ParallelCluster 버전 2.6.0 릴리스. 변경 사항: • Amazon Linux 2에 대한 지원이 추가되었습니다. • 이제 Amazon CloudWatch Logs가 클러스터 및 스케줄러 로그를 수집하는 데 사용됩니다. 자세한 내용은 Amazon CloudWatch Logs와 통합 항목을 참조하세요. • 새 Amazon FSx for Lustre 배포 유형 `SCRATCH_2` 및 `PERSISTENT_1` 에 대한 지원이 추가되었습니다. Ubuntu 18.04 및 Ubuntu 16.04에서 FSx for Lustre 지원 자세한 내용은 fsx를 참조하세요. • Ubuntu 18.04에서 Amazon DCV에 대한 지원이 추가되었습니다. 자세한 내용은 Amazon DCV를 통해 헤드 노드에 연결합니다. 단원을 참조하십시오.	2020년 2월 27일
AWS ParallelCluster 버전 2.5.1 릴리스	AWS ParallelCluster 버전 2.5.1 릴리스.	2019년 12월 13일
AWS ParallelCluster 버전 2.5.0 릴리스	AWS ParallelCluster 버전 2.5.0 릴리스.	2019년 11월 18일
AWS ParallelCluster 에서 Intel MPI에 대한 지원 도입	AWS ParallelCluster 버전 2.4.1에는 Intel MPI에 대한 지원이 도입되었습니다.	2019년 7월 29일

AWS ParallelCluster 에서 EFA에 대한 지원 도입	AWS ParallelCluster 버전 2.4.0에는 EFA(Elastic Fabric Adapter)에 대한 지원이 도입되었습니다.	2019년 6월 11일
AWS ParallelCluster 설명서 사이트에서 릴리스된 AWS 설명서	이제 AWS ParallelCluster 설명서는 10개 언어와 HTML 및 PDF 형식으로 제공됩니다.	2018년 5월 24일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.