



AWS ParallelCluster User Guide (v2)

AWS ParallelCluster



AWS ParallelCluster: AWS ParallelCluster User Guide (v2)

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

.....	xiii
What is AWS ParallelCluster	1
Pricing	1
Setting up AWS ParallelCluster	2
Installing AWS ParallelCluster	2
Installing AWS ParallelCluster in a virtual environment (recommended)	2
Installing AWS ParallelCluster in a non-virtual environment using pip	3
Steps to take after installation	3
Detailed instructions for each environment	4
Virtual environment	4
Linux	6
macOS	10
Windows	13
Configuring AWS ParallelCluster	15
Best practices	23
Best practices: master instance type selection	23
Best practices: network performance	24
Best practices: budget alerts	25
Best practices: moving a cluster to a new AWS ParallelCluster minor or patch version	25
Moving from CfnCluster to AWS ParallelCluster	26
Supported Regions	28
Using AWS ParallelCluster	30
Network configurations	30
AWS ParallelCluster in a single public subnet	31
AWS ParallelCluster using two subnets	32
AWS ParallelCluster in a single private subnet connected using AWS Direct Connect	33
AWS ParallelCluster with awsbatch scheduler	34
Custom Bootstrap Actions	35
Configuration	37
Arguments	37
Example	37
Working with Amazon S3	39
Examples	39
Working with Spot Instances	40

Scenario 1: Spot Instance with no running jobs is interrupted	40
Scenario 2: Spot Instance running single node jobs is interrupted	40
Scenario 3: Spot Instance running multi-node jobs is interrupted	42
AWS Identity and Access Management roles in AWS ParallelCluster	43
Default settings for cluster creation	44
Using an existing IAM role for Amazon EC2	44
AWS ParallelCluster example instance and user policies	44
Schedulers supported by AWS ParallelCluster	87
Son of Grid Engine	87
Slurm Workload Manager	88
Torque Resource Manager	100
AWS Batch	100
Tagging	107
Amazon CloudWatch dashboard	111
Integration with Amazon CloudWatch Logs	113
Elastic Fabric Adapter	115
Intel Select Solutions	116
Enable Intel MPI	118
Intel HPC Platform Specification	119
Arm Performance Libraries	120
Connect to the head node through Amazon DCV	121
Amazon DCV HTTPS certificate	122
Licensing Amazon DCV	123
Using <code>pcluster update</code>	123
AMI patching and EC2 instance replacement	125
Head node instance update or replacement	126
Instance store limitations	127
Instance store limitations workarounds	127
Stop and start a cluster's head node	128
AWS ParallelCluster CLI commands	131
<code>pcluster</code>	131
Arguments	131
Sub-commands:	131
<code>pcluster configure</code>	132
<code>pcluster create</code>	133
<code>pcluster createami</code>	135

pcluster dcv	138
pcluster delete	141
pcluster instances	143
pcluster list	144
pcluster ssh	144
pcluster start	146
pcluster status	147
pcluster stop	148
pcluster update	149
pcluster version	151
pcluster-config	151
Named arguments	152
Configuration	154
Layout	155
[global] section	155
cluster_template	155
update_check	156
sanity_check	156
[aws] section	156
[aliases] section	157
[cluster] section	158
additional_cfn_template	160
additional_iam_policies	160
base_os	161
cluster_resource_bucket	163
cluster_type	164
compute_instance_type	164
compute_root_volume_size	165
custom_ami	165
cw_log_settings	166
dashboard_settings	167
dcv_settings	167
desired_vcpus	168
disable_cluster_dns	168
disable_hyperthreading	169
ebs_settings	169

ec2_iam_role	170
efs_settings	170
enable_efa	171
enable_efa_gdr	171
enable_intel_hpc_platform	172
encrypted_ephemeral	173
ephemeral_dir	173
extra_json	173
fsx_settings	174
iam_lambda_role	174
initial_queue_size	175
key_name	176
maintain_initial_size	176
master_instance_type	177
master_root_volume_size	177
max_queue_size	178
max_vcpus	178
min_vcpus	179
placement	179
placement_group	180
post_install	180
post_install_args	181
pre_install	181
pre_install_args	181
proxy_server	182
queue_settings	182
raid_settings	183
s3_read_resource	183
s3_read_write_resource	184
scaling_settings	184
scheduler	184
shared_dir	185
spot_bid_percentage	186
spot_price	186
tags	187
template_url	188

vpc_settings	188
[compute_resource] section	188
initial_count	189
instance_type	190
max_count	190
min_count	190
spot_price	191
[cw_log] section	191
enable	192
retention_days	192
[dashboard] section	192
enable	193
[dcv] section	193
access_from	194
enable	194
port	195
[ebs] section	195
shared_dir	196
ebs_kms_key_id	197
ebs_snapshot_id	197
ebs_volume_id	197
encrypted	197
volume_iops	198
volume_size	199
volume_throughput	200
volume_type	200
[efs] section	201
efs_fs_id	202
efs_kms_key_id	203
encrypted	203
performance_mode	204
provisioned_throughput	204
shared_dir	205
throughput_mode	205
[fsx] section	205
auto_import_policy	207

automatic_backup_retention_days	208
copy_tags_to_backups	209
daily_automatic_backup_start_time	209
data_compression_type	210
deployment_type	210
drive_cache_type	212
export_path	212
fsx_backup_id	212
fsx_fs_id	213
fsx_kms_key_id	213
import_path	214
imported_file_chunk_size	214
per_unit_storage_throughput	215
shared_dir	215
storage_capacity	216
storage_type	217
weekly_maintenance_start_time	219
[queue] section	219
compute_resource_settings	220
compute_type	220
disable_hyperthreading	221
enable_efa	221
enable_efa_gdr	221
placement_group	222
[raid] section	223
shared_dir	224
ebs_kms_key_id	224
encrypted	224
num_of_raid_volumes	225
raid_type	225
volume_iops	225
volume_size	226
volume_throughput	227
volume_type	228
[scaling] section	229
scaledown_idletime	229

[vpc] section	229
additional_sg	230
compute_subnet_cidr	230
compute_subnet_id	230
master_subnet_id	231
ssh_from	231
use_public_ips	231
vpc_id	232
vpc_security_group_id	232
Examples	39
Slurm example	233
SGE and Torque example	234
AWS Batch example	235
How AWS ParallelCluster works	237
AWS ParallelCluster processes	237
SGE and Torque integration processes	238
Slurm integration processes	244
AWS services used by AWS ParallelCluster	244
AWS Auto Scaling	245
AWS Batch	246
CloudFormation	246
Amazon CloudWatch	246
Amazon CloudWatch Logs	247
AWS CodeBuild	247
Amazon DynamoDB	247
Amazon Elastic Block Store	247
Amazon Elastic Compute Cloud	248
Amazon Elastic Container Registry	248
Amazon EFS	248
Amazon FSx for Lustre	248
AWS Identity and Access Management	249
AWS Lambda	249
Amazon DCV	249
Amazon Route 53	250
Amazon Simple Notification Service	250
Amazon Simple Queue Service	250

Amazon Simple Storage Service	251
Amazon VPC	251
AWS ParallelCluster Auto Scaling	251
Scaling up	252
Scaling down	253
Static cluster	254
Tutorials	255
Running your first job on AWS ParallelCluster	255
Verifying your installation	255
Creating your first cluster	256
Logging into your head node	256
Running your first job using SGE	257
Building a Custom AWS ParallelCluster AMI	258
How to Customize the AWS ParallelCluster AMI	259
Modify an AMI	259
Build a Custom AWS ParallelCluster AMI	262
Use a Custom AMI at Runtime	263
Running an MPI job with AWS ParallelCluster and awsbatch scheduler	264
Creating the cluster	264
Logging into your head node	256
Running your first job using AWS Batch	266
Running an MPI job in a multi-node parallel environment	268
Disk encryption with a custom KMS Key	272
Creating the role	272
Give your key permissions	273
Creating the cluster	264
Multiple queue mode tutorial	274
Running your jobs on AWS ParallelCluster with multiple queue mode	274
Development	287
Setting up a custom AWS ParallelCluster cookbook	287
Steps	287
Setting up a custom AWS ParallelCluster node package	289
Steps	289
Troubleshooting	291
Retrieving and preserving logs	291
Troubleshooting stack deployment issues	292

Troubleshooting issues in multiple queue mode clusters	292
Key logs	293
Troubleshooting node initialization issues	294
Troubleshooting unexpected node replacements and terminations	296
Replacing, terminating, or powering down problem instances and nodes	297
Troubleshooting other known node and job issues	297
Troubleshooting issues in single queue mode clusters	298
Key logs	298
Troubleshooting failed launch and join operations	300
Troubleshooting scaling issues	300
Troubleshooting other cluster related issues	300
Placement groups and instance launch issues	301
Directories that cannot be replaced	301
Troubleshooting issues in Amazon DCV	302
Logs for Amazon DCV	302
Amazon DCV instance type memory	303
Ubuntu Amazon DCV issues	303
Troubleshooting issues in clusters with AWS Batch integration	303
Head node issues	303
AWS Batch multi-node parallel jobs submission issues	304
Compute issues	304
Job failures	304
Troubleshooting when a resource fails to create	304
Troubleshooting IAM policy size issues	305
Additional support	306
AWS ParallelCluster support policy	307
Security	308
Security information for services used by AWS ParallelCluster	308
Data protection	309
Data encryption	310
See also	311
Identity and Access Management	311
Compliance validation	312
Enforcing TLS 1.2	313
Determine Your Currently Supported Protocols	313
Compile OpenSSL and Python	315

Release notes and document history 317

This documentation references AWS ParallelCluster. For running HPC and AI workloads, we recommend [AWS Parallel Computing Service \(AWS PCS\)](#), our managed Slurm service that lets you focus on running your workloads rather than managing infrastructure.

What is AWS ParallelCluster

AWS ParallelCluster is an AWS supported open source cluster management tool that helps you to deploy and manage high performance computing (HPC) clusters in the AWS Cloud. It automatically sets up the required compute resources, scheduler, and shared filesystem. You can use AWS ParallelCluster with AWS Batch and Slurm schedulers.

With AWS ParallelCluster, you can quickly build and deploy proof of concept and production HPC compute environments. You can also build and deploy a high level workflow on top of AWS ParallelCluster, such as a genomics portal that automates an entire DNA sequencing workflow.

Pricing

When using the AWS ParallelCluster command line interface (CLI) or API, you only pay for the AWS resources that are created when you create or update AWS ParallelCluster images and clusters. For more information, see [AWS services used by AWS ParallelCluster](#).

Setting up AWS ParallelCluster

Topics

- [Installing AWS ParallelCluster](#)
- [Configuring AWS ParallelCluster](#)
- [Best practices](#)
- [Moving from CfnCluster to AWS ParallelCluster](#)
- [Supported Regions](#)

Installing AWS ParallelCluster

AWS ParallelCluster is distributed as a Python package and is installed using `pip`, the Python package manager. For more information on installing Python packages, see [Installing packages](#) in the *Python Packaging User Guide*.

Ways to install AWS ParallelCluster:

- [Using a virtual environment \(recommended\)](#)
- [Using pip](#)

You can find the version number of the most recent CLI on the [releases page on GitHub](#).

In this guide, the command examples assume that you have Python v3 installed. The `pip` command examples use the `pip3` version.

Installing AWS ParallelCluster in a virtual environment (recommended)

We recommend that you install AWS ParallelCluster in a virtual environment. If you encounter issues when you attempt to install AWS ParallelCluster with `pip3`, you can [install AWS ParallelCluster in a virtual environment](#) to isolate the tool and its dependencies. Or you can use a different version of Python than you normally do.

Installing AWS ParallelCluster in a non-virtual environment using pip

The primary distribution method for AWS ParallelCluster on Linux, Windows, and macOS is `pip`, which is a package manager for Python. It provides a way to install, upgrade, and remove Python packages and their dependencies.

Current AWS ParallelCluster Version

AWS ParallelCluster is updated regularly. To determine whether you have the latest version, see the [releases page on GitHub](#).

If you already have `pip` and a supported version of Python, you can install AWS ParallelCluster by using the following command. If you have Python version 3+ installed, we recommend that you use the **`pip3`** command.

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

Steps to take after installation

After you install AWS ParallelCluster, you might need to add the executable file path to your `PATH` variable. For platform-specific instructions, see the following topics:

- **Linux** – [Add the AWS ParallelCluster executable to your command line path](#)
- **macOS** – [Add the AWS ParallelCluster executable to your command line path](#)
- **Windows** – [Add the AWS ParallelCluster executable to your command line path](#)

You can verify that AWS ParallelCluster installed correctly by running `pcluster version`.

```
$ pcluster version
2.11.9
```

AWS ParallelCluster is updated regularly. To update to the latest version of AWS ParallelCluster, run the installation command again. For details about the latest version of AWS ParallelCluster, see the [AWS ParallelCluster release notes](#).

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

To uninstall AWS ParallelCluster, use `pip uninstall`.


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

If you don't have Python and pip, use the procedure for your environment.

Detailed instructions for each environment

- [Install AWS ParallelCluster in a virtual environment \(recommended\)](#)
- [Install AWS ParallelCluster on Linux](#)
- [Install AWS ParallelCluster on macOS](#)
- [Install AWS ParallelCluster on Windows](#)

Install AWS ParallelCluster in a virtual environment (recommended)

We recommend that you install AWS ParallelCluster in a virtual environment to avoid requirement version conflicts with other pip packages.

Prerequisites

- Verify that pip and Python are installed. We recommend pip3, and Python 3 version 3.8. If you are using Python 2, use pip instead of pip3 and virtualenv instead of venv.

To install AWS ParallelCluster in a virtual environment

1. If virtualenv is not installed, install virtualenv using pip3. If `python3 -m virtualenv help` displays help information, go to step 2.

Linux, macOS, or Unix

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

Run `exit` to leave the current terminal window and open a new terminal window to pick up changes to the environment.

Windows

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

Run `exit` to leave the current command prompt and open a new command prompt to pick up changes to the environment.

2. Create a virtual environment and name it.

Linux, macOS, or Unix

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

Alternatively, you can use the `-p` option to specify a specific version of Python.

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

Windows

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

3. Activate your new virtual environment.

Linux, macOS, or Unix

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

Windows

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

4. Install AWS ParallelCluster into your virtual environment.

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. Verify that AWS ParallelCluster is installed correctly.

Linux, macOS, or Unix

```
$ pcluster version  
2.11.9
```

Windows

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

You can use the `deactivate` command to exit the virtual environment. Each time you start a session, you must [reactivate the environment](#).

To upgrade to the latest version of AWS ParallelCluster, run the installation command again.

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"
```

Install AWS ParallelCluster on Linux

You can install AWS ParallelCluster and its dependencies on most Linux distributions by using `pip`, a package manager for Python. First, determine if Python and `pip` are installed:

1. To determine if your version of Linux includes Python and `pip`, run `pip --version`.

```
$ pip --version
```

If you have `pip` installed, go on to the [Install AWS ParallelCluster with `pip`](#) topic. Otherwise, continue with Step 2.

2. To determine if Python is installed, run `python --version`.

```
$ python --version
```

If you have Python 3 version 3.6+ or Python 2 version 2.7 installed, go on to the [Install AWS ParallelCluster with pip](#) topic. Otherwise, [install Python](#), and then return to this procedure to install pip.

3. Install pip by using the script that the *Python Packaging Authority* provides.
4. Use the `curl` command to download the installation script.

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

5. Run the script with Python to download and install the latest version of pip and other required support packages.

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

or

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

When you include the `--user` switch, the script installs pip to the path `~/.local/bin`.

6. To verify that the folder that contains pip is part of your PATH variable, do the following:
 - a. Find your shell's profile script in your user folder. If you're not sure which shell you have, run `basename $SHELL`.

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

- **Bash** – `.bash_profile`, `.profile`, or `.bash_login`
- **Zsh** – `.zshrc`
- **Tcsh** – `.tcshrc`, `.cshrc` or `.login`

- b. Add an export command at the end of your profile script that's similar to the following example.

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

The export command inserts the path, which is `~/local/bin` in this example, at the front of the existing PATH variable.

- c. To put these changes into effect, reload the profile into your current session.

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

7. Verify that pip is installed correctly.

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

Sections

- [Install AWS ParallelCluster with pip](#)
- [Add the AWS ParallelCluster executable to your command line path](#)
- [Installing Python on Linux](#)

Install AWS ParallelCluster with pip

Use pip to install AWS ParallelCluster.

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

When you use the `--user` switch, pip installs AWS ParallelCluster to `~/local/bin`.

Verify that AWS ParallelCluster installed correctly.

```
$ pcluster version
2.11.9
```

To upgrade to the latest version, run the installation command again.

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

Add the AWS ParallelCluster executable to your command line path

After installing with pip, you might need to add the `pcluster` executable to your operating system's PATH environment variable.

To verify the folder in which pip installed AWS ParallelCluster, run the following command.

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

If you omitted the `--user` switch when you installed AWS ParallelCluster, the executable might be in the `bin` folder of your Python installation. If you don't know where Python is installed, run this command.

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

Note that the output might be the path to a symlink, not to the actual executable. To see where the symlink points, run `ls -al`.

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

If this is the same folder that you added to the path in step 3 in [Installing AWS ParallelCluster](#), you're done with the installation. Otherwise, you must perform steps 3a – 3c again, adding this additional folder to the path.

Installing Python on Linux

If your distribution didn't come with Python, or came with an earlier version, install Python before installing pip and AWS ParallelCluster.

To install Python 3 on Linux

1. Check to see if Python is already installed.

```
$ python3 --version
```

or

```
$ python --version
```

Note

If your Linux distribution came with Python, you might need to install the Python developer package. The developer package includes the headers and libraries that are required to compile extensions and to install AWS ParallelCluster. Use your package manager to install the developer package. It is typically named `python-dev` or `python-devel`.

2. If Python 2.7 or later is not installed, install Python with your distribution's package manager. The command and package name varies:

- On Debian derivatives such as Ubuntu, use `apt`.

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

- On Red Hat and derivatives, use `yum`.

```
$ sudo yum install python3
```

- On SUSE and derivatives, use `zypper`.

```
$ sudo zypper install python3
```

3. To verify that Python installed correctly, open a command prompt or shell and run the following command.

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

Install AWS ParallelCluster on macOS

Sections

- [Prerequisites](#)
- [Install AWS ParallelCluster on macOS using pip](#)
- [Add the AWS ParallelCluster executable to your command line path](#)

Prerequisites

- Python 3 version 3.7+ or Python 2 version 2.7

Check your Python installation.

```
$ python --version
```

If your computer doesn't already have Python installed, or if you want to install a different version of Python, follow the procedure in [Install AWS ParallelCluster on Linux](#).

Install AWS ParallelCluster on macOS using pip

You can also use pip directly to install AWS ParallelCluster. If you don't have pip, follow the instructions in the main [installation topic](#). Run `pip3 --version` to see if your version of macOS already includes Python and pip3.

```
$ pip3 --version
```

To install AWS ParallelCluster on macOS

1. Download and install the latest version of Python from the [downloads page](#) of [Python.org](#).
2. Download and run the pip3 installation script provided by the Python Packaging Authority.

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

3. Use your newly installed pip3 to install AWS ParallelCluster. We recommend that if you use Python version 3+, you use the pip3 command.

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

4. Verify that AWS ParallelCluster is installed correctly.

```
$ pcluster version  
2.11.9
```

If the program isn't found, [add it to your command line path](#).

To upgrade to the latest version, run the installation command again.

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

Add the AWS ParallelCluster executable to your command line path

After installing with pip, you might need to add the `pcluster` program to your operating system's PATH environment variable. The location of the program depends on where Python is installed.

Example AWS ParallelCluster install location - macOS with Python 3.6 and pip (user mode)

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

Substitute the version of Python that you have for the version in the preceding example.

If you don't know where Python is installed, run `which python`.

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

The output might be the path to a symlink, not the path to the actual program. Run `ls -al` to see where it points.

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

pip installs programs in the same folder that contains the Python application. Add this folder to your PATH variable.

To modify your PATH variable (Linux, macOS, or Unix)

1. Find your shell's profile script in your user folder. If you're not sure which shell you have, run `echo $SHELL`.

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

- **Bash** – `.bash_profile`, `.profile`, or `.bash_login`

- **Zsh** – `.zshrc`
 - **Tcsh** – `.tcshrc`, `.cshrc`, or `.login`
2. Add an export command to your profile script.

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

This command adds a path, `~/.local/bin` in this example, to the current `PATH` variable.

3. Load the profile into your current session.

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

Install AWS ParallelCluster on Windows

You can install AWS ParallelCluster on Windows by using `pip`, which is a package manager for Python. If you already have `pip`, follow the instructions in the main [installation topic](#).

Sections

- [Install AWS ParallelCluster using Python and pip on Windows](#)
- [Add the AWS ParallelCluster executable to your command line path](#)

Install AWS ParallelCluster using Python and pip on Windows

The Python Software Foundation provides installers for Windows that include `pip`.

To install Python and pip (Windows)

1. Download the Python Windows x86-64 installer from the [downloads page](#) of [Python.org](#).
2. Run the installer.
3. Choose **Add Python 3 to PATH**.
4. Choose **Install Now**.

The installer installs Python in your user folder and adds its program folders to your user path.

To install AWS ParallelCluster with pip3 (Windows)

If you use Python version 3+, we recommend that you use the `pip3` command.

1. Open the **Command Prompt** from the **Start** menu.
2. Use the following commands to verify that Python and pip are both installed correctly.

```
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. Install AWS ParallelCluster using pip.

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

4. Verify that AWS ParallelCluster is installed correctly.

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

To upgrade to the latest version, run the installation command again.

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

Add the AWS ParallelCluster executable to your command line path

After installing AWS ParallelCluster with pip, add the `pcluster` program to your operating system's PATH environment variable.

You can find where the `pcluster` program is installed by running the following command.

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

If that command does not return any results, then you must add the path manually. Use the command line or Windows Explorer to discover where it is installed on your computer. Typical paths include:

- **Python 3 and pip3** – C:\Python38\Scripts\
- **Python 3 and pip3 --user option** – %APPDATA%\Python\Python38\Scripts

Note

Folder names that include version numbers can vary. The preceding examples show Python38. Replace as needed with the version number that you are using.

To modify your PATH variable (Windows)

1. Press the Windows key and enter **environment variables**.
2. Choose **Edit environment variables for your account**.
3. Choose **PATH**, and then choose **Edit**.
4. Add the path to the **Variable value** field. For example: **C:\new\path**
5. Choose **OK** twice to apply the new settings.
6. Close any running command prompts and reopen the command prompt window.

Configuring AWS ParallelCluster

After you install AWS ParallelCluster, complete the following configuration steps.

Verify that your AWS Account has a role that includes the permissions needed to run the [pcluster](#) CLI. For more information, see [AWS ParallelCluster example instance and user policies](#).

Set up your AWS credentials. For more information, see [Configuring the AWS CLI](#) in the *AWS CLI user guide*.

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

The AWS Region where the cluster is launched must have at least one Amazon EC2 key pair. For more information, see [Amazon EC2 key pairs](#) in the *Amazon EC2 User Guide*.

```
$ pcluster configure
```

The configure wizard prompts you for all of the information that's needed to create your cluster. The details of the sequence differ when using AWS Batch as the scheduler compared to using Slurm. For more information about a cluster configuration, see [Configuration](#).

 Note

Starting with version 2.11.5, AWS ParallelCluster doesn't support the use of SGE or Torque schedulers. You can continue using them in versions up to and including 2.11.4, but they aren't eligible for future updates or troubleshooting support from the AWS service and AWS Support teams.

Slurm

From the list of valid AWS Region identifiers, choose the AWS Region where you want your cluster to run.

 Note

The list of AWS Regions shown is based on the partition of your account, and only includes AWS Regions that are enabled for your account. For more information about enabling AWS Regions for your account, see [Managing AWS Regions](#) in the *AWS General Reference*. The example shown is from the AWS Global partition. If your account is in the AWS GovCloud (US) partition, only AWS Regions in that partition are listed (gov-us-east-1 and gov-us-west-1). Similarly, if your account is in the AWS China partition, only cn-north-1 and cn-northwest-1 are shown. For the complete list of AWS Regions supported by AWS ParallelCluster, see [Supported Regions](#).

Allowed values for the AWS Region 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 Region ID [ap-northeast-1]:
```

Choose the scheduler to use with your cluster.

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

Choose the operating system.

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

 Note

Support for `alinux2` was added in AWS ParallelCluster version 2.6.0.

The minimum and maximum size of the cluster of compute nodes is entered. This is measured in number of instances.

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

The head and compute nodes instance types are entered. For instance types, your account instance limits are large enough to meet your requirements. For more information, see [On-Demand Instance limits](#) in the *Amazon EC2 User Guide*.

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

The key pair is selected from the key pairs registered with Amazon EC2 in the selected AWS Region.

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

After the previous steps are completed, decide whether to use an existing VPC or let AWS ParallelCluster create a VPC for you. If you don't have a properly configured VPC, AWS ParallelCluster can create a new one. It either uses both the head and compute nodes in the same public subnet, or only the head node in a public subnet with all nodes in a private subnet. It's possible to reach your limit on number of VPCs in a AWS Region. The default limit is five VPCs for each AWS Region. For more information about this limit and how to request an increase, see [VPC and subnets](#) in the *Amazon VPC User Guide*.

If you let AWS ParallelCluster create a VPC, you must decide if all nodes should be in a public subnet.

Important

VPCs created by AWS ParallelCluster do not enable VPC Flow Logs by default. VPC Flow Logs enable you to capture information about the IP traffic going to and from network interfaces in your VPCs. For more information, see [VPC Flow Logs](#) in the *Amazon VPC User Guide*.

Note

If you choose 1. Master in a public subnet and compute fleet in a private subnet, AWS ParallelCluster creates a NAT gateway that results in additional cost, even if you specify free tier resources.

```
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
```

If you don't create a new VPC, you must select an existing VPC.

If you choose to have AWS ParallelCluster create the VPC, make a note of the VPC ID so you can use the AWS CLI to delete it later.

```
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
```

After the VPC has been selected, you need to decide whether to use existing subnets or create new ones.

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

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


AWS Batch

From the list of valid AWS Region identifiers, choose the AWS Region where you want your cluster to run.

Allowed values for AWS Region 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 Region ID [ap-northeast-1]:

Choose the scheduler to use with your cluster.

Allowed values for Scheduler:

1. slurm
2. awsbatch

Scheduler [awsbatch]:

When `awsbatch` is selected as the scheduler, `alinux2` is used as the operating system.

The minimum and maximum size of the cluster of compute nodes is entered. This is measured in vCPUs.

Minimum cluster size (vcpus) [0]:

Maximum cluster size (vcpus) [10]:

The head node instance type is entered. When using the `awsbatch` scheduler, the compute nodes use an instance type of `optimal`.

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

The Amazon EC2 key pair is selected from the key pairs registered with Amazon EC2 in the selected AWS Region.

```
Allowed values for EC2 Key Pair Name:
```

1. prod-uswest1-key
2. test-uswest1-key

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

Decide whether to use existing VPCs or let AWS ParallelCluster create VPCs for you. If you don't have a properly configured VPC, AWS ParallelCluster can create a new one. It either uses both the head and compute nodes in the same public subnet, or only the head node in a public subnet with all nodes in a private subnet. It's possible to reach your limit on number of VPCs in a AWS Region. The default number of VPCs is five. For more information about this limit and how to request an increase, see [VPC and subnets](#) in the *Amazon VPC User Guide*.

Important

VPCs created by AWS ParallelCluster do not enable VPC Flow Logs by default. VPC Flow Logs enable you to capture information about the IP traffic going to and from network interfaces in your VPCs. For more information, see [VPC Flow Logs](#) in the *Amazon VPC User Guide*.

If you let AWS ParallelCluster create a VPC, decide if all nodes should be in a public subnet.

Note

If you choose 1. Master in a public subnet and compute fleet in a private subnet, AWS ParallelCluster creates a NAT gateway that results in additional cost, even if you specify free tier resources.

```
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
```

If you don't create a new VPC, you must select an existing VPC.

If you choose to have AWS ParallelCluster create the VPC, make a note of the VPC ID so you can use the AWS CLI to delete it later.

```
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
```

After the VPC has been selected, decide whether to use existing subnets or create new ones.

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

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

When you have completed the preceding steps, a simple cluster launches into a VPC. The VPC uses an existing subnet that supports public IP addresses. The route table for the subnet is `0.0.0.0/0 => igw-xxxxxx`. Note the following conditions:

- The VPC must have DNS Resolution = yes and DNS Hostnames = yes.
- The VPC should also have DHCP options with the correct domain-name for the AWS Region. The default DHCP Option Set already specifies the required AmazonProvidedDNS. If specifying more than one domain name server, see [DHCP options sets](#) in the *Amazon VPC User Guide*. When using private subnets, use a NAT gateway or an internal proxy to enable web access for compute nodes. For more information, see [Network configurations](#).

When all settings contain valid values, you can launch the cluster by running the create command.

```
$ pcluster create mycluster
```

After the cluster reaches the "CREATE_COMPLETE" status, you can connect to it by using your normal SSH client settings. For more information about connecting to Amazon EC2 instances, see the [EC2 User Guide](#) in the *Amazon EC2 User Guide*.

To delete the cluster, run the following command.

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

To delete the network resources in the VPC, you can delete the CloudFormation networking stack. The stack name starts with "parallelclusternetworking-" and contains the creation time in "YYYYMMDDHHMMSS" format. You can list the stacks using the [list-stacks](#) command.

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

The stack can be deleted using the [delete-stack](#) command.

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

The VPC that [pcluster configure](#) creates for you is not created in the CloudFormation networking stack. You can delete that VPC manually in the console or by using the AWS CLI.

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

Best practices

Best practices: master instance type selection

Although the master node doesn't execute any job, its functions and its sizing are crucial to the overall performance of the cluster.

When choosing the instance type to use for your master node you want to evaluate the following items:

- **Cluster size:** the master node orchestrates the scaling logic of the cluster and is responsible of attaching new nodes to the scheduler. If you need to scale up and down the cluster of a considerable amount of nodes then you want to give the master node some extra compute capacity.
- **Shared file systems:** when using shared file systems to share artifacts between compute nodes and the master node take into account that the master is the node exposing the NFS server. For this reason you want to choose an instance type with enough network bandwidth and enough dedicated Amazon EBS bandwidth to handle your workflows.

Best practices: network performance

There are three hints that cover the whole range of possibilities to improve network communication.

- **Placement group:** a cluster placement group is a logical grouping of instances within a single Availability Zone. For more information on placement groups, see [placement groups](#) in the *Amazon EC2 User Guide*. You can configure the cluster to use your own placement group with `placement_group = your-placement-group-name` or let AWS ParallelCluster create a placement group with the "compute" strategy with `placement_group = DYNAMIC`. For more information, see [placement_group](#) for multiple queue mode and [placement_group](#) for single queue mode.
- **Enhanced networking:** consider to choose an instance type that supports Enhanced Networking. For more information, see [enhanced networking on Linux](#) in the *Amazon EC2 User Guide*.
- **Elastic Fabric Adapter:** To support high levels of scaleable inter-instance communication, consider choosing EFA network interfaces for your network. The EFA's custom-built operating system (OS) bypass hardware enhances inter-instance communications with the on-demand elasticity and flexibility of the AWS cloud. To configure a single Slurm cluster queue to use EFA, set `enable_efa = true`. For more information about using EFA with AWS ParallelCluster, see [Elastic Fabric Adapter](#) and [enable_efa](#). For more information about EFA, see [Elastic Fabric Adapter](#) in the *Amazon EC2 User Guide for Linux Instances*.
- **Instance bandwidth:** the bandwidth scales with instance size, please consider to choose the instance type which better suits your needs, see [Amazon EBS-optimized instances](#) and [Amazon EBS volume types](#) in the *Amazon EC2 User Guide*.

Best practices: budget alerts

To manage AWS ParallelCluster resource costs, we recommend that you use AWS Budgets actions to create a budget and defined budget threshold alerts for selected AWS resources. For more information, see [Configuring a budget action](#) in the *AWS Budgets User Guide*. You can also use Amazon CloudWatch to create a billing alarm. For more information, see [Creating a billing alarm to monitor your estimated AWS charges](#).

Best practices: moving a cluster to a new AWS ParallelCluster minor or patch version

Currently each AWS ParallelCluster minor version is self-contained along with its `pcluster` CLI. To move a cluster to a new minor or patch version, you must re-create the cluster using the new version's CLI.

To optimize the process of moving a cluster to a new minor version or to save your shared storage data for other reasons, we recommend that you use the following best practices.

- Save personal data in external volumes, such as Amazon EFS and FSx for Lustre. By doing this, you can easily move the data from one cluster to another.
- Create shared storage systems of the types listed below using the AWS CLI or AWS Management Console:
 - [\[ebs\] section](#)
 - [\[efs\] section](#)
 - [\[fsx\] section](#)

Add them to the new cluster configuration as existing file systems. This way, they are preserved when you delete the cluster and can be attached to a new cluster. Shared storage systems generally incur charges whether they are attached or detached from a cluster.

We recommend that you use Amazon EFS, or Amazon FSx for Lustre file systems because they can be attached to multiple clusters at the same time and you can attach them to the new cluster before deleting the old cluster. For more information, see [Mounting Amazon EFS file systems](#) in the *Amazon EFS User Guide* and [Accessing FSx for Lustre file systems](#) in the *Amazon FSx for Lustre User Guide*.

- Use [custom bootstrap actions](#) to customize your instances rather than a custom AMI. This optimizes the creation process because a new custom AMI doesn't need to be created for each new version.
- Recommended sequence.
 1. Update the cluster configuration to use existing file systems definitions.
 2. Verify the `pcluster` version and update it if needed.
 3. Create and test the new cluster.
 - Make sure your data is available in the new cluster.
 - Make sure your application works in the new cluster.
 4. If your new cluster is fully tested and operational and you're sure you aren't going to use the old cluster, delete it.

Moving from CfnCluster to AWS ParallelCluster

AWS ParallelCluster is an enhanced version of CfnCluster.

If you currently use CfnCluster, we recommend that you use AWS ParallelCluster instead and create new clusters with it. Even though you can continue to use CfnCluster, it's no longer being developed, and no new features or functionality will be added.

The main differences between CfnCluster and AWS ParallelCluster are described in the following sections.

AWS ParallelCluster CLI manages a different set of clusters

Clusters created with the `cfncluster` CLI can't be managed with the `pcluster` CLI. The following commands don't work on clusters created by CfnCluster:

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

To manage clusters that you created with CfnCluster, you must use the `cfncluster` CLI.

If you need a CfnCluster package to manage your old clusters, we recommend that you install and use it from a [Python virtual environment](#).

AWS ParallelCluster and CfnCluster use different IAM custom policies

Custom IAM policies that were previously used for CfnCluster cluster creation can't be used with AWS ParallelCluster. If you require custom policies for AWS ParallelCluster, you must create new ones. See the AWS ParallelCluster guide.

AWS ParallelCluster and CfnCluster use different configuration files

The AWS ParallelCluster configuration file resides in the `~/.parallelcluster` folder. The CfnCluster configuration file resides in the `~/.cfncluster` folder.

If you want to use an existing CfnCluster configuration file with AWS ParallelCluster, then you must complete the following actions:

1. Move the configuration file from `~/.cfncluster/config` to `~/.parallelcluster/config`.
2. If you use the [extra_json](#) configuration parameter, change it as shown.

CfnCluster setting:

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

AWS ParallelCluster setting:

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

In AWS ParallelCluster, ganglia is disabled by default

In AWS ParallelCluster, ganglia is disabled by default. To enable ganglia, complete these steps:

1. Set the [extra_json](#) parameter as shown:

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

2. Change the head security group to allow connections to port 80.

The `parallelcluster-<CLUSTER_NAME>-MasterSecurityGroup-<xxx>` security group must be modified by adding a new security group rule to allow Inbound connection to port 80 from your Public IP. For more information, see [Adding rules to a security group](#) in the *Amazon EC2 User Guide*.

Supported Regions

AWS ParallelCluster version 2.x is available in the following AWS Regions:

Region Name	Region
US East (Ohio)	us-east-2
US East (N. Virginia)	us-east-1
US West (N. California)	us-west-1
US West (Oregon)	us-west-2
Africa (Cape Town)	af-south-1
Asia Pacific (Hong Kong)	ap-east-1
Asia Pacific (Mumbai)	ap-south-1
Asia Pacific (Seoul)	ap-northeast-2
Asia Pacific (Singapore)	ap-southeast-1
Asia Pacific (Sydney)	ap-southeast-2
Asia Pacific (Tokyo)	ap-northeast-1
Canada (Central)	ca-central-1
China (Beijing)	cn-north-1
China (Ningxia)	cn-northwest-1
Europe (Frankfurt)	eu-central-1
Europe (Ireland)	eu-west-1
Europe (London)	eu-west-2
Europe (Milan)	eu-south-1

Region Name	Region
Europe (Paris)	eu-west-3
Europe (Stockholm)	eu-north-1
Middle East (Bahrain)	me-south-1
South America (São Paulo)	sa-east-1
AWS GovCloud (US-East)	us-gov-east-1
AWS GovCloud (US-West)	us-gov-west-1

Using AWS ParallelCluster

Topics

- [Network configurations](#)
- [Custom Bootstrap Actions](#)
- [Working with Amazon S3](#)
- [Working with Spot Instances](#)
- [AWS Identity and Access Management roles in AWS ParallelCluster](#)
- [Schedulers supported by AWS ParallelCluster](#)
- [AWS ParallelCluster resources and tagging](#)
- [Amazon CloudWatch dashboard](#)
- [Integration with Amazon CloudWatch Logs](#)
- [Elastic Fabric Adapter](#)
- [Intel Select Solutions](#)
- [Enable Intel MPI](#)
- [Intel HPC Platform Specification](#)
- [Arm Performance Libraries](#)
- [Connect to the head node through Amazon DCV](#)
- [Using pcluster update](#)
- [AMI patching and EC2 instance replacement](#)

Network configurations

AWS ParallelCluster uses Amazon Virtual Private Cloud (VPC) for networking. VPC provides a flexible and configurable networking platform where you can deploy clusters.

The VPC must have DNS Resolution = yes, DNS Hostnames = yes and DHCP options with the correct domain name for the Region. The default DHCP Option Set already specifies the required *AmazonProvidedDNS*. If specifying more than one domain name server, see [DHCP options sets](#) in the *Amazon VPC User Guide*.

AWS ParallelCluster supports the following high-level configurations:

- One subnet for both head and compute nodes.
- Two subnets, with the head node in one public subnet, and compute nodes in a private subnet.
The subnets can be either new or existing ones.

All of these configurations can operate with or without public IP addressing. AWS ParallelCluster can also be deployed to use an HTTP proxy for all AWS requests. The combinations of these configurations result in many deployment scenarios. For example, you can configure a single public subnet with all access over the internet. Or, you can configure a fully private network using AWS Direct Connect and HTTP proxy for all traffic.

See the following architecture diagrams for illustrations of some of these scenarios:

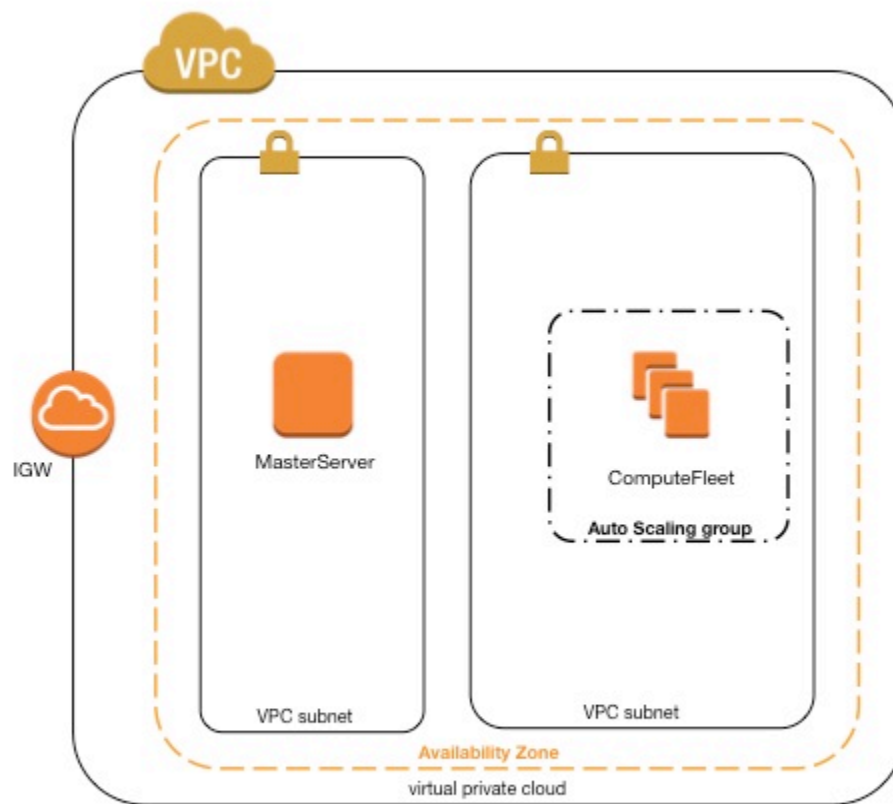
AWS ParallelCluster in a single public subnet

The configuration for this architecture requires the following settings:

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

The [use_public_ips](#) setting cannot be set to false, because the internet gateway requires that all instances have a globally unique IP address. For more information, see [Enabling internet access](#) in *Amazon VPC User Guide*.

AWS ParallelCluster using two subnets



The configuration to create a new private subnet for compute instances requires the following settings:

Note that all values are only provided as examples.

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

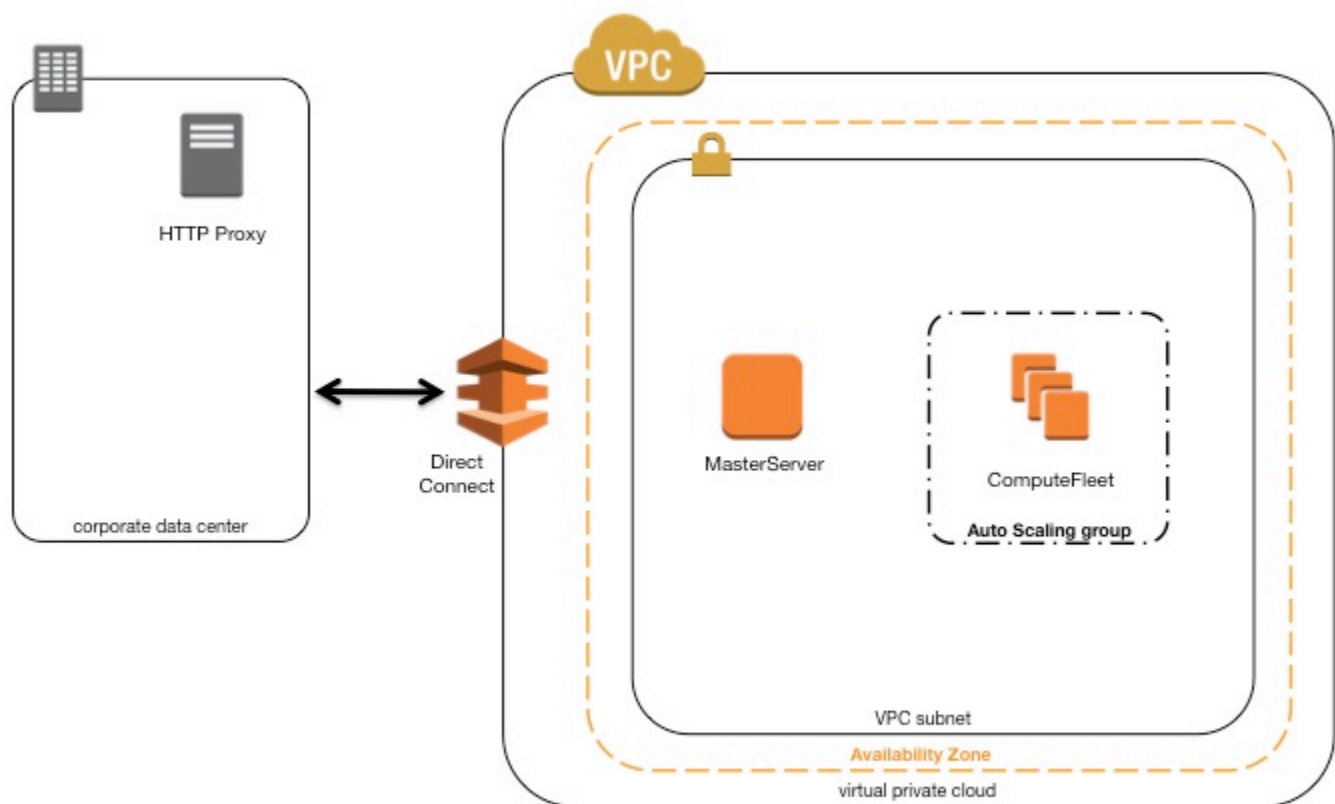
The configuration to use an existing private network requires the following settings:

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

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

Both of these configurations require a [NAT gateway](#) or an internal proxy to enable web access for compute instances.

AWS ParallelCluster in a single private subnet connected using AWS Direct Connect



The configuration for this architecture requires the following settings:

```
[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
```

When `use_public_ips` is set to `false`, the VPC must be correctly set up to use the Proxy for all traffic. Web access is required for both head and compute nodes.

AWS ParallelCluster with awsbatch scheduler

When you use `awsbatch` as the scheduler type, AWS ParallelCluster creates an AWS Batch managed compute environment. The AWS Batch environment takes care of managing Amazon Elastic Container Service (Amazon ECS) container instances, which are launched in the `compute_subnet`. For AWS Batch to function correctly, Amazon ECS container instances need external network access to communicate with the Amazon ECS service endpoint. This translates into the following scenarios:

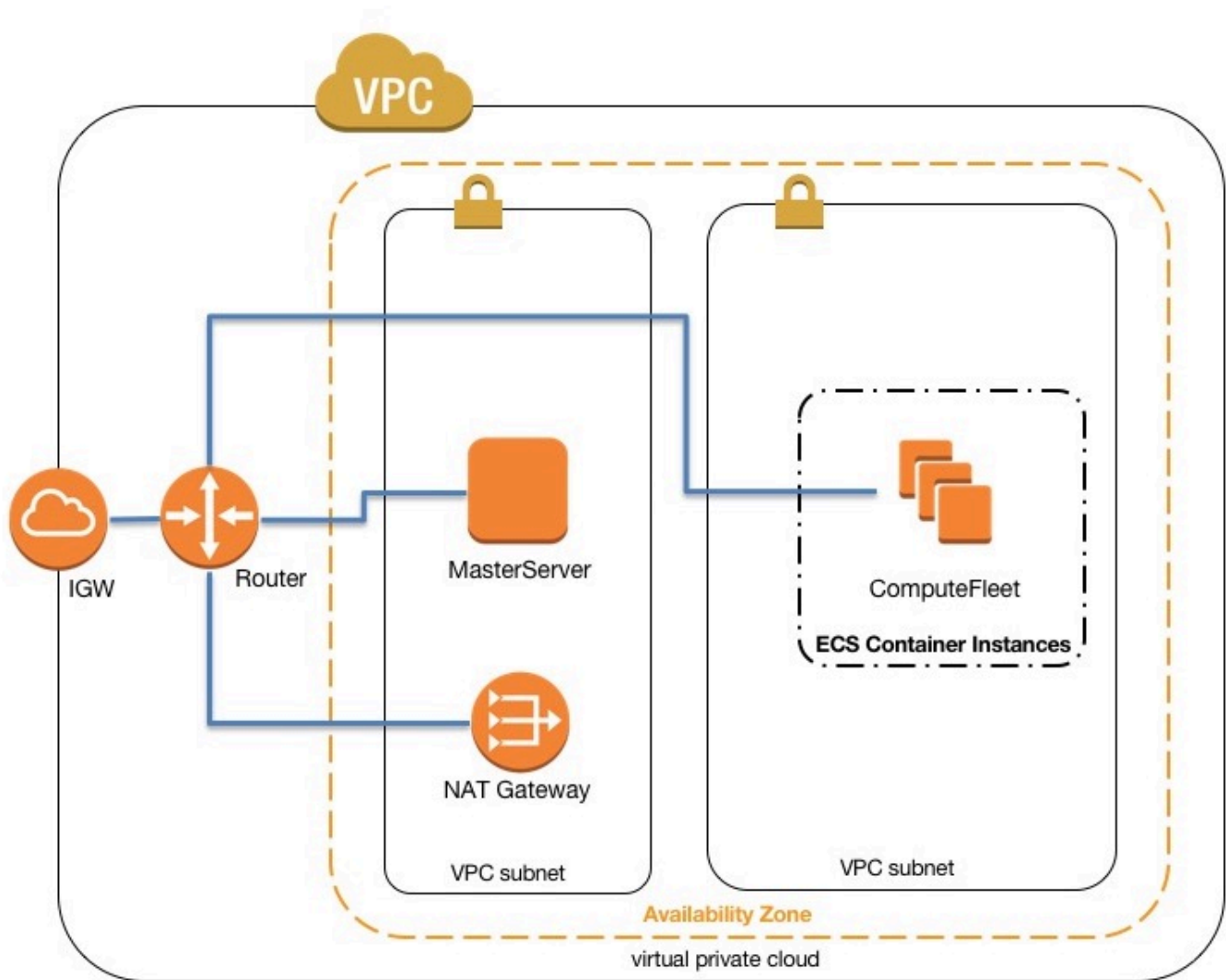
- The `compute_subnet` uses a NAT gateway to access the internet. (We recommended this approach.)
- Instances launched in the `compute_subnet` have public IP addresses and can reach the internet through an Internet Gateway.

Additionally, if you're interested in multi-node parallel jobs (from the [AWS Batch docs](#)):

AWS Batch multi-node parallel jobs use the Amazon ECS `awsvpc` network mode, which gives your multi-node parallel job containers the same networking properties as Amazon EC2 instances. Each multi-node parallel job container gets its own elastic network interface, a primary private IP address, and an internal DNS hostname. The network interface is created in the same Amazon VPC subnet as its host compute resource. Any security groups that are applied to your compute resources are also applied to it.

When using Amazon ECS Task Networking, the `awsvpc` network mode doesn't provide elastic network interfaces with public IP addresses for tasks that use the Amazon EC2 launch type. To access the internet, tasks that use the Amazon EC2 launch type must be launched in a private subnet that's configured to use a NAT gateway.

You must configure a NAT gateway in order to enable the cluster to run multi-node parallel jobs.



For more information, see the following topics:

- [AWS Batch managed compute environments](#)
- [AWS Batch multi-node parallel jobs](#)
- [Amazon ECS task networking with the awsvpc network mode](#)

Custom Bootstrap Actions

AWS ParallelCluster can run arbitrary code either before (pre-install) or after (post-install) the main bootstrap action when the cluster is created. In most cases, this code is stored in Amazon Simple Storage Service (Amazon S3) and accessed through an HTTPS connection. The code is run as root

and can be in any script language that's supported by the cluster OS. Often the code is in *Bash* or *Python*.

Pre-install actions are called before any cluster deployment bootstrap action is started, such as configuring NAT, Amazon Elastic Block Store (Amazon EBS) or the scheduler. Some pre-install actions include modifying storage, adding extra users, and adding packages.

Post-install actions are called after the cluster bootstrap processes are complete. Post-install actions are the last actions to occur before an instance is considered fully configured and complete. Some post-install actions include changing scheduler settings, modifying storage, and modifying packages.

You can pass argument to scripts by specifying them during configuration. For this, you pass them double-quoted to the pre-install or post-install actions.

If a pre-install or post-install action fails, the instance bootstrap also fails. Success is signaled with an exit code of zero (0). Any other exit code indicates the instance bootstrap failed.

You can differentiate between running head and compute nodes. Source the `/etc/parallelcluster/cfnconfig` file and evaluate the `cfn_node_type` environment variable that have a value of "MasterServer" and "ComputeFleet" for the head and compute nodes, respectively.

```
#!/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
```

Configuration

The following configuration settings are used to define pre-install and post-install actions and arguments.

```
# 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
```

Arguments

The first two arguments — \$0 and \$1 — are reserved for the script name and url.

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

Example

The following steps create a simple post-install script that installs the R packages in a cluster.

1. Create a script.

```
#!/bin/bash

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

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

2. Upload the script with the correct permissions to Amazon S3. If public read permissions aren't appropriate for you, use either [s3_read_resource](#) or [s3_read_write_resource](#) parameters to grant access. For more information, see [Working with Amazon S3](#).

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

Important

If the script was edited on Windows, line endings must be changed from CRLF to LF before the script is uploaded to Amazon S3.

3. Update the AWS ParallelCluster configuration to include the new post-install action.

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

If the bucket doesn't have public-read permission, use s3 as the URL protocol.

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

4. Launch the cluster.

```
$ pcluster create mycluster
```

5. Verify the output.

```
$ 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
```

Working with Amazon S3

To provide cluster resources permission to access to Amazon S3 buckets, specify the bucket ARNs in the [s3_read_resource](#) and [s3_read_write_resource](#) parameters in the AWS ParallelCluster configuration. For more information about controlling access with AWS ParallelCluster, see [AWS Identity and Access Management roles in 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/*
```

Both parameters accept either * or a valid Amazon S3 ARN. For information about specifying Amazon S3 ARNs, see [Amazon S3 ARN format](#) in the *AWS General Reference*.

Examples

The following example gives you read access to any object in the Amazon S3 bucket *my_corporate_bucket*.

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

This following example gives you read access to the bucket, but does **not** let you read items from the bucket.

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

This last example gives you read access to the bucket and to the items stored in the bucket.

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

Working with Spot Instances

AWS ParallelCluster uses Spot Instances if the cluster configuration has set `cluster_type = spot`. Spot Instances are more cost effective than On-Demand Instances, but they might be interrupted. The effect of the interruption varies depending on the specific scheduler used. It might help to take advantage of *Spot Instance interruption notices*, which provide a two minute warning before Amazon EC2 must stop or terminate your Spot Instance. For more information, see [Spot Instance interruptions](#) in *Amazon EC2 User Guide*. The following sections describe three scenarios in which Spot Instances can be interrupted.

Note

Using Spot Instances requires that the `AWSServiceRoleForEC2Spot` service-linked role exist in your account. To create this role in your account using the AWS CLI, run the following command:

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

For more information, see [Service-linked role for Spot Instance requests](#) in the *Amazon EC2 User Guide*.

Scenario 1: Spot Instance with no running jobs is interrupted

When this interruption occurs, AWS ParallelCluster tries to replace the instance if the scheduler queue has pending jobs that require additional instances, or if the number of active instances is lower than the `initial_queue_size` setting. If AWS ParallelCluster can't provision new instances, then a request for new instances is periodically repeated.

Scenario 2: Spot Instance running single node jobs is interrupted

The behavior of this interruption depends on the scheduler being used.

Slurm

The job fails with a state code of `NODE_FAIL`, and the job is requeued (unless `--no-requeue` is specified when the job is submitted). If the node is a static node, it's replaced. If the node is a dynamic node, the node is terminated and reset. For more information about `sbatch`, including the `--no-requeue` parameter, see [sbatch](#) in the *Slurm documentation*.

Note

This behavior changed in AWS ParallelCluster version 2.9.0. Earlier versions terminated the job with a state code of `NODE_FAIL` and the node was removed from the scheduler queue.

SGE

Note

This only applies to AWS ParallelCluster versions up to and including version 2.11.4. Starting with version 2.11.5, AWS ParallelCluster doesn't support the use of SGE or Torque schedulers.

The job is terminated. If the job has enabled the rerun flag (using either `qsub -r yes` or `qalter -r yes`) or the queue has the `rerun` configuration set to `TRUE`, then the job is rescheduled. The compute instance is removed from the scheduler queue. This behavior comes from these SGE configuration parameters:

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

Torque

Note

This only applies to AWS ParallelCluster versions up to and including version 2.11.4. Starting with version 2.11.5, AWS ParallelCluster doesn't support the use of SGE or Torque schedulers.

The job is removed from the system and the node is removed from the scheduler. The job isn't rerun. If multiple jobs are running on the instance when it is interrupted, Torque might time out during node removal. An error might display in the [sqswatcher](#) log file. This doesn't affect scaling logic, and a proper cleanup is performed by subsequent retries.

Scenario 3: Spot Instance running multi-node jobs is interrupted

The behavior of this interruption depends on the scheduler being used.

Slurm

The job fails with a state code of `NODE_FAIL`, and the job is requeued (unless `--no-requeue` was specified when the job was submitted). If the node is a static node, it's replaced. If the node is a dynamic node, the node is terminated and reset. Other nodes that were running the terminated jobs might be allocated to other pending jobs, or scaled down after the configured [scaledown_idletime](#) time has passed.

Note

This behavior changed in AWS ParallelCluster version 2.9.0. Earlier versions terminated the job with a state code of `NODE_FAIL` and the node was removed from the scheduler queue. Other nodes that were running the terminated jobs might be scaled down after the configured [scaledown_idletime](#) time has passed.

SGE

Note

This only applies to AWS ParallelCluster versions up to and including version 2.11.4. Starting with version 2.11.5, AWS ParallelCluster doesn't support the use of SGE or Torque schedulers.

The job isn't terminated and continues to run on the remaining nodes. The compute node is removed from the scheduler queue, but will appear in the hosts list as an orphaned and unavailable node.

The user must delete the job when this occurs (`qdel <jobid>`). The node still displays in the hosts list (`qhost`), although this doesn't affect AWS ParallelCluster. To remove the host from the list, run the following command after replacing the instance.

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

Torque

Note

This only applies to AWS ParallelCluster versions up to and including version 2.11.4. Starting with version 2.11.5, AWS ParallelCluster doesn't support the use of SGE or Torque schedulers.

The job is removed from the system and the node is removed from the scheduler. The job isn't rerun. If multiple jobs are running on the instance when it is interrupted, Torque might time out during node removal. An error might display in the [sqswatcher](#) log file. This doesn't affect scaling logic, and a proper cleanup is performed by subsequent retries.

For more information about Spot Instances, see [Spot Instances](#) in the *Amazon EC2 User Guide*.

AWS Identity and Access Management roles in AWS ParallelCluster

AWS ParallelCluster uses AWS Identity and Access Management (IAM) roles for Amazon EC2 to enable instances to access AWS services for the deployment and operation of a cluster. By default, the IAM role for Amazon EC2 is created when the cluster is created. This means that the user that creates the cluster must have the appropriate level of permissions, as described in the following sections.

AWS ParallelCluster uses multiple AWS services to deploy and operate a cluster. See the complete list in the [AWS Services used in AWS ParallelCluster](#) section.

You can track changes to the example policies in [AWS ParallelCluster documentation on GitHub](#).

Topics

- [Default settings for cluster creation](#)
- [Using an existing IAM role for Amazon EC2](#)
- [AWS ParallelCluster example instance and user policies](#)

Default settings for cluster creation

When you use the default settings for cluster creation, a default IAM role for Amazon EC2 is created by the cluster. The user that creates the cluster must have the right level of permissions to create all of the resources required to launch the cluster. This includes creating an IAM role for Amazon EC2. Typically, the user must have the permissions of an *AdministratorAccess* managed policy when using the default settings. For information about managed policies, see [AWS managed policies](#) in the *IAM User Guide*.

Using an existing IAM role for Amazon EC2

In place of the default settings, you can use an existing [ec2_iam_role](#) when creating a cluster, but you must define the IAM policy and role before attempting to launch the cluster. Typically, you choose an existing IAM role for Amazon EC2 to minimize the permissions that are granted to users as they launch clusters. The [AWS ParallelCluster example instance and user policies](#) include the minimum permissions required by AWS ParallelCluster and its features. You must create both policies and roles as individual policies in IAM and then attach the roles and policies to the appropriate resources. Some of the role policies might get large and cause quota errors. For more information, see [Troubleshooting IAM policy size issues](#). In the policies, replace *<REGION>*, *<AWS ACCOUNT ID>*, and similar strings with the appropriate values.

If your intent is to add extra policies to the default settings for cluster nodes, we recommend that you pass the additional custom IAM policies with the [additional_iam_policies](#) setting instead of using the [ec2_iam_role](#) settings.

AWS ParallelCluster example instance and user policies

The following example policies include Amazon Resource Names (ARNs) for the resources. If you're working in the AWS GovCloud (US) or AWS China partitions, the ARNs must be changed. Specifically, they must be changed from "arn:aws" to "arn:aws-us-gov" for the AWS GovCloud (US) partition or "arn:aws-cn" for the AWS China partition. For more information, see [Amazon Resource Names \(ARNs\) in AWS GovCloud \(US\) Regions](#) in the *AWS GovCloud (US) User Guide* and [ARNs for AWS services in China](#) in *Getting Started with AWS services in China*.

These policies include the minimum permissions currently required by AWS ParallelCluster, its features, and resources. Some of the role policies might get large and cause quota errors. For more information, see [Troubleshooting IAM policy size issues](#).

Topics

- [ParallelClusterInstancePolicy using SGE, Slurm, or Torque](#)
- [ParallelClusterInstancePolicy using awsbatch](#)
- [ParallelClusterUserPolicy using Slurm](#)
- [ParallelClusterUserPolicy using SGE or Torque](#)
- [ParallelClusterUserPolicy using awsbatch](#)
- [ParallelClusterLambdaPolicy using SGE, Slurm, or Torque](#)
- [ParallelClusterLambdaPolicy using awsbatch](#)
- [ParallelClusterUserPolicy for users](#)

ParallelClusterInstancePolicy using SGE, Slurm, or Torque

Note

Starting with version 2.11.5, AWS ParallelCluster doesn't support the use of SGE or Torque schedulers. You can continue using them in versions up to and including 2.11.4, but they aren't eligible for future updates or troubleshooting support from the AWS service and AWS Support teams.

Topics

- [ParallelClusterInstancePolicy using Slurm](#)
- [ParallelClusterInstancePolicy using SGE or Torque](#)

ParallelClusterInstancePolicy using Slurm

The following example sets the ParallelClusterInstancePolicy using Slurm as the scheduler.

JSON

```
{
  "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:us-east-1:111122223333:subnet/<COMPUTE SUBNET ID>",
        "arn:aws:ec2:us-east-1:111122223333:network-interface/*",
        "arn:aws:ec2:us-east-1:111122223333:instance/*",
        "arn:aws:ec2:us-east-1:111122223333:volume/*",
        "arn:aws:ec2:us-east-1::image/<IMAGE ID>",
        "arn:aws:ec2:us-east-1:111122223333:key-pair/<KEY NAME>",
        "arn:aws:ec2:us-east-1:111122223333:security-group/*",
        "arn:aws:ec2:us-east-1:111122223333:launch-template/*",
        "arn:aws:ec2:us-east-1:111122223333: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:us-east-1:111122223333:stack/parallelcluster-*/*"
    ],
    "Effect": "Allow",
    "Sid": "CloudFormation"
},
{
    "Action": [
        "dynamodb:PutItem",
        "dynamodb:Query",
        "dynamodb:GetItem",
        "dynamodb:BatchWriteItem",
        "dynamodb>DeleteItem",
        "dynamodb:DescribeTable"
    ],
    "Resource": [
        "arn:aws:dynamodb:us-east-1:111122223333:table/parallelcluster-*"
    ],
    "Effect": "Allow",
    "Sid": "DynamoDBTable"
},
{
    "Action": [
        "s3:GetObject"
    ],
    "Resource": [
        "arn:aws:s3:::us-east-1-aws-parallelcluster/*"
    ],
    "Effect": "Allow",
    "Sid": "S3GetObject"
}

```

```

    },
    {
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow",
      "Sid": "IAMPassRole",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": [
            "ec2.amazonaws.com"
          ]
        }
      }
    },
    {
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::dcv-license.us-east-1/*"
      ],
      "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"
  }
]
}

```

ParallelClusterInstancePolicy using SGE or Torque

The following example sets the ParallelClusterInstancePolicy using SGE or Torque as the scheduler.

Note

This policy only applies to AWS ParallelCluster versions up to and including version 2.11.4. Starting with version 2.11.5, AWS ParallelCluster doesn't support the use of SGE or Torque schedulers.

JSON

```
{
  "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:us-east-1:111122223333:subnet/<COMPUTE SUBNET ID>",
        "arn:aws:ec2:us-east-1:111122223333:network-interface/*",
        "arn:aws:ec2:us-east-1:111122223333:instance/*",
        "arn:aws:ec2:us-east-1:111122223333:volume/*",
        "arn:aws:ec2:us-east-1::image/<IMAGE ID>",
        "arn:aws:ec2:us-east-1:111122223333:key-pair/<KEY NAME>",
        "arn:aws:ec2:us-east-1:111122223333:security-group/*",
        "arn:aws:ec2:us-east-1:111122223333:launch-template/*",
        "arn:aws:ec2:us-east-1:111122223333: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:us-east-1:111122223333: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:us-east-1:111122223333:stack/
parallelcluster-*/*"
    ],
    "Effect": "Allow",
    "Sid": "CloudFormation"
},
{
    "Action": [
        "dynamodb:PutItem",
        "dynamodb:Query",
        "dynamodb:GetItem",
        "dynamodb:BatchWriteItem",
        "dynamodb>DeleteItem",
        "dynamodb:DescribeTable"
    ],
    "Resource": [
        "arn:aws:dynamodb:us-east-1:111122223333:table/parallelcluster-*"
    ],
    "Effect": "Allow",
    "Sid": "DynamoDBTable"
},
{
    "Action": [
        "s3:GetObject"
    ],
    "Resource": [
        "arn:aws:s3:::us-east-1-aws-parallelcluster/*"
    ],
    "Effect": "Allow",
    "Sid": "S3GetObject"
},
{
    "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.us-east-1/*"
    ],
    "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"
    }
  ]
}

```

ParallelClusterInstancePolicy using awsbatch

The following example sets the `ParallelClusterInstancePolicy` using `awsbatch` as the scheduler. You must include the same policies that are assigned to the `BatchUserRole` that is defined in the AWS Batch CloudFormation nested stack. The `BatchUserRole` ARN is provided as a stack output. In this example, "`<RESOURCES S3 BUCKET>`" is the value of the `cluster_resource_bucket` setting; if `cluster_resource_bucket` is not specified then "`<RESOURCES S3 BUCKET>`" is "parallelcluster-*". The following example is an overview of the required permissions:

JSON

```

{
  "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:us-east-1:111122223333:job-
definition/<AWS_BATCH_STACK - JOB_DEFINITION_SERIAL_NAME>:1",
    "arn:aws:batch:us-east-1:111122223333:job-
definition/<AWS_BATCH_STACK - JOB_DEFINITION_MNP_NAME>*",
    "arn:aws:batch:us-east-1:111122223333:job-queue/<AWS_BATCH_STACK
- JOB_QUEUE_NAME>",
    "arn:aws:cloudformation:us-east-1:111122223333:stack/<STACK
NAME>/*",
    "arn:aws:s3:::amzn-s3-demo-bucket/batch/*",
    "arn:aws:iam::111122223333:role/<AWS_BATCH_STACK - JOB_ROLE>",
    "arn:aws:ecs:us-east-1:111122223333:cluster/<ECS COMPUTE
ENVIRONMENT>",
    "arn:aws:ecs:us-east-1:111122223333:container-instance/*",
    "arn:aws:logs:us-east-1:111122223333:log-group:/aws/batch/
job:log-stream:*"
  ],
  "Effect": "Allow"
},
{
  "Action": [
    "s3:List*"
  ]
}

```

```

    ],
    "Resource": [
        "arn:aws:s3:::amzn-s3-demo-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"
  }
]
}

```

ParallelClusterUserPolicy using Slurm

The following example sets the ParallelClusterUserPolicy, using Slurm as the scheduler. In this example, "**<RESOURCES S3 BUCKET>**" is the value of the [cluster_resource_bucket](#) setting; if [cluster_resource_bucket](#) is not specified then "**<RESOURCES S3 BUCKET>**" is "parallelcluster-*".

Note

If you use a custom role, [ec2_iam_role](#) = **<role_name>**, you must change the IAM resource to include the name of that role from:

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

To:

"Resource": "arn:aws:iam::**<AWS ACCOUNT ID>**:role/**<role_name>**"

JSON

```
{
  "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:::amzn-s3-demo-bucket"
    ],
    "Effect": "Allow",
    "Sid": "S3ResourcesBucket"
},
{
    "Action": [
        "s3:Get*",
        "s3:List*"
    ],
    "Resource": [
        "arn:aws:s3:::us-east-1-aws-parallelcluster*"
    ],
    "Effect": "Allow",
    "Sid": "S3ParallelClusterReadOnly"
},
{
    "Action": [

```

```

        "s3:DeleteBucket",
        "s3:DeleteObject",
        "s3:DeleteObjectVersion"
    ],
    "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket"
    ],
    "Effect": "Allow",
    "Sid": "S3Delete"
},
{
    "Action": [
        "iam:PassRole",
        "iam:CreateRole",
        "iam>DeleteRole",
        "iam:GetRole",
        "iam:TagRole",
        "iam:SimulatePrincipalPolicy"
    ],
    "Resource": [
        "arn:aws:iam::111122223333:role/<PARALLELCLUSTER EC2 ROLE NAME>",
        "arn:aws:iam::111122223333: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::111122223333:role/aws-service-role/*",
    "Effect": "Allow",
    "Sid": "IAMServiceLinkedRole"
},
{
    "Action": [

```

```

        "iam:CreateInstanceProfile",
        "iam>DeleteInstanceProfile"
    ],
    "Resource": "arn:aws:iam::111122223333: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:us-east-1:111122223333:function:parallelcluster-
*",
            "arn:aws:lambda:us-east-1:111122223333:function:pcluster-*"
        ],
        "Effect": "Allow",

```

```

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

```

ParallelClusterUserPolicy using SGE or Torque

Note

This section only applies to AWS ParallelCluster versions up to and including version 2.11.4. Starting with version 2.11.5, AWS ParallelCluster doesn't support the use of SGE or Torque schedulers.

The following example sets the `ParallelClusterUserPolicy`, using SGE or Torque as the scheduler. In this example, "`<RESOURCES S3 BUCKET>`" is the value of the `cluster_resource_bucket` setting; if `cluster_resource_bucket` is not specified then "`<RESOURCES S3 BUCKET>`" is "parallelcluster-*".

Note

If you use a custom role, `ec2_iam_role = <role_name>`, you must change the IAM resource to include the name of that role from:

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

To:

"Resource": "arn:aws:iam::<AWS ACCOUNT ID>:role/<role_name>"

JSON

```
{
  "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:::amzn-s3-demo-bucket"
    ],
    "Effect": "Allow",
    "Sid": "S3ResourcesBucket"
},
{
    "Action": [
        "s3:Get*",
        "s3:List*"
    ],
    "Resource": [
        "arn:aws:s3:::us-east-1-aws-parallelcluster*"
    ],
    "Effect": "Allow",
    "Sid": "S3ParallelClusterReadOnly"
},
{
    "Action": [
        "s3:DeleteBucket",
        "s3:DeleteObject",
        "s3:DeleteObjectVersion"
    ]
}

```

```

    ],
    "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket"
    ],
    "Effect": "Allow",
    "Sid": "S3Delete"
},
{
    "Action": [
        "iam:PassRole",
        "iam:CreateRole",
        "iam>DeleteRole",
        "iam:GetRole",
        "iam:TagRole",
        "iam:SimulatePrincipalPolicy"
    ],
    "Resource": [
        "arn:aws:iam::111122223333:role/<PARALLELCLUSTER_EC2_ROLE_NAME>",
        "arn:aws:iam::111122223333: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::111122223333:role/aws-service-role/*",
    "Effect": "Allow",
    "Sid": "IAMServiceLinkedRole"
},
{
    "Action": [
        "iam:CreateInstanceProfile",
        "iam>DeleteInstanceProfile"
    ],

```

```

    "Resource": "arn:aws:iam::111122223333: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:us-east-1:111122223333:function:parallelcluster-
*",
        "arn:aws:lambda:us-east-1:111122223333:function:pcluster-*"
      ],
      "Effect": "Allow",
      "Sid": "Lambda"
    },
  ],
  {

```

```

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

```

ParallelClusterUserPolicy using awsbatch

The following example sets the ParallelClusterUserPolicy using awsbatch as the scheduler. In this example, "**<RESOURCES S3 BUCKET>**" is the value of the [cluster_resource_bucket](#) setting; if [cluster_resource_bucket](#) is not specified then "**<RESOURCES S3 BUCKET>**" is "parallelcluster-*".

JSON

```

{
  "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:us-east-1:111122223333: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:us-east-1:111122223333: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:us-east-1:111122223333: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::111122223333:role/parallelcluster-*",
        "arn:aws:iam::111122223333:role/<PARALLELCLUSTER_EC2_ROLE_NAME>"
    ],
    "Effect": "Allow",
    "Sid": "IAMRole"
},
{
    "Action": [
        "iam:CreateInstanceProfile",
        "iam>DeleteInstanceProfile",
        "iam:GetInstanceProfile",
        "iam:PassRole"
    ],

```

```

    "Resource": "arn:aws:iam::111122223333: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:::amzn-s3-demo-bucket"
    ],
    "Effect": "Allow",
    "Sid": "S3ResourcesBucket"
  },
  {
    "Action": [
      "s3:Get*",
      "s3:List*"
    ],
    "Resource": [
      "arn:aws:s3:::us-east-1-aws-parallelcluster/*"
    ],
    "Effect": "Allow",
    "Sid": "S3ParallelClusterReadOnly"
  },
  {
    "Action": [
      "s3:DeleteBucket",
      "s3:DeleteObject",

```

```

        "s3:DeleteObjectVersion"
    ],
    "Resource": [
        "arn:aws:s3:::amzn-s3-demo-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:us-east-1:111122223333:function:parallelcluster-
*",
        "arn:aws:lambda:us-east-1:111122223333:function:pcluster-*"
    ],
    "Effect": "Allow",
    "Sid": "Lambda"
},
{
    "Action": [
        "logs:*"
    ],
    "Resource": "arn:aws:logs:us-east-1:111122223333:*",
    "Effect": "Allow",
    "Sid": "Logs"
},
{
    "Action": [
        "codebuild:*"
    ],
    "Resource": "arn:aws:codebuild:us-east-1:111122223333: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": "*"
    }
]
}

```

ParallelClusterLambdaPolicy using SGE, Slurm, or Torque

The following example sets the ParallelClusterLambdaPolicy, using SGE, Slurm, or Torque as the scheduler.

Note

Starting with version 2.11.5, AWS ParallelCluster doesn't support the use of SGE or Torque schedulers.

JSON

```
{
```

```

"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:us-east-1:111122223333:table/parallelcluster-
**",
    "Effect": "Allow",
    "Sid": "DynamoDBTable"
},
{
    "Action": [
        "route53:ListResourceRecordSets",
        "route53:ChangeResourceRecordSets"
    ],
    "Resource": [
        "arn:aws:route53::hostedzone/*"
    ],
    "Effect": "Allow",
    "Sid": "Route53DeletePolicy"
}
]
}

```

ParallelClusterLambdaPolicy using awsbatch

The following example sets the ParallelClusterLambdaPolicy using awsbatch as the scheduler.

JSON

```

{
    "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"
}
]
}

```

ParallelClusterUserPolicy for users

The following example sets the `ParallelClusterUserPolicy` for users that don't need to create or update clusters. The following commands are supported.

- [pcluster dcv](#)
- [pcluster instances](#)
- [pcluster list](#)
- [pcluster ssh](#)

- [pcluster start](#)
- [pcluster status](#)
- [pcluster stop](#)
- [pcluster version](#)

JSON

```
{
  "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:us-east-1:111122223333:autoScalingGroup*:autoScalingGroupName/parallelcluster-*",
        "arn:aws:batch:us-east-1:111122223333:compute-environment/*",
        "arn:aws:cloudformation:us-east-1:111122223333:stack/<CLUSTERNAME>/*",
        "arn:aws:dynamodb:us-east-1:111122223333:table/<CLUSTERNAME>"
      ]
    },
    {
      "Sid": "Describe",
      "Action": [
        "cloudformation:DescribeStacks",
        "ec2:DescribeInstances",
        "ec2:DescribeInstanceStatus"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

```
]
}
```

Schedulers supported by AWS ParallelCluster

AWS ParallelCluster supports several schedulers, set using the [scheduler](#) setting.

Note

Starting with version 2.11.5, AWS ParallelCluster doesn't support the use of SGE or Torque schedulers. You can continue using them in versions up to and including 2.11.4, but they aren't eligible for future updates or troubleshooting support from the AWS service and AWS Support teams.

Topics

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

Son of Grid Engine (sge)

Note

Starting with version 2.11.5, AWS ParallelCluster doesn't support the use of SGE or Torque schedulers. You can continue using them in versions up to and including 2.11.4, but they aren't eligible for future updates or troubleshooting support from the AWS service and AWS Support teams.

AWS ParallelCluster versions 2.11.4 and earlier use Son of Grid Engine 8.1.9.

Slurm Workload Manager (slurm)

AWS ParallelCluster version 2.11.9 uses Slurm 20.11.9. For information about Slurm, see <https://slurm.schedmd.com/>. For downloads, see <https://github.com/SchedMD/slurm/tags>. For the source code, see <https://github.com/SchedMD/slurm>.

Important

AWS ParallelCluster is tested with Slurm configuration parameters, which are provided by default. Any changes that you make to these Slurm configuration parameters are done at your own risk. They are supported only on a best-effort basis.

AWS ParallelCluster version(s)	Supported Slurm version
2.11.7, 2.11.8, 2.11.9	20.11.9
2.11.4 to 2.11.6	20.11.8
2.11.0 to 2.11.3	20.11.7
2.10.4	20.02.7
2.9.0 to 2.10.3	20.02.4
2.6 to 2.8.1	19.05.5
2.5.0, 2.5.1	19.05.3-2
2.3.1 to 2.4.1	18.08.6-2
prior to 2.3.1	16.05.3-1

Multiple queue mode

AWS ParallelCluster version 2.9.0 introduced multiple queue mode. Multiple queue mode is supported when [scheduler](#) is set to `slurm` and the [queue_settings](#) setting is defined. This mode allows different instance types to coexist in the compute nodes. The compute resources that contain the different instance types can scale up or down as needed. In queue mode,

up to five (5) queues are supported, and each [\[queue\] section](#) can refer to up to three (3) [\[compute_resource\] sections](#). Each of these [\[queue\] sections](#) is a partition in Slurm Workload Manager. For more information, see [Slurm guide for multiple queue mode](#) and [Multiple queue mode tutorial](#).

Each [\[compute_resource\] section](#) in a queue must have a different instance type, and each of these [\[compute_resource\]](#) is further divided into static and dynamic nodes. Static nodes for each [\[compute_resource\]](#) are numbered from 1 to the value of [min_count](#). Dynamic nodes for each [\[compute_resource\]](#) are numbered from one (1) to ([max_count](#) - [min_count](#)). For example, if [min_count](#) is 2 and [max_count](#) is 10, the dynamic nodes for that [\[compute_resource\]](#) are numbered from one (1) to eight (8). At any time, there can be between zero (0) and the max number of dynamic nodes in a [\[compute_resource\]](#).

The instances that are launched into the compute fleet are dynamically assigned. To help manage this, hostnames are generated for each node. The format of the hostname is as follows:

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

- [\\$QUEUE](#) is the name of the queue. For example, if the section starts `[queue queue-name]` then "[\\$QUEUE](#)" is "*queue-name*".
- [\\$STATDYN](#) is st for static nodes or dy for dynamic nodes.
- [\\$INSTANCE_TYPE](#) is the instance type for the [\[compute_resource\]](#), from the [instance_type](#) setting.
- [\\$NODENUM](#) is the number of the node. [\\$NODENUM](#) is between one (1) and the value of [min_count](#) for static nodes and between one (1) and ([max_count](#) - [min_count](#)) for dynamic nodes.

Both hostnames and fully-qualified domain names (FQDN) are created using Amazon Route 53 hosted zones. The FQDN is `$HOSTNAME.$CLUSTERNAME.pcluster`, where [\\$CLUSTERNAME](#) is the name of the [\[cluster\] section](#) used for the cluster.

To convert your configuration to a queue mode, use the [pcluster-config convert](#) command. It writes an updated configuration with a single [\[queue\] section](#) named `[queue compute]`. That queue contains a single [\[compute_resource\] section](#) that is named `[compute_resource default]`. The `[queue compute]` and `[compute_resource default]` has settings migrated from the specified [\[cluster\] section](#).

Slurm guide for multiple queue mode

AWS ParallelCluster version 2.9.0 introduced multiple queue mode and a new scaling architecture for Slurm Workload Manager (Slurm).

The following sections provide a general overview on using a Slurm cluster with the newly introduced scaling architecture.

Overview

The new scaling architecture is based on Slurm's [Cloud Scheduling Guide](#) and power saving plugin. For more information about the power saving plugin, see [Slurm Power Saving Guide](#). In the new architecture, resources that can potentially be made available for a cluster are typically predefined in the Slurm configuration as cloud nodes.

Cloud node lifecycle

Throughout their lifecycle, cloud nodes enter several if not all of the following states:

POWER_SAVING, POWER_UP (pow_up), ALLOCATED (alloc), and POWER_DOWN (pow_dn). In some cases, a cloud node might enter the OFFLINE state. The following list details several aspects of these states in the cloud node lifecycle.

- A node in a POWER_SAVING state appears with a ~ suffix (for example idle~) in sinfo. In this state, there is no EC2 instance backing the node. However, Slurm can still allocate jobs to the node.
- A node transitioning to a POWER_UP state appears with a # suffix (for example idle#) in sinfo.
- When Slurm allocates job to a node in a POWER_SAVING state, the node automatically transfers to a POWER_UP state. Otherwise, nodes can be placed in the POWER_UP state manually using the `scontrol update nodename=nodename state=pow_up` command. In this stage, the `ResumeProgram` is invoked, and EC2 instances are launched and configured to back a POWER_UP node.
- A node that is currently available for use appears without any suffix (for example idle) in sinfo. After the node is set up and has joined the cluster, it becomes available to run jobs. In this stage, the node is properly configured and ready for use. As a general rule, we recommend that the number of instances in EC2 be the same as the number of available nodes. In most cases, static nodes are always available after the cluster is created.
- A node that is transitioning to a POWER_DOWN state appears with a % suffix (for example idle%) in sinfo. Dynamic nodes automatically enter POWER_DOWN state after [scaledown_idletime](#).

In contrast, static nodes in most cases aren't powered down. However, nodes can be placed in the `POWER_DOWN` state manually using the `scontrol update nodename=nodename state=powering_down` command. In this state, the instance associated with a node is terminated, and node is reset back to the `POWER_SAVING` state to future use after [scaledown_idletime](#). The `scaledown-idletime` setting is saved to the Slurm configuration as the `SuspendTimeout` setting.

- A node that is offline appears with a `*` suffix (for example `down*`) in `sinfo`. A node goes offline if Slurm controller can't contact the node or if the static nodes are disabled and the backing instances are terminated.

Now consider the node states shown in the following `sinfo` example.

```
$ 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]
```

The `spot-st-t2large-[1-2]` and `efa-st-c5n18xlarge-1` nodes already have backing instances set up and are available for use. The `ondemand-dy-c52xlarge-[1-2]` nodes are in the `POWER_UP` state, and they should be available within a few minutes. The `gpu-dy-g38xlarge-1` node is in the `POWER_DOWN` state, and it will transition into `POWER_SAVING` state after [scaledown_idletime](#) (defaults to 120 seconds).

All of the other nodes are in `POWER_SAVING` state with no EC2 instances backing them.

Working with an available node

An available node is backed by an EC2 instance. By default, the node name can be used to directly SSH into the instance (for example `ssh efa-st-c5n18xlarge-1`). The private IP address of the instance can be retrieved using the `scontrol show nodes nodename` command and checking the `NodeAddr` field. For nodes that aren't available, the `NodeAddr` field shouldn't point to a running EC2 instance. Rather, it should be the same as the node name.

Job states and submission

Jobs submitted in most cases are immediately allocated to nodes in the system, or placed in pending if all the nodes are allocated.

If nodes allocated for a job include any nodes in a `POWER_SAVING` state, the job starts out with a `CF`, or `CONFIGURING` state. At this time, the job waits for the nodes in the `POWER_SAVING` state to transition to `POWER_UP` state and become available.

After all nodes allocated for a job are available, the job enters the `RUNNING (R)` state.

By default, all jobs are submitted to the default queue (known as a partition in Slurm). This is signified by a `*` suffix after the queue name. You can select a queue using the `-p` job submission option.

All nodes are configured with the following features, which can be used in job submission commands:

- An instance type (for example `c5.xlarge`)
- A node type (This is either `dynamic` or `static`.)

You can see all of the features available for a particular node by using the `scontrol show nodes nodename` command and checking the `AvailableFeatures` list.

Another consideration is jobs. First consider the initial state of the cluster, which you can view by running the `sinfo` command.

```
$ 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]
```

Note that `spot` is the default queue. It is indicated by the `*` suffix.

Submit a job to one static node to the default queue (`spot`).

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

Submit a job to one dynamic node to the EFA queue.

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

Submit a job to eight (8) c5.2xlarge nodes and two (2) t2.xlarge nodes to the ondemand queue.

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

Submit a job to one GPU node to the gpu queue.

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

Now consider the state of the jobs using the squeue command.

```
$ squeue
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	ODELIST(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							

Jobs 7 and 8 (in the spot and efa queues) are already running (R). Jobs 12 and 13 are still configuring (CF), probably waiting for the instances to become available.

```
# Nodes states corresponds to state of running jobs
```

```
$ sinfo
```

PARTITION	AVAIL	TIMELIMIT	NODES	STATE	ODELIST
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

Node state and features

In most cases, node states are fully managed by AWS ParallelCluster according to the specific processes in the cloud node lifecycle described earlier in this topic.

However, AWS ParallelCluster also replaces or terminates unhealthy nodes in DOWN and DRAINED states and nodes that have unhealthy backing instances. For more information, see [clustermgtd](#).

Partition states

AWS ParallelCluster supports the following partition states. A Slurm partition is a queue in AWS ParallelCluster.

- UP: Indicates that the partition is in an active state. This is the default state of a partition. In this state, all nodes in the partition are active and available for use.
- INACTIVE: Indicates that the partition is in the inactive state. In this state, all instances backing nodes of an inactive partition are terminated. New instances aren't launched for nodes in an inactive partition.

pcluster start and stop

When [pcluster stop](#) is run, all partitions are placed in the INACTIVE state, and the AWS ParallelCluster processes keep the partitions in the INACTIVE state.

When [pcluster start](#) is run, all partitions are initially placed in the UP state. However, AWS ParallelCluster processes don't keep the partition in an UP state. You need to change partition states manually. All static nodes become available after a few minutes. Note that setting a partition to UP doesn't power up any dynamic capacity. If [initial_count](#) is greater than [max_count](#), then [initial_count](#) might not be satisfied when the partition state is changed to the UP state.

When [pcluster start](#) and [pcluster stop](#) are running, you can check the state of the cluster by running the [pcluster status](#) command and checking the ComputeFleetStatus. The following lists possible states:

- STOP_REQUESTED: The [pcluster stop](#) request is sent to the cluster.
- STOPPING: The pcluster process is currently stopping the cluster.
- STOPPED: The pcluster process finished the stopping process, all partitions are in INACTIVE state, and all compute instances are terminated.
- START_REQUESTED: The [pcluster start](#) request is sent to the cluster.
- STARTING: The pcluster process is currently starting the cluster
- RUNNING: The pcluster process finished the starting process, all partitions are in UP state, and static nodes will be available after a few minutes.

Manual control on queues

In some cases, you may want to have some manual control over the nodes or queue (known as a partition in Slurm) in a cluster. You can manage nodes in a cluster through the following common procedures.

- Power up dynamic nodes in POWER_SAVING state: Run the `scontrol update nodename=nodename state=power_up` command or submit a placeholder `sleep 1` job requesting for a certain number of nodes and rely on Slurm to power up required number of nodes.
- Power down dynamic nodes before [scaledown_idletime](#): Set dynamic nodes to DOWN with the `scontrol update nodename=nodename state=down` command. AWS ParallelCluster automatically terminates and resets the downed dynamic nodes. In general, we don't recommend setting nodes to POWER_DOWN directly using the `scontrol update nodename=nodename state=power_down` command. This is because AWS ParallelCluster automatically handles the power down process. No manual intervention is necessary. Therefore, we recommend that you try to set nodes to DOWN whenever possible.
- Disable a queue (partition) or stop all static nodes in specific partition: Set specific the queue to INACTIVE with the `scontrol update partition=queue name state=inactive` command. Doing this terminates all instances backing nodes in the partition.
- Enable a queue (partition): Set specific the queue to INACTIVE with the `scontrol update partition=queue name state=up` command.

Scaling behavior and adjustments

Here is an example of the normal scaling workflow:

- The scheduler receives a job that requires two nodes.
- The scheduler transitions two nodes to a `POWER_UP` state, and calls `ResumeProgram` with the node names (for example `queue1-dy-c5xlarge-[1-2]`).
- `ResumeProgram` launches two EC2 instances and assigns the private IP addresses and hostnames of `queue1-dy-c5xlarge-[1-2]`, waiting for `ResumeTimeout` (the default period is 60 minutes (1 hour)) before resetting the nodes.
- Instances are configured and join the cluster. Job starts running on instances.
- Job is done.
- After the configured `SuspendTime` has elapsed (which is set to [scaledown_idletime](#)), the instances are put in the `POWER_SAVING` state by the scheduler. Scheduler places `queue1-dy-c5xlarge-[1-2]` into `POWER_DOWN` state and calls `SuspendProgram` with the node names.
- `SuspendProgram` is called for two nodes. Nodes remain in the `POWER_DOWN` state, for example, by remaining `idle%` for a `SuspendTimeout` (the default period is 120 seconds (2 minutes)). After `clustermgtd` detects that nodes are powering down, it terminates the backing instances. Then, it configures `queue1-dy-c5xlarge-[1-2]` into idle state and resets private IP address and hostname so they can be power up for future jobs again.

Now, if things go wrong and an instance for a particular node can't be launched for some reason, then the following happens.

- Scheduler receives a job that requires two nodes.
- Scheduler places two cloud bursting nodes into `POWER_UP` state and calls `ResumeProgram` with the nodenames, (for example `queue1-dy-c5xlarge-[1-2]`).
- `ResumeProgram` launches only one (1) EC2 instance and configures `queue1-dy-c5xlarge-1`, but it failed to launch an instance for `queue1-dy-c5xlarge-2`.
- `queue1-dy-c5xlarge-1` will not be affected and will come online after reaching `POWER_UP` state.
- `queue1-dy-c5xlarge-2` is placed in `POWER_DOWN` state, and the job is requeued automatically because Slurm detects a node failure.
- `queue1-dy-c5xlarge-2` becomes available after `SuspendTimeout` (the default is 120 seconds (2 minutes)). In the meantime, the job is requeued and can start running on another node.
- The above process is repeated until the job can run on an available node without a failure occurring.

There are two timing parameters that can be adjusted if needed.

- **ResumeTimeout** (the default is 60 minutes (1 hour)): **ResumeTimeout** controls the time Slurm waits before putting the node the down state.
 - It might be useful to extend this if your pre/post installation process takes nearly that long.
 - This is also the maximum time that AWS ParallelCluster waits before replacing or resetting a node if there is an issue. Compute nodes self-terminate if any error occurs during launch or setup. Next, AWS ParallelCluster processes also replaces the node when it sees that the instance is terminated.
- **SuspendTimeout** (the default is 120 seconds (2 minutes)): **SuspendTimeout** controls how quickly nodes get placed back into the system and ready for use again.
 - A shorter **SuspendTimeout** would mean that nodes will be reset more quickly, and Slurm is able to try to launch instances more frequently.
 - A longer **SuspendTimeout** makes failed nodes reset more slowly. In the meantime, Slurm tires to use other nodes. If **SuspendTimeout** is more than a few minutes, Slurm tries to cycle through all nodes in the system. A longer **SuspendTimeout** might be beneficial for large-scale systems (over 1,000 nodes) for reducing stress on Slurm by frequently re-queuing failing jobs.
 - Note that **SuspendTimeout** doesn't refer to the time AWS ParallelCluster waited to terminate a backing instance for a node. Backing instances for power down nodes are immediately terminated. The terminate process usually is finished a few minutes. However, during this time, the node remains in the power down state and not available for use in the scheduler.

Logs for new architecture

The following list contains the key logs for the multiple queue architecture.

The log stream name used with Amazon CloudWatch Logs has the format

`{hostname}.{instance_id}.{logIdentifier}`, where *logIdentifier* follows the log names. For more information, see [Integration with Amazon CloudWatch Logs](#).

- **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)`

Common issues and how to debug:

Nodes that failed to launch, power up, or join the cluster:

- **Dynamic nodes:**
 - Check the `ResumeProgram` log to see if `ResumeProgram` was ever called with the node. If not, check the `slurmctld` log to determine if Slurm ever tried to call `ResumeProgram` with the node. Note that incorrect permissions on `ResumeProgram` might cause it to fail silently.
 - If `ResumeProgram` is called, check to see if an instance was launched for the node. If the instance can't be launched, there should be clear error message as to why the instance failed to launch.
 - If an instance was launched, there may have been some problem during the bootstrap process. Find the corresponding private IP address and instance ID from the `ResumeProgram` log and look at corresponding bootstrap logs for the specific instance in CloudWatch Logs.
- **Static nodes:**
 - Check the `clustermgtd` log to see if instances were launched for the node. If not, there should be clear errors on why the instances failed to launch.
 - If an instance was launched, there is some problem during bootstrap process. Find the corresponding private IP and instance ID from the `clustermgtd` log and look at corresponding bootstrap logs for the specific instance in CloudWatch Logs.

Nodes replaced or terminated unexpectedly, node failures

- Nodes replaced/terminated unexpectedly
 - In most cases, `clustermgtd` handles all node maintenance actions. To check if `clustermgtd` replaced or terminated a node, check the `clustermgtd` log.
 - If `clustermgtd` replaced or terminated the node, there should be a message indicating the reason for the action. If the reason is scheduler related (for example, the node was DOWN), check in the `slurmctld` log for more details. If the reason is EC2 related, use tools to check status or logs for that instance. For example, you can check if the instance had scheduled events or failed EC2 health status checks.
 - If `clustermgtd` didn't terminate the node, check if `computemgtd` terminated the node or if EC2 terminated the instance to reclaim a Spot Instance.
- Node failures
 - In most cases, jobs are automatically requeued if a node failed. Look in the `slurmctld` log to see why a job or a node failed and analyze the situation from there.

Failure when replacing or terminating instances, failure when powering down nodes

- In general, `clustermgtd` handles all expected instance termination actions. Look in the `clustermgtd` log to see why it failed to replace or terminate a node.
- For dynamic nodes failing [scaledown_idletime](#), look in the `SuspendProgram` log to see if a program by `slurmctld` with the specific node as argument. Note `SuspendProgram` doesn't actually perform any specific action. Rather, it only logs when it's called. All instance termination and `NodeAddr` reset are completed by `clustermgtd`. Slurm puts nodes into IDLE after `SuspendTimeout`.

Other issues

- AWS ParallelCluster doesn't make job allocation or scaling decisions. It simply tries to launch, terminate, and maintain resources according to Slurm's instructions.

For issues regarding job allocations, node allocation and scaling decision, look at the `slurmctld` log for errors.

Torque Resource Manager (torque)

Note

Starting with version 2.11.5, AWS ParallelCluster doesn't support the use of SGE or Torque schedulers. You can continue using them in versions up to and including 2.11.4, but they aren't eligible for future updates or troubleshooting support from the AWS service and AWS Support teams.

AWS ParallelCluster versions 2.11.4 and earlier use Torque Resource Manager 6.1.2. For more information about Torque Resource Manager 6.1.2, see <http://docs.adaptivecomputing.com/torque/6-1-2/releaseNotes/torquerelnote.htm>. For documentation, see <http://docs.adaptivecomputing.com/torque/6-1-2/adminGuide/torque.htm>. For the source code, see <https://github.com/adaptivecomputing/torque/tree/6.1.2>.

AWS ParallelCluster versions 2.4.0 and earlier use Torque Resource Manager 6.0.2. For release notes, see <http://docs.adaptivecomputing.com/torque/6-0-2/releaseNotes/torqueReleaseNotes6.0.2.pdf>. For documentation, see <http://docs.adaptivecomputing.com/torque/6-0-2/adminGuide/help.htm>. For the source code, see <https://github.com/adaptivecomputing/torque/tree/6.0.2>.

AWS Batch (awsbatch)

For information about AWS Batch, see [AWS Batch](#). For documentation, see the [AWS Batch User Guide](#).

AWS ParallelCluster CLI commands for AWS Batch

When you use the `awsbatch` scheduler, the AWS ParallelCluster CLI commands for AWS Batch are automatically installed in the AWS ParallelCluster head node. The CLI uses AWS Batch API operations and permits the following operations:

- Submit and manage jobs.
- Monitor jobs, queues, and hosts.
- Mirror traditional scheduler commands.

⚠ Important

AWS ParallelCluster doesn't support GPU jobs for AWS Batch. For more information, see [GPU jobs](#).

Topics

- [awsbsub](#)
- [awsbstat](#)
- [awsbout](#)
- [awsbkill](#)
- [awsbqueues](#)
- [awsbhosts](#)

awsbsub

Submits jobs to the job queue of the cluster.

```
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 doesn't support GPU jobs for AWS Batch. For more information, see [GPU jobs](#).

Positional Arguments***command***

Submits the job (the command specified must be available on the compute instances) or the file name to be transferred. See also `--command-file`.

arguments

(Optional) Specifies arguments for the command or the command-file.

Named Arguments

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

Names the job. The first character must be either a letter or number. The job name can contain letters (both uppercase and lowercase), numbers, hyphens, and underscores, and be up to 128 characters in length.

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

Specifies the cluster to use.

-cf, --command-file

Indicates that the command is a file to be transferred to the compute instances.

Default: False

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

Specifies the folder to use as the job's working directory. If a working directory isn't specified, the job is run in the `job-<AWS_BATCH_JOB_ID>` subfolder of the user's home directory. You can use either this parameter or the `--parent-working-dir` parameter.

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

Specifies the parent folder of the job's working directory. If a parent working directory isn't specified, it defaults to the user's home directory. A subfolder named `job-<AWS_BATCH_JOB_ID>` is created in the parent working directory. You can use either this parameter or the `--working-dir` parameter.

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

Specifies the file to be transferred to the compute instances, in the job's working directory. You can specify multiple input file parameters.

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

Specifies the number of vCPUs to reserve for the container. When used together with `-nodes`, it identifies the number of vCPUs for each node.

Default: 1

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

Specifies the hard limit of memory (in MiB) to provide for the job. If your job attempts to exceed the memory limit specified here, the job is ended.

Default: 128

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

Specifies a comma-separated list of environment variable names to export to the job environment. To export all environment variables, specify 'all'. Note that a list of 'all' environment variables doesn't include those listed in the `--env-blacklist` parameter, or variables starting with the `PCLUSTER_*` or `AWS_*` prefix.

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

Specifies a comma-separated list of environment variable names to **not** export to the job environment. By default, `HOME`, `PWD`, `USER`, `PATH`, `LD_LIBRARY_PATH`, `TERM`, and `TERMCAP` are not exported.

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

Specifies the number of times to move a job to the `RUNNABLE` status. You can specify between 1 and 10 attempts. If the value of attempts is greater than 1, the job is retried if it fails, until it has moved to a `RUNNABLE` status for the specified number of times.

Default: 1

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

Specifies the time duration in seconds (measured from the job attempt's `startedAt` timestamp) after which AWS Batch terminates your job if it hasn't finished. The timeout value must be at least 60 seconds.

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

Specifies the number of nodes to reserve for the job. Specify a value for this parameter to enable multi-node parallel submission.

 Note

Multi-node parallel jobs aren't supported when the [cluster_type](#) parameter is set to `spot`.

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

Indicates the size of the array. You can specify a value between 2 and 10,000. If you specify array properties for a job, it becomes an array job.

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

Specifies a semicolon-separated list of dependencies for a job. A job can depend upon a maximum of 20 jobs. You can specify a SEQUENTIAL type dependency without specifying a job ID for array jobs. A sequential dependency allows each child array job to complete sequentially, starting at index 0. You can also specify an N_TO_N type dependency with a job ID for array jobs. An N_TO_N dependency means that each index child of this job must wait for the corresponding index child of each dependency to complete before it can begin. The syntax for this parameter is "jobId=<*stringstring*

awsbstat

Shows the jobs that are submitted in the cluster's job queue.

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

Positional Arguments

job_ids

Specifies the space-separated list of job IDs to show in the output. If the job is a job array, all of the child jobs are displayed. If a single job is requested, it is shown in a detailed version.

Named Arguments

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

Indicates the cluster to use.

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

Specifies a comma-separated list of job statuses to include. The default job status is "active.". Accepted values are: SUBMITTED, PENDING, RUNNABLE, STARTING, RUNNING, SUCCEEDED, FAILED, and ALL.

Default: "SUBMITTED,PENDING,RUNNABLE,STARTING,RUNNING"

-e, --expand-children

Expands jobs with children (both array and multi-node parallel).

Default: False

-d, --details

Shows jobs details.

Default: False

awsbout

Shows the output of a given job.

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

Positional Arguments***job_id***

Specifies the job ID.

Named Arguments**-c *CLUSTER*, --cluster *CLUSTER***

Indicates the cluster to use.

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

Gets the first *HEAD* lines of the job output.

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

Gets the last <tail> lines of the job output.

-s, --stream

Gets the job output, and then waits for additional output to be produced. This argument can be used together with `-tail` to start from the latest <tail> lines of the job output.

Default: False

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

Sets the streaming period.

Default: 5

awsbkill

Cancels or terminates jobs submitted in the cluster.

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

Positional Arguments

job_ids

Specifies the space-separated list of job IDs to cancel or terminate.

Named Arguments

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

Indicates the name of the cluster to use.

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

Indicates the message to attach to a job, explaining the reason for canceling it.

Default: "Terminated by the user"

awsbqueues

Shows the job queue that is associated with the cluster.

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

Positional arguments

job_queues

Specifies the space-separated list of queue names to show. If a single queue is requested, it is shown in a detailed version.

Named arguments

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

Specifies the name of the cluster to use.

-d, --details

Indicates whether to show the details of the queues.

Default: False

awsbhosts

Shows the hosts that belong to the cluster's compute environment.

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

Positional Arguments

instance_ids

Specifies a space-separated list of instances IDs. If a single instance is requested, it is shown in a detailed version.

Named Arguments

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

Specifies the name of the cluster to use.

-d, --details

Indicates whether to show the details of the hosts.

Default: False

AWS ParallelCluster resources and tagging

With AWS ParallelCluster you can create tags to track and manage your AWS ParallelCluster resources. You define the tags that you want AWS CloudFormation to create and propagate to all

cluster resources in the [tags](#) section of the cluster configuration file. You can also use tags that AWS ParallelCluster automatically generates to track and manage your resources.

When you create a cluster, the cluster and its resources are tagged with the AWS ParallelCluster and AWS systems tags defined in this section.

AWS ParallelCluster applies tags to the cluster instances, volumes, and resources. To identify the cluster stack, AWS CloudFormation applies AWS system tags to the cluster instances. To identify the cluster EC2 launch templates, EC2 applies system tags to the instances. You can use these tags to view and manage your AWS ParallelCluster resources.

You can't modify AWS system tags. In order to avoid impacts to AWS ParallelCluster functionality, don't modify AWS ParallelCluster tags.

The following is an example of an AWS system tag for an AWS ParallelCluster resource. You can't modify them.

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

The following are examples of AWS ParallelCluster tags applied to a resource. Don't modify them.

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

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

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

You can view these tags in the EC2 section of the AWS Management Console.

View tags

1. Navigate the EC2 console at <https://console.aws.amazon.com/ec2/>.
2. To view all cluster tags, choose **Tags** in the navigation pane.
3. To view cluster tags by instance, choose **Instances** in the navigation pane.
4. Select a cluster instance.
5. Choose the **Manage tags** tab in the instance details and view the tags.

6. Choose the **Storage** tab in the instance details.
7. Select the **Volume ID**.
8. In **Volumes**, choose the volume.
9. Choose the **Tags** tab in the volume details and view the tags.

AWS ParallelCluster head node instance tags

Key	Tag value
ClusterName	<i>clustername</i>
Name	Master
Application	parallelcluster- <i>clustername</i>
aws:ec2launchtemplate:id	<i>lt-1234567890abcdef0</i>
aws:ec2launchtemplate:version	<i>1</i>
aws-parallelcluster-node-type	Master
aws:cloudformation:stack-name	parallelcluster- <i>clustername</i> - MasterServerSubstack- <i>ABCD1234E FGH</i>
aws:cloudformation:logical-id	MasterServer
aws:cloudformation:stack-id	arn:aws:cloudformation: <i>region- id</i> : <i>ACCOUNTID</i> :stack/parallelclu ster- <i>clustername</i> -MasterSe rverSubstack- <i>ABCD1234E FGH /1234abcd-12ab-12ab-12ab-123 4567890abcdef0</i>
Version	<i>2.11.9</i>

AWS ParallelCluster head node root volume tags

Tag key	Tag value
ClusterName	<i>clustername</i>
Application	parallelcluster- <i>clustername</i>
aws-parallelcluster-node-type	Master

AWS ParallelCluster compute node instance tags

Key	Tag value
ClusterName	<i>clustername</i>
aws-parallelcluster-node-type	Compute
aws:ec2launchtemplate:id	<i>lt-1234567890abcdef0</i>
aws:ec2launchtemplate:version	<i>1</i>
QueueName	<i>queue-name</i>
Version	<i>2.11.9</i>

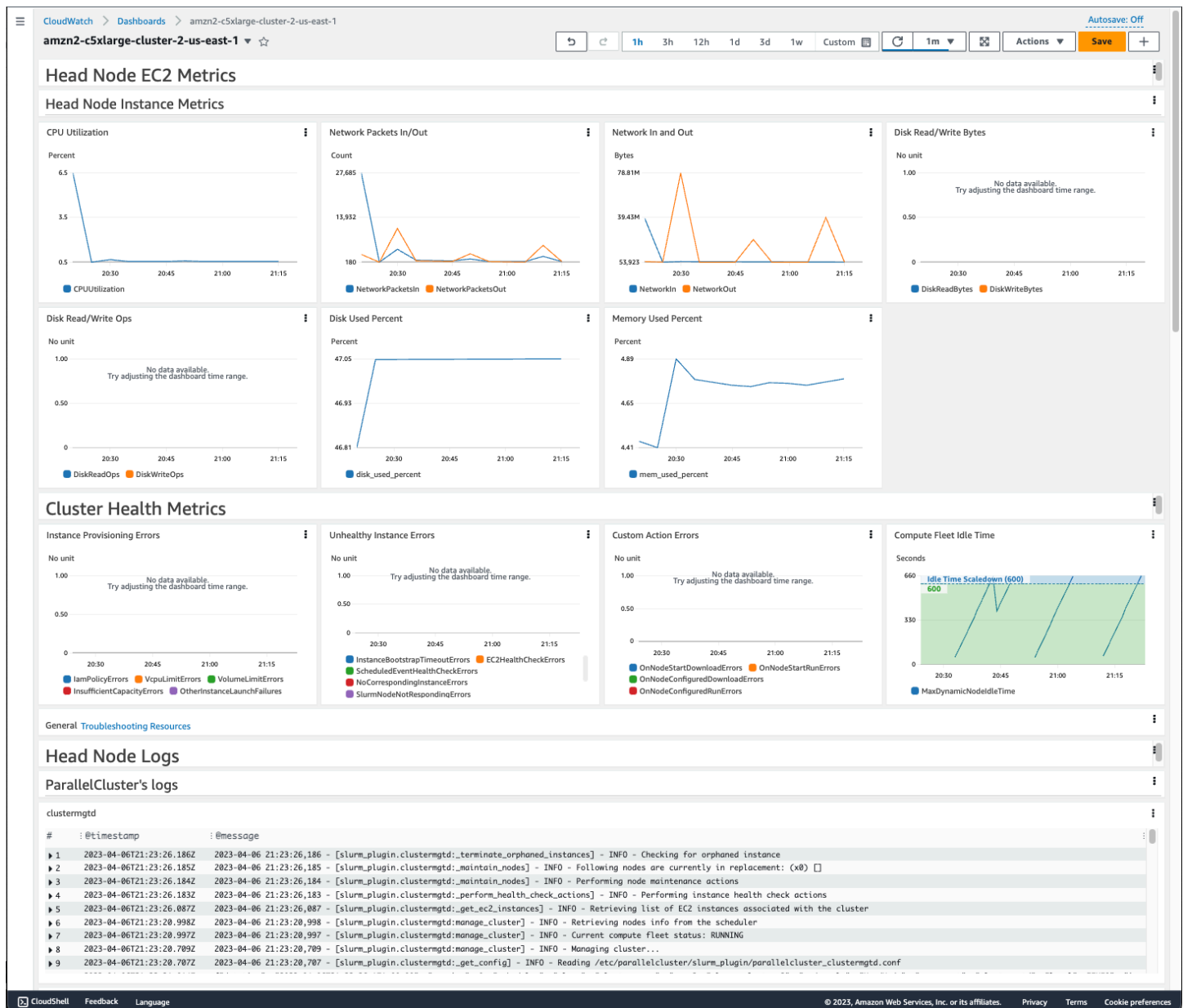
AWS ParallelCluster compute node root volume tags

Tag key	Tag value
ClusterName	<i>clustername</i>
Application	parallelcluster- <i>clustername</i>
aws-parallelcluster-node-type	Compute
QueueName	<i>queue-name</i>
Version	<i>2.11.9</i>

Amazon CloudWatch dashboard

Starting with AWS ParallelCluster version 2.10.0, an Amazon CloudWatch dashboard is created when the cluster is created. This makes it easier to monitor the nodes in your cluster, and to view the logs stored in Amazon CloudWatch Logs. The name of the dashboard is `parallelcluster-ClusterName-Region`. *ClusterName* is the name of your cluster and *Region* is the AWS Region of the cluster. You can access the dashboard in the console, or by opening `https://console.aws.amazon.com/cloudwatch/home?region=Region#dashboards:name=parallelcluster-ClusterName`.

The following image shows an example CloudWatch dashboard for a cluster.



The first section of the dashboard displays graphs of the Head Node EC2 metrics. If your cluster has shared storage, the next section shows shared storage metrics. The final section lists Head Node Logs grouped by ParallelCluster's logs, Scheduler's logs, NICE DCV integration logs, and System's logs.

For more information about Amazon CloudWatch dashboards, see [Using Amazon CloudWatch dashboards](#) in the *Amazon CloudWatch User Guide*.

If you don't want to create the Amazon CloudWatch dashboard, you must complete these steps: First, add a [\[dashboard\] section](#) to your configuration file, and then add the name of that

section as the value of the [dashboard_settings](#) setting in your [\[cluster\] section](#). In your [\[dashboard\] section](#), set [enable](#) = false.

For example, if your [\[dashboard\] section](#) is named myDashboard and your [\[cluster\] section](#) is named myCluster, your changes resemble this.

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

[dashboard MyDashboard]
enable = false
```

Integration with Amazon CloudWatch Logs

Starting with AWS ParallelCluster version 2.6.0, common logs are stored in CloudWatch Logs by default. For more information about CloudWatch Logs, see [Amazon CloudWatch Logs User Guide](#). To configure CloudWatch Logs integration, see the [\[cw_log\] section](#) and the [cw_log_settings](#) setting.

A log group is created for each cluster with a name `/aws/parallelcluster/cluster-name` (for example, `/aws/parallelcluster/testCluster`). Each log (or set of logs if the path contains a `*`) on each node has a log stream named `{hostname}.{instance_id}.{logIdentifier}`. (For example `ip-172-31-10-46.i-02587cf29cc3048f3.nodewatcher`.) Log data is sent to CloudWatch by the [CloudWatch agent](#), which runs as root on all cluster instances.

Starting with AWS ParallelCluster version 2.10.0, an Amazon CloudWatch dashboard is created when the cluster is created. This dashboard makes it easy to review the logs stored in CloudWatch Logs. For more information, see [Amazon CloudWatch dashboard](#).

This list contains the *logIdentifier* and path for the log streams available for platforms, schedulers, and nodes.

Log streams available for platforms, schedulers, and nodes

Platform	Schedulers	Nodes	Log streams
amazon awsbatch	HeadNode	dcv-authenticator	<code>/var/log/parallelcluster/pc-luster_dcv_authenticator.log</code>

Platform	Schedulers	Nodes	Log streams
centos ubuntu	slurm		dcv-ext-authenticator: /var/log/parallelcluster/parallelcluster_dcv_connect.log dcv-agent: /var/log/dcv/agent.*.log dcv-xsession: /var/log/dcv/dcv-xsession.*.log dcv-server: /var/log/dcv/server.log dcv-session-launcher: /var/log/dcv/sessionlauncher.log Xdcv: /var/log/dcv/Xdcv.*.log cfn-init: /var/log/cfn-init.log chef-client: /var/log/chef-client.log
amazon centos ubuntu	awsbatch slurm	Compute et HeadNode	cloud-init: /var/log/cloud-init.log supervisord: /var/log/supervisord.log
amazon centos ubuntu	slurm	Compute et	cloud-init-output: /var/log/cloud-init-output.log computemgtd: /var/log/parallelcluster/computemgtd slurmd: /var/log/slurmd.log
amazon centos ubuntu	slurm	HeadNode	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

Platform	Schedulers	Nodes	Log streams
amazon	awsbatch	Compute	system-messages: /var/log/messages
centos	slurm	HeadNode	
ubuntu	awsbatch	Compute	syslog: /var/log/syslog
	slurm	HeadNode	

Jobs in clusters that use AWS Batch store the output of jobs that reached a RUNNING, SUCCEEDED, or FAILED state in CloudWatch Logs. The log group is `/aws/batch/job`, and the log stream name format is `jobDefinitionName/default/ecs_task_id`. By default, these logs are set to never expire, but you can modify the retention period. For more information, see [Change log data retention in CloudWatch Logs](#) in the *Amazon CloudWatch Logs User Guide*.

Note

`chef-client`, `cloud-init-output`, `clustermgtd`, `computemgtd`, `slurm_resume`, and `slurm_suspend` were added in AWS ParallelCluster version 2.9.0. For AWS ParallelCluster version 2.6.0, `/var/log/cfn-init-cmd.log` (`cfn-init-cmd`) and `/var/log/cfn-wire.log` (`cfn-wire`) were also stored in CloudWatch Logs.

Elastic Fabric Adapter

Elastic Fabric Adapter (EFA) is a network device that has OS-bypass capabilities for low-latency network communications with other instances on the same subnet. EFA is exposed by using Libfabric, and can be used by applications using the Messaging Passing Interface (MPI).

To use EFA with AWS ParallelCluster, add the line `enable_efa = true` to the [\[queue\] section](#).

To view the list of EC2 instances that support EFA, see [Supported instance types](#) in the *Amazon EC2 User Guide for Linux Instances*.

For more information about the `enable_efa` setting, see [enable_efa](#) in the [\[queue\] section](#).

A cluster placement group should be used to minimize latencies between instances. For more information, see [placement](#) and [placement_group](#).

For more information, see [Elastic Fabric Adapter](#) in the *Amazon EC2 User Guide* and [Scale HPC workloads with elastic fabric adapter and AWS ParallelCluster](#) in the *AWS Open Source Blog*.

Note

By default, Ubuntu distributions enable ptrace (process trace) protection. Starting with AWS ParallelCluster 2.6.0, ptrace protection is disabled so that Libfabric works properly. For more information, see [Disable ptrace protection](#) in the *Amazon EC2 User Guide*.

Note

Support EFA on Arm-based Graviton2 instances was added in AWS ParallelCluster version 2.10.1.

Intel Select Solutions

AWS ParallelCluster is available as an Intel Select Solution for simulation and modeling. Configurations are verified to meet the standards set by the [Intel HPC Platform Specification](#), use specific Intel instance types, and are configured to use the [Elastic Fabric Adapter](#) (EFA) networking interface. AWS ParallelCluster is the first cloud solution to meet the requirements for the Intel Select Solutions program. Supported instance types include c5n.18xlarge, m5n.24xlarge, and r5n.24xlarge. A sample configuration compatible with the Intel Select Solutions standard is provided below.

Example Intel Select Solutions configuration

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

[aws]
aws_region_name = <Your AWS Region>

[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
```

For more information about AWS ParallelCluster and the Intel HPC Platform Specification, see [Intel HPC Platform Specification](#).

Enable Intel MPI

Intel MPI is available on the AWS ParallelCluster AMIs. To use Intel MPI, you must acknowledge and accept the terms of the [Intel simplified software license](#). By default, Open MPI is placed on the path. To enable Intel MPI instead of Open MPI, you must first load the Intel MPI module. Then, you need to install the latest version by using `module load intelmpi`. The exact name of the module changes with every update. To see which modules are available, run `module avail`. The output is as follows.

```
$ 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
```

To see which modules are loaded, run `module list`.

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

To verify that Intel MPI is enabled, run `mpirun --version`.

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

After the Intel MPI module has been loaded, multiple paths are changed to use the Intel MPI tools. To run code that was compiled by the Intel MPI tools, load the Intel MPI module first.

Note

Intel MPI isn't compatible with AWS Graviton-based instances.

Note

Before AWS ParallelCluster version 2.5.0, Intel MPI wasn't available on the AWS ParallelCluster AMIs in the China (Beijing) and China (Ningxia) Regions.

Intel HPC Platform Specification

AWS ParallelCluster is compliant with the Intel HPC Platform Specification. The Intel HPC Platform Specification provides a set of compute, fabric, memory, storage, and software requirements to help achieve a high standard of quality and compatibility with HPC workloads. For more information, see [Intel HPC Platform Specification](#) and [Applications Verified Compatible with the Intel HPC Platform Specification](#).

To be compliant with the Intel HPC Platform Specification, the following requirements must be met:

- The operating system must be CentOS 7 (`base_os` = centos7).
- The instance type for the compute nodes must have an Intel CPU and at least 64 GB of memory. For the c5 family of instance types, this means that the instance type must be at least a c5.9xlarge (`compute_instance_type` = c5.9xlarge).
- The head node must have at least 200 GB of storage.
- The End User License Agreement for Intel Parallel Studio must be accepted (`enable_intel_hpc_platform` = true).
- Each compute node must have at least 80 GB of storage (`compute_root_volume_size` = 80).

The storage can be local or on a network (NFS shared from the head node, Amazon EBS or FSx for Lustre), and it can be shared.

Arm Performance Libraries

Starting with AWS ParallelCluster version 2.10.1, Arm Performance Libraries are available on the AWS ParallelCluster AMIs for `alinux2`, `centos8`, `ubuntu1804`, and `ubuntu2004` values for the `base_os` setting. The Arm Performance Libraries provides optimized standard core math libraries for high-performance computing applications on Arm processors. To use Arm Performance Libraries, you must acknowledge and accept the terms of the [Arm Performance Libraries \(free version\) - End User License Agreement](#). For more information on Arm Performance Libraries, see [Free Arm Performance Libraries](#).

To enable Arm Performance Libraries, you must first load the Arm Performance Libraries module. `armpl-21.0.0` needs GCC-9.3 as a requirement, when you load the `armpl/21.0.0` module, the `gcc/9.3` module will also be loaded. The exact name of the module changes with every update. To see which modules are available, run `module avail`. Then, you need to install the latest version by using `module load armpl`. The output is as follows.

```
$ 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
```

To load a module, run `module load modulename`. You can add this to the script used to run `mpirun`.

```
$ 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
```

To see which modules are loaded, run `module list`.

```
$ 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

To verify that Arm Performance Libraries are enabled, run example tests.

```
$ 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
```

After the Arm Performance Libraries module has been loaded, multiple paths are changed to use the Arm Performance Libraries tools. To run code that was compiled by the Arm Performance Libraries tools, load the Arm Performance Libraries module first.

Note

AWS ParallelCluster versions between 2.10.1 and 2.10.4 use armpl/20.2.1.

Connect to the head node through Amazon DCV

Amazon DCV is a remote visualization technology that enables users to securely connect to graphic-intensive 3D applications that are hosted on a remote high-performance server. For more information, see [Amazon DCV](#).

Amazon DCV software is automatically installed on the head node when using `base_os = alinux2`, `base_os = centos7`, `base_os = ubuntu1804` or `base_os = ubuntu2004`.

If the head node is an ARM instance, the Amazon DCV software is automatically installed on it when using `base_os = alinux2`, `base_os = centos7`, or `base_os = ubuntu1804`.

To enable Amazon DCV on the head node, `dcv_settings` must contain the name of a `[dcv] section` that has `enable = master` and `base_os` must be set to `alinux2`, `centos7`, `ubuntu1804`, or `ubuntu2004`. If the head node is an ARM instance, `base_os` must be set to `alinux2`, `centos7`, or `ubuntu1804`. This way, AWS ParallelCluster sets the cluster configuration parameter `shared_dir` to the [DCV server storage folder](#).

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

For more information about Amazon DCV configuration parameters, see [dcv_settings](#). To connect to the Amazon DCV session, use the [pcluster dcv](#) command.

Note

Support for Amazon DCV on centos8 was removed in AWS ParallelCluster version 2.10.4. Support for Amazon DCV on centos8 was added in AWS ParallelCluster version 2.10.0. Support for Amazon DCV on AWS Graviton-based instances was added in AWS ParallelCluster version 2.9.0. Support for Amazon DCV on alinux2 and ubuntu1804 was added in AWS ParallelCluster version 2.6.0. Support for Amazon DCV on centos7 was added in AWS ParallelCluster version 2.5.0.

Note

Amazon DCV is not supported on AWS Graviton-based instances in AWS ParallelCluster versions 2.8.0 and 2.8.1.

Amazon DCV HTTPS certificate

Amazon DCV automatically generates a self-signed certificate to secure traffic between the Amazon DCV client and Amazon DCV server.

To replace the default self-signed Amazon DCV certificate with another certificate, first connect to the head node. Then, copy both the certificate and key to the `/etc/dcv` folder before running the [pcluster dcv](#) command.

For more information, see [Changing the TLS certificate](#) in the *Amazon DCV Administrator Guide*.

Licensing Amazon DCV

The Amazon DCV server doesn't require a license server when running on Amazon EC2 instances. However, the Amazon DCV server must periodically connect to an Amazon S3 bucket to determine if a valid license is available.

AWS ParallelCluster automatically adds the required permissions to the `ParallelClusterInstancePolicy`. When using a custom IAM Instance Policy, use the permissions described in [Amazon DCV on Amazon EC2](#) in the *Amazon DCV Administrator Guide*.

For troubleshooting tips, see [Troubleshooting issues in Amazon DCV](#).

Using `pcluster update`

Starting with AWS ParallelCluster version 2.8.0, [pcluster update](#) analyzes the settings used to create the current cluster and the settings in the configuration file for issues. If any issues are discovered, they are reported, and the steps to take to fix the issues are displayed. For example, if the [compute_instance_type](#) setting is changed to a different instance type, the compute fleet must be stopped before an update can proceed. This issue is reported when it is discovered. If no blocking issues are reported, you are prompted whether you want to apply the changes.

The documentation for each setting defines the update policy for that setting.

Update policy: These settings can be changed during an update., Update policy: This setting can be changed during an update.

These settings can be changed, and the cluster can be updated using [pcluster update](#).

Update policy: If this setting is changed, the update is not allowed.

These settings can't be changed if the existing cluster hasn't been deleted. Either the change must be reverted or the cluster must be deleted (using [pcluster delete](#)), and then a new cluster created (using [pcluster create](#)) in the old cluster's place.

Update policy: This setting is not analyzed during an update.

These settings can be changed, and the cluster updated using [pcluster update](#).

Update policy: The compute fleet must be stopped for this setting to be changed for an update.

These settings cannot be changed while the compute fleet exists. Either the change must be reverted or the compute fleet must be stopped (using [pcluster stop](#)), updated (using [pcluster update](#)), and then a new compute fleet created (using [pcluster start](#)).

Update policy: This setting can't be decreased during an update.

These settings can be changed, but they cannot be decreased. If these settings must be decreased, it is necessary to delete the cluster (using [pcluster delete](#)), and create a new cluster (using [pcluster create](#)).

Update policy: Reducing the size of a queue below the current number of nodes requires that the compute fleet be stopped first.

These settings can be changed, but if the change would reduce the size of the queue below the current size, the compute fleet must be stopped (using [pcluster stop](#)), updated (using [pcluster update](#)), and then a new compute fleet created (using [pcluster start](#)).

Update policy: Reducing the number of static nodes in a queue requires that the compute fleet be stopped first.

These settings can be changed, but if the change would reduce the number of static nodes in the queue below the current size, the compute fleet must be stopped (using [pcluster stop](#)), updated (using [pcluster update](#)), and then a new compute fleet created (using [pcluster start](#)).

Update policy: If this setting is changed, the update is not allowed. Updating this setting cannot be forced.

These settings can't be changed if the existing cluster hasn't been deleted. Either the change must be reverted or the cluster must be deleted (using [pcluster delete](#)), and then a new cluster created (using [pcluster create](#)) in the old cluster's place.

Update policy: If AWS ParallelCluster managed Amazon FSx for Lustre file systems aren't specified in the configuration, this setting can be changed during an update.

This setting can be changed if [\[cluster\] fsx_settings](#) isn't specified or if both [fsx_settings](#) and [fsx-fs-id](#) in [\[fsx fs\]](#) are specified to mount an existing external FSx for Lustre file system.

This example demonstrates a [pcluster update](#) with some changes that block the update.

```
$ 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 patching and EC2 instance replacement

To ensure that all dynamically launched cluster compute nodes behave in a consistent manner, AWS ParallelCluster disables cluster instance automatic OS updates. Additionally, a specific set of

AWS ParallelCluster AMIs are built for each version of AWS ParallelCluster and its associated CLI. This specific set of AMIs remain unchanged and they are only supported by the AWS ParallelCluster version they were built for. AWS ParallelCluster AMIs for released versions aren't updated.

However, due to emergent security issues, customers might want to add patches to these AMIs and then update their clusters with the patched AMI. This aligns with the [AWS ParallelCluster Shared Responsibility Model](#).

To view the specific set of AWS ParallelCluster AMIs supported by the AWS ParallelCluster CLI version you are currently using, run:

```
$ pcluster version
```

Then view [amis.txt](#) in AWS ParallelCluster's GitHub repository.

The AWS ParallelCluster head node is a static instance and you can manually update it. Restart and reboot of the head node is fully supported starting with AWS ParallelCluster version 2.11, if the instance type doesn't have an instance store. For more information, see [Instance types with instance store volumes](#) in the *Amazon EC2 User Guide for Linux Instances*. You can't update an AMI for an existing cluster.

Head node restart and reboot with AMI updates of cluster compute instances is fully supported starting with AWS ParallelCluster version 3.0.0. Consider upgrading to the most recent version to use these features.

Head node instance update or replacement

In some circumstances, you might be required to restart or reboot the head node. For example, this is required when you manually update the OS, or when there's an [AWS instance scheduled retirement](#) that imposes a head node instance restart.

If your instance doesn't have ephemeral drives, you can stop and start it again at any time. In case of a scheduled retirement, starting the stopped instance migrates it to use the new hardware.

Similarly, you can manually stop and start an instance that doesn't have instance stores. For this case and for other cases of instances without ephemeral volumes, continue to [Stop and start a cluster's head node](#).

If your instance has ephemeral drives and its been stopped, the data in the instance store is lost. You can determine if the instance type used for the head node has instance stores from the table found in [Instance store volumes](#).

The following sections describe the limitations in using instances with instance store volumes.

Instance store limitations

The limitations in using AWS ParallelCluster version 2.11 and instance types with an instance store are as follows:

- When ephemeral drives are not encrypted ([encrypted_ephemeral](#) parameter is set to `false` or not set), an AWS ParallelCluster instance isn't able to boot after an instance stop. This is because information on old non-existent ephemerals is written into `fstab` and the OS tries to mount non-existent storage.
- When ephemeral drives are encrypted ([encrypted_ephemeral](#) parameter is set to `true`), an AWS ParallelCluster instance can be started after a stop but the new ephemeral drives aren't setup, mounted, or available.
- When ephemeral drives are encrypted, an AWS ParallelCluster instance can be rebooted but old ephemeral drives (which survive the instance reboot) can't be accessed because the encryption key is created in the memory that's lost with the reboot.

The only supported case is the instance reboot, when ephemeral drives aren't encrypted. This is because the drive survives the reboot and is mounted back because of the entry written in `fstab`.

Instance store limitations workarounds

First, save your data. To check if you have data that needs to be preserved, view the content in the [ephemeral_dir](#) folder (`/scratch` by default). You can transfer the data to the root volume or the shared storage systems attached to the cluster, such as Amazon FSx, Amazon EFS, or Amazon EBS. Note that the data transfer to remote storage can incur additional costs.

The root cause of the limitations is in the logic that AWS ParallelCluster uses to format and mount instance store volumes. The logic adds an entry to `/etc/fstab` of the form:

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

`${ephemeral_dir}` is the value of the [ephemeral_dir](#) parameter from the `pcluster` configuration file (defaults to `/scratch`).

This line is added so that if or when a node is rebooted, the instance store volumes are re-mounted automatically. This is desirable because data in ephemeral drives persists through reboot. However,

data on the ephemeral drives doesn't persist through a start or stop cycle. This means they are formatted and mounted with no data.

The only supported case is the instance reboot when ephemeral drives aren't encrypted. This is because the drive survives the reboot and is mounted back because it's written in `fstab`.

To preserve the data in all other cases, you must remove the logical volume entry before stopping the instance. For example, remove `/dev/vg.01/lv_ephemeral` from `/etc/fstab` before stopping the instance. After doing this, you start the instance without mounting the ephemeral volumes. However, the instance store mount again won't be available after the stop or start of the instance.

After saving your data and then removing the `fstab` entry, continue to the next section.

Stop and start a cluster's head node

Note

Starting with AWS ParallelCluster version 2.11, head node stop and start is only supported if the instance type doesn't have an instance store.

1. Verify there aren't any running jobs in the cluster.

When using a Slurm scheduler:

- If the `sbatch --no-requeue` option isn't specified, running jobs are requeued.
- If the `--no-requeue` option is specified, running jobs fail.

2. Request a cluster compute fleet stop:

```
$ 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. Wait until the compute fleet status is STOPPED:

```
$ pcluster status cluster-name
```

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

4. For manual updates with an OS reboot or instance restart, you can use the AWS Management Console or AWS CLI. The following is an example of using the AWS CLI.

```
$ 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. Start the compute fleet of the cluster:

```
$ 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 commands

`pcluster` and `pcluster-config` are the AWS ParallelCluster CLI commands. You use `pcluster` to launch and manage HPC clusters in the AWS Cloud and `pcluster-config` to update your configuration.

To use `pcluster`, you must have an IAM role with the [permissions](#) required to run it.

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

Topics

- [pcluster](#)
- [pcluster-config](#)

pcluster

`pcluster` is the primary AWS ParallelCluster CLI command. You use `pcluster` to launch and manage HPC clusters in the AWS Cloud.

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

Arguments

`pcluster` *command*

Possible choices: [configure](#), [create](#), [createami](#), [dcw](#), [delete](#), [instances](#), [list](#), [ssh](#), [start](#), [status](#), [stop](#), [update](#), [version](#)

Sub-commands:

Topics

- [pcluster configure](#)
- [pcluster create](#)
- [pcluster createami](#)
- [pcluster dcw](#)
- [pcluster delete](#)
- [pcluster instances](#)
- [pcluster list](#)
- [pcluster ssh](#)
- [pcluster start](#)
- [pcluster status](#)
- [pcluster stop](#)
- [pcluster update](#)
- [pcluster version](#)

pcluster configure

Begins an AWS ParallelCluster configuration. For more information, see [Configuring AWS ParallelCluster](#).

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

Named arguments

-h, --help

Shows the help text for `pcluster configure`.

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

Specifies the full path of the alternative configuration file to use.

Defaults to `~/.parallelcluster/config`.

For more information, see [Configuring AWS ParallelCluster](#).

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

Specifies the AWS Region to use. If this is specified, the configuration skips AWS Region detection.

To delete the network resources in the VPC, you can delete the CloudFormation networking stack. The stack name starts with "parallelclusternetworking-" and contains the creation time in "YYYYMMDDHHMMSS" format. You can list the stacks using the [list-stacks](#) command.

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

The stack can be deleted using the [delete-stack](#) command.

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

The VPC that [pcluster configure](#) creates for you is not created in the CloudFormation networking stack. You can delete that VPC manually in the console or by using the AWS CLI.

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

pcluster create

Creates a new cluster.

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

Positional arguments

cluster_name

Defines the name of the cluster. The AWS CloudFormation stack name is parallelcluster-*cluster_name*.

Named arguments

-h, --help

Shows the help text for `pcluster create`.

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

Specifies the alternative configuration file to use.

Defaults to `~/.parallelcluster/config`.

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

Specifies the AWS Region to use. The priority order used to select the AWS Region for a new cluster is as follows:

1. `-r` or `--region` parameter to [pcluster create](#).
2. `AWS_DEFAULT_REGION` environment variable.
3. `aws_region_name` setting in `[aws]` section of AWS ParallelCluster config file (default location is `~/.parallelcluster/config`.) This is the location updated by the [pcluster configure](#) command.
4. `region` setting in `[default]` section of AWS CLI config file (`~/.aws/config`.)

-nw, --nowait

Indicates not to wait for stack events after running a stack command.

Defaults to `False`.

-nr, --norollback

Disables stack rollback on error.

Defaults to `False`.

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

Specifies a URL for the custom AWS CloudFormation template if it was used when created.

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

Indicates the cluster template to use.

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

Adds extra parameters to stack create.

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

Specifies additional tags to add to the stack.

When the command is called and begins polling for the status of that call, it's safe to use "Ctrl-C" to exit. You can return to viewing the current status by calling `pcluster status mycluster`.

Examples using AWS ParallelCluster version 2.11.7:

```
$ 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) Creates a custom AMI to use with 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 ]
```

Required dependencies

In addition to the AWS ParallelCluster CLI, the following dependency is required to run `pcluster createami`:

- **Packer:** Download the latest version from <https://developer.hashicorp.com/packer/downloads>.

Note

Before AWS ParallelCluster version 2.8.0, [Berkshelf](#) (installed using `gem install berkshelf`) was required to use `pcluster createami`.

Named arguments

-h, --help

Shows the help text for `pcluster createami`.

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

Specifies the base AMI to use for building the AWS ParallelCluster AMI.

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

Specifies the OS of the base AMI. Valid options are: `alinux2`, `ubuntu1804`, `ubuntu2004`, and `centos7`.

Note

OS support changes in different AWS ParallelCluster versions:

- Support for `centos8` was removed in AWS ParallelCluster version 2.10.4.
- Support for `centos8` was added, and support for `centos6` was removed in AWS ParallelCluster version 2.10.0.
- Support for `alinux2` was added in AWS ParallelCluster version 2.6.0.
- Support for `ubuntu1804` was added in AWS ParallelCluster version 2.5.0.

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

Specifies the instance type to use to create the AMI.

Defaults to `t2.xlarge`.

Note

Support for the `--instance-type` argument was added in AWS ParallelCluster version 2.4.1.

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

Specifies the prefix name of the resulting AWS ParallelCluster AMI.

Defaults to `custom-ami-`.

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

Specifies the cookbook to use to build the AWS ParallelCluster AMI.

--post-install *POST_INSTALL_SCRIPT*

Specifies the path to the post-install script. Paths must use a `s3://`, `https://`, or `file://` URL scheme. Examples include the following:

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

 Note

Support for the `--post-install` argument was added in AWS ParallelCluster version 2.10.0.

--no-public-ip

Do not associate a public IP address to the instance used to create the AMI. By default, a public IP address is associated with the instance.

 Note

Support for the `--no-public-ip` argument was added in AWS ParallelCluster version 2.5.0.

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

Specifies the alternative configuration file to use.

Defaults to `~/.parallelcluster/config`.

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

Specifies the [\[cluster\] section](#) of the *CONFIG_FILE* to use to retrieve the VPC and subnet settings.

Note

Support for the `--cluster-template` argument was added in AWS ParallelCluster version 2.4.0.

--vpc-id *VPC_ID*

Specifies the ID of the VPC to use to build the AWS ParallelCluster AMI.

Note

Support for the `--vpc-id` argument was added in AWS ParallelCluster version 2.5.0.

--subnet-id *SUBNET_ID*

Specifies the ID of the subnet to use to build the AWS ParallelCluster AMI.

Note

Support for the `--vpc-id` argument was added in AWS ParallelCluster version 2.5.0.

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

Specifies the AWS Region to use. Defaults to the AWS Region specified by using the [pcluster configure](#) command.

pcluster dcv

Interacts with the Amazon DCV server running on the head node.

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

pcluster dcv *command*

Possible choices: [connect](#)

Note

OS support changes for the `pcluster dcv` command in different AWS ParallelCluster versions:

- Support for the `pcluster dcv` command on centos8 was added in AWS ParallelCluster version 2.10.0.
- Support for the `pcluster dcv` command on AWS Graviton-based instances was added in AWS ParallelCluster version 2.9.0.
- Support for the `pcluster dcv` command on ubuntu1804 was added in AWS ParallelCluster version 2.6.0.
- Support for the `pcluster dcv` command on centos7 was added in AWS ParallelCluster version 2.5.0.

Named arguments

`-h, --help`

Shows the help text for `pcluster dcv`.

Sub-commands

`pcluster dcv connect`

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

⚠ Important

The URL expires 30 seconds after it's issued. If the connection isn't made before the URL expires, run `pcluster dcv connect` again to generate a new URL.

Positional arguments

cluster_name

Specifies the name of the cluster to connect to.

Named arguments

-h, --help

Shows the help text for `pcluster dcv connect`.

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

Key path of the SSH key to use for the connection.

The key must be the one specified at cluster creation time in the [key_name](#) configuration parameter. This argument is optional, but if it's not specified, then the key must be available by default for the SSH client. For example, add it to the ssh-agent with `ssh-add`.

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

Specifies the AWS Region to use. Defaults to the AWS Region specified by using the [pcluster configure](#) command.

-s, --show-url

Displays a one-time URL for connecting to the Amazon DCV session. The default browser isn't opened when this option is specified.

Note

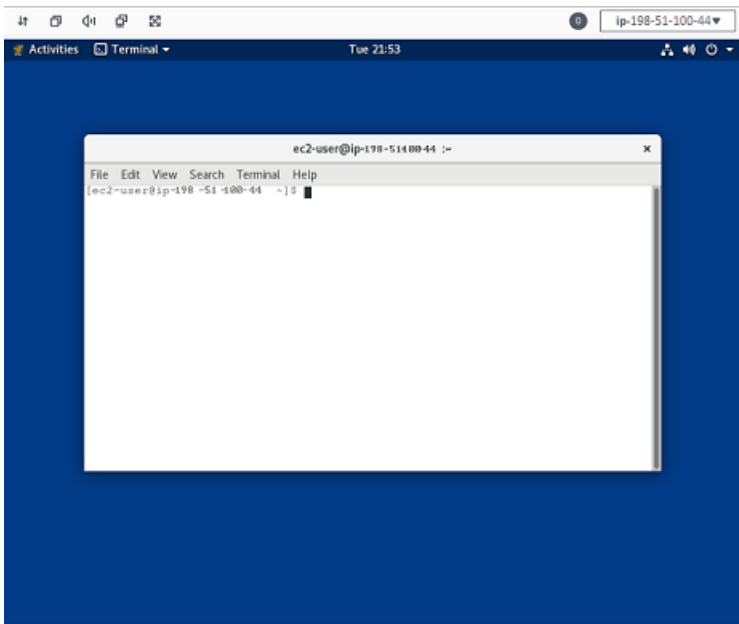
Support for the `--show-url` argument was added in AWS ParallelCluster version 2.5.1.

Example using AWS ParallelCluster version 2.11.7:

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

Opens the default browser to connect to the Amazon DCV session running on the head node.

A new Amazon DCV session is created if one isn't already started.



pcluster delete

Deletes a cluster.

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

Positional arguments

cluster_name

Specifies the name of the cluster to delete.

Named arguments

-h, --help

Shows the help text for `pcluster delete`.

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

Specifies the alternative configuration file to use.

Defaults to `~/.parallelcluster/config`.

--keep-logs

Keep the CloudWatch Logs data after deleting the cluster. The log group remains until you delete it manually, but the log events expire based on the [retention_days](#) setting. The setting defaults to 14 days.

Note

Support for the **--keep-logs** argument was added in AWS ParallelCluster version 2.6.0.

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

Specifies the AWS Region to use. Defaults to the AWS Region specified by using the [pcluster configure](#) command.

When the command is called and begins polling for the status of that call, it's safe to use "Ctrl-C" to exit. You can return to viewing the current status by calling `pcluster status mycluster`.

Example using AWS ParallelCluster version 2.11.7:

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

To delete the network resources in the VPC, you can delete the CloudFormation networking stack. The stack name starts with "parallelclusternetworking-" and contains the creation time in "YYYYMMDDHHMMSS" format. You can list the stacks using the [list-stacks](#) command.

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

The stack can be deleted using the [delete-stack](#) command.

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

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

The VPC that [pcluster configure](#) creates for you is not created in the CloudFormation networking stack. You can delete that VPC manually in the console or by using the AWS CLI.

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

pcluster instances

Displays a list of all instances in a cluster.

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

Positional arguments

cluster_name

Displays the instances for the cluster with the provided name.

Named arguments

-h, --help

Shows the help text for `pcluster instances`.

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

Specifies the alternative configuration file to use.

Defaults to `~/.parallelcluster/config`.

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

Specifies the AWS Region to use. Defaults to the AWS Region specified by using the [pcluster configure](#) command.

Example using AWS ParallelCluster version 2.11.7:

```
$ pcluster instances -c path/to/config -r us-east-1 mycluster  
MasterServer      i-1234567890abcdef0  
ComputeFleet      i-abcdef01234567890
```

pcluster list

Displays a list of stacks that are associated with AWS ParallelCluster.

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

Named arguments

-h, --help

Shows the help text for `pcluster list`.

--color

Displays the cluster status in color.

Defaults to False.

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

Specifies the alternative configuration file to use.

Defaults to `c`.

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

Specifies the AWS Region to use. Defaults to the AWS Region specified by using the [pcluster configure](#) command.

Lists the name of any AWS CloudFormation stacks named `parallelcluster-*`.

Example using AWS ParallelCluster version 2.11.7:

```
$ 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

Runs an `ssh` command with the user name and IP address of the cluster pre-populated. Arbitrary arguments are appended to the end of the `ssh` command. This command can be customized in the aliases section of the configuration file.

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

Positional arguments

cluster_name

Specifies the name of the cluster to connect to.

Named arguments

-h, --help

Shows the help text for `pcluster ssh`.

-d, --dryrun

Prints the command that would be run and exits.

Defaults to False.

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

Specifies the AWS Region to use. Defaults to the Region specified by using the [pcluster configure](#) command.

Examples using AWS ParallelCluster version 2.11.7:

```
$ 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
```

Runs `ssh` command with the user name and IP address of the cluster pre-populated:

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

The `ssh` command is defined in the global configuration file under the [\[aliases\] section](#). It can be customized as follows.

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

Variables substituted:

CFN_USER

The user name for the [base_os](#) that is selected.

MASTER_IP

The IP address of the head node.

ARGS

Optional arguments to pass to the ssh command.

pcluster start

Starts the compute fleet for a cluster that has been stopped.

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

Positional arguments

cluster_name

Starts the compute fleet of the provided cluster name.

Named arguments

-h, --help

Shows the help text for pcluster start.

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

Specifies the alternative configuration file to use.

Defaults to ~/.parallelcluster/config.

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

Specifies the AWS Region to use. Defaults to the AWS Region specified by using the [pcluster configure](#) command.

Example using AWS ParallelCluster version 2.11.7:

```
$ 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
```

This command sets the Auto Scaling Group parameters to one of the following:

- The initial configuration values (max_queue_size and initial_queue_size) from the template that was used to create the cluster.
- The configuration values that were used to update the cluster since it was first created.

pcluster status

Pulls the current status of the cluster.

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

Positional arguments

cluster_name

Shows the status of the cluster with the provided name.

Named arguments

-h, --help

Shows the help text for pcluster status.

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

Specifies the alternative configuration file to use.

Defaults to `~/.parallelcluster/config`.

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

Specifies the AWS Region to use. Defaults to the AWS Region specified by using the [pcluster configure](#) command.

-nw, --nowait

Indicates not to wait for stack events after processing a stack command.

Defaults to `False`.

Example using AWS ParallelCluster version 2.11.7:

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

pcluster stop

Stops the compute fleet, leaving the head node running.

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

Positional arguments

cluster_name

Stops the compute fleet of the provided cluster name.

Example using AWS ParallelCluster version 2.11.7:

Named arguments

-h, --help

Shows the help text for `pcluster stop`.

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

Specifies the alternative configuration file to use.

Defaults to `~/.parallelcluster/config`.

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

Specifies the AWS Region to use. Defaults to the AWS Region specified by using the [pcluster configure](#) command.

```
$ 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

Sets the Auto Scaling group parameters to min/max/desired = 0/0/0, and terminates the compute fleet. The head remains running. To terminate all EC2 resources and avoid EC2 charges, consider deleting the cluster.

pcluster update

Analyzes the configuration file to determine if the cluster can be safely updated. If the analysis determines the cluster can be updated, you are prompted to confirm the change. If the analysis shows the cluster can't be updated, the configuration settings that are the source of the conflicts are enumerated with details. For more information, see [Using pcluster update](#).

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

Positional arguments

cluster_name

Specifies the name of the cluster to update.

Named arguments

-h, --help

Shows the help text for pcluster update.

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

Specifies the alternative configuration file to use.

Defaults to `~/.parallelcluster/config`.

--force

Enables an update even if one or more settings has a blocking change or if an outstanding action is required (such as stopping the compute fleet) before the update can proceed. This shouldn't be combined with the `--yes` argument.

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

Specifies the AWS Region to use. Defaults to the AWS Region specified by using the [pcluster configure](#) command.

-nr, --norollback

Disables AWS CloudFormation stack rollback on error.

Defaults to False.

-nw, --nowait

Indicates not to wait for stack events after processing a stack command.

Defaults to False.

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

Specifies the section of the cluster template to use.

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

Adds extra parameters to a stack update.

-rd, --reset-desired

Resets the current capacity of an Auto Scaling Group to the initial configuration values.

Defaults to False.

--yes

Automatically assumes that the answer to all prompts is yes. This should not be combined with the `--force` argument.

```
$ 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

```

When the command is called and begins polling for the status of that call, it's safe to use "Ctrl-C" to exit. You can return to viewing the current status by calling `pcluster status mycluster`.

pcluster version

Displays the AWS ParallelCluster version.

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

For command-specific flags, run: `pcluster [command] --help`.

Named arguments

-h, --help

Shows the help text for `pcluster version`.

When the command is called and begins polling for the status of that call, it's safe to use "Ctrl-C" to exit. You can return to viewing the current status by calling `pcluster status mycluster`.

```

$ pcluster version
2.11.7

```

pcluster-config

Updates the AWS ParallelCluster configuration file.

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

For command-specific flags, run: `pcluster-config [command] -h`.

Named arguments

-h, --help

Shows the help text for `pcluster-config`.

Note

The `pcluster-config` command was added in AWS ParallelCluster version 2.9.0.

Sub-commands

pcluster-config convert

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

Named arguments

-h, --help

Shows the help text for `pcluster-config convert`.

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

Specifies the path of the configuration file to read.

Defaults to `~/.parallelcluster/config`.

For more information, see [Configuring AWS ParallelCluster](#).

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

Indicates the [\[cluster\] section](#) to use. If this argument is not specified, `pcluster-config convert` will use the [cluster_template](#) setting in the [\[global\] section](#). If that isn't specified, then the `[cluster default]` section is used.

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

Specifies the path of the converted configuration file to be written. By default, the output is written to STDOUT.

Example:

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

Converts the cluster configuration specified in the [cluster alpha] section of ~/.parallelcluster/config, writing the converted configuration file to ~/.parallelcluster/multiinstance.

Configuration

By default, AWS ParallelCluster uses the `~/.parallelcluster/config` file for all configuration parameters. You can specify a custom configuration file by using the `-c` or `--config` command line option or the `AWS_PCLUSTER_CONFIG_FILE` environment variable.

An example configuration file is installed with AWS ParallelCluster in the Python directory at `site-packages/aws-parallelcluster/examples/config`. The example configuration file is also available on GitHub, at <https://github.com/aws/aws-parallelcluster/blob/v2.11.9/cli/src/pcluster/examples/config>.

Current AWS ParallelCluster 2 version: 2.11.9.

Topics

- [Layout](#)
- [\[global\] section](#)
- [\[aws\] section](#)
- [\[aliases\] section](#)
- [\[cluster\] section](#)
- [\[compute_resource\] section](#)
- [\[cw_log\] section](#)
- [\[dashboard\] section](#)
- [\[dcv\] section](#)
- [\[ebs\] section](#)
- [\[efs\] section](#)
- [\[fsx\] section](#)
- [\[queue\] section](#)
- [\[raid\] section](#)
- [\[scaling\] section](#)
- [\[vpc\] section](#)
- [Examples](#)

Layout

An AWS ParallelCluster configuration is defined in multiple sections.

The following sections are required: [\[global\] section](#) and [\[aws\] section](#).

You also must include at least one [\[cluster\] section](#) and one [\[vpc\] section](#).

A section starts with the section name in brackets, followed by parameters and configuration.

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

[global] section

Specifies global configuration options related to pcluster.

```
[global]
```

Topics

- [cluster_template](#)
- [update_check](#)
- [sanity_check](#)

cluster_template

Defines the name of the `cluster` section that's used for the cluster by default. For additional information about `cluster` sections, see [\[cluster\] section](#). The cluster name must start with a letter, contain no more than 60 characters, and only contain letters, numbers, and hyphens (-).

For example, the following setting specifies that the section that starts `[cluster default]` is used by default.

```
cluster_template = default
```

Update policy: This setting is not analyzed during an update.

update_check

(Optional) Checks for updates to pcluster.

The default value is `true`.

```
update_check = true
```

Update policy: This setting is not analyzed during an update.

sanity_check

(Optional) Attempts to validate the configuration of the resources that are defined in the cluster parameters.

The default value is `true`.

Warning

If `sanity_check` is set to `false`, important checks are skipped. This might cause your configuration to not function as intended.

```
sanity_check = true
```

Note

Before AWS ParallelCluster version 2.5.0, [sanity_check](#) defaulted to `false`.

Update policy: This setting is not analyzed during an update.

[aws] section

(Optional) Used to select the AWS Region.

Cluster creation uses this priority order to select the AWS Region for a new cluster:

1. `-r` or `--region` parameter to [pcluster create](#).

2. AWS_DEFAULT_REGION environment variable.
3. aws_region_name setting in [aws] section of AWS ParallelCluster config file (default location is ~/.parallelcluster/config.) This is the location updated by the [pcluster configure](#) command.
4. region setting in [default] section of AWS CLI config file (~/.aws/config.)

Note

Before AWS ParallelCluster version 2.10.0, these settings were required and applied to all clusters.

To store credentials, you can use the environment, IAM roles for Amazon EC2, or the [AWS CLI](#), rather than saving credentials into the AWS ParallelCluster config file.

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

Update policy: This setting is not analyzed during an update.

[aliases] section

Specifies aliases, and enables you to customize the ssh command.

Note the following default settings:

- CFN_USER is set to the default user name for the OS
- MASTER_IP is set to the IP address of the head node
- ARGS is set to whatever arguments the user provides after *pcluster ssh cluster_name*

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

Update policy: This setting is not analyzed during an update.

[cluster] section

Defines a cluster template that can be used to create a cluster. A config file can contain multiple [cluster] sections.

The same cluster template can be used to create multiple clusters.

The format is [cluster *cluster-template-name*]. The [\[cluster\] section](#) named by the [cluster_template](#) setting in the [\[global\] section](#) is used by default, but can be overridden on the [pcluster](#) command line.

cluster-template-name must start with a letter, contain no more than 30 characters, and only contain letters, numbers, hyphens (-), and underscores (_).

```
[cluster default]
```

Topics

- [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

(Optional) Defines an additional AWS CloudFormation template to launch along with the cluster. This additional template is used for creating resources that are outside of the cluster but are part of the cluster's lifecycle.

The value must be an HTTP URL to a public template, with all parameters provided.

There is no default value.

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

Update policy: If this setting is changed, the update is not allowed.

additional_iam_policies

(Optional) Specifies a list of Amazon Resource Names (ARNs) of IAM policies for Amazon EC2. This list is attached to the root role used in the cluster in addition to the permissions required by AWS ParallelCluster separated by commas. An IAM policy name and its ARN are different. Names can't be used as an argument to `additional_iam_policies`.

If your intent is to add extra policies to the default settings for cluster nodes, we recommend that you pass the additional custom IAM policies with the `additional_iam_policies` setting instead of using the [ec2_iam_role](#) settings to add your specific EC2 policies. This is because `additional_iam_policies` are added to the default permissions that AWS ParallelCluster requires. An existing [ec2_iam_role](#) must include all permissions required. However, because the permissions required often change from release to release as features are added, an existing [ec2_iam_role](#) can become obsolete.

There is no default value.

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

Note

Support for [additional_iam_policies](#) was added in AWS ParallelCluster version 2.5.0.

Update policy: This setting can be changed during an update.

base_os

(Required) Specifies which OS type is used in the cluster.

Available options are:

- alinux2
- centos7
- ubuntu1804
- ubuntu2004

Note

For AWS Graviton-based instances, only alinux2, ubuntu1804, or ubuntu2004 are supported.

Note

Support for centos8 was removed in AWS ParallelCluster version 2.11.4. Support for ubuntu2004 was added and support for alinux and ubuntu1604 was removed in AWS ParallelCluster version 2.11.0. Support for centos8 was added and support for centos6 was removed in AWS ParallelCluster version 2.10.0. Support for alinux2 was added in AWS ParallelCluster version 2.6.0. Support for ubuntu1804 was added, and support for ubuntu1404 was removed in AWS ParallelCluster version 2.5.0.

Other than the specific AWS Regions mentioned in the following table that don't support centos7. All other AWS commercial Regions support all of the following operating systems.

Partition (AWS Regions)	alinux2	centos7	ubuntu1804 and ubuntu2004
Commercial (All AWS Regions not specifically mentioned)	True	True	True
AWS GovCloud (US-East) (us-gov-east-1)	True	False	True
AWS GovCloud (US-West) (us-gov-west-1)	True	False	True
China (Beijing) (cn-north-1)	True	False	True
China (Ningxia) (cn-northwest-1)	True	False	True

Note

The [base_os](#) parameter also determines the user name that's used to log into the cluster.

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

Note

Before AWS ParallelCluster version 2.7.0, the [base_os](#) parameter was optional, and the default was `alinux`. Starting with AWS ParallelCluster version 2.7.0, the [base_os](#) parameter is required.

Note

If the [scheduler](#) parameter is awsbatch, only alinux2 is supported.

```
base_os = alinux2
```

Update policy: If this setting is changed, the update is not allowed.

cluster_resource_bucket

(Optional) Specifies the name of the Amazon S3 bucket that's used to host resources that are generated when the cluster is created. The bucket must have versioning enabled. For more information, see [Using versioning](#) in the *Amazon Simple Storage Service User Guide*. This bucket can be used for multiple clusters. The bucket must be in the same Region as the cluster.

If this parameter isn't specified, a new bucket is created when the cluster is created. The new bucket has the name of parallelcluster-*random_string*. In this name, *random_string* is a random string of alphanumeric characters. All cluster resources are stored in this bucket in a path with the form *bucket_name/resource_directory*. *resource_directory* has the form *stack_name-random_string*, where *stack_name* is the name of one of the CloudFormation stacks used by AWS ParallelCluster. The value of *bucket_name* can be found in the ResourcesS3Bucket value in the output of the parallelcluster-*clustername* stack. The value of *resource_directory* can be found in the value of the ArtifactS3RootDirectory output from the same stack.

The default value is parallelcluster-*random_string*.

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

Note

Support for [cluster_resource_bucket](#) was added in AWS ParallelCluster version 2.10.0.

Update policy: If this setting is changed, the update is not allowed. Updating this setting cannot be forced.

cluster_type

(Optional) Defines the type of cluster to launch. If the [queue_settings](#) setting is defined, then this setting must be replaced by the [compute_type](#) settings in the [\[queue\] sections](#).

Valid options are: ondemand, and spot.

The default value is ondemand.

For more information about Spot Instances, see [Working with Spot Instances](#).

Note

Using Spot Instances requires that the AWSServiceRoleForEC2Spot service-linked role exist in your account. To create this role in your account using the AWS CLI, run the following command:

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

For more information, see [Service-linked role for Spot Instance requests](#) in the *Amazon EC2 User Guide*.

```
cluster_type = ondemand
```

Update policy: The compute fleet must be stopped for this setting to be changed for an update.

compute_instance_type

(Optional) Defines the Amazon EC2 instance type that's used for the cluster compute nodes. The architecture of the instance type must be the same as the architecture used for the [master_instance_type](#) setting. If the [queue_settings](#) setting is defined, then this setting must be replaced by the [instance_type](#) settings in the [\[compute_resource\] sections](#).

If you're using the awsbatch scheduler, see the Compute Environments creation in the AWS Batch UI for a list of supported instance types.

Defaults to t2.micro, optimal when the scheduler is awsbatch.

```
compute_instance_type = t2.micro
```

Note

Support for AWS Graviton-based instances (including A1 and C6g instances) was added in AWS ParallelCluster version 2.8.0.

Update policy: The compute fleet must be stopped for this setting to be changed for an update.

compute_root_volume_size

(Optional) Specifies the ComputeFleet root volume size in gibibytes (GiB). The AMI must support growroot.

The default value is 35.

Note

For AWS ParallelCluster versions between 2.5.0 and 2.10.4, the default was 25. Before AWS ParallelCluster version 2.5.0, the default was 20.

```
compute_root_volume_size = 35
```

Update policy: The compute fleet must be stopped for this setting to be changed for an update.

custom_ami

(Optional) Specifies the ID of a custom AMI to use for the head and compute nodes instead of the default [published AMIs](#). For more information, see [Modify an AMI](#) or [Build a Custom AWS ParallelCluster AMI](#).

There is no default value.

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

If the custom AMI requires additional permissions for its launch, these permissions must be added to both the user and head node policies.

For example, if a custom AMI has an encrypted snapshot associated with it, the following additional policies are required in both the user and head node policies:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:DescribeKey",
        "kms:ReEncrypt*",
        "kms:CreateGrant",
        "kms:Decrypt"
      ],
      "Resource": [
        "arn:aws:kms:us-east-1:111122223333:key/<AWS_KMS_KEY_ID>"
      ]
    }
  ]
}
```

Update policy: If this setting is changed, the update is not allowed.

cw_log_settings

(Optional) Identifies the [cw_log] section with the CloudWatch Logs configuration. The section name must start with a letter, contain no more than 30 characters, and only contain letters, numbers, hyphens (-), and underscores (_).

For more information, see the [\[cw_log\] section](#), [Amazon CloudWatch dashboard](#), and [Integration with Amazon CloudWatch Logs](#).

For example, the following setting specifies that the section that starts [cw_log custom-cw] is used for the CloudWatch Logs configuration.

```
cw_log_settings = custom-cw
```

Note

Support for [cw_log_settings](#) was added in AWS ParallelCluster version 2.6.0.

Update policy: If this setting is changed, the update is not allowed.

dashboard_settings

(Optional) Identifies the [dashboard] section with the CloudWatch dashboard configuration. The section name must start with a letter, contain no more than 30 characters, and only contain letters, numbers, hyphens (-), and underscores (_).

For more information, see the [\[dashboard\] section](#).

For example, the following setting specifies that the section that starts [dashboard custom-dashboard] is used for the CloudWatch dashboard configuration.

```
dashboard_settings = custom-dashboard
```

Note

Support for [dashboard_settings](#) was added in AWS ParallelCluster version 2.10.0.

Update policy: This setting can be changed during an update.

dcv_settings

(Optional) Identifies the [dcv] section with the Amazon DCV configuration. The section name must start with a letter, contain no more than 30 characters, and only contain letters, numbers, hyphens (-), and underscores (_).

For more information, see the [\[dcv\] section](#).

For example, the following setting specifies that the section that starts [dcv custom-dcv] is used for the Amazon DCV configuration.

```
dcv_settings = custom-dcv
```

Note

On AWS Graviton-based instances, Amazon DCV is only supported on `alinux2`.

Note

Support for [dcv_settings](#) was added in AWS ParallelCluster version 2.5.0.

Update policy: If this setting is changed, the update is not allowed.

desired_vcpus

(Optional) Specifies the desired number of vCPUs in the compute environment. Used only if the scheduler is awsbatch.

The default value is 4.

```
desired_vcpus = 4
```

Update policy: This setting is not analyzed during an update.

disable_cluster_dns

(Optional) Specifies if the DNS entries for the cluster shouldn't be created. By default, AWS ParallelCluster creates a Route 53 hosted zone. If `disable_cluster_dns` is set to `true`, the hosted zone isn't created.

The default value is `false`.

```
disable_cluster_dns = true
```

Warning

A name resolution system is required for the cluster to operate properly. If `disable_cluster_dns` is set to `true`, an additional name resolution system must also be provided.

Important

[disable_cluster_dns](#) = `true` is only supported if the [queue_settings](#) setting is specified.

Note

Support for [disable_cluster_dns](#) was added in AWS ParallelCluster version 2.9.1.

Update policy: If this setting is changed, the update is not allowed.

disable_hyperthreading

(Optional) Disables hyperthreading on the head and compute nodes. Not all instance types can disable hyperthreading. For a list of instance types that support disabling hyperthreading, see [CPU cores and threads for each CPU core for each instance type](#) in the *Amazon EC2 User Guide*. If the [queue_settings](#) setting is defined, either this setting can be defined, or the [disable_hyperthreading](#) settings in the [\[queue\] sections](#) can be defined.

The default value is false.

```
disable_hyperthreading = true
```

Note

[disable_hyperthreading](#) only affects the head node when [scheduler](#) = awsbatch.

Note

Support for [disable_hyperthreading](#) was added in AWS ParallelCluster version 2.5.0.

Update policy: If this setting is changed, the update is not allowed.

ebs_settings

(Optional) Identifies the [ebs] sections with the Amazon EBS volumes that are mounted on the head node. When using multiple Amazon EBS volumes, enter these parameters in a list with each one separated by a comma. The section name must start with a letter, contain no more than 30 characters, and only contain letters, numbers, hyphens (-), and underscores (_).

Up to five (5) additional Amazon EBS volumes are supported.

For more information, see the [\[ebs\] section](#).

For example, the following setting specifies that the sections that start [ebs custom1] and [ebs custom2] are used for the Amazon EBS volumes.

```
ebs_settings = custom1, custom2
```

Update policy: If this setting is changed, the update is not allowed.

ec2_iam_role

(Optional) Defines the name of an existing IAM role for Amazon EC2 that's attached to all instances in the cluster. An IAM role name and its Amazon Resource Name (ARN) are different. ARNs can't be used as an argument to ec2_iam_role.

If this option is specified, the [additional_iam_policies](#) setting is ignored. If your intent is to add extra policies to the default settings for cluster nodes, we recommend that you pass the additional custom IAM policies with the [additional_iam_policies](#) setting instead of using the ec2_iam_role settings.

If this option isn't specified, the default AWS ParallelCluster IAM role for Amazon EC2 is used. For more information, see [AWS Identity and Access Management roles in AWS ParallelCluster](#).

There is no default value.

```
ec2_iam_role = ParallelClusterInstanceRole
```

Update policy: If this setting is changed, the update is not allowed.

efs_settings

(Optional) Specifies settings related to the Amazon EFS file system. The section name must start with a letter, contain no more than 30 characters, and only contain letters, numbers, hyphens (-), and underscores (_).

For more information, see the [\[efs\] section](#).

For example, the following setting specifies that the section that starts [efs customfs] is used for the Amazon EFS file system configuration.

```
efs_settings = customfs
```

Update policy: If this setting is changed, the update is not allowed.

enable_efa

(Optional) If present, specifies that Elastic Fabric Adapter (EFA) is enabled for the compute nodes. To view the list of EC2 instances that support EFA, see [Supported instance types](#) in the *Amazon EC2 User Guide for Linux Instances*. For more information, see [Elastic Fabric Adapter](#). If the [queue_settings](#) setting is defined, either this setting can be defined, or the [enable_efa](#) settings in the [\[queue\] section](#) can be defined. A cluster placement group should be used to minimize latencies between instances. For more information, see [placement](#) and [placement_group](#).

```
enable_efa = compute
```

Note

Support for EFA on Arm-based Graviton2 instances was added in AWS ParallelCluster version 2.10.1.

Update policy: If this setting is changed, the update is not allowed.

enable_efa_gdr

(Optional) Starting with AWS ParallelCluster version 2.11.3, this setting has no effect. Elastic Fabric Adapter (EFA) support for GPUDirect RDMA (remote direct memory access) is always enabled if it's supported by both the instance type and the operating system.

Note

AWS ParallelCluster version 2.10.0 through 2.11.2: If `compute`, specifies that Elastic Fabric Adapter (EFA) support for GPUDirect RDMA (remote direct memory access) is enabled for the compute nodes. Setting this setting to `compute` requires that the [enable_efa](#) setting is set to `compute`. EFA support for GPUDirect RDMA is supported by specific instance types (p4d.24xlarge) on specific operating systems ([base_os](#) is `alinux2`, `centos7`,

ubuntu1804, or ubuntu2004). If the [queue_settings](#) setting is defined, either this setting can be defined, or the [enable_efa_gdr](#) settings in the [\[queue\] sections](#) can be defined. A cluster placement group should be used to minimize latencies between instances. For more information, see [placement](#) and [placement_group](#).

```
enable_efa_gdr = compute
```

Note

Support for `enable_efa_gdr` was added in AWS ParallelCluster version 2.10.0.

Update policy: The compute fleet must be stopped for this setting to be changed for an update.

`enable_intel_hpc_platform`

(Optional) If present, indicates that the [End user license agreement](#) for Intel Parallel Studio is accepted. This causes Intel Parallel Studio to be installed on the head node and shared with the compute nodes. This adds several minutes to the time it takes the head node to bootstrap. The [enable_intel_hpc_platform](#) setting is only supported on CentOS 7 ([base_os](#) = `centos7`).

The default value is `false`.

```
enable_intel_hpc_platform = true
```

Note

The [enable_intel_hpc_platform](#) parameter isn't compatible with AWS Graviton-based instances.

Note

Support for [enable_intel_hpc_platform](#) was added in AWS ParallelCluster version 2.5.0.

Update policy: If this setting is changed, the update is not allowed.

encrypted_ephemeral

(Optional) Encrypts the ephemeral instance store volumes with non-recoverable in-memory keys, using LUKS (Linux Unified Key Setup).

For more information, see <https://gitlab.com/cryptsetup/cryptsetup/blob/master/README.md>.

The default value is false.

```
encrypted_ephemeral = true
```

Update policy: If this setting is changed, the update is not allowed.

ephemeral_dir

(Optional) Defines the path where instance store volumes are mounted if they are used.

The default value is /scratch.

```
ephemeral_dir = /scratch
```

Update policy: If this setting is changed, the update is not allowed.

extra_json

(Optional) Defines the extra JSON that's merged into the Chef dna.json. For more information, see [Building a Custom AWS ParallelCluster AMI](#).

The default value is {}.

```
extra_json = {}
```

Note

Starting with AWS ParallelCluster version 2.6.1, most of the install recipes are skipped by default when launching nodes to improve start up times. To run all of the install recipes for better backwards compatibility at the expense of startup times, add

"skip_install_recipes" : "no" to the cluster key in the [extra_json](#) setting. For example:

```
extra_json = { "cluster" : { "skip_install_recipes" : "no" } }
```

Update policy: The compute fleet must be stopped for this setting to be changed for an update.

fsx_settings

(Optional) Specifies the section that defines the FSx for Lustre configuration. The section name must start with a letter, contain no more than 30 characters, and only contain letters, numbers, hyphens (-), and underscores (_).

For more information, see the [\[fsx\] section](#).

For example, the following setting specifies that the section that starts [fsx fs] is used for the FSx for Lustre configuration.

```
fsx_settings = fs
```

Update policy: If this setting is changed, the update is not allowed.

iam_lambda_role

(Optional) Defines the name of an existing AWS Lambda execution role. This role is attached to all Lambda functions in the cluster. For more information, see [AWS Lambda execution role](#) in the *AWS Lambda Developer Guide*.

Note

Starting with version 2.11.5, AWS ParallelCluster doesn't support the use of SGE or Torque schedulers.

An IAM role name and its Amazon Resource Name (ARN) are different. ARNs can't be used as an argument to `iam_lambda_role`. If both [ec2_iam_role](#) and `iam_lambda_role` are defined, and the [scheduler](#) is sge, slurm, or torque, then there will be no roles created. If

the [scheduler](#) is awsbatch, then there will be roles created during [pcluster start](#). For example policies, see [ParallelClusterLambdaPolicy using SGE, Slurm, or Torque](#) and [ParallelClusterLambdaPolicy using awsbatch](#).

There is no default value.

```
iam_lambda_role = ParallelClusterLambdaRole
```

Note

Support for `iam_lambda_role` was added in AWS ParallelCluster version 2.10.1.

Update policy: This setting can be changed during an update.

initial_queue_size

(Optional) Sets the initial number of Amazon EC2 instances to launch as compute nodes in the cluster. If the [queue_settings](#) setting is defined, then this setting must be removed and replaced by the [initial_count](#) settings in the [\[compute_resource\] sections](#).

Note

Starting with version 2.11.5, AWS ParallelCluster doesn't support the use of SGE or Torque schedulers.

This setting is applicable only for traditional schedulers (SGE, Slurm, and Torque). If the [maintain_initial_size](#) setting is true, then the [initial_queue_size](#) setting must be at least one (1).

If the scheduler is awsbatch, use [min_vcpus](#) instead.

Defaults to 2.

```
initial_queue_size = 2
```

Update policy: This setting can be changed during an update.

key_name

(Optional) Names an existing Amazon EC2 key pair with which to enable SSH access to the instances.

```
key_name = mykey
```

Note

Before AWS ParallelCluster version 2.11.0, `key_name` was a required setting.

Update policy: If this setting is changed, the update is not allowed.

maintain_initial_size

Note

Starting with version 2.11.5, AWS ParallelCluster doesn't support the use of SGE or Torque schedulers.

(Optional) Maintains the initial size of the Auto Scaling group for traditional schedulers (SGE, Slurm, and Torque).

If the scheduler is `awsbatch`, use [desired_vcpus](#) instead.

This setting is a Boolean flag. If set to `true`, the Auto Scaling group doesn't ever have fewer members than the value of [initial_queue_size](#), and the value of [initial_queue_size](#) must be one (1) or greater. The cluster can still scale up to the value of [max_queue_size](#). If `cluster_type = spot` then the Auto Scaling group can have instances interrupted and the size can drop under [initial_queue_size](#).

If set to `false`, the Auto Scaling group can scale down to zero (0) members to prevent resources from sitting idle when they aren't needed.

If the [queue_settings](#) setting is defined then this setting must be removed and replaced by the [initial_count](#) and [min_count](#) settings in the [\[compute_resource\]](#) sections.

Defaults to false.

```
maintain_initial_size = false
```

Update policy: [This setting can be changed during an update.](#)

master_instance_type

(Optional) Defines the Amazon EC2 instance type that's used for the head node. The architecture of the instance type must be the same as the architecture used for the [compute_instance_type](#) setting.

In AWS Regions that have a Free Tier, defaults to the Free Tier instance type (t2.micro or t3.micro). In AWS Regions that do not have a Free Tier, defaults to t3.micro. For more information about the AWS Free Tier, see [AWS Free Tier FAQs](#).

```
master_instance_type = t2.micro
```

Note

Before AWS ParallelCluster version 2.10.1, defaulted to t2.micro in all AWS Regions. In AWS ParallelCluster version 2.10.0, the p4d.24xlarge wasn't supported for the head node. Support for AWS Graviton-based instances (such as A1 and C6g) was added in AWS ParallelCluster version 2.8.0.

Update policy: [If this setting is changed, the update is not allowed.](#)

master_root_volume_size

(Optional) Specifies the head node root volume size in gibibytes (GiB). The AMI must support growroot.

The default value is 35.

Note

For AWS ParallelCluster versions between 2.5.0 and 2.10.4, the default was 25. Before AWS ParallelCluster version 2.5.0, the default was 20.

```
master_root_volume_size = 35
```

Update policy: If this setting is changed, the update is not allowed.

max_queue_size

(Optional) Sets the maximum number of Amazon EC2 instances that can be launched in the cluster. If the [queue_settings](#) setting is defined, then this setting must be removed and replaced by the [max_count](#) settings in the [\[compute_resource\]](#) sections.

Note

Starting with version 2.11.5, AWS ParallelCluster doesn't support the use of SGE or Torque schedulers.

This setting is applicable only for traditional schedulers (SGE, Slurm, and Torque).

If the scheduler is awsbatch, use [max_vcpus](#) instead.

Defaults to 10.

```
max_queue_size = 10
```

Update policy: This setting can be changed during an update, but the compute fleet should be stopped if the value is reduced. Otherwise, existing nodes may be terminated.

max_vcpus

(Optional) Specifies the maximum number of vCPUs in the compute environment. Used only if the scheduler is awsbatch.

The default value is 20.

```
max_vcpus = 20
```

Update policy: This setting can't be decreased during an update.

min_vcpus

(Optional) Maintains the initial size of the Auto Scaling group for the awsbatch scheduler.

Note

Starting with version 2.11.5, AWS ParallelCluster doesn't support the use of SGE or Torque schedulers.

If the scheduler is SGE, Slurm, or Torque, use [maintain_initial_size](#) instead.

The compute environment never has fewer members than the value of [min_vcpus](#).

Defaults to 0.

```
min_vcpus = 0
```

Update policy: This setting can be changed during an update.

placement

(Optional) Defines the cluster placement group logic, enabling either the whole cluster or only the compute instances to use the cluster placement group.

If the [queue_settings](#) setting is defined, then this setting should be removed and replaced with [placement_group](#) settings for each of the [\[queue\] sections](#). If the same placement group is used for different instance types, it's more likely that the request might fail due to an insufficient capacity error. For more information, see [Insufficient instance capacity](#) in the *Amazon EC2 User Guide*. Multiple queues can only share a placement group if it's created in advance and configured in the [placement_group](#) setting for each queue. If each [\[queue\] sections](#) defines a [placement_group](#) setting, then the head node can't be in the placement group for a queue.

Valid options are `cluster` or `compute`.

This parameter isn't used when the scheduler is awsbatch.

The default value is `compute`.

```
placement = compute
```


Update policy: If this setting is changed, the update is not allowed.

placement_group

(Optional) Defines the cluster placement group. If the [queue_settings](#) setting is defined, then this setting should be removed and replaced by the [placement_group](#) settings in the [\[queue\] sections](#).

Valid options are the following values:

- DYNAMIC
- An existing Amazon EC2 cluster placement group name

When set to DYNAMIC, a unique placement group is created and deleted as part of the cluster stack.

This parameter isn't used when the scheduler is awsbatch.

For more information about placement groups, see [Placement groups](#) in the *Amazon EC2 User Guide*. If the same placement group is used for different instance types, it's more likely that the request might fail due to an insufficient capacity error. For more information, see [Insufficient instance capacity](#) in the *Amazon EC2 User Guide*.

There is no default value.

Not all instance types support cluster placement groups. For example, the default instance type of t3.micro doesn't support cluster placement groups. For information about the list of instance types that support cluster placement groups, see [Cluster placement group rules and limitations](#) in the *Amazon EC2 User Guide*. See [Placement groups and instance launch issues](#) for tips when working with placement groups.

```
placement_group = DYNAMIC
```

Update policy: If this setting is changed, the update is not allowed.

post_install

(Optional) Specifies the URL of a post-install script that's run after all of the node bootstrap actions are complete. For more information, see [Custom Bootstrap Actions](#).

When using `awsbatch` as the scheduler, the post-install script is run only on the head node.

The parameter format can be either `http://hostname/path/to/script.sh` or `s3://bucket-name/path/to/script.sh`.

There is no default value.

```
post_install = s3://<bucket-name>/my-post-install-script.sh
```

Update policy: The compute fleet must be stopped for this setting to be changed for an update.

post_install_args

(Optional) Specifies a quoted list of arguments to pass to the post-install script.

There is no default value.

```
post_install_args = "argument-1 argument-2"
```

Update policy: The compute fleet must be stopped for this setting to be changed for an update.

pre_install

(Optional) Specifies the URL of a pre-install script that's run before any node deployment bootstrap action is started. For more information, see [Custom Bootstrap Actions](#).

When using `awsbatch` as the scheduler, the pre-install script is run only on the head node.

The parameter format can be either `http://hostname/path/to/script.sh` or `s3://bucket-name/path/to/script.sh`.

There is no default value.

```
pre_install = s3://bucket-name/my-pre-install-script.sh
```

Update policy: The compute fleet must be stopped for this setting to be changed for an update.

pre_install_args

(Optional) Specifies a quoted list of arguments to pass to the pre-install script.

There is no default value.

```
pre_install_args = "argument-3 argument-4"
```

Update policy: The compute fleet must be stopped for this setting to be changed for an update.

proxy_server

(Optional) Defines an HTTP or HTTPS proxy server, typically `http://x.x.x.x:8080`.

There is no default value.

```
proxy_server = http://10.11.12.13:8080
```

Update policy: If this setting is changed, the update is not allowed.

queue_settings

(Optional) Specifies that the cluster uses queues instead of a homogeneous compute fleet, and which [\[queue\] sections](#) are used. The first [\[queue\] section](#) listed is the default scheduler queue. The queue section names must start with a lowercase letter, contain no more than 30 characters, and only contain lowercase letters, numbers, and hyphens (-).

Important

[queue_settings](#) is only supported when [scheduler](#) is set to slurm.

The [cluster_type](#), [compute_instance_type](#), [initial_queue_size](#), [maintain_initial_size](#), [max_queue_size](#), [placement](#), [placement_group](#), and [spot_price](#) settings must not be specified. The [disable_hyperthreading](#) and [enable_efa](#) settings can either be specified in the [\[cluster\] section](#) or the [\[queue\] sections](#), but not both.

Up to five (5) [\[queue\] sections](#) are supported.

For more information, see the [\[queue\] section](#).

For example, the following setting specifies that the sections that start `[queue q1]` and `[queue q2]` are used.

```
queue_settings = q1, q2
```

Note

Support for [queue_settings](#) was added in AWS ParallelCluster version 2.9.0.

Update policy: The compute fleet must be stopped for this setting to be changed for an update.

raid_settings

(Optional) Identifies the [raid] section with the Amazon EBS volume RAID configuration. The section name must start with a letter, contain no more than 30 characters, and only contain letters, numbers, hyphens (-), and underscores (_).

For more information, see the [\[raid\] section](#).

For example, the following setting specifies that the section that starts [raid rs] be used for the Auto Scaling configuration.

```
raid_settings = rs
```

Update policy: If this setting is changed, the update is not allowed.

s3_read_resource

(Optional) Specifies an Amazon S3 resource to which AWS ParallelCluster nodes are granted read-only access.

For example, `arn:aws:s3:::my_corporate_bucket*` provides read-only access to the *my_corporate_bucket* bucket and to the objects in the bucket.

See [working with Amazon S3](#) for details on format.

There is no default value.

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

Update policy: This setting can be changed during an update.

s3_read_write_resource

(Optional) Specifies an Amazon S3 resource which AWS ParallelCluster nodes are granted read/write access to.

For example, `arn:aws:s3:::my_corporate_bucket/Development/*` provides read/write access to all objects in the Development folder of the *my_corporate_bucket* bucket.

See [working with Amazon S3](#) for details on format.

There is no default value.

```
s3_read_write_resource = arn:aws:s3:::my_corporate_bucket/*
```

Update policy: This setting can be changed during an update.

scaling_settings

Identifies the [scaling] section with the Auto Scaling configuration. The section name must start with a letter, contain no more than 30 characters, and only contain letters, numbers, hyphens (-), and underscores (_).

For more information, see the [\[scaling\] section](#).

For example, the following setting specifies that the section that starts [scaling custom] is used for the Auto Scaling configuration.

```
scaling_settings = custom
```

Update policy: If this setting is changed, the update is not allowed.

scheduler

(Required) Defines the cluster scheduler.

Valid options are the following values:

awsbatch

AWS Batch

For more information about the awsbatch scheduler, see [networking setup](#) and [AWS Batch \(awsbatch\)](#).

sgc

Note

Starting with version 2.11.5, AWS ParallelCluster doesn't support the use of SGE or Torque schedulers.

Son of Grid Engine (SGE)

slurm

Slurm Workload Manager (Slurm)

torque

Note

Starting with version 2.11.5, AWS ParallelCluster doesn't support the use of SGE or Torque schedulers.

Torque Resource Manager (Torque)

Note

Before AWS ParallelCluster version 2.7.0, the `scheduler` parameter was optional, and the default was `sgc`. Starting with AWS ParallelCluster version 2.7.0, the `scheduler` parameter is required.

```
scheduler = slurm
```

Update policy: If this setting is changed, the update is not allowed.

shared_dir

(Optional) Defines the path where the shared Amazon EBS volume is mounted.

Don't use this option with multiple Amazon EBS volumes. Instead, provide [shared_dir](#) values under each [\[ebs\] section](#).

See the [\[ebs\] section](#) for details on working with multiple Amazon EBS volumes.

The default value is `/shared`.

The following example shows a shared Amazon EBS volume mounted at `/myshared`.

```
shared_dir = myshared
```

Update policy: If this setting is changed, the update is not allowed.

spot_bid_percentage

(Optional) Sets the on-demand percentage used to calculate the maximum Spot price for the ComputeFleet, when `awsbatch` is the scheduler.

If unspecified, the current spot market price is selected, capped at the On-Demand price.

```
spot_bid_percentage = 85
```

Update policy: This setting can be changed during an update.

spot_price

Note

Starting with version 2.11.5, AWS ParallelCluster doesn't support the use of SGE or Torque schedulers.

(Optional) Sets the maximum Spot price for the ComputeFleet on traditional schedulers (SGE, Slurm, and Torque). Used only when the [cluster_type](#) setting is set to `spot`. If you don't specify a value, you are charged the Spot price, capped at the On-Demand price. If the [queue_settings](#) setting is defined, then this setting must be removed and replaced by the [spot_price](#) settings in the [\[compute_resource\] sections](#).

If the scheduler is `awsbatch`, use [spot_bid_percentage](#) instead.

For assistance finding a Spot Instance that meets your needs, see the [Spot Instance advisor](#).

```
spot_price = 1.50
```

Note

In AWS ParallelCluster version 2.5.0, if `cluster_type = spot` but [spot_price](#) isn't specified, the instance launches of the ComputeFleet fail. This was fixed in AWS ParallelCluster version 2.5.1.

Update policy: This setting can be changed during an update.

tags

(Optional) Defines tags to be used by CloudFormation.

If command line tags are specified via `--tags`, they are merged with config tags.

Command line tags overwrite config tags that have the same key.

Tags are JSON formatted. Don't use quotes outside of the curly braces.

For more information, see [CloudFormation resource tags type](#) in the *AWS CloudFormation User Guide*.

```
tags = {"key" : "value", "key2" : "value2"}
```

Update policy: If this setting is changed, the update is not allowed.

Note

The update policy did not support changing the tags setting for AWS ParallelCluster version 2.8.0 through version 2.9.1.

For versions 2.10.0 through version 2.11.7, the listed update policy that supported changing the tags setting isn't accurate. A cluster update when modifying this setting isn't supported.

template_url

(Optional) Defines the path to the AWS CloudFormation template that's used to create the cluster.

Updates use the template that was originally used to create the stack.

Defaults to `https://aws_region_name-aws-parallelcluster.s3.amazonaws.com/templates/aws-parallelcluster-version.cfn.json`.

Warning

This is an advanced parameter. Any change to this setting is done at your own risk.

```
template_url = https://us-east-1-aws-parallelcluster.s3.amazonaws.com/templates/aws-parallelcluster-2.11.9.cfn.json
```

Update policy: This setting is not analyzed during an update.

vpc_settings

(Required) Identifies the [vpc] section with the Amazon VPC configuration where the cluster is deployed. The section name must start with a letter, contain no more than 30 characters, and only contain letters, numbers, hyphens (-), and underscores (_).

For more information, see the [\[vpc\] section](#).

For example, the following setting specifies that the section that starts [vpc public] is used for the Amazon VPC configuration.

```
vpc_settings = public
```

Update policy: If this setting is changed, the update is not allowed.

[compute_resource] section

Defines configuration settings for a compute resource. [\[compute_resource\] sections](#) are referenced by the [compute_resource_settings](#) setting in the [\[queue\] section](#). [\[compute_resource\] sections](#) are only supported when [scheduler](#) is set to slurm.

The format is `[compute_resource <compute-resource-name>]`. *compute-resource-name* must start with a letter, contain no more than 30 characters, and only contain letters, numbers, hyphens (-), and underscores (_).

```
[compute_resource cr1]
instance_type = c5.xlarge
min_count = 0
initial_count = 2
max_count = 10
spot_price = 0.5
```

Note

Support for the [\[compute_resource\] section](#) was added in AWS ParallelCluster version 2.9.0.

Topics

- [initial_count](#)
- [instance_type](#)
- [max_count](#)
- [min_count](#)
- [spot_price](#)

initial_count

(Optional) Sets the initial number of Amazon EC2 instances to launch for this compute resource. Cluster creation doesn't complete until at least this many nodes have been launched into the compute resource. If the [compute_type](#) setting for the queue is spot and there aren't enough Spot Instances available, the cluster creation might time out and fail. Any count larger than the [min_count](#) setting is dynamic capacity subject to the [scaledown_idletime](#) setting. This setting replaces the [initial_queue_size](#) setting.

Defaults to 0.

```
initial_count = 2
```

Update policy: The compute fleet must be stopped for this setting to be changed for an update.

instance_type

(Required) Defines the Amazon EC2 instance type that's used for this compute resource. The architecture of the instance type must be the same as the architecture used for the [master_instance_type](#) setting. The `instance_type` setting must be unique for each [\[compute_resource\] section](#) referenced by a [\[queue\] section](#). This setting replaces the [compute_instance_type](#) setting.

```
instance_type = t2.micro
```

Update policy: The compute fleet must be stopped for this setting to be changed for an update.

max_count

(Optional) Sets the maximum number of Amazon EC2 instances that can be launched in this compute resource. Any count larger than the [initial_count](#) setting is started in a power down mode. This setting replaces the [max_queue_size](#) setting.

Defaults to 10.

```
max_count = 10
```

Update policy: Reducing the size of a queue below the current number of nodes requires that the compute fleet be stopped first.

Note

The update policy did not support changing the `max_count` setting until the compute fleet was stopped for AWS ParallelCluster version 2.0.0 through version 2.9.1.

min_count

(Optional) Sets the minimum number of Amazon EC2 instances that can be launched in this compute resource. These nodes are all static capacity. Cluster creation doesn't complete until at least this number of nodes has been launched into the compute resource.

Defaults to 0.

```
min_count = 1
```

Update policy: Reducing the number of static nodes in a queue requires that the compute fleet be stopped first.

Note

The update policy did not support changing the `min_count` setting until the compute fleet was stopped for AWS ParallelCluster version 2.0.0 through version 2.9.1.

spot_price

(Optional) Sets the maximum Spot price for this compute resource. Used only when the [compute_type](#) setting for the queue containing this compute resources is set to `spot`. This setting replaces the [spot_price](#) setting.

If you don't specify a value, you're charged the Spot price, capped at the On-Demand price.

For assistance finding a Spot Instance that meets your needs, see the [Spot Instance advisor](#).

```
spot_price = 1.50
```

Update policy: The compute fleet must be stopped for this setting to be changed for an update.

[cw_log] section

Defines configuration settings for CloudWatch Logs.

The format is `[cw_log cw-log-name]`. *cw-log-name* must start with a letter, contain no more than 30 characters, and only contain letters, numbers, hyphens (-), and underscores (_).

```
[cw_log custom-cw-log]  
enable = true  
retention_days = 14
```

For more information, see [Integration with Amazon CloudWatch Logs](#), [Amazon CloudWatch dashboard](#), and [Integration with Amazon CloudWatch Logs](#).

Note

Support for `cw_log` was added in AWS ParallelCluster version 2.6.0.

enable

(Optional) Indicates whether CloudWatch Logs is enabled.

The default value is `true`. Use `false` to disable CloudWatch Logs.

The following example enables CloudWatch Logs.

```
enable = true
```

Update policy: If this setting is changed, the update is not allowed.

retention_days

(Optional) Indicates how many days CloudWatch Logs retains individual log events.

The default value is 14. The supported values are 1, 3, 5, 7, 14, 30, 60, 90, 120, 150, 180, 365, 400, 545, 731, 1827, and 3653.

The following example configures CloudWatch Logs to retain log events for 30 days.

```
retention_days = 30
```

Update policy: This setting can be changed during an update.

[dashboard] section

Defines configuration settings for the CloudWatch dashboard.

The format is `[dashboard dashboard-name]`. *dashboard-name* must start with a letter, contain no more than 30 characters, and only contain letters, numbers, hyphens (-), and underscores (_).

```
[dashboard custom-dashboard]  
enable = true
```

Note

Support for dashboard was added in AWS ParallelCluster version 2.10.0.

enable

(Optional) Indicates whether the CloudWatch dashboard is enabled.

The default value is `true`. Use `false` to disable the CloudWatch dashboard.

The following example enables the CloudWatch dashboard.

```
enable = true
```

[Update policy: This setting can be changed during an update.](#)

[dcv] section

Defines configuration settings for the Amazon DCV server running on the head node.

To create and configure a Amazon DCV server, specify the cluster [dcv_settings](#) with the name you define in the dcv section, and set [enable](#) to `master`, and [base_os](#) to `alinux2`, `centos7`, `ubuntu1804` or `ubuntu2004`. If the head node is an ARM instance, set [base_os](#) to `alinux2`, `centos7`, or `ubuntu1804`.

The format is `[dcv dcv-name]`. *dcv-name* must start with a letter, contain no more than 30 characters, and only contain letters, numbers, hyphens (-), and underscores (_).

```
[dcv custom-dcv]  
enable = master  
port = 8443  
access_from = 0.0.0.0/0
```

For more information, see [Connect to the head node through Amazon DCV](#)

Important

By default the Amazon DCV port setup by AWS ParallelCluster is open to all IPv4 addresses. However, you can connect to a Amazon DCV port only if you have the URL for the Amazon DCV session and connect to the Amazon DCV session within 30 seconds of when the URL is returned from `pcluster dcv connect`. Use the [access_from](#) setting to further restrict access to the Amazon DCV port with a CIDR-formatted IP range, and use the [port](#) setting to set a nonstandard port.

Note

Support for the [\[dcv\] section](#) on centos8 was removed in AWS ParallelCluster version 2.10.4. Support for the [\[dcv\] section](#) on centos8 was added in AWS ParallelCluster version 2.10.0. Support for the [\[dcv\] section](#) on AWS Graviton-based instances was added in AWS ParallelCluster version 2.9.0. Support for the [\[dcv\] section](#) on alinux2 and ubuntu1804 was added in AWS ParallelCluster version 2.6.0. Support for the [\[dcv\] section](#) on centos7 was added in AWS ParallelCluster version 2.5.0.

access_from

(Optional, Recommended) Specifies the CIDR-formatted IP range for connections to Amazon DCV. This setting is used only when AWS ParallelCluster creates the security group.

The default value is `0.0.0.0/0`, which allows access from any internet address.

```
access_from = 0.0.0.0/0
```

Update policy: This setting can be changed during an update.

enable

(Required) Indicates whether Amazon DCV is enabled on the head node. To enable Amazon DCV on the head node and configure the required security group rule, set the `enable` setting to `master`.

The following example enables Amazon DCV on the head node.

```
enable = master
```

Note

Amazon DCV automatically generates a self-signed certificate that's used to secure traffic between the Amazon DCV client and Amazon DCV server running on the head node. To configure your own certificate, see [Amazon DCV HTTPS certificate](#).

Update policy: If this setting is changed, the update is not allowed.

port

(Optional) Specifies the port for Amazon DCV.

The default value is 8443.

```
port = 8443
```

Update policy: If this setting is changed, the update is not allowed.

[ebs] section

Defines Amazon EBS volume configuration settings for volumes that are mounted on the head node and shared to the compute nodes through NFS.

To learn how to include Amazon EBS volumes in your cluster definition, see [\[cluster\] section / ebs_settings](#).

To use an existing Amazon EBS volume for long-term permanent storage that's independent of the cluster life cycle, specify [ebs_volume_id](#).

If you don't specify [ebs_volume_id](#), AWS ParallelCluster creates the EBS volume from the [ebs] settings when it creates the cluster and deletes the volume and data when the cluster is deleted.

For more information, see [Best practices: moving a cluster to a new AWS ParallelCluster minor or patch version](#).

The format is `[ebs ebs-name]`. *ebs-name* must start with a letter, contain no more than 30 characters, and only contain letters, numbers, hyphens (-), and underscores (_).

```
[ebs custom1]
shared_dir = vol1
ebs_snapshot_id = snap-xxxxx
volume_type = io1
volume_iops = 200
...

[ebs custom2]
shared_dir = vol2
...

...
```

Topics

- [shared_dir](#)
- [ebs_kms_key_id](#)
- [ebs_snapshot_id](#)
- [ebs_volume_id](#)
- [encrypted](#)
- [volume_iops](#)
- [volume_size](#)
- [volume_throughput](#)
- [volume_type](#)

shared_dir

(Required) Specifies the path where the shared Amazon EBS volume is mounted.

This parameter is required when using multiple Amazon EBS volumes.

When you use one Amazon EBS volume, this option overwrites the [shared_dir](#) that's specified under the [\[cluster\] section](#). In the following example, the volume mounts to /vol1.

```
shared_dir = vol1
```

Update policy: If this setting is changed, the update is not allowed.

ebs_kms_key_id

(Optional) Specifies a custom AWS KMS key to use for encryption.

This parameter must be used together with `encrypted = true`. It also must have a custom [ec2_iam_role](#).

For more information, see [Disk encryption with a custom KMS Key](#).

```
ebs_kms_key_id = xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

Update policy: If this setting is changed, the update is not allowed.

ebs_snapshot_id

(Optional) Defines the Amazon EBS snapshot ID if you're using a snapshot as the source for the volume.

There is no default value.

```
ebs_snapshot_id = snap-xxxxx
```

Update policy: If this setting is changed, the update is not allowed.

ebs_volume_id

(Optional) Defines the volume ID of an existing Amazon EBS volume to attach to the head node.

There is no default value.

```
ebs_volume_id = vol-xxxxxx
```

Update policy: If this setting is changed, the update is not allowed.

encrypted

(Optional) Specifies whether the Amazon EBS volume is encrypted. Note: Do *not* use with snapshots.

The default value is false.

```
encrypted = false
```

Update policy: If this setting is changed, the update is not allowed.

volume_iops

(Optional) Defines the number of IOPS for io1, io2, and gp3 type volumes.

The default value, supported values, and volume_iops to volume_size ratio varies by [volume_type](#) and [volume_size](#).

volume_type = io1

Default volume_iops = 100

Supported values volume_iops = 100–64000 †

Maximum volume_iops to volume_size ratio = 50 IOPS for each GiB. 5000 IOPS requires a volume_size of at least 100 GiB.

volume_type = io2

Default volume_iops = 100

Supported values volume_iops = 100–64000 (256000 for io2 Block Express volumes) †

Maximum volume_iops to volume_size ratio = 500 IOPS for each GiB. 5000 IOPS requires a volume_size of at least 10 GiB.

volume_type = gp3

Default volume_iops = 3000

Supported values volume_iops = 3000–16000

Maximum volume_iops to volume_size ratio = 500 IOPS for each GiB. 5000 IOPS requires a volume_size of at least 10 GiB.

```
volume_iops = 200
```

Update policy: This setting can be changed during an update.

† Maximum IOPS is guaranteed only on [Instances built on the Nitro System](#) provisioned with more than 32,000 IOPS. Other instances guarantee up to 32,000 IOPS. Unless you [modify the volume](#), earlier io1 volumes might not reach full performance. io2 Block Express volumes support volume_iops values up to 256,000. For more information, see [io2 Block Express volumes \(In preview\)](#) in the *Amazon EC2 User Guide*.

volume_size

(Optional) Specifies the size of the volume to be created, in GiB (if you're not using a snapshot).

The default value and supported values varies by [volume_type](#).

volume_type = standard

Default volume_size = 20 GiB

Supported values volume_size = 1–1024 GiB

volume_type = gp2, io1, io2, and gp3

Default volume_size = 20 GiB

Supported values volume_size = 1–16384 GiB

volume_type = sc1 and st1

Default volume_size = 500 GiB

Supported values volume_size = 500–16384 GiB

```
volume_size = 20
```

Note

Before AWS ParallelCluster version 2.10.1, the default value for all volume types was 20 GiB.

Update policy: If this setting is changed, the update is not allowed.

volume_throughput

(Optional) Defines the throughput for gp3 volume types, in MiB/s.

The default value is 125.

Supported values volume_throughput = 125–1000 MiB/s

The ratio of volume_throughput to volume_iops can be no more than 0.25. The maximum throughput of 1000 MiB/s requires that the volume_iops setting is at least 4000.

```
volume_throughput = 1000
```

Note

Support for volume_throughput was added in AWS ParallelCluster version 2.10.1.

Update policy: If this setting is changed, the update is not allowed.

volume_type

(Optional) Specifies the [Amazon EBS volume type](#) of the volume that you want to launch.

Valid options are the following volume types:

gp2, gp3

General purpose SSD

io1, io2

Provisioned IOPS SSD

st1

Throughput optimized HDD

sc1

Cold HDD

standard

Previous generation magnetic

For more information, see [Amazon EBS volume types](#) in the *Amazon EC2 User Guide*.

The default value is gp2.

```
volume_type = io2
```

Note

Support for gp3 and io2 was added in AWS ParallelCluster version 2.10.1.

Update policy: If this setting is changed, the update is not allowed.

[efs] section

Defines the configuration settings for the Amazon EFS that's mounted on the head and compute nodes. For more information, see [CreateFileSystem](#) in the *Amazon EFS API Reference*.

To learn how to include Amazon EFS file systems in your cluster definition, see [\[cluster\] section / efs_settings](#).

To use an existing Amazon EFS file system for long-term permanent storage that's independent of the cluster life cycle, specify [efs_fs_id](#).

If you don't specify [efs_fs_id](#), AWS ParallelCluster creates the Amazon EFS file system from the [efs] settings when it creates the cluster and deletes the file system and data when the cluster is deleted.

For more information, see [Best practices: moving a cluster to a new AWS ParallelCluster minor or patch version](#).

The format is [efs *efs-name*]. *efs-name* must start with a letter, contain no more than 30 characters, and only contain letters, numbers, hyphens (-), and underscores (_).

```
[efs customfs]
```

```
shared_dir = efs
encrypted = false
performance_mode = generalPurpose
```

Topics

- [efs_fs_id](#)
- [efs_kms_key_id](#)
- [encrypted](#)
- [performance_mode](#)
- [provisioned_throughput](#)
- [shared_dir](#)
- [throughput_mode](#)

efs_fs_id

(Optional) Defines the Amazon EFS file system ID for an existing file system.

Specifying this option voids all other Amazon EFS options except for [shared_dir](#).

If you set this option, it only supports the following types of file systems:

- File systems that don't have a mount target in the stack's Availability Zone.
- File systems that have an existing mount target in the stack's Availability Zone with both inbound and outbound NFS traffic allowed from 0.0.0.0/0.

The sanity check for validating [efs_fs_id](#) requires the IAM role to have the following permissions:

- elasticfilesystem:DescribeMountTargets
- elasticfilesystem:DescribeMountTargetSecurityGroups
- ec2:DescribeSubnets
- ec2:DescribeSecurityGroups
- ec2:DescribeNetworkInterfaceAttribute

To avoid errors, you must add these permissions to your IAM role, or set `sanity_check = false`.

⚠ Important

When you set a mount target with inbound and outbound NFS traffic allowed from `0.0.0.0/0`, it exposes the file system to NFS mounting requests from anywhere in the mount target's Availability Zone. AWS doesn't recommend creating a mount target in the stack's Availability Zone. Instead, let AWS handle this step. If you want to have a mount target in the stack's Availability Zone, consider using a custom security group by providing a [vpc_security_group_id](#) option under the [\[vpc\] section](#). Then, add that security group to the mount target and turn off `sanity_check` to create the cluster.

There is no default value.

```
efs_fs_id = fs-12345
```

Update policy: If this setting is changed, the update is not allowed.

efs_kms_key_id

(Optional) Identifies the AWS Key Management Service (AWS KMS) customer managed key to be used to protect the encrypted file system. If this is set, the [encrypted](#) setting must be set to `true`. This corresponds to the [KmsKeyId](#) parameter in the *Amazon EFS API Reference*.

There is no default value.

```
efs_kms_key_id = 1234abcd-12ab-34cd-56ef-1234567890ab
```

Update policy: If this setting is changed, the update is not allowed.

encrypted

(Optional) Indicates whether the file system is encrypted. This corresponds to the [Encrypted](#) parameter in the *Amazon EFS API Reference*.

The default value is `false`.

```
encrypted = true
```

Update policy: If this setting is changed, the update is not allowed.

performance_mode

(Optional) Defines the performance mode of the file system. This corresponds to the [PerformanceMode](#) parameter in the *Amazon EFS API Reference*.

Valid options are the following values:

- generalPurpose
- maxIO

Both values are case sensitive.

We recommend the generalPurpose performance mode for most file systems.

File systems that use the maxIO performance mode can scale to higher levels of aggregate throughput and operations per second. However, there's a trade-off of slightly higher latencies for most file operations.

After the file system is created, this parameter can't be changed.

The default value is generalPurpose.

```
performance_mode = generalPurpose
```

Update policy: If this setting is changed, the update is not allowed.

provisioned_throughput

(Optional) Defines the provisioned throughput of the file system, measured in MiB/s. This corresponds to the [ProvisionedThroughputInMibps](#) parameter in the *Amazon EFS API Reference*.

If you use this parameter, you must set [throughput_mode](#) to provisioned.

The quota on throughput is 1024 MiB/s. To request a quota increase, contact Support.

The minimum value is 0.0 MiB/s.

```
provisioned_throughput = 1024
```

Update policy: This setting can be changed during an update.

shared_dir

(Required) Defines the Amazon EFS mount point on the head and compute nodes.

This parameter is required. The Amazon EFS section is used only if [shared_dir](#) is specified.

Don't use NONE or /NONE as the shared directory.

The following example mounts Amazon EFS at /efs.

```
shared_dir = efs
```

Update policy: If this setting is changed, the update is not allowed.

throughput_mode

(Optional) Defines the throughput mode of the file system. This corresponds to the [ThroughputMode](#) parameter in the *Amazon EFS API Reference*.

Valid options are the following values:

- bursting
- provisioned

The default value is bursting.

```
throughput_mode = provisioned
```

Update policy: This setting can be changed during an update.

[fsx] section

Defines configuration settings for an attached FSx for Lustre file system. For more information, see [Amazon FSx CreateFileSystem](#) in the *Amazon FSx API Reference*.

If the [base_os](#) is `alinux2`, `centos7`, `ubuntu1804`, or `ubuntu2004`, FSx for Lustre is supported.

When using Amazon Linux, the kernel must be `4.14.104-78.84.amzn1.x86_64` or a later version. For instructions, see [Installing the lustre client](#) in the *Amazon FSx for Lustre User Guide*.

Note

FSx for Lustre isn't currently supported when using `awsbatch` as a scheduler.

Note

Support for FSx for Lustre on `centos8` was removed in AWS ParallelCluster version 2.10.4. Support for FSx for Lustre on `ubuntu2004` was added in AWS ParallelCluster version 2.11.0. Support for FSx for Lustre on `centos8` was added in AWS ParallelCluster version 2.10.0. Support for FSx for Lustre on `alinux2`, `ubuntu1604`, and `ubuntu1804` was added in AWS ParallelCluster version 2.6.0. Support for FSx for Lustre on `centos7` was added in AWS ParallelCluster version 2.4.0.

If using an existing file system, it must be associated to a security group that allows inbound TCP traffic to port 988. Setting the source to `0.0.0.0/0` on a security group rule provides client access from all the IP ranges within your VPC security group for the protocol and port range for that rule. To further limit access to your file systems, we recommend using more restrictive sources for your security group rules. For example, you can use more specific CIDR ranges, IP addresses, or security group IDs. This is done automatically when not using [vpc_security_group_id](#).

To use an existing Amazon FSx file system for long-term permanent storage that's independent of the cluster life cycle, specify [fsx_fs_id](#).

If you don't specify [fsx_fs_id](#), AWS ParallelCluster creates the FSx for Lustre file system from the `[fsx]` settings when it creates the cluster and deletes the file system and data when the cluster is deleted.

For more information, see [Best practices: moving a cluster to a new AWS ParallelCluster minor or patch version](#).

The format is `[fsx fsx-name]`. *fsx-name* must start with a letter, contain no more than 30 characters, and only contain letters, numbers, hyphens (-), and underscores (_).

```
[fsx fs]
shared_dir = /fsx
fsx_fs_id = fs-073c3803dca3e28a6
```

To create and configure a new file system, use the following parameters:

```
[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
```

Topics

- [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

(Optional) Specifies the automatic import policy for reflecting changes in the S3 bucket used to create the FSx for Lustre file system. The possible values are the following:

NEW

FSx for Lustre automatically imports directory listings of any new objects that are added to the linked S3 bucket that don't currently exist in the FSx for Lustre file system.

NEW_CHANGED

FSx for Lustre automatically imports file and directory listings of any new objects that are added to the S3 bucket and any existing objects that are changed in the S3 bucket.

This corresponds to the [AutoImportPolicy](#) property. For more information, see [Automatically import updates from your S3 bucket](#) in the *Amazon FSx for Lustre User Guide*. When the [auto_import_policy](#) parameter is specified, the [automatic_backup_retention_days](#), [copy_tags_to_backups](#), [daily_automatic_backup_start_time](#), and [fsx_backup_id](#) parameters must not be specified.

If the `auto_import_policy` setting isn't specified, automatic imports are disabled. FSx for Lustre only updates file and directory listings from the linked S3 bucket when the file system is created.

```
auto_import_policy = NEW_CHANGED
```

Note

Support for [auto_import_policy](#) was added in AWS ParallelCluster version 2.10.0.

Update policy: If this setting is changed, the update is not allowed.

automatic_backup_retention_days

(Optional) Specifies the number of days to retain automatic backups. This is only valid for use with `PERSISTENT_1` deployment types. When the [automatic_backup_retention_days](#) parameter is specified, the [auto_import_policy](#), [export_path](#), [import_path](#), and [imported_file_chunk_size](#) parameters must not be specified. This corresponds to the [AutomaticBackupRetentionDays](#) property.

The default value is 0. This setting disables automatic backups. The possible values are integers between 0 and 35, inclusive.

```
automatic_backup_retention_days = 35
```

Note

Support for [automatic_backup_retention_days](#) was added in AWS ParallelCluster version 2.8.0.

Update policy: [This setting can be changed during an update.](#)

copy_tags_to_backups

(Optional) Specifies whether tags for the filesystem are copied to the backups. This is only valid for use with PERSISTENT_1 deployment types. When the [copy_tags_to_backups](#) parameter is specified, the [automatic_backup_retention_days](#) must be specified with a value greater than 0, and the [auto_import_policy](#), [export_path](#), [import_path](#), and [imported_file_chunk_size](#) parameters must not be specified. This corresponds to the [CopyTagsToBackups](#) property.

The default value is false.

```
copy_tags_to_backups = true
```

Note

Support for [copy_tags_to_backups](#) was added in AWS ParallelCluster version 2.8.0.

Update policy: [If this setting is changed, the update is not allowed.](#)

daily_automatic_backup_start_time

(Optional) Specifies the time of day (UTC) to start automatic backups. This is only valid for use with PERSISTENT_1 deployment types. When the [daily_automatic_backup_start_time](#) parameter is specified, the [automatic_backup_retention_days](#) must be specified with a value greater than 0, and the [auto_import_policy](#), [export_path](#), [import_path](#), and [imported_file_chunk_size](#) parameters must not be specified. This corresponds to the [DailyAutomaticBackupStartTime](#) property.

The format is HH:MM, where HH is the zero-padded hour of the day (0-23), and MM is the zero-padded minute of the hour. For example, 1:03 A.M. UTC is the following.

```
daily_automatic_backup_start_time = 01:03
```

The default value is a random time between 00:00 and 23:59.

Note

Support for [daily_automatic_backup_start_time](#) was added in AWS ParallelCluster version 2.8.0.

Update policy: This setting can be changed during an update.

data_compression_type

(Optional) Specifies the FSx for Lustre data compression type. This corresponds to the [DataCompressionType](#) property. For more information, see [FSx for Lustre data compression](#) in the *Amazon FSx for Lustre User Guide*.

The only valid value is LZ4. To disable data compression, remove the [data_compression_type](#) parameter.

```
data_compression_type = LZ4
```

Note

Support for [data_compression_type](#) was added in AWS ParallelCluster version 2.11.0.

Update policy: This setting can be changed during an update.

deployment_type

(Optional) Specifies the FSx for Lustre deployment type. This corresponds to the [DeploymentType](#) property. For more information, see [FSx for Lustre deployment options](#) in the *Amazon FSx for*

Lustre User Guide. Choose a scratch deployment type for temporary storage and shorter-term processing of data. SCRATCH_2 is the latest generation of scratch file systems. It offers higher burst throughput over baseline throughput and the in-transit encryption of data.

The valid values are SCRATCH_1, SCRATCH_2, and PERSISTENT_1.

SCRATCH_1

The default deployment type for FSx for Lustre. With this deployment type, the [storage_capacity](#) setting has possible values of 1200, 2400, and any multiple of 3600. Support for SCRATCH_1 was added in AWS ParallelCluster version 2.4.0.

SCRATCH_2

The latest generation of scratch file systems. It supports up to six times the baseline throughput for spiky workloads. It also supports in-transit encryption of data for supported instance types in supported AWS Regions. For more information, see [Encrypting data in transit](#) in the *Amazon FSx for Lustre User Guide*. With this deployment type, the [storage_capacity](#) setting has possible values of 1200 and any multiple of 2400. Support for SCRATCH_2 was added in AWS ParallelCluster version 2.6.0.

PERSISTENT_1

Designed for longer-term storage. The file servers are highly available and the data is replicated within the file systems' AWS Availability Zone. It supports in-transit encryption of data for supported instance types. With this deployment type, the [storage_capacity](#) setting has possible values of 1200 and any multiple of 2400. Support for PERSISTENT_1 was added in AWS ParallelCluster version 2.6.0.

The default value is SCRATCH_1.

```
deployment_type = SCRATCH_2
```

Note

Support for [deployment_type](#) was added in AWS ParallelCluster version 2.6.0.

Update policy: If this setting is changed, the update is not allowed.

drive_cache_type

(Optional) Specifies that the file system has an SSD drive cache. This can only be set if the [storage_type](#) setting is set to HDD. This corresponds to the [DriveCacheType](#) property. For more information, see [FSx for Lustre deployment options](#) in the *Amazon FSx for Lustre User Guide*.

The only valid value is READ. To disable the SSD drive cache, don't specify the `drive_cache_type` setting.

```
drive_cache_type = READ
```

Note

Support for [drive_cache_type](#) was added in AWS ParallelCluster version 2.10.0.

Update policy: If this setting is changed, the update is not allowed.

export_path

(Optional) Specifies the Amazon S3 path where the root of your file system is exported. When the [export_path](#) parameter is specified, the [automatic_backup_retention_days](#), [copy_tags_to_backups](#), [daily_automatic_backup_start_time](#), and [fsx_backup_id](#) parameters must not be specified. This corresponds to the [ExportPath](#) property. File data and metadata isn't automatically exported to the `export_path`. For information about exporting data and metadata, see [Exporting changes to the data repository](#) in the *Amazon FSx for Lustre User Guide*.

The default value is `s3://import-bucket/FSxLustre[creation-timestamp]`, where *import-bucket* is the bucket provided in the [import_path](#) parameter.

```
export_path = s3://bucket/folder
```

Update policy: If this setting is changed, the update is not allowed.

fsx_backup_id

(Optional) Specifies the ID of the backup to use for restoring the file system from an existing backup. When the [fsx_backup_id](#) parameter is specified, the

[auto_import_policy](#), [deployment_type](#), [export_path](#), [fsx_kms_key_id](#), [import_path](#), [imported_file_chunk_size](#), [storage_capacity](#), and [per_unit_storage_throughput](#) parameters must not be specified. These parameters are read from the backup. Additionally, the [auto_import_policy](#), [export_path](#), [import_path](#), and [imported_file_chunk_size](#) parameters must not be specified.

This corresponds to the [BackupId](#) property.

```
fsx_backup_id = backup-fedcba98
```

Note

Support for [fsx_backup_id](#) was added in AWS ParallelCluster version 2.8.0.

Update policy: If this setting is changed, the update is not allowed.

fsx_fs_id

(Optional) Attaches an existing FSx for Lustre file system.

If this option is specified, only the [shared_dir](#) and [fsx_fs_id](#) settings in the [\[fsx\] section](#) are used and any other settings in the [\[fsx\] section](#) are ignored.

```
fsx_fs_id = fs-073c3803dca3e28a6
```

Update policy: If this setting is changed, the update is not allowed.

fsx_kms_key_id

(Optional) Specifies the key ID of your AWS Key Management Service (AWS KMS) customer managed key.

This key is used to encrypt the data in your file system at rest.

This must be used with a custom [ec2_iam_role](#). For more information, see [Disk encryption with a custom KMS Key](#). This corresponds to the [KmsKeyId](#) parameter in the *Amazon FSx API Reference*.

```
fsx_kms_key_id = xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxxx
```

Note

Support for [fsx_kms_key_id](#) was added in AWS ParallelCluster version 2.6.0.

Update policy: If this setting is changed, the update is not allowed.

import_path

(Optional) Specifies the S3 bucket to load data from into the file system and serve as the export bucket. For more information, see [export_path](#). If you specify the [import_path](#) parameter, the [automatic_backup_retention_days](#), [copy_tags_to_backups](#), [daily_automatic_backup_start_time](#), and [fsx_backup_id](#) parameters must not be specified. This corresponds to the [ImportPath](#) parameter in the *Amazon FSx API Reference*.

Import occurs on cluster creation. For more information, see [Importing data from your data repository](#) in the *Amazon FSx for Lustre User Guide*. On import, only file metadata (name, ownership, timestamp, and permissions) is imported. File data isn't imported from the S3 bucket until the file is first accessed. For information about pre-loading the file contents, see [Pre-loading files into your file system](#) in the *Amazon FSx for Lustre User Guide*.

If a value isn't provided, the file system is empty.

```
import_path = s3://bucket
```

Update policy: If this setting is changed, the update is not allowed.

imported_file_chunk_size

(Optional) Determines the stripe count and the maximum amount of data for each file (in MiB) stored on a single physical disk for files that are imported from a data repository (using [import_path](#)). The maximum number of disks that a single file can be striped across is limited by the total number of disks that make up the file system. When the [imported_file_chunk_size](#) parameter is specified, the [automatic_backup_retention_days](#), [copy_tags_to_backups](#), [daily_automatic_backup_start_time](#), and [fsx_backup_id](#) parameters must not be specified. This corresponds to the [ImportedFileChunkSize](#) property.

The chunk size default is 1024 (1 GiB), and it can go as high as 512,000 MiB (500 GiB). Amazon S3 objects have a maximum size of 5 TB.

```
imported_file_chunk_size = 1024
```

Update policy: If this setting is changed, the update is not allowed.

per_unit_storage_throughput

(Required for PERSISTENT_1 deployment types) For the [deployment_type](#) = PERSISTENT_1 deployment type, describes the amount of read and write throughput for each 1 tebibyte (TiB) of storage, in MB/s/TiB. File system throughput capacity is calculated by multiplying file system storage capacity (TiB) by the [per_unit_storage_throughput](#) (MB/s/TiB). For a 2.4 TiB file system, provisioning 50 MB/s/TiB of [per_unit_storage_throughput](#) yields 120 MB/s of file system throughput. You pay for the amount of throughput that you provision. This corresponds to the [PerUnitStorageThroughput](#) property.

The possible values depend on the value of the [storage_type](#) setting.

[storage_type](#) = SSD

The possible values are 50, 100, 200.

[storage_type](#) = HDD

The possible values are 12, 40.

```
per_unit_storage_throughput = 200
```

Note

Support for [per_unit_storage_throughput](#) was added in AWS ParallelCluster version 2.6.0.

Update policy: If this setting is changed, the update is not allowed.

shared_dir

(Required) Defines the mount point for the FSx for Lustre file system on the head and compute nodes.

Don't use NONE or /NONE as the shared directory.

The following example mounts the file system at /fsx.

```
shared_dir = /fsx
```

Update policy: If this setting is changed, the update is not allowed.

storage_capacity

(Required) Specifies the storage capacity of the file system, in GiB. This corresponds to the [StorageCapacity](#) property.

The storage capacity possible values vary based on the [deployment_type](#) setting.

SCRATCH_1

The possible values are 1200, 2400, and any multiple of 3600.

SCRATCH_2

The possible values are 1200 and any multiple of 2400.

PERSISTENT_1

The possible values vary based on the values of other settings.

[storage_type](#) = SSD

The possible values are 1200 and any multiple of 2400.

[storage_type](#) = HDD

The possible values vary based on the setting of the [per_unit_storage_throughput](#) setting.

[per_unit_storage_throughput](#) = 12

The possible values are any multiple of 6000.

[per_unit_storage_throughput](#) = 40

The possible values are any multiple of 1800.

```
storage_capacity = 7200
```

Note

For AWS ParallelCluster version 2.5.0 and 2.5.1, [storage_capacity](#) supported possible values of 1200, 2400, and any multiple of 3600. For versions earlier than AWS ParallelCluster version 2.5.0, [storage_capacity](#) had a minimum size of 3600.

Update policy: If this setting is changed, the update is not allowed.

storage_type

(Optional) Specifies the storage type of the file system. This corresponds to the [StorageType](#) property. The possible values are SSD and HDD. The default is SSD.

The storage type changes the possible values of other settings.

```
storage_type = SSD
```

Specifies a sold-state drive (SSD) storage type.

`storage_type = SSD` changes the possible values of several other settings.

[drive_cache_type](#)

This setting cannot be specified.

[deployment_type](#)

This setting can be set to SCRATCH_1, SCRATCH_2, or PERSISTENT_1.

[per_unit_storage_throughput](#)

This setting must be specified if [deployment_type](#) is set to PERSISTENT_1. The possible values are 50, 100, or 200.

[storage_capacity](#)

This setting must be specified. The possible values vary based on [deployment_type](#).

```
deployment_type = SCRATCH_1
```

[storage_capacity](#) can be 1200, 2400, or any multiple of 3600.

`deployment_type = SCRATCH_2` or `deployment_type = PERSISTENT_1`

[storage_capacity](#) can be 1200 or any multiple of 2400.

`storage_type = HDD`

Specifies a hard disk drive (HDD) storage type.

`storage_type = HDD` changes the possible values of other settings.

[drive_cache_type](#)

This setting can be specified.

[deployment_type](#)

This setting must be set to `PERSISTENT_1`.

[per_unit_storage_throughput](#)

This setting must be specified. The possible values are 12, or 40.

[storage_capacity](#)

This setting must be specified. The possible values vary based on the

[per_unit_storage_throughput](#) setting.

`storage_capacity = 12`

[storage_capacity](#) can be any multiple of 6000.

`storage_capacity = 40`

[storage_capacity](#) can be any multiple of 1800.

```
storage_type = SSD
```

Note

Support for the [storage_type](#) setting was added in AWS ParallelCluster version 2.10.0.

Update policy: If this setting is changed, the update is not allowed.

weekly_maintenance_start_time

(Optional) Specifies a preferred time to perform weekly maintenance, in the UTC time zone. This corresponds to the [WeeklyMaintenanceStartTime](#) property.

The format is [day of week]:[hour of day]:[minute of hour]. For example, Monday at Midnight is as follows.

```
weekly_maintenance_start_time = 1:00:00
```

Update policy: This setting can be changed during an update.

[queue] section

Defines configuration settings for a single queue. [\[queue\] sections](#) are only supported when [scheduler](#) is set to slurm.

The format is [queue *<queue-name>*]. *queue-name* must start with a lowercase letter, contain no more than 30 characters, and only contain lowercase letters, numbers, and hyphens (-).

```
[queue q1]
compute_resource_settings = i1,i2
placement_group = DYNAMIC
enable_efa = true
disable_hyperthreading = false
compute_type = spot
```

Note

Support for the [\[queue\] section](#) was added in AWS ParallelCluster version 2.9.0.

Topics

- [compute_resource_settings](#)
- [compute_type](#)
- [disable_hyperthreading](#)
- [enable_efa](#)
- [enable_efa_gdr](#)

- [placement_group](#)

compute_resource_settings

(Required) Identifies the [\[compute_resource\] sections](#) containing the compute resources configurations for this queue. The section names must start with a letter, contain no more than 30 characters, and only contain letters, numbers, hyphens (-), and underscores (_).

Up to three (3) [\[compute_resource\] sections](#) are supported for each [\[queue\] section](#).

For example, the following setting specifies that the sections that start [compute_resource cr1] and [compute_resource cr2] are used.

```
compute_resource_settings = cr1, cr2
```

Update policy: If this setting is changed, the update is not allowed.

compute_type

(Optional) Defines the type of instances to launch for this queue. This setting replaces the [cluster_type](#) setting.

Valid options are: ondemand, and spot.

The default value is ondemand.

For more information about Spot Instances, see [Working with Spot Instances](#).

Note

Using Spot Instances requires that the AWSServiceRoleForEC2Spot service-linked role exist in your account. To create this role in your account using the AWS CLI, run the following command:

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

For more information, see [Service-linked role for Spot Instance requests](#) in the *Amazon EC2 User Guide*.

The following example uses SpotInstances for the compute nodes in this queue.

```
compute_type = spot
```

Update policy: [The compute fleet must be stopped for this setting to be changed for an update.](#)

disable_hyperthreading

(Optional) Disables hyperthreading on the nodes in this queue. Not all instance types can disable hyperthreading. For a list of instance types that support disabling hyperthreading, see [CPU cores and threads for each CPU core per instance type](#) in the *Amazon EC2 User Guide*. If the [disable_hyperthreading](#) setting in the [\[cluster\] section](#) is defined, then this setting cannot be defined.

The default value is false.

```
disable_hyperthreading = true
```

Update policy: [The compute fleet must be stopped for this setting to be changed for an update.](#)

enable_efa

(Optional) If set to true, specifies that Elastic Fabric Adapter (EFA) is enabled for the nodes in this queue. To view the list of EC2 instances that support EFA, see [Supported instance types](#) in the *Amazon EC2 User Guide for Linux Instances*. If the [enable_efa](#) setting in the [\[cluster\] section](#) is defined, then this setting cannot be defined. A cluster placement group should be used to minimize latencies between instances. For more information, see [placement](#) and [placement_group](#).

```
enable_efa = true
```

Update policy: [The compute fleet must be stopped for this setting to be changed for an update.](#)

enable_efa_gdr

(Optional) Starting with AWS ParallelCluster version 2.11.3, this setting has no effect. Elastic Fabric Adapter (EFA) support for GPUDirect RDMA (remote direct memory access) is enabled for the compute nodes is always enabled if it's supported by the instance type.

Note

AWS ParallelCluster version 2.10.0 through 2.11.2: If `true`, specifies that Elastic Fabric Adapter (EFA) GPUDirect RDMA (remote direct memory access) is enabled for the nodes in this queue. Setting this to `true` requires that the [enable_efa](#) setting is set to `true`. EFA GPUDirect RDMA is supported by the following instance types (p4d.24xlarge) on these operating systems (alinux2, centos7, ubuntu1804, or ubuntu2004). If the [enable_efa_gdr](#) setting in the [\[cluster\] section](#) is defined, then this setting cannot be defined. A cluster placement group should be used to minimize latencies between instances. For more information, see [placement](#) and [placement_group](#).

The default value is `false`.

```
enable_efa_gdr = true
```

Note

Support for `enable_efa_gdr` was added in AWS ParallelCluster version 2.10.0.

Update policy: The compute fleet must be stopped for this setting to be changed for an update.

placement_group

(Optional) If present, defines the placement group for this queue. This setting replaces the [placement_group](#) setting.

Valid options are the following values:

- DYNAMIC
- An existing Amazon EC2 cluster placement group name

When set to DYNAMIC, a unique placement group for this queue is created and deleted as part of the cluster stack.

For more information about placement groups, see [Placement groups](#) in the *Amazon EC2 User Guide*. If the same placement group is used for different instance types, it's more likely that the

request might fail due to an insufficient capacity error. For more information, see [Insufficient instance capacity](#) in the *Amazon EC2 User Guide*.

There is no default value.

Not all instance types support cluster placement groups. For example, `t2.micro` doesn't support cluster placement groups. For information about the list of instance types that support cluster placement groups, see [Cluster placement group rules and limitations](#) in the *Amazon EC2 User Guide*. See [Placement groups and instance launch issues](#) for tips when working with placement groups.

```
placement_group = DYNAMIC
```

Update policy: The compute fleet must be stopped for this setting to be changed for an update.

[raid] section

Defines configuration settings for a RAID array that's built from a number of identical Amazon EBS volumes. The RAID drive is mounted on the head node and is exported to compute nodes with NFS.

The format is `[raid raid-name]`. *raid-name* must start with a letter, contain no more than 30 characters, and only contain letters, numbers, hyphens (-), and underscores (_).

```
[raid rs]
shared_dir = raid
raid_type = 1
num_of_raid_volumes = 2
encrypted = true
```

Topics

- [shared_dir](#)
- [ebs_kms_key_id](#)
- [encrypted](#)
- [num_of_raid_volumes](#)
- [raid_type](#)
- [volume_iops](#)
- [volume_size](#)

- [volume_throughput](#)
- [volume_type](#)

shared_dir

(Required) Defines the mount point for the RAID array on the head and compute nodes.

The RAID drive is created only if this parameter is specified.

Don't use NONE or /NONE as the shared directory.

The following example mounts the array at /raid.

```
shared_dir = raid
```

Update policy: If this setting is changed, the update is not allowed.

ebs_kms_key_id

(Optional) Specifies a custom AWS KMS key to use for encryption.

This parameter must be used together with `encrypted = true`, and it must have a custom [ec2_iam_role](#).

For more information, see [Disk encryption with a custom KMS Key](#).

```
ebs_kms_key_id = xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxxx
```

Update policy: If this setting is changed, the update is not allowed.

encrypted

(Optional) Specifies whether the file system is encrypted.

The default value is false.

```
encrypted = false
```

Update policy: If this setting is changed, the update is not allowed.

num_of_raid_volumes

(Optional) Defines the number of Amazon EBS volumes to assemble the RAID array from.

Minimum number of volumes is 2.

Maximum number of volumes is 5.

The default value is 2.

```
num_of_raid_volumes = 2
```

Update policy: If this setting is changed, the update is not allowed.

raid_type

(Required) Defines the RAID type for the RAID array.

The RAID drive is created only if this parameter is specified.

Valid options are the following values:

- 0
- 1

For more information on RAID types, see [RAID info](#) in the *Amazon EC2 User Guide*.

The following example creates a RAID 0 array:

```
raid_type = 0
```

Update policy: If this setting is changed, the update is not allowed.

volume_iops

(Optional) Defines the number of IOPS for io1, io2, and gp3 type volumes.

The default value, supported values, and volume_iops to volume_size ratio varies by [volume_type](#) and [volume_size](#).

`volume_type = io1`

Default `volume_iops` = 100

Supported values `volume_iops` = 100–64000 †

Maximum `volume_iops` to `volume_size` ratio = 50 IOPS per GiB. 5000 IOPS requires a `volume_size` of at least 100 GiB.

`volume_type = io2`

Default `volume_iops` = 100

Supported values `volume_iops` = 100–64000 (256000 for io2 Block Express volumes) †

Maximum `volume_iops` to `volume_size` ratio = 500 IOPS per GiB. 5000 IOPS requires a `volume_size` of at least 10 GiB.

`volume_type = gp3`

Default `volume_iops` = 3000

Supported values `volume_iops` = 3000–16000

Maximum `volume_iops` to `volume_size` ratio = 500 IOPS per GiB. 5000 IOPS requires a `volume_size` of at least 10 GiB.

```
volume_iops = 3000
```

Update policy: This setting can be changed during an update.

† Maximum IOPS is guaranteed only on [Instances built on the Nitro System](#) provisioned with more than 32,000 IOPS. Other instances guarantee up to 32,000 IOPS. Older io1 volumes might not reach full performance unless you [modify the volume](#). io2 Block Express volumes support `volume_iops` values up to 256000. For more information, see [io2 Block Express volumes \(In preview\)](#) in the *Amazon EC2 User Guide*.

volume_size

(Optional) Defines the size of the volume to be created, in GiB.

The default value and supported values varies by [volume_type](#).

`volume_type = standard`

Default `volume_size` = 20 GiB

Supported values `volume_size` = 1–1024 GiB

`volume_type` = gp2, io1, io2, and gp3

Default `volume_size` = 20 GiB

Supported values `volume_size` = 1–16384 GiB

`volume_type` = sc1 and st1

Default `volume_size` = 500 GiB

Supported values `volume_size` = 500–16384 GiB

```
volume_size = 20
```

Note

Before AWS ParallelCluster version 2.10.1, the default value for all volume types was 20 GiB.

Update policy: If this setting is changed, the update is not allowed.

volume_throughput

(Optional) Defines the throughput for gp3 volume types, in MiB/s.

The default value is 125.

Supported values `volume_throughput` = 125–1000 MiB/s

The ratio of `volume_throughput` to `volume_iops` can be no more than 0.25. The maximum throughput of 1000 MiB/s requires that the `volume_iops` setting is at least 4000.

```
volume_throughput = 1000
```


Note

Support for `volume_throughput` was added in AWS ParallelCluster version 2.10.1.

Update policy: If this setting is changed, the update is not allowed.

`volume_type`

(Optional) Defines the type of volume to build.

Valid options are the following values:

`gp2`, `gp3`

General purpose SSD

`io1`, `io2`

Provisioned IOPS SSD

`st1`

Throughput optimized HDD

`sc1`

Cold HDD

`standard`

Previous generation magnetic

For more information, see [Amazon EBS volume types](#) in the *Amazon EC2 User Guide*.

The default value is `gp2`.

```
volume_type = io2
```

Note

Support for `gp3` and `io2` was added in AWS ParallelCluster version 2.10.1.

Update policy: If this setting is changed, the update is not allowed.

[scaling] section

Topics

- [scaledown_idletime](#)

Specifies settings that define how the compute nodes scale.

The format is [scaling *scaling-name*]. *scaling-name* must start with a letter, contain no more than 30 characters, and only contain letters, numbers, hyphens (-), and underscores (_).

```
[scaling custom]
scaledown_idletime = 10
```

scaledown_idletime

(Optional) Specifies the amount of time in minutes without a job, after which the compute node terminates.

This parameter isn't used if awsbatch is the scheduler.

The default value is 10.

```
scaledown_idletime = 10
```

Update policy: The compute fleet must be stopped for this setting to be changed for an update.

[vpc] section

Specifies Amazon VPC configuration settings. For more information about VPCs, see [What is Amazon VPC?](#) and [Security best practices for your VPC](#) in the *Amazon VPC User Guide*.

The format is [vpc *vpc-name*]. *vpc-name* must start with a letter, contain no more than 30 characters, and only contain letters, numbers, hyphens (-), and underscores (_).

```
[vpc public]
vpc_id = vpc-xxxxxxx
```

```
master_subnet_id = subnet-xxxxxx
```

Topics

- [additional_sg](#)
- [compute_subnet_cidr](#)
- [compute_subnet_id](#)
- [master_subnet_id](#)
- [ssh_from](#)
- [use_public_ips](#)
- [vpc_id](#)
- [vpc_security_group_id](#)

additional_sg

(Optional) Provides an additional Amazon VPC security group Id for all instances.

There is no default value.

```
additional_sg = sg-xxxxxx
```

compute_subnet_cidr

(Optional) Specifies a Classless Inter-Domain Routing (CIDR) block. Use this parameter if you want AWS ParallelCluster to create a compute subnet.

```
compute_subnet_cidr = 10.0.100.0/24
```

Update policy: If this setting is changed, the update is not allowed.

compute_subnet_id

(Optional) Specifies the ID of an existing subnet in which to provision the compute nodes.

If not specified, [compute_subnet_id](#) uses the value of [master_subnet_id](#).

If the subnet is private, you must set up NAT for web access.

```
compute_subnet_id = subnet-xxxxxx
```

Update policy: The compute fleet must be stopped for this setting to be changed for an update.

master_subnet_id

(Required) Specifies the ID of an existing subnet in which to provision the head node.

```
master_subnet_id = subnet-xxxxxx
```

Update policy: If this setting is changed, the update is not allowed.

ssh_from

(Optional) Specifies a CIDR-formatted IP range to allow SSH access from.

This parameter is used only when AWS ParallelCluster creates the security group.

The default value is 0.0.0.0/0.

```
ssh_from = 0.0.0.0/0
```

Update policy: This setting can be changed during an update.

use_public_ips

(Optional) Defines whether to assign public IP addresses to compute instances.

If set to `true`, an Elastic IP address is associated to the head node.

If set to `false`, the head node has a public IP (or not) according to the value of the "Auto-assign Public IP" subnet configuration parameter.

For examples, see [networking configuration](#).

The default value is `true`.

```
use_public_ips = true
```

⚠ Important

By default, all AWS accounts are limited to five (5) Elastic IP addresses for each AWS Region. For more information, see [Elastic IP address limit](#) in *Amazon EC2 User Guide*.

Update policy: The compute fleet must be stopped for this setting to be changed for an update.

vpc_id

(Required) Specifies the ID of the Amazon VPC in which to provision the cluster.

```
vpc_id = vpc-xxxxxx
```

Update policy: If this setting is changed, the update is not allowed.

vpc_security_group_id

(Optional) Specifies the use of an existing security group for all instances.

There is no default value.

```
vpc_security_group_id = sg-xxxxxx
```

The security group created by AWS ParallelCluster allows SSH access using port 22 from the addresses specified in the [ssh_from](#) setting, or all IPv4 addresses (0.0.0.0/0) if the [ssh_from](#) setting isn't specified. If Amazon DCV is enabled, then the security group allows access to Amazon DCV using port 8443 (or whatever the [port](#) setting specifies) from the addresses specified in the [access_from](#) setting, or all IPv4 addresses (0.0.0.0/0) if the [access_from](#) setting isn't specified.

⚠ Warning

You can change the value of this parameter and update the cluster if [\[cluster\] fsx_settings](#) isn't specified or both `fsx_settings` and an external existing FSx for Lustre file system is specified for [fsx-fs-id](#) in [\[fsx fs\]](#).
You can't change the value of this parameter if an AWS ParallelCluster managed FSx for Lustre file system is specified in `fsx_settings` and [\[fsx fs\]](#).

Update policy: If AWS ParallelCluster managed Amazon FSx for Lustre file systems aren't specified in the configuration, this setting can be changed during an update.

Examples

The following example configurations demonstrate AWS ParallelCluster configurations using Slurm, Torque, and AWS Batch schedulers.

Note

Starting with version 2.11.5, AWS ParallelCluster doesn't support the use of SGE or Torque schedulers.

Contents

- [Slurm Workload Manager \(slurm\)](#)
- [Son of Grid Engine \(sge\) and Torque Resource Manager \(torque\)](#)
- [AWS Batch \(awsbatch\)](#)

Slurm Workload Manager (slurm)

The following example launches a cluster with the `slurm` scheduler. The example configuration launches 1 cluster with 2 job queues. The first queue, `spot`, initially has 2 `t3.micro` Spot instances available. It can scale up to a maximum of 10 instances, and scale down to a minimum of 1 instance when no jobs have been run for 10 minutes (adjustable using the [scaledown_idletime](#) setting). The second queue, `ondemand`, starts with no instances and can scale up to a maximum of 5 `t3.micro` On-Demand instances.

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

[aws]
aws_region_name = <your AWS Region>

[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) and Torque Resource Manager (torque)

Note

This example only applies to AWS ParallelCluster versions up to and including version 2.11.4. Starting with version 2.11.5, AWS ParallelCluster doesn't support the use of SGE or Torque schedulers.

The following example launches a cluster with the `torque` or `sge` scheduler. To use SGE, change `scheduler = torque` to `scheduler = sge`. The example configuration allows a maximum of 5 concurrent nodes, and scales down to two when no jobs have run for 10 minutes.

```
[global]
update_check = true
```

```

sanity_check = true
cluster_template = torque

[aws]
aws_region_name = <your AWS Region>

[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

Starting with version 2.11.5, AWS ParallelCluster doesn't support the use of SGE or Torque schedulers. If you use these versions, you can continue using them, or troubleshooting support from the AWS service and AWS Support teams.

AWS Batch (awsbatch)

The following example launches a cluster with the `awsbatch` scheduler. It's set to select the better instance type based on your job resource needs.

The example configuration allows a maximum of 40 concurrent vCPUs, and scales down to zero when no jobs have run for 10 minutes (adjustable using the [scaledown_idletime](#) setting).

```

[global]
update_check = true
sanity_check = true
cluster_template = awsbatch

[aws]

```



```
aws_region_name = <your AWS Region>

[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
```

How AWS ParallelCluster works

AWS ParallelCluster was built not only as a way to manage clusters, but as a reference on how to use AWS services to build your HPC environment.

Topics

- [AWS ParallelCluster processes](#)
- [AWS services used by AWS ParallelCluster](#)
- [AWS ParallelCluster Auto Scaling](#)

AWS ParallelCluster processes

This section applies only to HPC clusters that are deployed with one of the supported traditional job schedulers (SGE, Slurm, or Torque). When used with these schedulers, AWS ParallelCluster manages compute node provisioning and removal by interacting with both the Auto Scaling group and the underlying job scheduler.

For HPC clusters that are based on AWS Batch, AWS ParallelCluster relies on the capabilities provided by the AWS Batch for the compute node management.

Note

Starting with version 2.11.5, AWS ParallelCluster doesn't support the use of SGE or Torque schedulers. You can continue using them in versions up to and including 2.11.4, but they aren't eligible for future updates or troubleshooting support from the AWS service and AWS Support teams.

Topics

- [SGE and Torque integration processes](#)
- [Slurm integration processes](#)

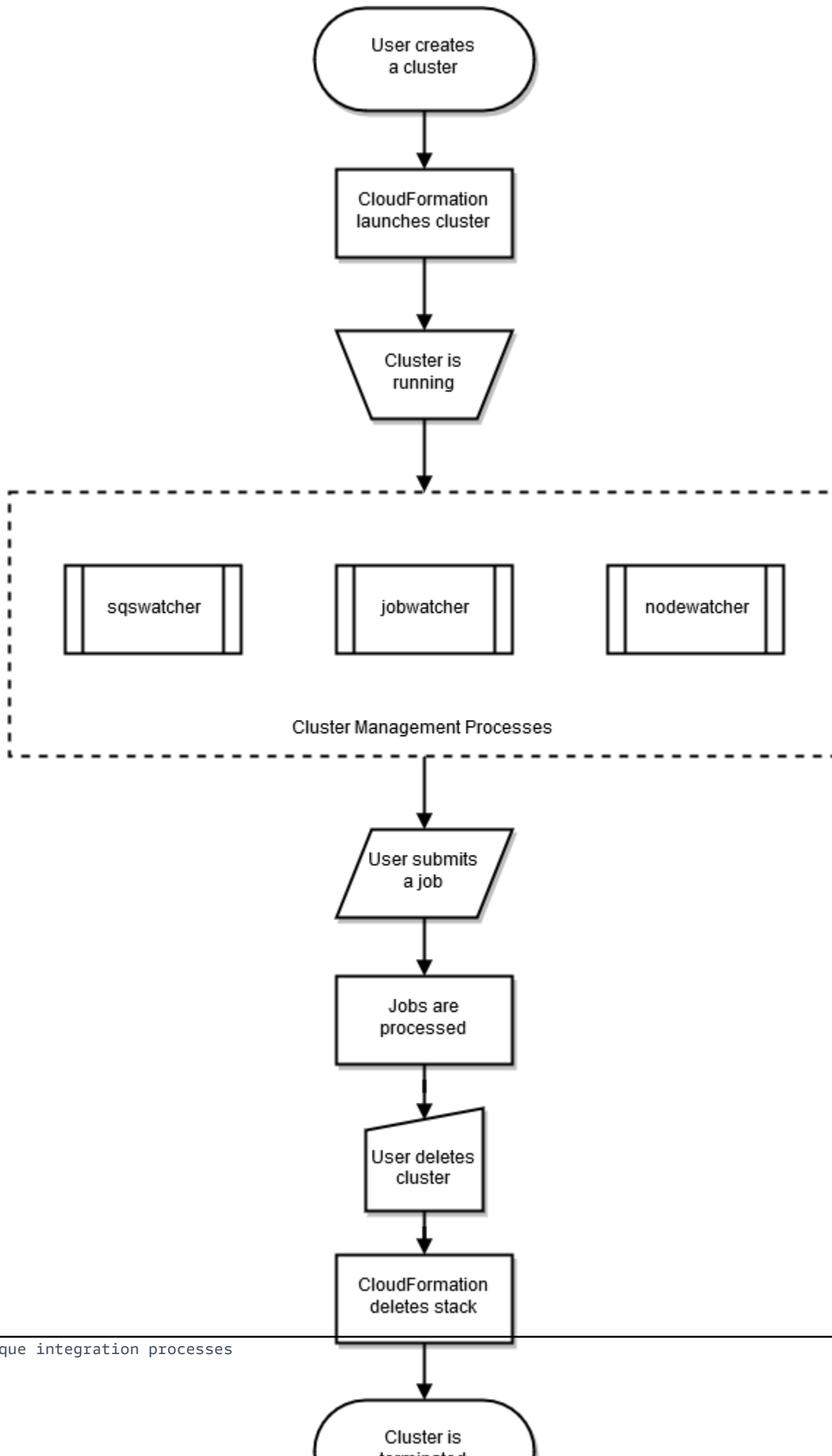
SGE and Torque integration processes

Note

This section only applies to AWS ParallelCluster versions up to and including version 2.11.4. Starting with version 2.11.5, AWS ParallelCluster doesn't support the use of SGE and Torque schedulers, Amazon SNS, and Amazon SQS.

General overview

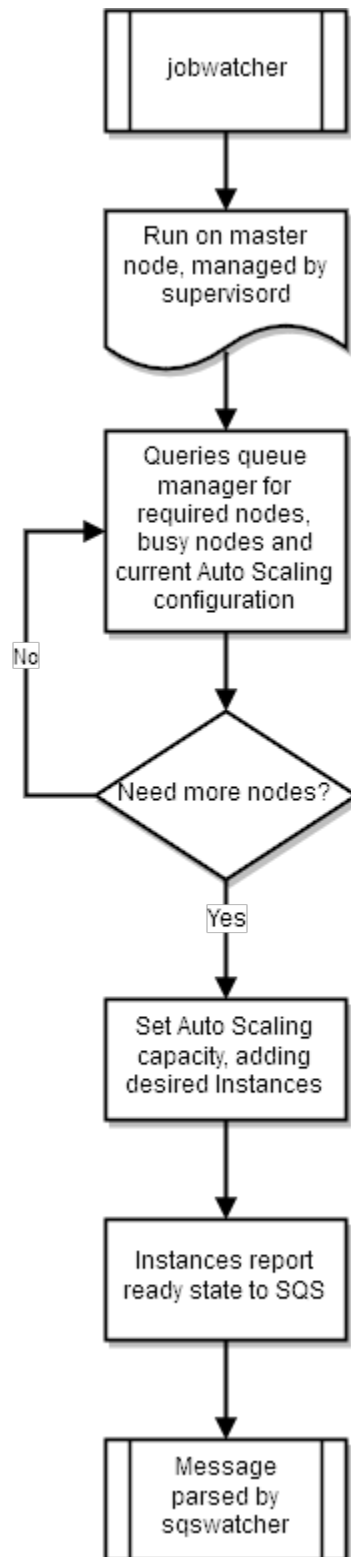
A cluster's lifecycle begins after it is created by a user. Typically, a cluster is created from the Command Line Interface (CLI). After it's created, a cluster exists until it's deleted. AWS ParallelCluster daemons run on the cluster nodes, mainly to manage the HPC cluster elasticity. The following diagram shows a user workflow and the cluster lifecycle. The sections that follow describe the AWS ParallelCluster daemons that are used to manage the cluster.



With SGE and Torque schedulers, AWS ParallelCluster uses `nodewatcher`, `jobwatcher`, and `sqswatcher` processes.

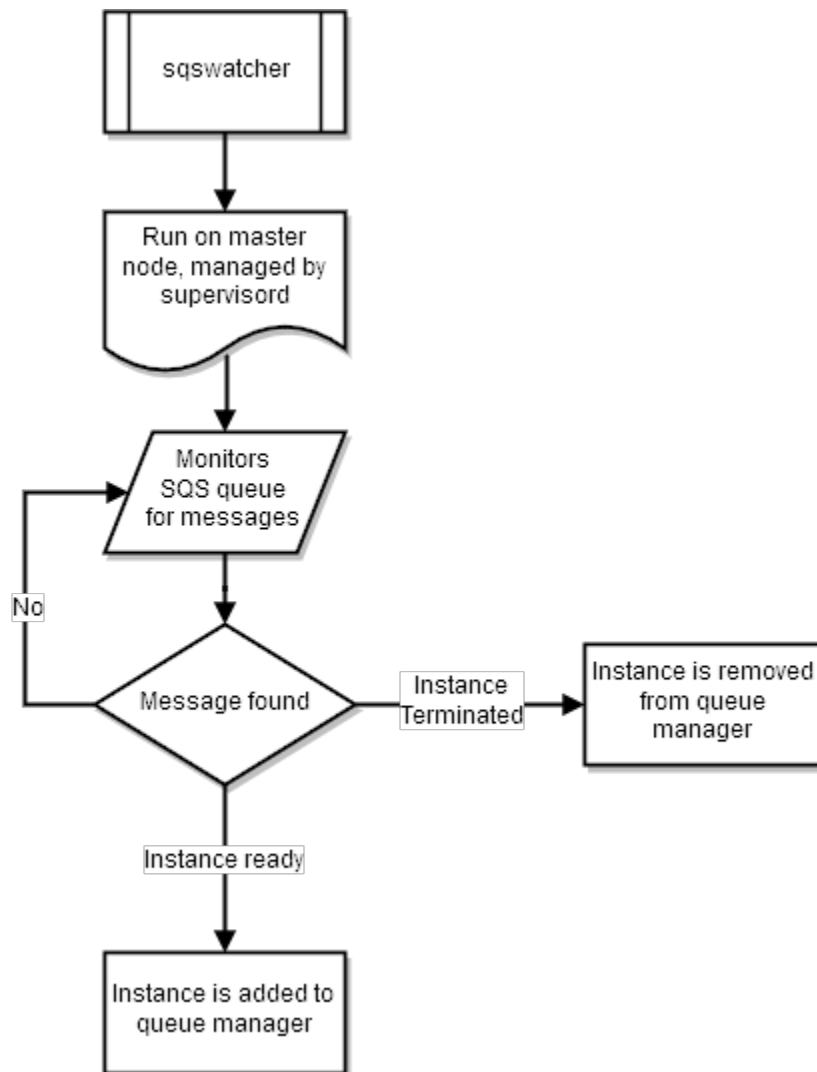
jobwatcher

When a cluster is running, a process owned by the root user monitors the configured scheduler (SGE or Torque). Each minute it evaluates the queue in order to decide when to scale up.



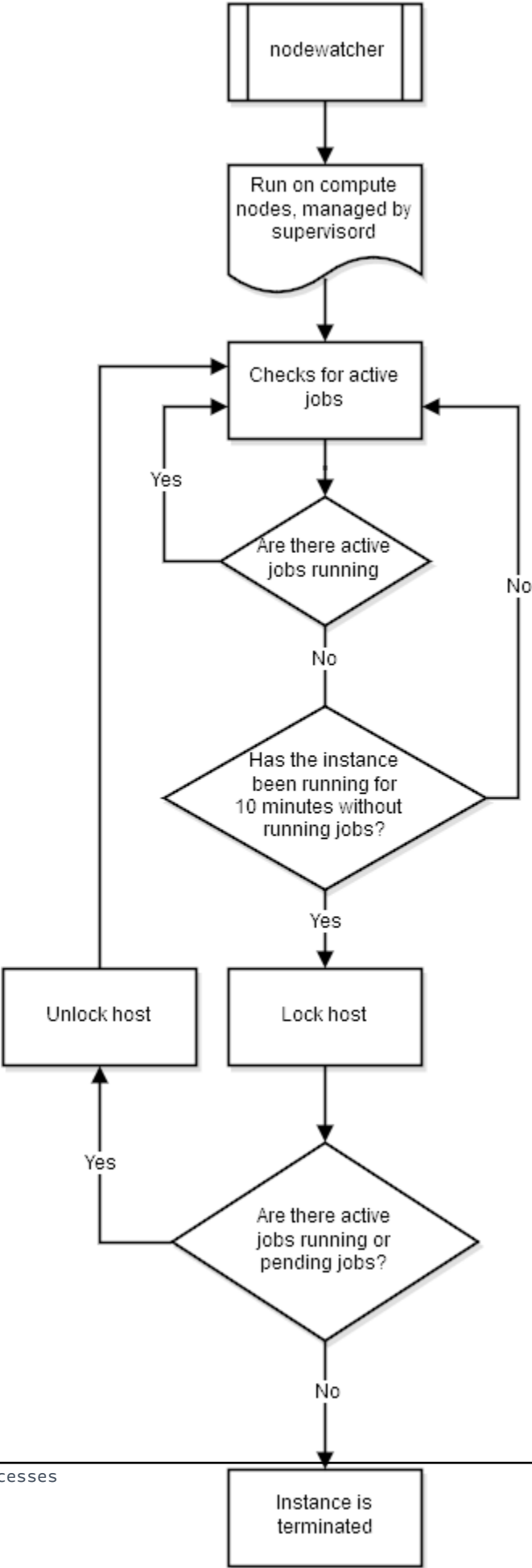
sqswatcher

The `sqswatcher` process monitors for Amazon SQS messages that are sent by Auto Scaling to notify you of state changes within the cluster. When an instance comes online, it submits an "instance ready" message to Amazon SQS. This message is picked up by `sqswatcher`, running on the head node. These messages are used to notify the queue manager when new instances come online or are terminated, so they can be added or removed from the queue.



nodewatcher

The `nodewatcher` process runs on each node in the compute fleet. After the `scaledown_idletime` period, as defined by the user, the instance is terminated.



Slurm integration processes

With Slurm schedulers, AWS ParallelCluster uses `clustermgtd` and `computemgt` processes.

`clustermgtd`

Clusters that run in heterogeneous mode (indicated by specifying a [queue_settings](#) value) have a cluster management daemon (`clustermgtd`) process that runs on the head node. These tasks are performed by the cluster management daemon.

- Inactive partition clean-up
- Static capacity management: make sure static capacity is always up and healthy
- Sync scheduler with Amazon EC2.
- Orphaned instance clean-up
- Restore scheduler node status on Amazon EC2 termination that happens outside of the suspend workflow
- Unhealthy Amazon EC2 instances management (failing Amazon EC2 health checks)
- Scheduled maintenance events management
- Unhealthy Scheduler nodes management (failing Scheduler health checks)

`computemgtd`

Clusters that run in heterogeneous mode (indicated by specifying a [queue_settings](#) value) have compute management daemon (`computemgtd`) processes that run on each of the compute node. Every five (5) minutes, the compute management daemon confirms that the head node can be reached and is healthy. If five (5) minutes pass during which the head node cannot be reached or is not healthy, the compute node is shut down.

AWS services used by AWS ParallelCluster

The following Amazon Web Services (AWS) services are used by AWS ParallelCluster.

Topics

- [AWS Auto Scaling](#)
- [AWS Batch](#)
- [CloudFormation](#)

- [Amazon CloudWatch](#)
- [Amazon CloudWatch Logs](#)
- [AWS CodeBuild](#)
- [Amazon DynamoDB](#)
- [Amazon Elastic Block Store](#)
- [Amazon Elastic Compute Cloud](#)
- [Amazon Elastic Container Registry](#)
- [Amazon EFS](#)
- [Amazon FSx for Lustre](#)
- [AWS Identity and Access Management](#)
- [AWS Lambda](#)
- [Amazon DCV](#)
- [Amazon Route 53](#)
- [Amazon Simple Notification Service](#)
- [Amazon Simple Queue Service](#)
- [Amazon Simple Storage Service](#)
- [Amazon VPC](#)

AWS Auto Scaling

Note

This section only applies to AWS ParallelCluster versions up to and including version 2.11.4. Starting with version 2.11.5, AWS ParallelCluster doesn't support the use of AWS Auto Scaling.

AWS Auto Scaling is a service that monitors your applications and automatically adjusts capacity based on your specific and changing service requirements. This service manages your ComputeFleet instances as an Auto Scaling group. The group can be elastically driven by your changing workload or statically fixed by your initial instance configurations.

AWS Auto Scaling is used with ComputeFleet instances but is not used with AWS Batch clusters.

For more information about AWS Auto Scaling, see <https://aws.amazon.com/autoscaling/> and <https://docs.aws.amazon.com/autoscaling/>.

AWS Batch

AWS Batch is an AWS managed job scheduler service. It dynamically provisions the optimal quantity and type of compute resources (for example, CPU or memory-optimized instances) in AWS Batch clusters. These resources are provisioned based on the specific requirements of your batch jobs, including volume requirements. With AWS Batch, you don't need to install or manage additional batch computing software or server clusters to run your jobs effectively.

AWS Batch is used only with AWS Batch clusters.

For more information about AWS Batch, see <https://aws.amazon.com/batch/> and <https://docs.aws.amazon.com/batch/>.

CloudFormation

CloudFormation is an infrastructure-as-code service that provides a common language to model and provision AWS and third-party application resources in your cloud environment. It is the main service used by AWS ParallelCluster. Each cluster in AWS ParallelCluster is represented as a stack, and all resources required by each cluster are defined within the AWS ParallelCluster CloudFormation template. In most cases, AWS ParallelCluster CLI commands directly correspond to CloudFormation stack commands, such as create, update, and delete commands. Instances that are launched within a cluster make HTTPS calls to the CloudFormation endpoint in the AWS Region where the cluster is launched.

For more information about CloudFormation, see <https://aws.amazon.com/cloudformation/> and <https://docs.aws.amazon.com/cloudformation/>.

Amazon CloudWatch

Amazon CloudWatch (CloudWatch) is a monitoring and observability service that provides you with data and actionable insights. These insights can be used to monitor your applications, respond to performance changes and service exceptions, and optimize resource utilization. In AWS ParallelCluster, CloudWatch is used for a dashboard, to monitor and log Docker image build steps and the output of the AWS Batch jobs.

Before AWS ParallelCluster version 2.10.0, CloudWatch was used only with AWS Batch clusters.

For more information about CloudWatch, see <https://aws.amazon.com/cloudwatch/> and <https://docs.aws.amazon.com/cloudwatch/>.

Amazon CloudWatch Logs

Amazon CloudWatch Logs (CloudWatch Logs) is one of the core features of Amazon CloudWatch. You can use it to monitor, store, view, and search the log files for many of the components used by AWS ParallelCluster.

Before AWS ParallelCluster version 2.6.0, CloudWatch Logs was only used with AWS Batch clusters.

For more information, see [Integration with Amazon CloudWatch Logs](#).

AWS CodeBuild

AWS CodeBuild (CodeBuild) is an AWS managed continuous integration service that complies source code, runs tests, and produces software packages that are ready to deploy. In AWS ParallelCluster, CodeBuild is used to automatically and transparently build Docker images when clusters are created.

CodeBuild is used only with AWS Batch clusters.

For more information about CodeBuild, see <https://aws.amazon.com/codebuild/> and <https://docs.aws.amazon.com/codebuild/>.

Amazon DynamoDB

Amazon DynamoDB (DynamoDB) is a fast and flexible NoSQL database service. It is used to store the minimal state information of the cluster. The head node tracks provisioned instances in a DynamoDB table.

DynamoDB is not used with AWS Batch clusters.

For more information about DynamoDB, see <https://aws.amazon.com/dynamodb/> and <https://docs.aws.amazon.com/dynamodb/>.

Amazon Elastic Block Store

Amazon Elastic Block Store (Amazon EBS) is a high-performance block storage service that provides persistent storage for shared volumes. All Amazon EBS settings can be passed through the configuration. Amazon EBS volumes can either be initialized empty or from an existing Amazon EBS snapshot.

For more information about Amazon EBS, see <https://aws.amazon.com/ebs/> and <https://docs.aws.amazon.com/ebs/>.

Amazon Elastic Compute Cloud

Amazon Elastic Compute Cloud (Amazon EC2) provides the computing capacity for AWS ParallelCluster. The head and compute nodes are Amazon EC2 instances. Any instance type that support HVM can be selected. The head and compute nodes can be different instance types. Moreover, if multiple queues are used, some or all of compute nodes can also be launched as a Spot Instance. Instance store volumes found on the instances are mounted as striped LVM volumes.

For more information about Amazon EC2, see <https://aws.amazon.com/ec2/> and <https://docs.aws.amazon.com/ec2/>.

Amazon Elastic Container Registry

Amazon Elastic Container Registry (Amazon ECR) is a fully managed Docker container registry that makes it easy to store, manage, and deploy Docker container images. In AWS ParallelCluster, Amazon ECR stores the Docker images that are built when clusters are created. The Docker images are then used by AWS Batch to run the containers for the submitted jobs.

Amazon ECR is used only with AWS Batch clusters.

For more information, see <https://aws.amazon.com/ecr/> and <https://docs.aws.amazon.com/ecr/>.

Amazon EFS

Amazon Elastic File System (Amazon EFS) provides a simple, scalable, and fully managed elastic NFS file system for use with AWS Cloud services and on-premises resources. Amazon EFS is used when the [efs_settings](#) setting is specified and refers to an [\[efs\] section](#). Support for Amazon EFS was added in AWS ParallelCluster version 2.1.0.

For more information about Amazon EFS, see <https://aws.amazon.com/efs/> and <https://docs.aws.amazon.com/efs/>.

Amazon FSx for Lustre

FSx for Lustre provides a high-performance file system that uses the open-source Lustre file system. FSx for Lustre is used when the [fsx_settings](#) setting is specified and refers to an [\[fsx\] section](#). Support for FSx for Lustre was added in AWS ParallelCluster version 2.2.1.

For more information about FSx for Lustre, see <https://aws.amazon.com/fsx/lustre/> and <https://docs.aws.amazon.com/fsx/>.

AWS Identity and Access Management

AWS Identity and Access Management (IAM) is used within AWS ParallelCluster to provide a least privileged IAM role for Amazon EC2 for the instance that is specific to each individual cluster. AWS ParallelCluster instances are given access only to the specific API calls that are required to deploy and manage the cluster.

With AWS Batch clusters, IAM roles are also created for the components that are involved with the Docker image building process when clusters are created. These components include the Lambda functions that are allowed to add and delete Docker images to and from the Amazon ECR repository. They also include the functions allowed to delete the Amazon S3 bucket that is created for the cluster and CodeBuild project. There are also roles for AWS Batch resources, instances, and jobs.

For more information about IAM, see <https://aws.amazon.com/iam/> and <https://docs.aws.amazon.com/iam/>.

AWS Lambda

AWS Lambda (Lambda) runs the functions that orchestrate the creation of Docker images. Lambda also manages the cleanup of custom cluster resources, such as Docker images stored in the Amazon ECR repository and on Amazon S3.

For more information about Lambda, see <https://aws.amazon.com/lambda/> and <https://docs.aws.amazon.com/lambda/>.

Amazon DCV

Amazon DCV is a high-performance remote display protocol that provides a secure way to deliver remote desktops and application streaming to any device over varying network conditions. Amazon DCV is used when the [dcv_settings](#) setting is specified and refers to an [\[dcv\] section](#). Support for Amazon DCV was added in AWS ParallelCluster version 2.5.0.

For more information about Amazon DCV, see <https://aws.amazon.com/hpc/dcv/> and <https://docs.aws.amazon.com/dcv/>.

Amazon Route 53

Amazon Route 53 (Route 53) is used to create hosted zones with hostnames and fully qualified domain names for each of the compute nodes.

For more information about Route 53, see <https://aws.amazon.com/route53/> and <https://docs.aws.amazon.com/route53/>.

Amazon Simple Notification Service

Note

This section only applies to AWS ParallelCluster versions up to and including version 2.11.4. Starting with version 2.11.5, AWS ParallelCluster doesn't support the use of Amazon Simple Notification Service.

Amazon Simple Notification Service (Amazon SNS) receives notifications from Auto Scaling. These events are called lifecycle events and are generated when an instance launches or terminates in an Auto Scaling group. Within AWS ParallelCluster, the Amazon SNS topic for the Auto Scaling group is subscribed to an Amazon SQS queue.

Amazon SNS is not used with AWS Batch clusters.

For more information about Amazon SNS, see <https://aws.amazon.com/sns/> and <https://docs.aws.amazon.com/sns/>.

Amazon Simple Queue Service

Note

This section only applies to AWS ParallelCluster versions up to and including version 2.11.4. Starting with version 2.11.5, AWS ParallelCluster doesn't support the use of Amazon Simple Queue Service.

Amazon Simple Queue Service (Amazon SQS) holds notification sent from Auto Scaling, notifications sent through Amazon SNS, and notifications sent from the compute nodes. Amazon

SQS decouples the sending of notifications from the receiving of notifications. This allows the head node to handle notifications through a polling process. In this process, the head node runs Amazon SQSwatcher and polls the queue. Auto Scaling and the compute nodes post messages to the queue.

Amazon SQS is not used with AWS Batch clusters.

For more information about Amazon SQS, see <https://aws.amazon.com/sqs/> and <https://docs.aws.amazon.com/sqs/>.

Amazon Simple Storage Service

Amazon Simple Storage Service (Amazon S3) stores AWS ParallelCluster templates located in each AWS Region. AWS ParallelCluster can be configured to allow CLI/SDK tools to use Amazon S3.

When you use AWS Batch cluster, an Amazon S3 bucket in your account is used for storing related data. For example, the bucket stores artifacts created when a Docker image and scripts are created from submitted jobs.

For more information, see <https://aws.amazon.com/s3/> and <https://docs.aws.amazon.com/s3/>.

Amazon VPC

Amazon VPC defines a network used by the nodes in your cluster. The VPC settings for the cluster are defined in the [\[vpc\] section](#).

For more information about Amazon VPC, see <https://aws.amazon.com/vpc/> and <https://docs.aws.amazon.com/vpc/>.

AWS ParallelCluster Auto Scaling

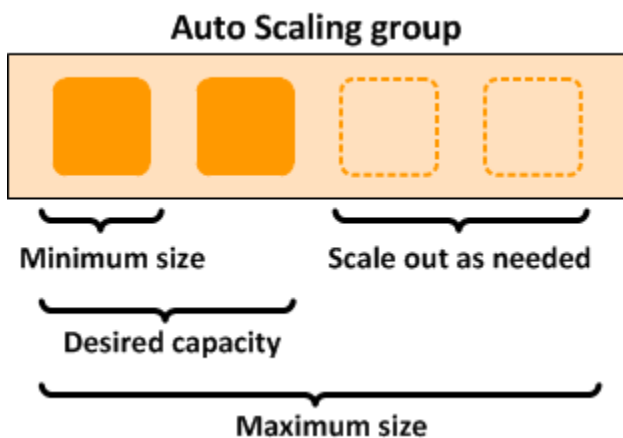
Note

This section only applies to AWS ParallelCluster versions up to and including version 2.11.4. Starting with version 2.11.5, AWS ParallelCluster doesn't support the use of SGE or Torque schedulers. You can continue using them in versions up to and including 2.11.4, but they aren't eligible for future updates or troubleshooting support from the AWS service and AWS Support teams.

Starting with AWS ParallelCluster version 2.9.0, Auto Scaling isn't supported for use with Slurm Workload Manager (Slurm). To learn about Slurm and multiple queue scaling, see the [Multiple queue mode tutorial](#).

The auto scaling strategy that's described in this topic applies to HPC clusters that are deployed with either Son of Grid Engine (SGE) or Torque Resource Manager (Torque). When deployed with one of these schedulers, AWS ParallelCluster implements the scaling capabilities by managing the Auto Scaling group of the compute nodes, and then changing the scheduler configuration as needed. For HPC clusters that are based on AWS Batch, AWS ParallelCluster relies on the elastic scaling capabilities provided by the AWS managed job scheduler. For more information, see [What is Amazon EC2 Auto Scaling](#) in the *Amazon EC2 Auto Scaling User Guide*.

Clusters deployed with AWS ParallelCluster are elastic in several ways. Setting the [initial_queue_size](#) specifies the minimum size value of the ComputeFleet Auto Scaling group, and also the desired capacity value. Setting the [max_queue_size](#) specifies the maximum size value of the ComputeFleet Auto Scaling group.



Scaling up

Every minute, a process called [jobwatcher](#) runs on the head node. It evaluates the current number of instances required by the pending jobs in the queue. If the total number of busy nodes and requested nodes is greater than the current desired value in the Auto Scaling group, it adds more instances. If you submit more jobs, the queue is re-evaluated and the Auto Scaling group is updated, up to the specified [max_queue_size](#).

With an SGE scheduler, each job requires a number of slots to run (one slot corresponds to one processing unit, for example, a vCPU). To evaluate the number of instances that are required to

serve the currently pending jobs, the `jobwatcher` divides the total number of requested slots by the capacity of a single compute node. The capacity of a compute node that corresponds to the number of available vCPUs depends on the Amazon EC2 instance type that's specified in the cluster configuration.

With Slurm (before AWS ParallelCluster version 2.9.0) and Torque schedulers, each job might require both a number of nodes and a number of slots for every node, depending on circumstance. For each request, the `jobwatcher` determines the number of compute nodes that are needed to fulfill the new computational requirements. For example, let's assume a cluster with `c5.2xlarge` (8 vCPU) as the compute instance type, and three queued pending jobs with the following requirements:

- job1: 2 nodes / 4 slots each
- job2: 3 nodes / 2 slots each
- job3: 1 node / 4 slots each

In this example, the `jobwatcher` requires three new compute instances in the Auto Scaling group to serve the three jobs.

Current limitation: auto scale up logic doesn't consider partially loaded busy nodes. For example, a node that's running a job is considered busy even if there are empty slots.

Scaling down

On each compute node, a process called [nodewatcher](#) runs and evaluates the idle time of the node. An instance is terminated when both of the following conditions are met:

- An instance has no jobs for a period of time longer than the [scaledown_idletime](#) (the default setting is 10 minutes)
- There are no pending jobs in the cluster

To terminate an instance, `nodewatcher` calls the [TerminateInstanceInAutoScalingGroup](#) API operation, which removes an instance if the size of the Auto Scaling group is at least the minimum Auto Scaling group size. This process scales down a cluster without affecting running jobs. It also enables an elastic cluster with a fixed base number of instances.

Static cluster

The value of auto scaling is the same for HPC as with any other workloads. The only difference is that AWS ParallelCluster has code that makes it interact more intelligently. For example, if a static cluster is required, you set the [initial_queue_size](#) and [max_queue_size](#) parameters to the exact size of cluster that's required,. and then you set the [maintain_initial_size](#) parameter to true. This causes the ComputeFleet Auto Scaling group to have the same value for minimum, maximum, and desired capacity.

Tutorials

The following tutorials show you how to get started with AWS ParallelCluster, and provide best practice guidance for some common tasks.

Topics

- [Running your first job on AWS ParallelCluster](#)
- [Building a Custom AWS ParallelCluster AMI](#)
- [Running an MPI job with AWS ParallelCluster and awsbatch scheduler](#)
- [Disk encryption with a custom KMS Key](#)
- [Multiple queue mode tutorial](#)

Running your first job on AWS ParallelCluster

This tutorial walks you through running your first Hello World job on AWS ParallelCluster.

Prerequisites

- AWS ParallelCluster [is installed](#).
- The AWS CLI [is installed and configured](#).
- You have an [EC2 key pair](#).
- You have an IAM role with the [permissions](#) required to run the [pcluster](#) CLI.

Verifying your installation

First, we verify that AWS ParallelCluster is correctly installed and configured.

```
$ pcluster version
```

This returns the running version of AWS ParallelCluster. If the output gives you a message about configuration, you need to run the following to configure AWS ParallelCluster:

```
$ pcluster configure
```

Creating your first cluster

Now it's time to create your first cluster. Because the workload for this tutorial isn't performance intensive, we can use the default instance size of `t2.micro`. (For production workloads, you choose an instance size that best fits your needs.)

Let's call your cluster `hello-world`.

```
$ pcluster create hello-world
```

When the cluster is created, you see output similar to the following:

```
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/
```

The message `CREATE_COMPLETE` shows that the cluster created successfully. The output also provides us with the public and private IP addresses of our head node. We need this IP to log in.

Logging into your head node

Use your OpenSSH pem file to log into your head node.

```
pcluster ssh hello-world -i /path/to/keyfile.pem
```

After you log in, run the command `qhost` to verify that your compute nodes are set up and configured.

```
$ qhost
```

HOSTNAME	ARCH	NCPU	NSOC	NCOR	NTHR	LOAD	MEMTOT	MEMUSE	SWAPTOT
global	-	-	-	-	-	-	-	-	-
ip-192-168-1-125	lx-amd64	2	1	2	2	0.15	3.7G	130.8M	1024.0M

ip-192-168-1-126	1x-amd64	2	1	2	2	0.15	3.7G	130.8M	1024.0M
0.0									

The output shows that we have two compute nodes in our cluster, both with two threads available to them.

Running your first job using SGE

Note

This example only applies to AWS ParallelCluster versions up to and including version 2.11.4. Starting with version 2.11.5, AWS ParallelCluster doesn't support the use of SGE or Torque schedulers.

Next, we create a job that sleeps for a little while and then outputs its own hostname.

Create a file called `hellojob.sh`, with the following contents.

```
#!/bin/bash
sleep 30
echo "Hello World from $(hostname)"
```

Next, submit the job using `qsub`, and verify that it runs.

```
$ qsub hellojob.sh
Your job 1 ("hellojob.sh") has been submitted
```

Now, you can view your queue and check the status of the job.

```
$ 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
```

The output shows that the job is currently in a running state. Wait 30 seconds for the job to finish, and then run `qstat` again.

```
$ qstat
$
```

Now that there are no jobs in the queue, we can check for output in our current directory.

```
$ 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
```

In the output, we see an "e1" and "o1" file in our job script. Because the e1 file is empty, there was no output to stderr. If we view the o1 file, we can see output from our job.

```
$ cat hellojob.sh.o1
Hello World from ip-192-168-1-125
```

The output also shows that our job ran successfully on instance ip-192-168-1-125.

To learn more about creating and using clusters, see [Best practices](#).

Building a Custom AWS ParallelCluster AMI

Important

We don't recommend building a custom AMI as an approach for customizing AWS ParallelCluster.

This is because, after you build your own AMI, you no longer receive updates or bug fixes with future releases of AWS ParallelCluster. Moreover, if you build a custom AMI, you must repeat the steps that you used to create your custom AMI with each new AWS ParallelCluster release.

Before reading any further, we recommend that you first check out the [Custom Bootstrap Actions](#) section to determine if the modifications you want to make can be scripted and supported with future AWS ParallelCluster releases.

Even though building a custom AMI isn't ideal (because of the reasons mentioned earlier), there are still scenarios where building a custom AMI for AWS ParallelCluster is necessary. This tutorial guides you through the process of building a custom AMI for these scenarios.

Note

Starting with AWS ParallelCluster version 2.6.1, most of the install recipes are skipped by default when launching nodes. This is to improve startup times. To run all of the install recipes for better backwards compatibility at the expense of startup times, add "skip_install_recipes" : "no" to the cluster key in the [extra_json](#) setting. For example:

```
extra_json = { "cluster" : { "skip_install_recipes" : "no" } }
```

Prerequisites

- AWS ParallelCluster [is installed](#).
- The AWS CLI [is installed and configured](#).
- You have an [EC2 key pair](#).
- You have an IAM role with the [permissions](#) required to run the [pcluster](#) CLI.

How to Customize the AWS ParallelCluster AMI

There are three ways to use a custom AWS ParallelCluster AMI described in the next sections. Two of these three methods require you to build a new AMI that's available under your AWS account. The third method (Use a Custom AMI at Runtime) doesn't require that you build anything in advance, but does add risk to the deployment. Choose the method that best fits your needs.

Modify an AMI

This is the safest and most recommended method. Because the base AWS ParallelCluster AMI is often updated with new releases, this AMI has all of the components required for AWS ParallelCluster to function when it's installed and configured. You can start with this as the base.

New EC2 console

1. In the AWS ParallelCluster AMI list, find the AMI that corresponds to the specific AWS Region that you use. The AMI list that you choose must match the version of AWS ParallelCluster that you use. Run `pcluster version` to verify the version. For AWS ParallelCluster version 2.11.9, go to <https://github.com/aws/aws-parallelcluster/blob/v2.11.9/amis.txt>. To select another version, use the same link, choose the **Tag: 2.11.9** button, select the **Tags** tab, and then select the appropriate version.
2. Sign in to the AWS Management Console and open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
3. In the **Amazon EC2 Dashboard**, choose **Launch instance**.
4. In **Application and OS images**, choose **Browse more AMIs**, navigate to **Community AMIs**, and enter the AWS ParallelCluster AMI ID for your AWS Region into the search box.
5. **Select** the AMI, choose your **Instance type** and properties, select your **Key pair**, and **Launch instance**.
6. Log into your instance using the OS user and your SSH key. For more information, navigate to **Instances**, select the new instance, and **Connect**.
7. Customize your instance as required.
8. Run the following command to prepare your instance for AMI creation:

```
sudo /usr/local/sbin/ami_cleanup.sh
```

9. Navigate to **Instances**, choose the new instance, select **Instance state**, and **Stop instance**.
10. Create a new AMI from the instance using the EC2 console or AWS CLI [create-image](#).

From the EC2 console

- a. Choose **Instances** in the navigation pane.
 - b. Choose the instance you created and modified.
 - c. In **Actions**, choose **Image and templates**, and then **Create image**.
 - d. Choose **Create Image**.
11. Enter the new AMI id in the [custom_ami](#) field in your cluster configuration.

Old EC2 console

1. In the AWS ParallelCluster AMI list, find the AMI that corresponds to the specific AWS Region that you use. The AMI list that you choose must match the version of AWS ParallelCluster that you use. Run `pcluster version` to verify the version. For AWS ParallelCluster version 2.11.9, go to <https://github.com/aws/aws-parallelcluster/blob/v2.11.9/amis.txt>. To select another version, use the same link, choose the **Tag: 2.11.9** button, select the **Tags** tab, and then select the appropriate version.
2. Sign in to the AWS Management Console and open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
3. In the **Amazon EC2 Dashboard**, choose **Launch instance**.
4. Choose **Community AMIs**, search for the AWS ParallelCluster AMI ID, and **Select** it.
5. Choose your instance type and select **Next: Configure Instance Details**, or **Review and Launch** to launch your instance.
6. Choose **Launch**, select your **Key pair**, and **Launch Instances**.
7. Log into your instance using the OS user and your SSH key. For more information, navigate to **Instances**, select the new instance, and **Connect**.
8. Customize your instance as required.
9. Run the following command to prepare your instance for AMI creation:

```
sudo /usr/local/sbin/ami_cleanup.sh
```

10. Navigate to **Instances**, choose the new instance, select **Instance State**, and **Stop**.
11. Create a new AMI from the instance using the EC2 console or AWS CLI [create-image](#).

From the EC2 console

- a. Choose **Instances** in the navigation pane.
 - b. Choose the instance you created and modified.
 - c. In **Actions**, choose **Image**, and then **Create Image**.
 - d. Choose **Create Image**.
12. Enter the new AMI id in the [custom_ami](#) field in your cluster configuration.

Build a Custom AWS ParallelCluster AMI

If you have a customized AMI and software already in place, you can apply the changes needed by AWS ParallelCluster on top of it.

1. Install the following in your local system, together with the AWS ParallelCluster CLI:
 - Packer: find the latest version for your OS from the [Packer website](#), and install it. The version must be at least 1.4.0, but the latest version is recommended. Verify that the `packer` command is available in your PATH.

Note

Before AWS ParallelCluster version 2.8.0, [Berkshelf](#) (which is installed by using `gem install berkshelf`) was required to use `pcluster createami`.

2. Configure your AWS account credentials so that Packer can make calls to AWS API operations on your behalf. The minimal set of required permissions necessary for Packer to work are documented in the [IAM Task or Instance Role](#) section of the *Amazon AMI Builder* topic in the Packer documentation.
3. Use the command `createami` in the AWS ParallelCluster CLI to build an AWS ParallelCluster AMI starting from the one that you provide as base:

```
pcluster createami --ami-id <BASE_AMI> --os <BASE_AMI_OS>
```

Important

You shouldn't use an AWS ParallelCluster AMI from a running cluster as `<BASE_AMI>` for the `createami` command. Otherwise, the command fails.

For other parameters, see [pcluster createami](#).

4. The command in Step 4 runs Packer, which specifically does the following:
 - a. Launches an instance using the base AMI provided.
 - b. Applies the AWS ParallelCluster cookbook to the instance to install relevant software and perform other necessary configuration tasks.
 - c. Stops the instance.

- d. Creates a new AMI from the instance.
 - e. Terminates the instance after the AMI is created.
 - f. Outputs the new AMI ID string to use to create your cluster.
5. To create your cluster, enter the AMI ID in the [custom_ami](#) field within your cluster configuration.

Note

The instance type used to build a custom AWS ParallelCluster AMI is `t2.xlarge`. This instance type doesn't qualify for the AWS free tier, so you're charged for any instances that are created when you build this AMI.

Use a Custom AMI at Runtime

Warning

To avoid the risk of using an AMI that's not compatible with AWS ParallelCluster, we recommend that you avoid using this method.

When compute nodes are launched with potentially untested AMIs at runtime, incompatibilities with the runtime installation of AWS ParallelCluster's required software might cause AWS ParallelCluster to stop working.

If you don't want to create anything in advance, you can use your AMI and create an AWS ParallelCluster from that AMI.

With this method, it takes longer for the AWS ParallelCluster to be created because all the software that's needed by AWS ParallelCluster when the cluster is created must be installed. Moreover, scaling up also takes a longer time.

- Enter the AMI id in the [custom_ami](#) field within your cluster configuration.

Running an MPI job with AWS ParallelCluster and awsbatch scheduler

This tutorial walks you through running an MPI job with awsbatch as a scheduler.

Prerequisites

- AWS ParallelCluster [is installed](#).
- The AWS CLI [is installed and configured](#).
- You have an [EC2 key pair](#).
- You have an IAM role with the [permissions](#) required to run the [pcluster](#) CLI.

Creating the cluster

First, let's create a configuration for a cluster that uses awsbatch as the scheduler. Make sure to insert the missing data in the vpc section and the key_name field with the resources that you created at configuration time.

```
[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-#####

```

You can now start the creation of the cluster. Let's call our cluster *awsbatch-tutorial*.

```

$ pcluster create -c /path/to/the/created/config/aws_batch.config -t awsbatch awsbatch-tutorial

```

When the cluster is created, you see output similar to the following:

```

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

```

Logging into your head node

The [AWS ParallelCluster Batch CLI](#) commands are all available on the client machine where AWS ParallelCluster is installed. However, we are going to SSH into the head node and submit the jobs from there. This allows us to take advantage of the NFS volume that is shared between the head and all Docker instances that run AWS Batch jobs.

Use your SSH pem file to log into your head node.

```

$ pcluster ssh awsbatch-tutorial -i /path/to/keyfile.pem

```

When you are logged in, run the commands `awsbqueues` and `awsbhosts` to show the configured AWS Batch queue and the running Amazon ECS instances.

```

[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

```

As you can see from the output, we have one single running host. This is due to the value we chose for `min_vcpus` in the configuration. If you want to display additional details about the AWS Batch queue and hosts, add the `-d` flag to the command.

Running your first job using AWS Batch

Before moving to MPI, let's create a dummy job that sleeps for a little while and then outputs its own hostname, greeting the name passed as a parameter.

Create a file called "hellojob.sh" with the following content.

```

#!/bin/bash

sleep 30
echo "Hello $1 from $HOSTNAME"
echo "Hello $1 from $HOSTNAME" > "/shared/secret_message_for_${1}_by_
${AWS_BATCH_JOB_ID}"

```

Next, submit the job using `awsbsub` and verify that it runs.

```

$ awsbsub -jn hello -cf hellojob.sh Luca
Job 6efe6c7c-4943-4c1a-baf5-edbfecab5d2 (hello) has been submitted.

```

View your queue, and check the status of the job.

```

$ awsbstat
jobId              jobName    status    startedAt
stoppedAt    exitCode
-----
-----
6efe6c7c-4943-4c1a-baf5-edbfecab5d2  hello      RUNNING   2018-11-12 09:41:29  -
-

```

The output provides detailed information for the job.

```

$ awsbstat 6efe6c7c-4943-4c1a-baf5-edbfecab5d2

```

```

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
-----

```

Note that the job is currently in a RUNNING state. Wait 30 seconds for the job to finish, and then run `awsbststat` again.

```

$ awsbststat
jobId           jobName      status      startedAt
stoppedAt      exitCode
-----
-----
-----
-----

```

Now you can see that the job is in the SUCCEEDED status.

```

$ awsbststat -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

```


Because there are no jobs in the queue now, we can check for output through the `awsbout` command.

```
$ 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
```

We can see that our job successfully ran on instance "ip-172-31-4-234".

If you look into the `/shared` directory, you find a secret message for you.

To explore all of the available features that are not part of this tutorial, see the [AWS ParallelCluster Batch CLI documentation](#). When you are ready to continue the tutorial, let's move on and see how to submit an MPI job.

Running an MPI job in a multi-node parallel environment

While still logged into the head node, create a file in the `/shared` directory named `mpi_hello_world.c`. Add the following MPI program to the file:

```
// 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();
}

```

Now save the following code as `submit_mpi.sh`:

```

#!/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

```

We are now ready to submit our first MPI job and make it run concurrently on three nodes:

```
$ awsbsub -n 3 -cf submit_mpi.sh
```

Now let's monitor the job status, and wait for it to enter the RUNNING status:

```
$ watch awsbstat -d
```

When the job enters the RUNNING status, we can look at its output. To show the output of the main node, append #0 to the job id. To show the output of the compute nodes, use #1 and #2:

```

[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!
```

We can now confirm that the job completed successfully:

```
[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  -
```

Note: if you want to terminate a job before it ends, you can use the `awsbkill` command.

Disk encryption with a custom KMS Key

AWS ParallelCluster supports the configuration options `ebs_kms_key_id` and `fsx_kms_key_id`. These options allow you to provide a custom AWS KMS key for Amazon EBS Disk encryption or FSx for Lustre. To use them, you specify an `ec2_iam_role`.

In order for the cluster to create, the AWS KMS key must know the name of the cluster's role. This prevents you from using the role created on cluster create, requiring a custom `ec2_iam_role`.

Prerequisites

- AWS ParallelCluster [is installed](#).
- The AWS CLI [is installed and configured](#).
- You have an [EC2 key pair](#).
- You have an IAM role with the [permissions](#) required to run the [pcluster](#) CLI.

Creating the role

First you create a policy:

1. Go to the IAM Console: <https://console.aws.amazon.com/iam/home>.
2. Under **Policies**, **Create policy**, click the **JSON** tab.
3. As the policy's body, paste in the [Instance Policy](#). Make sure to replace all occurrences of `<AWS ACCOUNT ID>` and `<REGION>`.

4. Name the policy `ParallelClusterInstancePolicy`, and then click **Create Policy**.

Next create a role:

1. Under **Roles**, create a role.
2. Click EC2 as the trusted entity.
3. Under **Permissions**, search for the `ParallelClusterInstancePolicy` role that you just created, and attach it.
4. Name the role `ParallelClusterInstanceRole`, and then click **Create Role**.

Give your key permissions

In the AWS KMS Console > **Customer managed keys** > click your key's **Alias** or **Key ID**.

Click the **Add** button in the **Key users** box, underneath the **Key policy** tab, and search for the `ParallelClusterInstanceRole` you just created. Attach it.

Creating the cluster

Now create a cluster. The following is an example of a cluster with encrypted Raid 0 drives:

```
[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-xxxxxxxxxxxxxx
```

The following is an example with the FSx for Lustre file system:

```
[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-xxxxxxxxxxxxxx
```

Similar configurations apply to Amazon EBS and Amazon FSx based file systems.

Multiple queue mode tutorial

Running your jobs on AWS ParallelCluster with multiple queue mode

This tutorial walks you through running your first Hello World job on AWS ParallelCluster with [Multiple queue mode](#).

Prerequisites

- AWS ParallelCluster [is installed](#).
- The AWS CLI [is installed and configured](#).
- You have an [EC2 key pair](#).
- You have an IAM role with the [permissions](#) required to run the [pcluster](#) CLI.

Note

Multiple queue mode is only supported for AWS ParallelCluster version 2.9.0 or later.

Configuring your cluster

First, verify that AWS ParallelCluster is correctly installed by running the following command.

```
$ pcluster version
```

For more information about `pcluster` version, see [pcluster version](#).

This command returns the running version of AWS ParallelCluster.

Next, run `pcluster configure` to generate a basic configuration file. Follow all the prompts that follow this command.

```
$ pcluster configure
```

For more information about the `pcluster configure` command, see [pcluster configure](#).

After you complete this step, you should have a basic configuration file under `~/.parallelcluster/config`. This file should contain a basic cluster configurations and a VPC section.

This next part of the tutorial outlines how to modify your newly created configuration and launch a cluster with multiple queues.

Note

Some instances used in this tutorial aren't free-tier eligible.

For this tutorial, use the following configuration.

```
[global]
update_check = true
sanity_check = true
cluster_template = multi-queue

[aws]
aws_region_name = <Your AWS Region>

[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

```

Creating your cluster

This section details how to create the multiple queue mode cluster.

First, name your cluster `multi-queue-hello-world`, and create the cluster according to the `multi-queue` cluster section defined in the previous section.

```
$ pcluster create multi-queue-hello-world -t multi-queue
```

For more information about `pcluster create`, see [pcluster create](#).

When the cluster is created, the following output is displayed:

```
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
```

The message `CREATE_COMPLETE` indicates that the cluster was successfully created. The output also provides the public and private IP addresses of the head node.

Logging into your head node

Use your private SSH key file to log into your head node.

```
$ pcluster ssh multi-queue-hello-world -i ~/path/to/keyfile.pem
```

For more information about `pcluster ssh`, see [pcluster ssh](#).

After logging in, run the `sinfo` command to verify that your scheduler queues are set up and configured.

For more information about `sinfo`, see [sinfo](#) in the *Slurm documentation*.

```
$ 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
```

The output shows that you have two `t2.micro` compute nodes in the `idle` state that are available in your cluster.

Note

- `spot-st-t2micro-1` is a static node with `st` in its name. This node is always available and corresponds to the [min_count](#) = 1 in your cluster configuration.

- `spot-dy-t2micro-1` is a dynamic node with `dy` in its name. This node is currently available because it corresponds to `initial_count - min_count = 1` according to your cluster configuration. This node scales down after your custom `scaledown_idletime` of five minutes.

Other nodes are all in power saving state, shown by the `~` suffix in node state, with no EC2 instances backing them. The default queue is designated by a `*` suffix after its queue name, so `spot` is your default job queue.

Running job in multiple queue mode

Next, try to run a job to sleep for a while. The job will later output its own hostname. Make sure this script can be run by the current user.

```
$ 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
```

Submit the job using the `sbatch` command. Request two nodes for this job with the `-N 2` option, and verify that the job submits successfully. For more information about `sbatch`, see [sbatch](#) in the *Slurm documentation*.

```
$ sbatch -N 2 --wrap "srun hellojob.sh"
Submitted batch job 2
```

You can view your queue and check the status of the job with the `squeue` command. Note that, because you didn't specify a specific queue, the default queue (`spot`) is used. For more information about `squeue`, see [squeue](#) in the *Slurmdocumentation*.

```
$ 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								

The output shows that the job is currently in a running state. Wait 30 seconds for the job to finish, and then run `squeue` again.

```
$ squeue
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
-------	-----------	------	------	----	------	-------	------------------

Now that the jobs in the queue have all finished, look for the output file `slurm-2.out` in your current directory.

```
$ cat slurm-2.out
Hello World from spot-dy-t2micro-1
Hello World from spot-st-t2micro-1
```

The output also shows that our job ran successfully on the `spot-st-t2micro-1` and `spot-st-t2micro-2` nodes.

Now submit the same job by specifying constraints for specific instances with the following commands.

```
$ sbatch -N 3 -p spot -C "[c5.xlarge*1&t2.micro*2]" --wrap "srun hellojob.sh"
Submitted batch job 3
```

You used these parameters for `sbatch`.

- `-N 3`— requests three nodes
- `-p spot`— submits the job to the spot queue. You can also submit a job to the ondemand queue by specifying `-p ondemand`.
- `-C "[c5.xlarge*1&t2.micro*2]"`— specifies the specific node constraints for this job. This requests one (1) `c5.xlarge` node and two (2) `t2.micro` nodes to be used for this job.

Run the `sinfo` command to view the nodes and queues. (Queues in AWS ParallelCluster are called partitions in Slurm.)

```
$ sinfo
```

PARTITION	AVAIL	TIMELIMIT	NODES	STATE	NODELIST
ondemand	up	infinite	10	idle~	ondemand-dy-c52xlarge-[1-10]
spot*	up	infinite	1	mix#	spot-dy-c5xlarge-1
spot*	up	infinite	17	idle~	spot-dy-c5xlarge-[2-10],spot-dy-t2micro-[2-9]

```
spot*      up    infinite    2  alloc spot-dy-t2micro-1,spot-st-t2micro-1
```

The nodes are powering up. This is signified by the # suffix on the node state. Run the `squeue` command to view information about the jobs in the cluster.

```
$ 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
```

Your job is in the CF (CONFIGURING) state, waiting for instances to scale up and join the cluster.

After about three minutes, the nodes should be available and the job enters the R (RUNNING) state.

```
$ 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
```

The job finishes, and all three nodes are in the idle state.

```
$ 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
```

Then, after there are no jobs remaining in the queue, you can check for `slurm-3.out` in your local directory.

```
$ 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
```

The output also shows that the job ran successfully on the corresponding nodes.

You can observe the scale down process. In your cluster configuration that you specified a custom [scaledown_idletime](#) of 5 minutes. After five minutes in an idle state, your dynamic nodes `spot-dy-c5xlarge-1` and `spot-dy-t2micro-1` automatically scale down and enter into the `POWER_DOWN` mode. Note that static node `spot-st-t2micro-1` doesn't scale down.

```
$ 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
```

From the above code, you can see that `spot-dy-c5xlarge-1` and `spot-dy-t2micro-1` are in `POWER_DOWN` mode. This is indicated by the `%` suffix. The corresponding instances are immediately terminated, but nodes remain in the `POWER_DOWN` state and aren't available for use for 120 seconds (two minutes). After this time, the nodes return in power saving and are available for use again. For more information, see [Slurm guide for multiple queue mode](#).

This should be the final state of the cluster:

```
$ 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
```

After logging off of the cluster, you can clean up by running `pcluster delete`. For more information, about `pcluster list` and `pcluster delete`, see [pcluster list](#) and [pcluster delete](#).

```
$ pcluster list
multi-queue CREATE_COMPLETE 2.11.9
$ pcluster delete multi-queue
Deleting: multi-queue
```

...

Running jobs on cluster with EFA and GPU instances

This part of the tutorial details how to modify the configuration and launch a cluster with multiple queues that contains instances with EFA networking and GPU resources. Note that instances used in this tutorial are higher priced instances.

Check your account limits to make sure that you're authorized to use these instances before proceeding with the steps outlined in this tutorial.

Modify the configuration file by using the following.

```
[global]
update_check = true
sanity_check = true
cluster_template = multi-queue-special

[aws]
aws_region_name = <Your AWS Region>

[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
```

Create the cluster

```
$ pcluster create multi-queue-special -t multi-queue-special
```

After the cluster is created, use your private SSH key file to log into your head node.

```
$ pcluster ssh multi-queue-special -i ~/path/to/keyfile.pem
```

This should be the initial state of the cluster:

```
$ 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]
```

This section describes how to submit some jobs to check that nodes have EFA or GPU resources.

First, write the job scripts. `efa_job.sh` will sleep for 30 seconds. After which, look for EFA in the output of the `lspci` command. `gpu_job.sh` will sleep for 30 seconds. After which, run `nvidia-smi` to show the GPU information about the node.

```
$ 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
```


Submit the job with sbatch,

```
$ 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]

After a few minutes, you should see the nodes online and jobs running.

```
[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								

After the job completes, check the output. From the output in the `slurm-2.out` file, you can see that EFA is present on the `efa-dy-c5n18xlarge-1` node. . From the output in the `slurm-3.out` file, you can see the `nvidia-smi` output contains GPU information for the `gpu-dy-g38xlarge-1` node.

```
$ 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                                         |
+-----+

```

You can observe the scale down process. In the cluster configuration, you previously specified a custom [scaledown_idletime](#) of five minutes. As a result, after five minutes in an idle state, your dynamic nodes, `spot-dy-c5xlarge-1` and `spot-dy-t2micro-1`, automatically scale down and enter into the `POWER_DOWN` mode. Eventually, the nodes enter the power saving mode and are available for use again.

```

$ 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]

```

After logging off of the cluster, you can clean up by running `pcluster delete <cluster name>`.

```
$ pcluster list
multi-queue-special CREATE_COMPLETE 2.11.9
$ pcluster delete multi-queue-special
Deleting: multi-queue-special
...
```

For more information, see [Slurm guide for multiple queue mode](#).

Development

You can use the following sections to get started with the development of AWS ParallelCluster.

Important

The following sections include instructions for using a custom version of the cookbook recipes and a custom AWS ParallelCluster node package. This information covers an advanced method of customizing AWS ParallelCluster, with potential issues that can be hard to debug. The AWS ParallelCluster team highly recommends using the scripts in [Custom Bootstrap Actions](#) for customization, because post-install hooks are generally easier to debug and more portable across releases of AWS ParallelCluster.

Topics

- [Setting up a custom AWS ParallelCluster cookbook](#)
- [Setting up a custom AWS ParallelCluster node package](#)

Setting up a custom AWS ParallelCluster cookbook

Important

The following are instructions for using a custom version of the AWS ParallelCluster cookbook recipes. This is an advanced method of customizing AWS ParallelCluster, with potential issues that can be hard to debug. The AWS ParallelCluster team highly recommends using the scripts in [Custom Bootstrap Actions](#) for customization, because post-install hooks are generally easier to debug and more portable across releases of AWS ParallelCluster.

Steps

1. Identify the AWS ParallelCluster Cookbook working directory where you have cloned the [AWS ParallelCluster cookbook](#) code.

```
_cookbookDir=<path to cookbook>
```

2. Detect the current version of the AWS ParallelCluster Cookbook.

```
_version=$(grep version ${_cookbookDir}/metadata.rb|awk '{print $2}'| tr -d \')
```

3. Create an archive of the AWS ParallelCluster Cookbook and calculate its md5.

```
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. Create an Amazon S3 bucket and upload the archive, its md5, and its last modified date into the bucket. Give public readable permission through a public-read ACL.

```
_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. Add the following variables to the AWS ParallelCluster configuration file, under the [\[cluster\]](#) section.

```
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

Starting with AWS ParallelCluster version 2.6.1, most of the install recipes are skipped by default when launching nodes to improve startup times. To skip most of the install

recipes for better startup times at the expense of backwards compatibility, remove "skip_install_recipes" : "no" from the cluster key in the [extra_json](#) setting.

Setting up a custom AWS ParallelCluster node package

Warning

The following are instructions for using a custom version of the AWS ParallelCluster node package. This is an advanced method of customizing AWS ParallelCluster, with potential issues that can be hard to debug. The AWS ParallelCluster team highly recommends using the scripts in [Custom Bootstrap Actions](#) for customization, because post-install hooks are generally easier to debug and more portable across releases of AWS ParallelCluster.

Steps

1. Identify the AWS ParallelCluster node working directory where you have cloned the AWS ParallelCluster node code.

```
_nodeDir=<path to node package>
```

2. Detect the current version of the AWS ParallelCluster node.

```
_version=$(grep "version = \"${_nodeDir}/setup.py |awk '{print $3}' | tr -d \")
```

3. Create an archive of the AWS ParallelCluster Node.

```
cd "${_nodeDir}"
_stashName=$(git stash create)
git archive --format tar --prefix="aws-parallelcluster-node-${_version}/"
"${_stashName}:-HEAD" | gzip > "aws-parallelcluster-node-${_version}.tgz"
```

4. Create an Amazon S3 bucket and upload the archive into the bucket. Give public readable permission through a public-read ACL.

```
_bucket=<the bucket name>
```

```
aws s3 cp --acl public-read aws-parallelcluster-node-${_version}.tgz s3://${_bucket}/  
node/aws-parallelcluster-node-${_version}.tgz
```

5. Add the following variable to the AWS ParallelCluster configuration file, under the [\[cluster\]](#) [section](#).

```
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

Starting with AWS ParallelCluster version 2.6.1, most of the install recipes are skipped by default when launching nodes to improve startup times. To skip most of the install recipes for better startup times at the expense of backwards compatibility, remove "skip_install_recipes" : "no" from the cluster key in the [extra_json](#) setting.

AWS ParallelCluster troubleshooting

The AWS ParallelCluster community maintains a Wiki page that provides many troubleshooting tips on the [AWS ParallelCluster GitHub Wiki](#). For a list of known issues, see [Known issues](#).

Topics

- [Retrieving and preserving logs](#)
- [Troubleshooting stack deployment issues](#)
- [Troubleshooting issues in multiple queue mode clusters](#)
- [Troubleshooting issues in single queue mode clusters](#)
- [Placement groups and instance launch issues](#)
- [Directories that cannot be replaced](#)
- [Troubleshooting issues in Amazon DCV](#)
- [Troubleshooting issues in clusters with AWS Batch integration](#)
- [Troubleshooting when a resource fails to create](#)
- [Troubleshooting IAM policy size issues](#)
- [Additional support](#)

Retrieving and preserving logs

Logs are a useful resource for troubleshooting issues. Before you can use logs to troubleshoot issues for your AWS ParallelCluster resources, you should first create an archive of cluster logs. Follow the steps described in the [Creating an Archive of a Cluster's Logs](#) topic on the [AWS ParallelCluster GitHub Wiki](#) to start this process.

If one of your running clusters is experiencing issues, you should place the cluster in a STOPPED state by running the `pcluster stop <cluster_name>` command before you begin to troubleshoot. This prevents incurring any unexpected costs.

If `pcluster` stops functioning or if you want to delete a cluster while still preserving its logs, run the `pcluster delete --keep-logs <cluster_name>` command. Running this command deletes the cluster yet retains the log group that's stored in Amazon CloudWatch. For more information about this command, see the [pcluster delete](#) documentation.

Troubleshooting stack deployment issues

If your cluster fails to be created and rolls back stack creation, you can look through the following log files to diagnose the issue. You want to look for the output of `ROLLBACK_IN_PROGRESS` in these logs. The failure message should look like the following:

```
$ 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
```

To diagnose the issue, create the cluster again using [pcluster create](#), including the `--norollback` flag. Then, SSH into the cluster:

```
$ pcluster create mycluster --norollback
...
$ pcluster ssh mycluster
```

After you're logged into the head node, you should find three primary log files that you can use to pinpoint the error.

- `/var/log/cfn-init.log` is the log for the `cfn-init` script. First check this log. You're likely to see an error like `Command chef failed` in this log. Look at the lines immediately before this line for more specifics connected with the error message. For more information, see [cfn-init](#).
- `/var/log/cloud-init.log` is the log for [cloud-init](#). If you don't see anything in `cfn-init.log`, then try checking this log next.
- `/var/log/cloud-init-output.log` is the output of commands that were run by [cloud-init](#). This includes the output from `cfn-init`. In most cases, you don't need to look at this log to troubleshoot this type of issue.

Troubleshooting issues in multiple queue mode clusters

This section is relevant to clusters that were installed using AWS ParallelCluster version 2.9.0 and later with the Slurm job scheduler. For more information about multiple queue mode, see [Multiple queue mode](#).

Topics

- [Key logs](#)
- [Troubleshooting node initialization issues](#)
- [Troubleshooting unexpected node replacements and terminations](#)
- [Replacing, terminating, or powering down problem instances and nodes](#)
- [Troubleshooting other known node and job issues](#)

Key logs

The following table provides an overview of the key logs for the head node:

`/var/log/cfn-init.log`

This is the CloudFormation init log. It contains all commands that were run when an instance is set up. It's useful for troubleshooting initialization issues.

`/var/log/chef-client.log`

This is the Chef client log. It contains all commands that were run through Chef/CINC. It's useful for troubleshooting initialization issues.

`/var/log/parallelcluster/slurm_resume.log`

This is a ResumeProgram log. It launches instances for dynamic nodes and useful for troubleshooting dynamic nodes launch issues.

`/var/log/parallelcluster/slurm_suspend.log`

This is the SuspendProgram log. It's called when instances are terminated for dynamic nodes, and useful for troubleshooting dynamic nodes termination issues. When you check this log, you should also check the `clustermgtd` log.

`/var/log/parallelcluster/clustermgtd`

This is the `clustermgtd` log. It runs as the centralized daemon that manages most cluster operation actions. It's useful for troubleshooting any launch, termination, or cluster operation issue.

`/var/log/slurmctld.log`

This is the Slurm control daemon log. AWS ParallelCluster doesn't make scaling decisions. Rather, it only attempts to launch resources to satisfy the Slurm requirements. It's useful

for scaling and allocation issues, job-related issues, and any scheduler-related launch and termination issues.

These are the Key notes for the Compute nodes:

`/var/log/cloud-init-output.log`

This is the [cloud-init](#) log. It contains all commands that were run when an instance is set up. It's useful for troubleshooting initialization issues.

`/var/log/parallelcluster/computemgtd`

This is the `computemgtd` log. It runs on each compute node to monitor the node in the rare event that `clustermgtd` daemon on the head node is offline. It's useful for troubleshooting unexpected termination issues.

`/var/log/slurmd.log`

This is the Slurm compute daemon log. It's useful for troubleshooting initialization and compute failure related issues.

Troubleshooting node initialization issues

This section covers how you can troubleshoot node initialization issues. This includes issues where the node fails to launch, power up, or join a cluster.

Head node:

Applicable logs:

- `/var/log/cfn-init.log`
- `/var/log/chef-client.log`
- `/var/log/parallelcluster/clustermgtd`
- `/var/log/parallelcluster/slurm_resume.log`
- `/var/log/slurmctld.log`

Check the `/var/log/cfn-init.log` and `/var/log/chef-client.log` logs. These logs should contain all the actions that were run when the head node is set up. Most errors that occur during setup should have a error messages located in the `/var/log/chef-client.log` log. If pre-

install or post-install scripts are specified in the configuration of the cluster, double check that the script runs successfully through log messages.

When a cluster is created, the head node needs to wait for the compute nodes to join the cluster before it can join the cluster. As such, if the compute nodes fail to join the cluster, then the head node also fails. You can follow one of these sets of procedures, depending on the type of compute nodes you use, to troubleshoot this type of issue:

Dynamic compute nodes:

- Search the ResumeProgram log (`/var/log/parallelcluster/slurm_resume.log`) for your compute node name to see if ResumeProgram was ever called with the node. (If ResumeProgram wasn't ever called, you can check the `slurmctld` log (`/var/log/slurmctld.log`) to determine if Slurm ever tried to call ResumeProgram with the node.)
- Note that incorrect permissions for ResumeProgram might cause ResumeProgram to fail silently. If you're using a custom AMI with modification to ResumeProgram setup, check that the ResumeProgram is owned by the `slurm` user and has the `744 (rwxr--r--)` permission.
- If ResumeProgram is called, check to see if an instance is launched for the node. If no instance was launched, you should be able to see an error message that describes the launch failure.
- If the instance is launched, then there might have been a problem during the setup process. You should see the corresponding private IP address and instance ID from the ResumeProgram log. Moreover, you can look at corresponding setup logs for the specific instance. For more information about troubleshooting a setup error with a compute node, see the next section.

Static compute nodes:

- Check the `clustermgtd` (`/var/log/parallelcluster/clustermgtd`) log to see if instances were launched for the node. If they weren't launched, there should be clear error message detailing the launch failure.
- If instance is launched, there's some issue during setup process. You should see the corresponding private IP address and instance ID from the ResumeProgram log. Moreover, you can look at the corresponding setup logs for the specific instance.
- **Compute nodes:**
 - **Applicable logs:**
 - `/var/log/cloud-init-output.log`

- `/var/log/slurmd.log`
- If compute node is launched, first check `/var/log/cloud-init-output.log`, which should contain the setup logs similar to the `/var/log/chef-client.log` log on the head node. Most errors that occur during setup should have error messages located at the `/var/log/cloud-init-output.log` log. If pre-install or post-install scripts are specified in cluster configuration, check that they ran successfully.
- If you're using a custom AMI with modification to Slurm configuration, then there might be a Slurm related error that prevents the compute node from joining the cluster. For scheduler related errors, check the `/var/log/slurmd.log` log.

Troubleshooting unexpected node replacements and terminations

This section continues to explore how you can troubleshoot node related issues, specifically when a node is replaced or terminated unexpectedly.

- **Applicable logs:**
 - `/var/log/parallelcluster/clustermgtd` (head node)
 - `/var/log/slurmctld.log` (head node)
 - `/var/log/parallelcluster/computemgtd` (compute node)
- **Nodes replaced or terminated unexpectedly**
 - Check in the `clustermgtd` log (`/var/log/parallelcluster/clustermgtd`) to see if `clustermgtd` took the action to replace or terminate a node. Note that `clustermgtd` handles all normal node maintenance action.
 - If `clustermgtd` replaced or terminated the node, there should be a message detailing why this action was taken on the node. If the reason is scheduler related (for example, because the node is in DOWN), check in `slurmctld` log for more information. If the reason is Amazon EC2 related, there should be informative message detailing the Amazon EC2 related issue that required the replacement.
 - If `clustermgtd` didn't terminate the node, first check if this was an expected termination by Amazon EC2, more specifically a spot termination. `computemgtd`, running on a Compute node, can also take an action to terminate a node if `clustermgtd` is determined as unhealthy. Check `computemgtd` log (`/var/log/parallelcluster/computemgtd`) to see if `computemgtd` terminated the node.
- **Nodes failed**
 - Check in `slurmctld` log (`/var/log/slurmctld.log`) to see why a job or a node failed. Note that jobs are automatically re-queued if a node failed.

- If `slurm_resume` reports that node is launched and `clustermgtd` reports after several minutes that there's no corresponding instance in Amazon EC2 for that node, the node might fail during setup. To retrieve the log from a compute (`/var/log/cloud-init-output.log`), do the following steps:
 - Submit a job to let Slurm spin up a new node.
 - After the node starts, enable termination protection using this command.

```
aws ec2 modify-instance-attribute --instance-id i-xyz --disable-api-termination
```

- Retrieve the console output from the node with this command.

```
aws ec2 get-console-output --instance-id i-xyz --output text
```

Replacing, terminating, or powering down problem instances and nodes

- **Applicable logs:**

- `/var/log/parallelcluster/clustermgtd` (head node)
- `/var/log/parallelcluster/slurm_suspend.log` (head node)
- In most cases, `clustermgtd` handles all expected instance termination action. Check in the `clustermgtd` log to see why it failed to replace or terminate a node.
- For dynamic nodes failing [scaledown_idletime](#), check in the `SuspendProgram` log to see if `SuspendProgram` was called by `slurmctld` with the specific node as argument. Note that `SuspendProgram` doesn't actually perform any action. Rather, it only logs when it's called. All instance termination and `NodeAddr` reset is done by `clustermgtd`. Slurm puts nodes back into a `POWER_SAVING` state after `SuspendTimeout` automatically.

Troubleshooting other known node and job issues

Another type of known issue is that AWS ParallelCluster might fail to allocate jobs or make scaling decisions. With this type of issue, AWS ParallelCluster only launches, terminates, or maintains resources according to Slurm instructions. For these issues, check the `slurmctld` log to troubleshoot these issues.

Troubleshooting issues in single queue mode clusters

Note

Starting with version 2.11.5, AWS ParallelCluster doesn't support the use of SGE or Torque schedulers.

This section applies to clusters that don't have multiple queue mode with one of the following two configurations:

- Launched using a AWS ParallelCluster version earlier than 2.9.0 and SGE, Torque, or Slurm job schedulers.
- Launched using AWS ParallelCluster version 2.9.0 or later and SGE or Torque job schedulers.

Topics

- [Key logs](#)
- [Troubleshooting failed launch and join operations](#)
- [Troubleshooting scaling issues](#)
- [Troubleshooting other cluster related issues](#)

Key logs

The following log files are the key logs for the head node.

For AWS ParallelCluster version 2.9.0 or later:

`/var/log/chef-client.log`

This is the CINC (chef) client log. It contains all commands that were run through CINC. It's useful for troubleshooting initialization issues.

For all AWS ParallelCluster versions:

`/var/log/cfn-init.log`

This is the `cfn-init` log. It contains all commands that were run when an instance is set up, and is therefore useful for troubleshooting initialization issues. For more information, see [cfn-init](#).

`/var/log/clustermgtd.log`

This is the `clustermgtd` log for Slurm schedulers. `clustermgtd` runs as the centralized daemon that manages most cluster operation actions. It's useful for troubleshooting any launch, termination, or cluster operation issue.

`/var/log/jobwatcher`

This is the `jobwatcher` log for SGE and Torque schedulers. `jobwatcher` monitors the scheduler queue and updates the Auto Scaling Group. It's useful for troubleshooting issues related to scaling up nodes.

`/var/log/sqswatcher`

This is the `sqswatcher` log for SGE and Torque schedulers. `sqswatcher` processes the instance ready event sent by a compute instance after successful initialization. It also adds compute nodes to the scheduler configuration. This log is useful for troubleshooting why a node or nodes failed to join a cluster.

The following are the key logs for the compute nodes.

AWS ParallelCluster version 2.9.0 or later

`/var/log/cloud-init-output.log`

This is the Cloud init log. It contains all commands that were run when an instance is set up. It's useful for troubleshooting initialization issues.

AWS ParallelCluster versions before 2.9.0

`/var/log/cfn-init.log`

This is the CloudFormation init log. It contains all commands that were run when an instance is set up. It's useful for troubleshooting initialization issues

All versions

`/var/log/nodewatcher`

This is the `nodewatcher` log. `nodewatcher` daemons that run on each Compute node when using SGE and Torque schedulers. They scale down a node if it's idle. This log is useful for any issues related to scaling down resources.

Troubleshooting failed launch and join operations

- **Applicable logs:**
 - `/var/log/cfn-init-cmd.log` (head node and compute node)
 - `/var/log/sqswatcher` (head node)
- If nodes failed to launch, check in the `/var/log/cfn-init-cmd.log` log to see the specific error message. In most cases, node launch failures are due to a setup failure.
- If compute nodes failed to join the scheduler configuration despite successful setup, check the `/var/log/sqswatcher` log to see whether `sqswatcher` processed the event. These issues in most cases are because `sqswatcher` didn't process the event.

Troubleshooting scaling issues

- **Applicable logs:**
 - `/var/log/jobwatcher` (head node)
 - `/var/log/nodewatcher` (compute node)
- **Scale up issues:** For the head node, check the `/var/log/jobwatcher` log to see if the `jobwatcher` daemon calculated the proper number of required nodes and updated the Auto Scaling Group. Note that `jobwatcher` monitors the scheduler queue and updates the Auto Scaling Group.
- **Scale down issues:** For compute nodes, check the `/var/log/nodewatcher` log on the problem node to see why the node was scaled down. Note that `nodewatcher` daemons scale down a compute node if it's idle.

Troubleshooting other cluster related issues

One known issue is random compute node fails on large-scale clusters, specifically those with 500 or more compute nodes. This issue is related to a limitation of the scaling architecture of single queue cluster. If you want to use a large scale cluster, are using AWS ParallelCluster version v2.9.0

or later, are using Slurm, and want to avoid this issue, you should upgrade and switch to a multiple queue mode supported cluster. You can do so by running [pcluster-config convert](#).

For ultra-large-scale clusters, additional tuning to your system might be required. For more information, contact Support.

Placement groups and instance launch issues

To get the lowest inter-node latency, use a *placement group*. A placement group guarantees that your instances are on the same networking backbone. If there aren't enough instances available when a request is made, an `InsufficientInstanceCapacity` error is returned. To reduce the possibility of receiving this error when using cluster placement groups, set the [placement_group](#) parameter to `DYNAMIC` and set the [placement](#) parameter to `compute`.

If you require a high performance shared filesystem, consider using [FSx for Lustre](#).

If the head node must be in the placement group, use the same instance type and subnet for both the head as well as all of the compute nodes. By doing this, the [compute_instance_type](#) parameter has the same value as the [master_instance_type](#) parameter, the [placement](#) parameter is set to `cluster`, and the [compute_subnet_id](#) parameter isn't specified. With this configuration, the value of the [master_subnet_id](#) parameter is used for the compute nodes.

For more information, see [Troubleshooting instance launch issues](#) and [Placement groups roles and limitations](#) in the *Amazon EC2 User Guide*

Directories that cannot be replaced

The following directories are shared between the nodes and cannot be replaced.

`/home`

This includes the default user home folder (`/home/ec2_user` on Amazon Linux, `/home/centos` on CentOS, and `/home/ubuntu` on Ubuntu).

`/opt/intel`

This includes Intel MPI, Intel Parallel Studio, and related files.

/opt/sge

Note

Starting with version 2.11.5, AWS ParallelCluster doesn't support the use of SGE or Torque schedulers.

This includes Son of Grid Engine and related files. (Conditional, only if [scheduler](#) = sge.)

/opt/slurm

This includes Slurm Workload Manager and related files. (Conditional, only if [scheduler](#) = slurm.)

/opt/torque

Note

Starting with version 2.11.5, AWS ParallelCluster doesn't support the use of SGE or Torque schedulers.

This includes Torque Resource Manager and related files. (Conditional, only if [scheduler](#) = torque.)

Troubleshooting issues in Amazon DCV

Topics

- [Logs for Amazon DCV](#)
- [Amazon DCV instance type memory](#)
- [Ubuntu Amazon DCV issues](#)

Logs for Amazon DCV

The logs for Amazon DCV are written to files in the `/var/log/dcv/` directory. Reviewing these logs can help to troubleshoot issues.

Amazon DCV instance type memory

The instance type should have at least 1.7 gibibyte (GiB) of RAM to run Amazon DCV. Nano and micro instance types don't have enough memory to run Amazon DCV.

Ubuntu Amazon DCV issues

When running Gnome Terminal over a DCV session on Ubuntu, you might not automatically have access to the user environment that AWS ParallelCluster makes available through the login shell. The user environment provides environment modules such as openmpi or intelmpi, and other user settings.

Gnome Terminal's default settings prevent the shell from starting as a login shell. This means that shell profiles aren't automatically sourced and the AWS ParallelCluster user environment isn't loaded.

To properly source the shell profile and access the AWS ParallelCluster user environment, do one of the following:

- **Change the default terminal settings:**
 1. Choose the **Edit** menu in the Gnome terminal.
 2. Select **Preferences**, then **Profiles**.
 3. Choose **Command** and select **Run Command as login shell**.
 4. Open a new terminal.
- **Use the command line to source the available profiles:**

```
$ source /etc/profile && source $HOME/.bashrc
```

Troubleshooting issues in clusters with AWS Batch integration

This section is relevant to clusters with AWS Batch scheduler integration.

Head node issues

Head node related setup issues can be troubleshooted in the same way as single queue cluster. For more information about these issues, see [Troubleshooting issues in single queue mode clusters](#).

AWS Batch multi-node parallel jobs submission issues

If you have problems submitting multi-node parallel jobs when using AWS Batch as the job scheduler, you should upgrade to AWS ParallelCluster version 2.5.0. If that's not feasible, you can use the workaround that's detailed in the topic: [Self patch a cluster used for submitting multi-node parallel jobs through AWS Batch](#).

Compute issues

AWS Batch manages the scaling and compute aspects of your services. If you encounter compute related issues, see the AWS Batch [troubleshooting](#) documentation for help.

Job failures

If a job fails, you can run the [awsbcout](#) command to retrieve the job output. You can also run the [awsbststat](#) -d command to obtain a link to the job logs stored by Amazon CloudWatch.

Troubleshooting when a resource fails to create

This section is relevant to cluster resources when they fail to create.

When a resource fails to create, ParallelCluster returns an error message like the following.

```
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)
}

```

As an example, if you see the status message shown in the previous command response, you must use instance types that won't exceed your current vCPU limit or request more vCPU capacity.

You can also use the CloudFormation console to see information about the "Cluster creation failed" status.

View CloudFormation error messages from the console.

1. Log in to the AWS Management Console and navigate to <https://console.aws.amazon.com/cloudformation>.
2. Select the stack named parallelcluster-*cluster_name*.
3. Choose the **Events** tab.
4. Check the **Status** for the resource that failed to create by scrolling through the list of resource events by **Logical ID**. If a subtask failed to create, work backwards to find the failed resource event.
5. An example of a AWS CloudFormation error message:

```

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].

```

Troubleshooting IAM policy size issues

Refer to [IAM and AWS STS quotas, name requirements, and character limits](#) to verify quotas on managed policies attached to roles. If a managed policy size exceeds the quota, split the policy into

two or more policies. If you exceed the quota on the number of policies attached to an IAM role, create additional roles and distribute the policies among them to meet the quota.

Additional support

For a list of known issues, see the main [GitHub Wiki](#) page or the [issues](#) page. For more urgent issues, contact Support or open a [new GitHub issue](#).

AWS ParallelCluster support policy

AWS ParallelCluster supports multiple releases at the same time. Every AWS ParallelCluster release has a scheduled End of Support Life (EOSL) date. After the EOSL date, no further support or maintenance is provided for that release.

AWS ParallelCluster uses a `major.minor.patch` version scheme. New features, performance improvements, security updates, and bug fixes are included in new minor version releases for the latest major version release. Minor versions are backward compatible within a major version. For critical issues, AWS provides fixes through patch releases, but only for the latest minor versions of releases that have not reached EOSL. If you want to use the updates from a new version release, you need to upgrade to the new minor or patch version.

AWS ParallelCluster versions	End of supported life (EOSL) date
2.10.4 and earlier	12/31/2021
2.11.x	12/31/2022

Security in AWS ParallelCluster

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security sensitive organizations.

Security is a shared responsibility between AWS and you. The [shared responsibility model](#) describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the [AWS Compliance Programs](#). To learn about the compliance programs that apply to AWS ParallelCluster, see [AWS Services in Scope by Compliance Program](#).
- **Security in the cloud** – Your responsibility is determined by the specific AWS service or services that you use. You are also responsible for several other related factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation describes how you should apply the shared responsibility model when using AWS ParallelCluster. The following topics show you how to configure AWS ParallelCluster to meet your security and compliance objectives. You also learn how to use AWS ParallelCluster in a way that helps you to monitor and secure your AWS resources.

Topics

- [Security information for services used by AWS ParallelCluster](#)
- [Data protection in AWS ParallelCluster](#)
- [Identity and Access Management for AWS ParallelCluster](#)
- [Compliance validation for AWS ParallelCluster](#)
- [Enforcing a Minimum Version of TLS 1.2](#)

Security information for services used by AWS ParallelCluster

- [Security in Amazon EC2](#)
- [Security in Amazon API Gateway](#)

- [Security in AWS Batch](#)
- [Security in CloudFormation](#)
- [Security in Amazon CloudWatch](#)
- [Security in AWS CodeBuild](#)
- [Security in Amazon DynamoDB](#)
- [Security in Amazon ECR](#)
- [Security in Amazon ECS](#)
- [Security in Amazon EFS](#)
- [Security in FSx for Lustre](#)
- [Security in AWS Identity and Access Management \(IAM\)](#)
- [Security in EC2 Image Builder](#)
- [Security in AWS Lambda](#)
- [Security in Amazon Route 53](#)
- [Security in Amazon SNS](#)
- [Security in Amazon SQS \(For AWS ParallelCluster version 2.x.\)](#)
- [Security in Amazon S3](#)
- [Security in Amazon VPC](#)

Data protection in AWS ParallelCluster

The AWS [shared responsibility model](#) applies to data protection in . As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. You are also responsible for the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see [Data Privacy FAQ](#). For information about data protection in Europe, see the [General Data Protection Regulation \(GDPR\) Center](#).

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.

- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail. For information about using CloudTrail trails to capture AWS activities, see [Working with CloudTrail trails](#) in the *AWS CloudTrail User Guide*.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-3 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see [Federal Information Processing Standard \(FIPS\) 140-3](#).

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Data encryption

A key feature of any secure service is that information is encrypted when it is not being actively used.

Encryption at rest

AWS ParallelCluster does not itself store any customer data other than the credentials it needs to interact with the AWS services on the user's behalf.

For data on the nodes in the cluster, data can be encrypted at rest.

For Amazon EBS volumes, encryption is configured using the [ebs_kms_key_id](#) settings in the [\[ebs\] section](#) for AWS ParallelCluster version 2.x.) For more information, see [Amazon EBS encryption](#) in the Amazon EC2 User Guide.

For Amazon EFS volumes, encryption is configured using the [encrypted](#) and [efs_kms_key_id](#) settings in the [\[efs\] section](#) in AWS ParallelCluster version 2.x). For more information, see [How encryption at rest works](#) in the *Amazon Elastic File System User Guide*.

For FSx for Lustre file systems, encryption of data at rest is automatically enabled when creating an Amazon FSx file system. For more information, see [Encrypting data at rest](#) in the *Amazon FSx for Lustre User Guide*.

For instance types with NVMe volumes, the data on NVMe instance store volumes is encrypted using an XTS-AES-256 cipher implemented on a hardware module on the instance. The encryption keys are generated using the hardware module and are unique to each NVMe instance storage device. All encryption keys are destroyed when the instance is stopped or terminated and cannot be recovered. You cannot disable this encryption and you cannot provide your own encryption key. For more information, see [Encryption at rest](#) in the *Amazon EC2 User Guide*.

If you use AWS ParallelCluster to invoke an AWS service that transmits customer data to your local computer for storage, then refer to the Security and Compliance chapter in that service's User Guide for information on how that data is stored, protected, and encrypted.

Encryption in transit

By default, all data transmitted from the client computer running AWS ParallelCluster and AWS service endpoints is encrypted by sending everything through a HTTPS/TLS connection. Traffic between the nodes in the cluster can be automatically encrypted, depending on the instance types selected. For more information, see [Encryption in transit](#) in the *Amazon EC2 User Guide*.

See also

- [Data protection in Amazon EC2](#)
- [Data protection in EC2 Image Builder](#)
- [Data protection in CloudFormation](#)
- [Data protection in Amazon EFS](#)
- [Data protection in Amazon S3](#)
- [Data protection in FSx for Lustre](#)

Identity and Access Management for AWS ParallelCluster

AWS ParallelCluster uses roles to access your AWS resources and their services. The instance and user policies that AWS ParallelCluster uses to grant permissions are documented at [AWS Identity and Access Management roles in AWS ParallelCluster](#).

The only major difference is how you authenticate when using a standard user and long-term credentials. Although an user requires a password to access an AWS service's console, that same user requires an access key pair to perform the same operations using AWS ParallelCluster. All other short-term credentials are used in the same way they are used with the console.

The credentials used by AWS ParallelCluster are stored in plaintext files and are **not** encrypted.

- The `$HOME/.aws/credentials` file stores long-term credentials required to access your AWS resources. These include your access key ID and secret access key.
- Short-term credentials, such as those for roles that you assume, or that are for AWS IAM Identity Center services, are also stored in the `$HOME/.aws/cli/cache` and `$HOME/.aws/sso/cache` folders, respectively.

Mitigation of Risk

- We strongly recommend that you configure your file system permissions on the `$HOME/.aws` folder and its child folders and files to restrict access to only authorized users.
- Use roles with temporary credentials wherever possible to reduce the opportunity for damage if the credentials are compromised. Use long-term credentials only to request and refresh short-term role credentials.

Compliance validation for AWS ParallelCluster

Third-party auditors assess the security and compliance of AWS services as part of multiple AWS compliance programs. Using AWS ParallelCluster to access a service does not alter that service's compliance.

For a list of AWS services in scope of specific compliance programs, see [AWS services in scope by compliance program](#). For general information, see [AWS compliance programs](#).

You can download third-party audit reports using the AWS Artifact. For more information, see [Downloading reports in AWS Artifact](#).

Your compliance responsibility when using AWS ParallelCluster is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- [Security and compliance quick start guides](#) – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- [Architecting for HIPAA security and Compliance on Amazon Web Services AWS Whitepaper](#) – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- [AWS compliance resources](#) – This collection of workbooks and guides might apply to your industry and location.
- [Evaluating resources with rules](#) in the *AWS Config Developer Guide* – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- [AWS Security Hub CSPM](#) – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

Enforcing a Minimum Version of TLS 1.2

To add increased security when communicating with AWS services, you should configure your AWS ParallelCluster to use TLS 1.2 or later. When you use AWS ParallelCluster, Python is used to set the TLS version.

To ensure AWS ParallelCluster uses no TLS version earlier than TLS 1.2, you might need to recompile OpenSSL to enforce this minimum and then recompile Python to use the newly built OpenSSL.

Determine Your Currently Supported Protocols

First, create a self-signed certificate to use for the test server and the Python SDK using OpenSSL.

```
$ openssl req -subj '/CN=localhost' -x509 -newkey rsa:4096 -nodes -keyout key.pem -out cert.pem -days 365
```

Then spin up a test server using OpenSSL.

```
$ openssl s_server -key key.pem -cert cert.pem -www
```

In a new terminal window, create a virtual environment and install the Python SDK.

```
$ python3 -m venv test-env
source test-env/bin/activate
pip install botocore
```

Create a new Python script named `check.py` that uses the SDK's underlying HTTP library.

```
$ 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'))
```

Run your new script.

```
$ python check.py
```

This displays details about the connection made. Search for "Protocol : " in the output. If the output is "TLSv1.2" or later, the SDK defaults to TLS v1.2 or later. If it's an earlier version, you need to recompile OpenSSL and recompile Python.

However, even if your installation of Python defaults to TLS v1.2 or later, it's still possible for Python to renegotiate to a version earlier than TLS v1.2 if the server doesn't support TLS v1.2 or later. To check that Python doesn't automatically renegotiate to earlier versions, restart the test server with the following.

```
$ openssl s_server -key key.pem -cert cert.pem -no_tls1_3 -no_tls1_2 -www
```

If you're using an earlier version of OpenSSL, you might not have the `-no_tls1_3` flag available. If this is the case, remove the flag because the version of OpenSSL you're using doesn't support TLS v1.3. Then rerun the Python script.

```
$ python check.py
```

If your installation of Python correctly doesn't renegotiate for versions earlier than TLS 1.2, you should receive an SSL error.

```
$ 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)')))
```

If you're able to make a connection, you need to recompile OpenSSL and Python to disable negotiation of protocols earlier than TLS v1.2.

Compile OpenSSL and Python

To ensure that AWS ParallelCluster doesn't negotiate for anything earlier than TLS 1.2, you need to recompile OpenSSL and Python. To do this, copy the following content to create a script and run it.

```
#!/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
```

This compiles a version of Python that has a statically linked OpenSSL that doesn't automatically negotiate anything earlier than TLS 1.2. This also installs OpenSSL in the `/opt/openssl-with-`

min-tls1_2 directory and installs Python in the /opt/python-with-min-tls1_2 directory. After you run this script, confirm installation of the new version of Python.

```
$ /opt/python-with-min-tls1_2/bin/python3 --version
```

This should print out the following.

```
Python 3.8.1
```

To confirm this new version of Python doesn't negotiate a version earlier than TLS 1.2, rerun the steps from [Determine Your Currently Supported Protocols](#) using the newly installed Python version (that is, /opt/python-with-min-tls1_2/bin/python3).

Release notes and document history

The following table describes the major updates and new features for the *AWS ParallelCluster User Guide*. We also update the documentation frequently to address the feedback that you send us.

Change	Description	Date
Documentation only release	AWS ParallelCluster version 2 specific user guide published.	July 17, 2023
	Documentation only release:	
	<ul style="list-style-type: none">• AWS ParallelCluster version 2 has its own separate user guide.	
AWS ParallelCluster version 2.11.9 released	AWS ParallelCluster version 2.11.9 released.	December 2, 2022
	Bug fixes:	
	<ul style="list-style-type: none">• Prevent replacement of managed FSx for Lustre file systems and loss of data on cluster updates that include changes to <code>vpc_security_group_id</code> .	
	For details of the changes, see the <code>CHANGELOG</code> file for the aws-parallelcluster package on GitHub.	

[AWS ParallelCluster version 2.11.8 released](#)

AWS ParallelCluster version 2.11.8 released.

November 14, 2022

Changes:

- Upgrade Intel MPI Library to Version 2021 Update 6 (updated from Version 2021 Update 4). For more information, see [Intel® MPI Library 2021 Update 6](#).
- Upgrade EFA installer to 1.19.0
 - Efa-driver: efa-1.16.0-1
 - Efa-config: efa-config-1.11-1 (from efa-config-1.9-1)
 - Efa-profile: efa-profile-1.5-1 (no change)
 - Libfabric-aws: libfabric-aws-1.16.0-1 (from libfabric-1.13.2)
 - Rdma-core: rdma-core-41.0-2 (from rdma-core-37.0)
 - Open MPI: openmpi40-aws-4.1.4-3 (from openmpi40-aws-4.1.1-2)
- Upgrade Python runtime, that's used by Lambda

functions in the AWS Batch integration, to python3.9.

Bug fixes:

- Prevent cluster tags from being changed during an update because it's not supported.

For details of the changes, see the CHANGELOG files for the [aws-parallelcluster](#) package on GitHub.

[AWS ParallelCluster version 2.11.7 released](#)

AWS ParallelCluster version 2.11.7 released.

May 13, 2022

Changes:

- Upgrade Slurm to version 20.11.9.

For details of the changes, see the CHANGELOG files for the [aws-parallelcluster](#) package on GitHub.

[AWS ParallelCluster version 2.11.6 released](#)

AWS ParallelCluster version 2.11.6 released.

April 19, 2022

Enhancements:

- Improve exception management in case of missing networking.

Changes:

- OS package updates and security fixes.

For details of the changes, see the `CHANGELOG` files for the [aws-parallelcluster](#) package on GitHub.

[AWS ParallelCluster version 2.11.5 released](#)

AWS ParallelCluster version 2.11.5 released.

March 1, 2022

Enhancements:

- Add support for NEW_CHANGED_DELETED as value of FSx for Lustre AutoImportPolicy option.
- Remove support for SGE and Torque schedulers.
- Disable log4j-cve-2021-44228-hotpatch service on Amazon Linux to avoid incurring potential performance degradation.

Changes:

- Upgrade NVIDIA driver to version 470.103.01 (from 470.82.01).
- Upgrade NVIDIA Fabric manager to version 470.103.01 (from 470.82.01).
- Upgrade CUDA library to version 11.4.4 (from 11.4.3).
- [Intel MPI](#) updated to Version 2021 Update 4 (updated from Version

2019 Update 8). For more information, see [Intel® MPI Library 2021 Update 4](#).

- Extend head node creation timeout to one hour.

Bug fixes:

- Fix DCV connection through browsers.
- Fix YAML quoting to prevent custom Tags from being parsed as numbers.

For details of the changes, see the CHANGELOG files for the [aws-parallelcluster](#) package on GitHub.

[AWS ParallelCluster version 2.11.4 released](#)

AWS ParallelCluster version 2.11.4 released.

December 20, 2021

Changes include:

- CentOS 8 support removed. CentOS 8 reaches end of life (EOL) on December 31, 2021.
- Upgrade Slurm Workload Manager to version 20.11.8.
- Upgrade Cinc Client to 17.2.29.
- [Amazon DCV](#) updated to Amazon DCV 2021.2-11190. For more information, see [DCV 2021.2-11190 — October 11, 2021](#) in the *Amazon DCV Administrator Guide*.
- Upgrade NVIDIA driver to version 470.82.01 (from 460.73.01).
- Upgrade CUDA library to version 11.4.3 (from 11.3.0).
- Upgrade NVIDIA Fabric Manager to 470.82.01 .
- Disable package update at instance launch time on Amazon Linux 2.
- Disable unattended package update on Ubuntu and Amazon Linux 2.

- Install Python 3 version of [CloudFormation helper scripts](#) on CentOS 7 and Ubuntu 18.04. (These were already used on Amazon Linux 2 and Ubuntu 20.04.)

Fixes include:

- Disable update of [ec2_iam_role](#) parameter.
- Fix the CpuOptions configuration in the launch template for T2 instances.

For details of the changes, see the CHANGELOG files for the [aws-parallelcluster](#), [aws-parallelcluster-cookbook](#) and [aws-parallelcluster-node](#) packages on GitHub.

[AWS ParallelCluster version 2.11.3 released](#)

AWS ParallelCluster version 2.11.3 released.

November 3, 2021

- Fix [pcluster createami](#) failure due to Son of Grid Engine sources not being available at `arc.liv.ac.uk`.

Upgrade [Elastic Fabric Adapter](#) installer to 1.14.1 (from 1.13.0)

- EFA config: `efa-config-1.9-1` (from `efa-config-1.9`)
- EFA profile: `efa-profile-1.5-1` (no change)
- EFA Kernel module: `efa-1.14.2` (from `efa-1.13.0`)
- RDMA core: `rdma-core-37.0` (from `rdma-core-35.0amzn`)
- Libfabric: `libfabric-1.13.2` (from `libfabric-1.13.0amazon1.0`)
- Open MPI: `openmpi40-aws-4.1.1-2` (no change)

GPUDirect RDMA is always enabled if supported by the instance type.

- The [enable_efa_gdr](#) and [enable_efa_gdr](#) configuration options have no effect.

For details of the changes, see the CHANGELOG files for the [aws-parallelcluster](#), [aws-parallelcluster-cookbook](#) and [aws-parallelcluster-node](#) packages on GitHub.

[AWS ParallelCluster version 2.11.2 released](#)

AWS ParallelCluster version 2.11.2 released.

August 27, 2021

Changes include:

- Do not install EFA with GPUDirect RDMA (GDR) enabled at bootstrap time if EFA is installed in the base AMI.
- Lock version of `nvidia-fabricmanager` package to remain in sync with the NVIDIA driver version installed by AWS ParallelCluster.
- Slurm: Fix issue caused when cluster was stopped and restarted while a node was powering up.
- [Elastic Fabric Adapter](#) installer updated to 1.13.0:
 - EFA config: `efa-config-1.9` (no change)
 - EFA profile: `efa-profile-1.5-1` (no change)
 - EFA Kernel module: `efa-1.13.0` (no change)
 - RDMA core: `rdma-core-35.0amzn` (from `rdma-core-32.1amzn`)
 - Libfabric: `libfabric-1.13.0amzn1.0`

```
(from libfabric  
-1.11.2amzn1.1 )
```

- Open MPI: openmpi40
-aws-4.1.1-2 (no
change)
- When using a custom
AMI with a pre-installed
EFA package, no changes
are made to EFA at node
bootstrap time. The original
EFA package deployment is
preserved.

For more details of the
changes, see the CHANGELOG
files for the [aws-parallelcluste
r](#) and [aws-parallelcluste
r-cookbook](#) packages on
GitHub.

[AWS ParallelCluster version 2.11.1 released](#)

AWS ParallelCluster version 2.11.1 released.

July 23, 2021

Changes include:

- Mount file systems using the `noatime` mount option to stop recording last access time when a file is read. This improves the performance of the remote file system.
- [Elastic Fabric Adapter](#) installer updated to 1.12.3:
 - EFA config: `efa-config-1.9` (from `efa-config-1.8-1`)
 - EFA profile: `efa-profile-1.5-1` (no change)
 - EFA Kernel module: `efa-1.13.0` (from `efa-1.12.3`)
 - RDMA core: `rdma-core-32.1amzn` (no change)
 - Libfabric: `libfabric-1.11.2amzn1.1` (no change)
 - Open MPI: `openmpi40-aws-4.1.1-2` (no change)
- Retry installations of the `aws-parallelcluster` package on the head node

when using AWS Batch as the scheduler.

- Avoid failures when building SGE on an instance type with more than 31 vCPUs.
- Pinned to version 1.247347.6 of the Amazon CloudWatch Agent to avoid issues seen in version 1.247348.0.

For more details of the changes, see the CHANGELOG files for the [aws-parallelcluster](#) and [aws-parallelcluster-cookbook](#) packages on GitHub.

[AWS ParallelCluster version 2.11.0 released](#)

AWS ParallelCluster version 2.11.0 released.

July 1, 2021

Changes include:

- Added support for Ubuntu 20.04 (ubuntu2004) and removed support for Ubuntu 16.04 (ubuntu1604) and Amazon Linux (alinux). Amazon Linux 2 (alinux2) remains fully supported. For more information, see [base_os](#).
- Removed support for Python versions below 3.6.
- Default root volume size increased to 35 gibibytes (GiB). For more information, see [compute_root_volume_size](#) and [master_root_volume_size](#).
- [Elastic Fabric Adapter](#) installer updated to 1.12.2:
 - EFA config: efa-config-1.8-1 (from efa-config-1.7)
 - EFA profile: efa-profile-1.5-1 (from efa-profile-1.4)
 - EFA Kernel module: efa-1.12.3 (from efa-1.10.2)

- RDMA core: `rdma-core-32.1amzn` (from `rdma-core-31.2amzn`)
- Libfabric: `libfabric-1.11.2amzn1.1` (from `libfabric-1.11.1amzn1.0`)
- Open MPI: `openmpi40-aws-4.1.1-2` (from `openmpi40-aws-4.1.0`)
- Upgraded Slurm to version 20.11.7 (from 20.02.7).
- Install SSM Agent on centos7 and centos8. (SSM Agent is pre-installed in `alinux2`, `ubuntu1804` , and `ubuntu2004` .)
- SGE: Always use the shortname as the hostname filter with `qstat`.
- Use instance metadata service Version 2 (IMDSv2) instead of instance metadata service Version 1 (IMDSv1) to retrieve instance metadata. For more information, see [Instance metadata and user data](#) in the *Amazon EC2 User Guide*.
- Upgrade NVIDIA driver to version 460.73.01 (from 450.80.02).

- Upgrade CUDA library to version 11.3.0 (from 11.0).
- Upgrade NVIDIA Fabric Manager to nvidia-fabricmanager-460 .
- Upgrade Python used in AWS ParallelCluster virtualenvs to 3.7.10 (from 3.6.13).
- Upgrade Cinc Client to 16.13.16.
- Upgrade third-party dependencies of [aws-parallelcluster-cookbook](#):
 - apt-7.4.0 (from apt-7.3.0).
 - iptables-8.0.0 (from iptables-7.1.0).
 - line-4.0.1 (from line-2.9.0).
 - openssh-2.9.1 (from openssh-2.8.1).
 - pyenv-3.4.2 (from pyenv-3.1.1).
 - selinux-3.1.1 (from selinux-2.1.1).
 - ulimit-1.1.1 (from ulimit-1.0.0).
 - yum-6.1.1 (from yum-5.1.0).

- `yum-epel-4.1.2`
(from `yum-epel-3.3.0`).

For more details of the changes, see the CHANGELOG files for the [aws-parallelcluster](#), [aws-parallelcluster-cookbook](#), and [aws-parallelcluster-node](#) packages on GitHub.

[AWS ParallelCluster version 2.10.4 released](#)

AWS ParallelCluster version 2.10.4 released.

May 15, 2021

Changes include:

- Upgraded Slurm to version `20.02.7` (from `20.02.4`).

For more details of the changes, see the CHANGELOG file for the [aws-parallelcluster](#) package on GitHub.

[AWS ParallelCluster version 2.10.3 released](#)

AWS ParallelCluster version 2.10.3 released.

March 18, 2021

Changes include:

- Added support for Ubuntu 18.04 and Amazon Linux 2 on Arm-based AWS Graviton instances in the AWS China and AWS GovCloud (US) AWS Regions.
- [Elastic Fabric Adapter](#) installer updated to 1.11.2:
 - EFA config: `efa-config-1.7` (no change)
 - EFA profile: `efa-profile-1.4` (from `efa-profile-1.3`)
 - EFA Kernel module: `efa-1.10.2` (no change)
 - RDMA core: `rdma-core-31.2amzn` (no change)
 - Libfabric: `libfabric-1.11.1amzn1.0` (no change)
 - Open MPI: `openmpi40-aws-4.1.0` (no change)

For more details of the changes, see the CHANGELOG file for the [aws-parallelcluster](#) package on GitHub.

[AWS ParallelCluster version 2.10.2 released](#)

AWS ParallelCluster version 2.10.2 released.

March 2, 2021

Changes include:

- Improve cluster config validation to use the cluster target AMI when invoking the Amazon EC2 [RunInstances](#) API operation in `--dry-run` mode.
- Update the Python version used in the AWS ParallelCluster virtual environments to 3.6.13.
- Fix [sanity_check](#) for Arm instance types.
- Fix `enable_efa` when using centos8 with the Slurm scheduler or Arm instance types.
- Run `apt update` in non-interactive mode (`-y`).
- Fix [encrypted_ephemeral](#) = true with `alinux2` and `centos8`.

For more details of the changes, see the CHANGELOG file for the [aws-parallelcluster](#) package on GitHub.

[AWS ParallelCluster version 2.10.1 released](#)

AWS ParallelCluster version 2.10.1 released.

December 22, 2020

Changes include:

- Added support for the Africa (Cape Town) (af-south-1), Europe (Milan) (me-south-1), and Middle East (Bahrain) (me-south-1) AWS Regions. At launch, support is limited in the following ways:
 - FSx for Lustre and Arm-based Graviton instances aren't supported in any of these AWS Regions.
 - AWS Batch isn't supported in Africa (Cape Town).
 - Amazon EBS io2 and gp3 volume types aren't supported in the Africa (Cape Town) and Europe (Milan) AWS Regions.
- Added support for the Amazon EBS io2 and gp3 volume types. For more information, see [\[ebs\] section](#) and [\[raid\] section](#).
- Added support for [Elastic Fabric Adapter](#) on Arm-based Graviton2 instances running `alinux2`,

ubuntu1804 , or
ubuntu2004 . For more
information, see [Elastic
Fabric Adapter](#).

- Install the Arm Performance Libraries 20.2.1 on Arm AMIs (alinux2, centos8, and ubuntu1804). For more information, see [Arm Performance Libraries](#).
- [Intel MPI](#) updated to Version 2019 Update 8 (updated from Version 2019 Update 7). For more information, see [Intel® MPI Library 2019 Update 8](#).
- Removed the CloudFormation DescribeStacks API operation call from the AWS Batch Docker entrypoint to end the job failures caused by throttling by CloudFormation.
- Improved the calls to Amazon EC2 DescribeInstanceTypes API operation call when validating a cluster configuration.
- Amazon Linux 2 Docker images are pulled from Amazon ECR Public when building the Docker image for the awsbatch scheduler.

- The default instance type changed from the hardcoded `t2.micro` instance type to the Free Tier instance type for the AWS Region (`t2.micro` or `t3.micro`, depending on the AWS Region). AWS Regions that do not have a Free Tier default to the `t3.micro` instance type.
- [Elastic Fabric Adapter](#) installer updated to 1.11.1:
 - EFA config: `efa-config-1.7` (from `efa-config-1.5`)
 - EFA profile: `efa-profile-1.3` (from `efa-profile-1.1`)
 - EFA Kernel module: `efa-1.10.2` (no change)
 - RDMA core: `rdma-core-31.2amzn` (from `rdma-core-31.amzn0`)
 - Libfabric: `libfabric-1.11.1amzn1.0` (from `libfabric-1.10.1amzn1.1`)
 - Open MPI: `openmpi40-aws-4.1.0` (from `openmpi40-aws-4.0.5`)

- The [vpc_settings](#) , [vpc_id](#), and [master_subnet_id](#) parameters are now required.
- The nfsd daemon in the head node is now set to use at least 8 threads. If there are more than 8 cores, it will use as many threads as there are cores. When `ubuntu1604` is used, the setting only changes after the node is rebooted.
- [Amazon DCV](#) updated to Amazon DCV 2020.2-9662. For more information, see [DCV 2020.2-9662—December 04, 2020](#) in the *Amazon DCV Administrator Guide*.
- The Intel MPI and HPC packages for AWS ParallelCluster are pulled from Amazon S3. They are no longer pulled from Intel yum repos.
- Changed the default systemd runlevel to `multi-user.target` on all OSs during creation of official AWS ParallelCluster AMIs. The runlevel is set to `graphical.target` on the head node only when DCV is enabled. This prevents graphical

services (such as x/gdm) from running when they are not required.

- Enabled support for p4d.24xlarge instances on the head node.
- Increase the maximum number of retry attempts when registering Slurm nodes in Amazon Route 53.

For more details of the changes, see the CHANGELOG files for the [aws-parallelcluster](#), [aws-parallelcluster-cookbook](#), and [aws-parallelcluster-node](#) packages on GitHub.

[AWS ParallelCluster version 2.10.0 released](#)

AWS ParallelCluster version 2.10.0 released.

November 18, 2020

Changes include:

- Added support for CentOS 8 in all AWS Regions (outside of the AWS China and AWS GovCloud (US) Regions). Removed support for CentOS 6.
- Added support for p4d.24xlarge instances for compute nodes.
- Added support for NVIDIA GPUDirect RDMA on EFA by using the new [enable_ea_gdr](#) setting.
- Added support for Amazon FSx for Lustre features.
 - Configure your Amazon FSx for Lustre file system to import preferences using the [auto_import_policy](#) setting.
 - Added support for HDD-based Amazon FSx for Lustre file systems using the [storage_type](#) and [drive_cache_type](#) settings.
- Added a Amazon CloudWatch dashboard , including head node metrics and easy access to cluster logs. For more

information, see [Amazon CloudWatch dashboard](#).

- Added support for using an existing Amazon S3 bucket to store cluster configuration information, using the [cluster_resource_bucket](#) setting.
- Enhanced the [pcluster createami](#) command.
 - Added `--post-install` parameter to use a post-installation script when build an AMI.
 - Added a validation step to fail when using a base AMI created by a different version of AWS ParallelCluster.
 - Added a validation step to fail when if the select OS is different from the OS in the base AMI.
 - Added support for using a AWS ParallelCluster base AMI.
- Enhanced the [pcluster update](#) command.
 - The [tags](#) setting can now be changed during an update.
 - Queues can now be resized during an update without stopping the compute fleet

- Added `all_or_nothing_batch` configuration parameter for `slurm_resume` script. When `True`, `slurm_resume` will succeed only if all the instances required by all the pending jobs in Slurm will be available. For more information, see [Introducing all_or_nothing_batch launches](#) in the AWS ParallelCluster Wiki on GitHub.
- [Elastic Fabric Adapter](#) installer updated to 1.10.1:
 - EFA config: `efa-config-1.5` (from `efa-config-1.4`)
 - EFA profile: `efa-profile-1.1` (from `efa-profile-1.0.0`)
 - EFA Kernel module: `efa-1.10.2` (from `efa-1.6.0`)
 - RDMA core: `rdma-core-31.amzn0` (from `rdma-core-28.amzn0`)
 - Libfabric: `libfabric-1.11.1amzn1.0` (from `libfabric-1.10.1amzn1.1`)
 - Open MPI: `openmpi40-aws-4.0.5` (from

```
openmpi40-aws-4.0.  
3 )
```

- In the AWS GovCloud (US) Regions, enable support for Amazon DCV and AWS Batch.
- In the AWS China Regions, enable support for Amazon FSx for Lustre.
- Upgrade NVIDIA driver to version 450.80.02 (from 450.51.05).
- Install NVIDIA Fabric Manager to enable NVIDIA NVSwitch on supported platforms.
- Removed default AWS Region of `us-east-1`. The default uses this lookup order.
 - AWS Region specified in `-r` or `--region` argument.
 - `AWS_DEFAULT_REGION` environment variable.
 - `aws_region_name` setting in the [\[aws\]](#) [section](#) of the AWS ParallelCluster configuration file (defaults to `~/.parallelcluster/config`).
 - `region` setting in the `[default]` section of the AWS CLI configura

tion file (defaults to ~/aws/config).

For more details of the changes, see the CHANGELOG files for the [aws-parallelcluster](#), [aws-parallelcluster-cookbook](#), and [aws-parallelcluster-node](#) packages on GitHub.

[AWS ParallelCluster version 2.9.0 released](#)

AWS ParallelCluster version 2.9.0 released.

September 11, 2020

Changes include:

- Added support for multiple queues and multiple instance types in the compute fleet when used with Slurm Workload Manager. When using queues, Auto Scaling groups are no longer used on Slurm. An Amazon Route 53 hosted zone is now created with the cluster and is used for DNS resolution of compute nodes when the Slurm scheduler is used. For more information, see [Multiple queue mode](#).
- Added support for [Amazon DCV](#) on Arm-based AWS Graviton-based instances.
- Added support for disabling hyperthreading on instance types that do not support CPU options in launch templates (for example *.metal instance types).
- Added support for NFS 4 for file systems shared from the head node.
- Removed dependency on [cfn-init](#) when bootstrap

ping compute nodes
to avoid throttling by
CloudFormation when a
large number of nodes join
the cluster.

- [Elastic Fabric Adapter](#)
installer updated to 1.9.5:
 - EFA config: `efa-config-1.4` (from `efa-config-1.3`)
 - EFA profile: `efa-profile-1.0.0` (new)
 - Kernel module:
`efa-1.6.0` (no change)
 - RDMA core: `rdma-core-28.amzn0` (no change)
 - Libfabric: `libfabric-1.10.1amzn1.1` (no change)
 - Open MPI: `openmpi40-aws-4.0.3` (no change)
- Upgraded Slurm to version `20.02.4` (from `19.05.5`).
- [Amazon DCV](#) updated to Amazon DCV 2020.1-9012. For more information, see [DCV 2020.1-9012— August 24, 2020 Release Notes](#) in the *Amazon DCV Administrator Guide*.
- When mounting shared NFS drives, use the head node

private IP address instead of the hostname.

- Added new log streams to CloudWatch Logs: `chef-client` , `clustermgtd` , `computemgtd` , `slurm_resume` , and `slurm_suspend` .
- Added support for queue names in pre-install and post-install scripts.
- In the AWS GovCloud (US) AWS Regions, use the Amazon DynamoDB on-demand billing option. For more information, see [On-Demand Mode](#) in the *Amazon DynamoDB Developer Guide*.

For more details of the changes, see the CHANGELOG files for the [aws-parallelcluster](#), [aws-parallelcluster-cookbook](#), and [aws-parallelcluster-node](#) packages on GitHub.

[AWS ParallelCluster version 2.8.1 released](#)

AWS ParallelCluster version 2.8.1 released.

August 4, 2020

Changes include:

- Disable screen lock for Amazon DCV sessions to prevent users from being locked out.
- Fix [pcluster configure](#) when including an Arm-based AWS Graviton-based instance type.

For more details of the changes, see the CHANGELOG files for the [aws-parallelcluster](#), [aws-parallelcluster-cookbook](#), and [aws-parallelcluster-node](#) packages on GitHub.

[AWS ParallelCluster version 2.8.0 released](#)

AWS ParallelCluster version 2.8.0 released.

July 23, 2020

Changes include:

- Added support for Arm-based AWS Graviton-based instances (like the A1 and C6g).
- Added support for the automatic daily backup features of Amazon FSx for Lustre. For more information, see [automatic_backup_retention_days](#), [copy_tags_to_backups](#), [daily_automatic_backup_start_time](#), and [fsx_backup_id](#).
- Removed dependency on Berkshelf from [pcluster createami](#).
- Improved the robustness and user experience of [pcluster update](#). For more information, see [Using pcluster update](#).
- [Elastic Fabric Adapter](#) installer updated to 1.9.4:
 - Kernel module: efa-1.6.0 (updated from efa-1.5.1)
 - RDMA core: rdma-core-28.amzn0

(updated from `rdma-core-25.0`)

- Libfabric: `libfabric-1.10.1amzn1.1`
(updated from `libfabric-aws-1.9.0amzn1.1`)
- Open MPI: `openmpi40-aws-4.0.3` (no change)
- Upgrade NVIDIA driver to Tesla version 440.95.01 on CentOS 6 and version 450.51.05 on all other distributions.
- Upgrade CUDA library to version 11.0 on all distributions other than CentOS 6.

For more details of the changes, see the CHANGELOG files for the [aws-parallelcluster](#), [aws-parallelcluster-cookbook](#), and [aws-parallelcluster-node](#) packages on GitHub.

[AWS ParallelCluster version 2.7.0 released](#)

AWS ParallelCluster version 2.7.0 released.

May 19, 2020

Changes include:

- [base_os](#) is now a required parameter.
- [scheduler](#) is now a required parameter.
- [Amazon DCV](#) updated to Amazon DCV 2020.0. For more information, see [Amazon DCV releases version 2020.0 with surround sound 7.1 and stylus support](#).

[Intel MPI](#) updated to Version 2019 Update 7 (updated from Version 2019 Update 6). For more information, see [Intel® MPI Library 2019 Update 7](#).

[Elastic Fabric Adapter](#) installer updated to 1.8.4:

- Kernel module:
efa-1.5.1 (no change)
- RDMA core: rdma-core-25.0 (no change)
- Libfabric: libfabric-aws-1.9.0amzn1.1 (no change)
- Open MPI: openmpi40-aws-4.0.3 (updated

from openmpi40-aws-4.0.2)

- Upgrade CentOS 7 AMI to version 7.8-2003 (updated from 7.7-1908). For more information, see [CentOS-7 \(2003\) Release Notes](#).

[AWS ParallelCluster version 2.6.1 released](#)

AWS ParallelCluster version 2.6.1 released.

April 17, 2020

Changes include:

- Removed `cfn-init-cmd` and `cfn-wire` from logs stored in Amazon CloudWatch Logs. For more information, see [Integration with Amazon CloudWatch Logs](#).

[AWS ParallelCluster version 2.6.0 released](#)

AWS ParallelCluster version 2.6.0 released.

February 27, 2020

Changes include:

- Added support for Amazon Linux 2.
- Now Amazon CloudWatch Logs is used to collect cluster and scheduler logs. For more information, see [Integration with Amazon CloudWatch Logs](#).
- Added support for new Amazon FSx for Lustre deployment types SCRATCH_2 and PERSISTENT_1. Support for FSx for Lustre on Ubuntu 18.04 and Ubuntu 16.04. For more information, see [fsx](#).
- Added support for Amazon DCV on Ubuntu 18.04. For more information, see [Connect to the head node through Amazon DCV](#).

[AWS ParallelCluster version 2.5.1 released](#)

AWS ParallelCluster version 2.5.1 released.

December 13, 2019

[AWS ParallelCluster version 2.5.0 released](#)

AWS ParallelCluster version 2.5.0 released.

November 18, 2019

[AWS ParallelCluster introduces support for Intel MPI](#)

AWS ParallelCluster version 2.4.1 introduces support for Intel MPI.

July 29, 2019

[AWS ParallelCluster introduces support for EFA](#)

AWS ParallelCluster version 2.4.0 introduces support for Elastic Fabric Adapter (EFA).

June 11, 2019

[AWS ParallelCluster documentation released on AWS documentation site](#)

The AWS ParallelCluster documentation is now available in 10 languages and in both HTML and PDF formats.

May 24, 2018