
AWS ParallelCluster

AWS ParallelCluster User Guide



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

What is AWS ParallelCluster	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	2
Steps to take after installation	3
Detailed instructions for each environment	3
Virtual environment	3
Linux	5
macOS	8
Windows	10
Configuring AWS ParallelCluster	11
Moving from CfnCluster to AWS ParallelCluster	16
Supported regions	17
Using AWS ParallelCluster	19
AWS ParallelCluster CLI commands	19
pcluster	19
pcluster-config	30
Network configurations	31
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	33
Custom Bootstrap Actions	35
Configuration	35
Arguments	36
Example	36
Working with Amazon S3	37
Examples	37
Working with Spot Instances	37
Scenario 1: Spot Instance with no running jobs is interrupted	38
Scenario 2: Spot Instance running single node jobs is interrupted	38
Scenario 3: Spot Instance running multi-node jobs is interrupted	38
AWS Identity and Access Management roles in AWS ParallelCluster	39
Defaults	39
Using an existing IAM role for Amazon EC2	39
AWS ParallelCluster example instance and user policies	39
Schedulers supported by AWS ParallelCluster	56
Son of Grid Engine	56
Slurm Workload Manager	56
Torque Resource Manager	56
AWS Batch	57
Multiple queue mode	62
Integration with Amazon CloudWatch Logs	63
Elastic Fabric Adapter	64
Enable Intel MPI	64
Intel HPC Platform Specification	65
Connect to the head node through NICE DCV	65
NICE DCV HTTPS certificate	66
Licensing NICE DCV	66
Using pcluster update	66
Configuration	35
Layout	69
[global] section	70

cluster_template	70
update_check	70
sanity_check	70
[aws] section	70
[aliases] section	71
[cluster] section	71
additional_cfn_template	73
additional_iam_policies	73
base_os	73
cluster_type	74
compute_instance_type	75
compute_root_volume_size	75
custom_ami	75
cw_log_settings	75
dcv_settings	76
desired_vcpus	76
disable_hyperthreading	76
ebs_settings	77
ec2_iam_role	77
efs_settings	77
enable_efa	77
enable_intel_hpc_platform	78
encrypted_ephemeral	78
ephemeral_dir	78
extra_json	79
fsx_settings	79
initial_queue_size	79
key_name	80
maintain_initial_size	80
master_instance_type	80
master_root_volume_size	80
max_queue_size	81
max_vcpus	81
min_vcpus	81
placement	81
placement_group	82
post_install	82
post_install_args	83
pre_install	83
pre_install_args	83
proxy_server	83
queue_settings	84
raid_settings	84
s3_read_resource	84
s3_read_write_resource	85
scaling_settings	85
scheduler	85
shared_dir	86
spot_bid_percentage	86
spot_price	86
tags	87
template_url	87
vpc_settings	87
[compute_resource] section	88
initial_count	88
instance_type	88
max_count	89

min_count	89
spot_price	89
[cw_log] section	89
enable	90
retention_days	90
[dcv] section	90
access_from	91
enable	91
port	91
[ebs] section	92
shared_dir	92
ebs_snapshot_id	92
volume_type	93
volume_size	93
volume_iops	93
encrypted	93
ebs_kms_key_id	93
ebs_volume_id	94
[efs] section	94
efs_fs_id	94
efs_kms_key_id	95
encrypted	95
performance_mode	96
provisioned_throughput	96
shared_dir	96
throughput_mode	97
[fsx] section	97
automatic_backup_retention_days	98
copy_tags_to_backups	98
daily_automatic_backup_start_time	99
deployment_type	99
export_path	100
fsx_backup_id	100
fsx_fs_id	100
fsx_kms_key_id	101
import_path	101
imported_file_chunk_size	101
per_unit_storage_throughput	102
shared_dir	102
storage_capacity	102
weekly_maintenance_start_time	103
[queue] section	103
compute_resource_settings	103
compute_type	104
disable_hyperthreading	104
enable_efa	104
placement_group	104
[raid] section	105
shared_dir	105
raid_type	106
num_of_raid_volumes	106
volume_type	106
volume_size	107
volume_iops	107
encrypted	107
ebs_kms_key_id	107
[scaling] section	107

scaledown_idletime	108
[vpc] section	108
additional_sg	108
compute_subnet_cidr	109
compute_subnet_id	109
master_subnet_id	109
ssh_from	109
use_public_ips	109
vpc_id	110
vpc_security_group_id	110
Example	37
How AWS ParallelCluster works	112
AWS ParallelCluster processes	112
General overview	112
jobwatcher	113
sqswatcher	114
nodewatcher	115
clustermgtd	116
computemgtd	117
AWS services used in AWS ParallelCluster	117
AWS Auto Scaling	117
AWS Batch	118
AWS CloudFormation	118
Amazon CloudWatch	118
Amazon CloudWatch Logs	118
AWS CodeBuild	119
Amazon DynamoDB	119
Amazon Elastic Block Store	119
Amazon Elastic Compute Cloud	119
Amazon Elastic Container Registry	119
Amazon EFS	119
Amazon FSx for Lustre	120
AWS Identity and Access Management	120
AWS Lambda	120
NICE DCV	120
Amazon Route 53	120
Amazon Simple Notification Service	121
Amazon Simple Queue Service	121
Amazon Simple Storage Service	121
Amazon VPC	121
AWS ParallelCluster Auto Scaling	121
Scaling up	122
Scaling down	123
Static cluster	123
Tutorials	124
Running your first job on AWS ParallelCluster	124
Verifying your installation	124
Creating your first cluster	124
Logging into your head node	125
Running your first job using SGE	125
Building a Custom AWS ParallelCluster AMI	126
How to Customize the AWS ParallelCluster AMI	126
Running an MPI job with AWS ParallelCluster and awsbatch scheduler	128
Creating the cluster	128
Logging into your head node	125
Running your first job using AWS Batch	130
Running an MPI job in a multi-node parallel environment	131

Disk encryption with a custom KMS Key	134
Creating the role	134
Give your key permissions	134
Creating the cluster	128
Security	136
Security information for services used by AWS ParallelCluster	136
Data protection	137
Data encryption	137
See also	138
Identity and Access Management	138
Compliance validation	139
Enforcing TLS 1.2	139
Determine Your Currently Supported Protocols	139
Compile OpenSSL and Python	140
Development	142
Setting up a custom AWS ParallelCluster cookbook	142
Steps	142
Setting up a custom AWS ParallelCluster node package	143
Steps	142
Troubleshooting	145
Failure submitting AWS Batch multi-node parallel jobs	145
Placement groups and instance launch issues	145
Directories that cannot be replaced	145
NICE DCV troubleshooting	146
Document history	147

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. Built on the open source CfnCluster project, AWS ParallelCluster enables you to quickly build an HPC compute environment in AWS. It automatically sets up the required compute resources and shared filesystem. You can use AWS ParallelCluster with a batch schedulers, such as AWS Batch and Slurm. AWS ParallelCluster facilitates quick start proof of concept deployments and production deployments. You can also build higher level workflows, such as a genomics portal that automates an entire DNA sequencing workflow, on top of AWS ParallelCluster.

Setting up AWS ParallelCluster

Topics

- [Installing AWS ParallelCluster \(p. 2\)](#)
- [Configuring AWS ParallelCluster \(p. 11\)](#)
- [Moving from CfnCluster to AWS ParallelCluster \(p. 16\)](#)
- [Supported regions \(p. 17\)](#)

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 \(recommended\) \(p. 2\)](#)
- [Using `pip` \(p. 2\)](#)

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 \(p. 3\)](#) 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 --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 \(p. 7\)](#)
- **macOS** – [Add the AWS ParallelCluster executable to your command line path \(p. 9\)](#)
- **Windows** – [Add the AWS ParallelCluster executable to your command line path \(p. 11\)](#)

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

```
$ pcluster version
2.9.1
```

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 --upgrade --user
```

To uninstall AWS ParallelCluster, use `pip uninstall`.

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

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 \(recommended\) \(p. 3\)](#)
- [Install AWS ParallelCluster on Linux \(p. 5\)](#)
- [Install AWS ParallelCluster on macOS \(p. 8\)](#)
- [Install AWS ParallelCluster on Windows \(p. 10\)](#)

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
```

Windows

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

5. Verify that AWS ParallelCluster is installed correctly.

Linux, macOS, or Unix

```
$ pcluster version
```

```
2.9.1
```

Windows

```
(apc-ve) C:\>pcluster version  
2.9.1
```

You can use the `deactivate` command to exit the virtual environment. Each time you start a session, you must [reactivate the environment \(p. 4\)](#).

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
```

Windows

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

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 \(p. 2\)](#) 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 \(p. 2\)](#) topic. Otherwise, [install Python \(p. 7\)](#), 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 ensure 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 20.2.3 from ~/.local/lib/python3.6/site-packages (python 3.6)
```

Sections

- [Install AWS ParallelCluster with pip \(p. 6\)](#)
- [Add the AWS ParallelCluster executable to your command line path \(p. 7\)](#)
- [Installing Python on Linux \(p. 7\)](#)

Install AWS ParallelCluster with pip

Use `pip` to install AWS ParallelCluster.

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

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

Verify that AWS ParallelCluster installed correctly.

```
$ pcluster version  
2.9.1
```

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

```
$ python3 -m pip install aws-parallelcluster --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 \(p. 2\)](#), 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 3.8.5
```

Install AWS ParallelCluster on macOS

Sections

- [Prerequisites \(p. 8\)](#)
- [Install AWS ParallelCluster on macOS using pip \(p. 8\)](#)
- [Add the AWS ParallelCluster executable to your command line path \(p. 9\)](#)

Prerequisites

- Python 3 version 3.6+ 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 \(p. 5\)](#).

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 \(p. 2\)](#). 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 --upgrade --user
```

4. Verify that AWS ParallelCluster is installed correctly.

```
$ pcluster version  
2.9.1
```

If the program isn't found, [add it to your command line path \(p. 9\)](#).

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

```
$ pip3 install aws-parallelcluster --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 \(p. 2\)](#).

Sections

- [Install AWS ParallelCluster using Python and pip on Windows \(p. 10\)](#)
- [Add the AWS ParallelCluster executable to your command line path \(p. 11\)](#)

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 3.8.5
C:\>pip3 --version
pip 20.2.3 from c:\python38\lib\site-packages\pip (python 3.8)
```

3. Install AWS ParallelCluster using `pip`.

```
C:\>pip3 install aws-parallelcluster
```

4. Verify that AWS ParallelCluster is installed correctly.

```
C:\>pcluster version
2.9.1
```

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

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

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.

First, 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 region name [us-east-1]: us-east-1
Default output format [None]:
```

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

```
$ 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 SGE, Slurm, or Torque.

Warning

Starting on December 31, 2021, AWS will no longer include SGE and Torque support for all released versions of AWS ParallelCluster. Previous versions of AWS ParallelCluster that support SGE and Torque will still be available for download and use. However, these versions will not be eligible for future updates or troubleshooting support from AWS service and customer support teams. Moreover, future releases of AWS ParallelCluster made before and after December 31, 2021 will not include support for either SGE or Torque.

SGE, Slurm, or Torque

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

```
Allowed values for the 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. sge
2. torque
3. slurm
4. awsbatch
Scheduler [sge]:
```

Choose the operating system.

```
Allowed values for Operating System:
1. alinux
2. alinux2
3. centos6
4. centos7
5. ubuntu1604
6. ubuntu1804
Operating System [alinux]:
```

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 for Linux Instances*.

```
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 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 is possible to reach your limit on number of VPCs in a Region. The default limit is five VPCs per 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 whether all nodes should be in a public subnet.

```
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 do not create a new VPC, you must select an existing VPC.

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

AWS Batch

From the list of valid AWS Region identifiers, choose the Region in which 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. sge
2. torque
3. slurm
4. awsbatch
Scheduler [sge]:
```

When `awsbatch` is selected as the scheduler, either `alinux` or `alinux2` can be 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 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 is possible to reach your limit on number of VPCs in a 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*.

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

```
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 do not create a new VPC, you must select an existing VPC

```
Automate VPC creation? (y/n) [n]: n
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, using an existing subnet that supports public IP's (the route table for the subnet is 0.0.0.0/0 => *igw-xxxxxxx*). Note the following:

- 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 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 \(p. 31\)](#).

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 details on connecting to Amazon EC2 instances, see the [EC2 User Guide](#) in the *Amazon EC2 User Guide for Linux Instances*.

If [pcluster configure \(p. 20\)](#) created a new VPC, you can delete that VPC by deleting the AWS CloudFormation stack it created. The name will start with "parallelclusternetworking-" and contain the creation time in a "YYYYMMDDHHMMSS" format. You can list the stacks using the [list-stacks](#) command.

```
$ aws --region us-east-2 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-west-2 cloudformation delete-stack \  
  --stack-name parallelclusternetworking-pubpriv-20191029205804
```

Moving from CfnCluster to AWS ParallelCluster

AWS ParallelCluster is an enhanced version of CfnCluster.

If you currently use CfnCluster, we encourage you to start using and creating new clusters with AWS ParallelCluster instead. Although you can continue to use CfnCluster, it is 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 cannot be managed with the `pcluster` CLI. The following commands do not 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 cannot 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, you must:

1. Move the configuration file from `~/.cfncluster/config` to `~/.parallelcluster/config`.
2. If you use the [extra_json](#) (p. 79) configuration parameter, change it as shown.

CfnCluster setting:

```
extra_json = { "cfnccluster" : { } }
```

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:

1. Set the [extra_json](#) (p. 79) 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 for Linux Instances*.

Supported regions

AWS ParallelCluster 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
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

Region Name	Region
Europe (London)	eu-west-2
Europe (Paris)	eu-west-3
Europe (Stockholm)	eu-north-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

- [AWS ParallelCluster CLI commands \(p. 19\)](#)
- [Network configurations \(p. 31\)](#)
- [Custom Bootstrap Actions \(p. 35\)](#)
- [Working with Amazon S3 \(p. 37\)](#)
- [Working with Spot Instances \(p. 37\)](#)
- [AWS Identity and Access Management roles in AWS ParallelCluster \(p. 39\)](#)
- [Schedulers supported by AWS ParallelCluster \(p. 56\)](#)
- [Multiple queue mode \(p. 62\)](#)
- [Integration with Amazon CloudWatch Logs \(p. 63\)](#)
- [Elastic Fabric Adapter \(p. 64\)](#)
- [Enable Intel MPI \(p. 64\)](#)
- [Intel HPC Platform Specification \(p. 65\)](#)
- [Connect to the head node through NICE DCV \(p. 65\)](#)
- [Using pcluster update \(p. 66\)](#)

AWS ParallelCluster CLI commands

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

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

Topics

- [pcluster \(p. 19\)](#)
- [pcluster-config \(p. 30\)](#)

pcluster

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

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

Arguments

`pcluster` *command*

Possible choices: [configure \(p. 20\)](#), [create \(p. 21\)](#), [createami \(p. 22\)](#),
[dcv \(p. 23\)](#), [delete \(p. 24\)](#), [instances \(p. 25\)](#), [list \(p. 25\)](#),

[ssh](#) (p. 26), [start](#) (p. 27), [status](#) (p. 27), [stop](#) (p. 28), [update](#) (p. 29),
[version](#) (p. 30)

Sub-commands:

Topics

- [pcluster configure](#) (p. 20)
- [pcluster create](#) (p. 21)
- [pcluster createami](#) (p. 22)
- [pcluster dcv](#) (p. 23)
- [pcluster delete](#) (p. 24)
- [pcluster instances](#) (p. 25)
- [pcluster list](#) (p. 25)
- [pcluster ssh](#) (p. 26)
- [pcluster start](#) (p. 27)
- [pcluster status](#) (p. 27)
- [pcluster stop](#) (p. 28)
- [pcluster update](#) (p. 29)
- [pcluster version](#) (p. 30)

pcluster configure

Begins an AWS ParallelCluster configuration.

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

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](#) (p. 11).

If [pcluster configure](#) (p. 20) created a new VPC, you can delete that VPC by deleting the AWS CloudFormation stack it created. The name will start with "parallelclusternetworking-" and contain the creation time in a "YYYYMMDDHHMMSS" format. You can list the stacks using the [list-stacks](#) command.

```
$ aws --region us-east-2 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-west-2 cloudformation delete-stack \  
  --stack-name parallelclusternetworking-pubpriv-20191029205804
```

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. Defaults to the Region specified by using the `pcluster configure` (p. 20) command.

-nw, --nowait

Indicates not to wait for stack events after executing 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 at creation time.

-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 is safe to use "Ctrl-C" to exit. You can return to viewing the current status by calling `pcluster status mycluster`.

Examples:

```
$ pcluster create mycluster
$ 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 ]
                  [ -c CONFIG_FILE ] [ -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://www.packer.io/downloads.html>.

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: `alinux`, `alinux2`, `ubuntu1604`, `ubuntu1804`, `centos6`, and `centos7`.

Note

Support for `alinux2` was added in AWS ParallelCluster version 2.6.0. Support for `ubuntu1804` was added in AWS ParallelCluster version 2.5.0.

-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.

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

Specifies the alternative configuration file to use.

Defaults to `~/.parallelcluster/config`.

-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.

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

Specifies the Region to connect to.

pcluster dcv

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

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

pcluster dcv *command*

Possible choices: `connect` (p. 23)

Note

Support for NICE DCV 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 ] cluster_name
```

Important

The URL expires 30 seconds after it is issued. If the connection is not 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](#) (p. 80) configuration parameter. This argument is optional, but if it is 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`.

-s, --show-url

Displays a one-time URL for connecting to the NICE DCV session. The default browser is not opened when this option is specified.

Note

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

Example:

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

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

A new NICE DCV session is created if one is not 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 will expire based on the [retention_days](#) (p. 90) 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 Region to connect to.

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

If `pcluster configure` (p. 20) created a new VPC, you can delete that VPC by deleting the AWS CloudFormation stack it created. The name will start with "parallelclusternetworking-" and contain the creation time in a "YYYYMMDDHHMMSS" format. You can list the stacks using the `list-stacks` command.

```
$ aws --region us-east-2 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-west-2 cloudformation delete-stack \
  --stack-name parallelclusternetworking-pubpriv-20191029205804
```

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 Region to connect to.

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 Region to connect to.

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

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 ] 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`.

Example:

```
$ pcluster ssh -d mycluster -i ~/.ssh/id_rsa
```

Returns an `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 \(p. 71\)](#). 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` (p. 73) 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 Region to connect to.

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 Region to connect to.

-nw, --nowait

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

Defaults to `False`.

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.

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 Region to connect to.

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 cannot be updated, the configuration settings that are the source of the conflicts are enumerated with details. For more information, see [Using pcluster update \(p. 66\)](#).

```
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 should not be combined with the `--yes` argument.

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

Specifies the Region to connect to.

-nr, --norollback

Disables AWS CloudFormation stack rollback on error.

Defaults to `False`.

-nw, --nowait

Indicates not to wait for stack events after executing 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 the answer to all prompts is yes. This should not be combined with the `--force` argument.

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

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 \(p. 11\)](#).

-t CLUSTER_TEMPLATE, --cluster-template CLUSTER_TEMPLATE

Indicates the `[cluster]` section (p. 71) to use. If this argument is not specified, `pcluster-config convert` will use the `cluster_template` (p. 70) setting in the `[global]` section (p. 70). If that is not specified, then the `[cluster default]` section will be used.

-o OUTPUT_FILE, --output OUTPUT_FILE

Specifies the path of the converted configuration file to be written. By default the output will be 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`.

Network configurations

AWS ParallelCluster uses Amazon Virtual Private Cloud (VPC) for networking. VPC provides a flexible and configurable networking platform in which to 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 new or existing.

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:

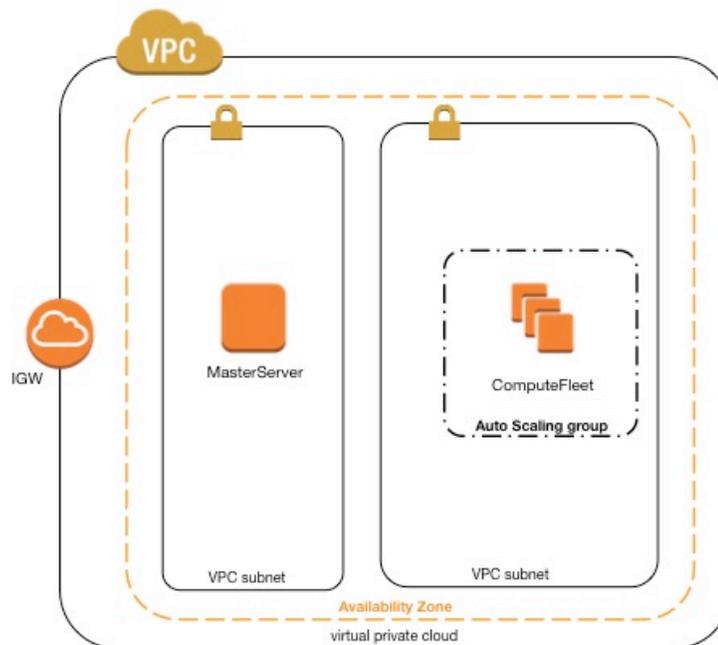
AWS ParallelCluster in a single public subnet

The configuration for this architecture requires the following settings:

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

The `use_public_ips` (p. 109) 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 examples only

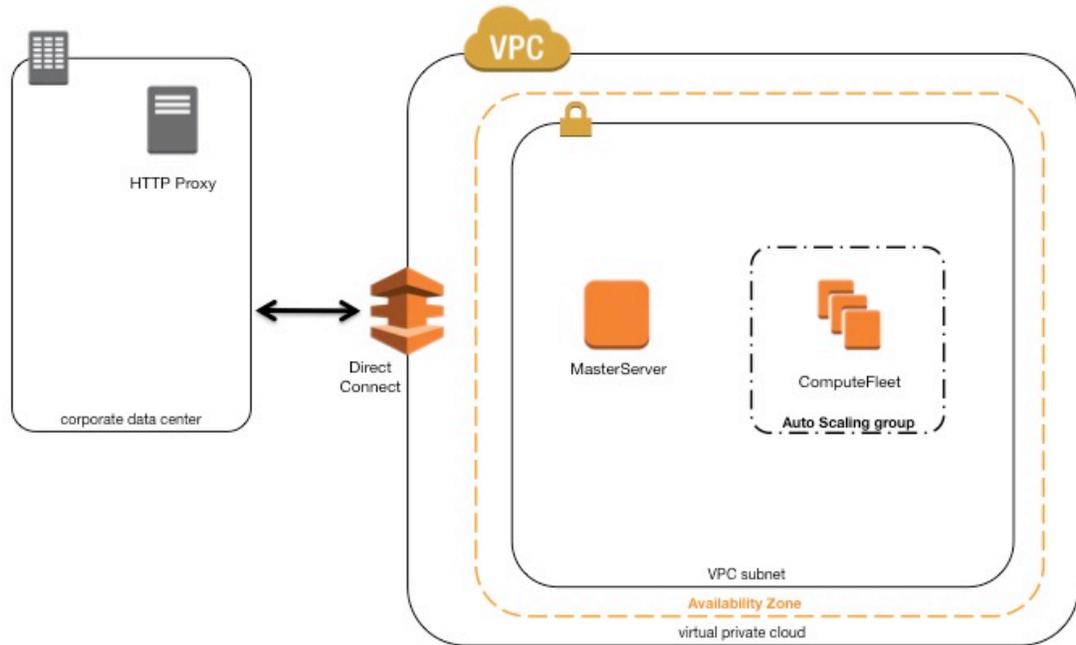
```
[vpc public-private-new]
vpc_id = vpc-xxxxxxx
master_subnet_id = subnet-<public>
compute_subnet_cidr = 10.0.1.0/24
```

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

```
[vpc public-private-existing]
vpc_id = vpc-xxxxxxx
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-xxxxxxx
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`. In order 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:

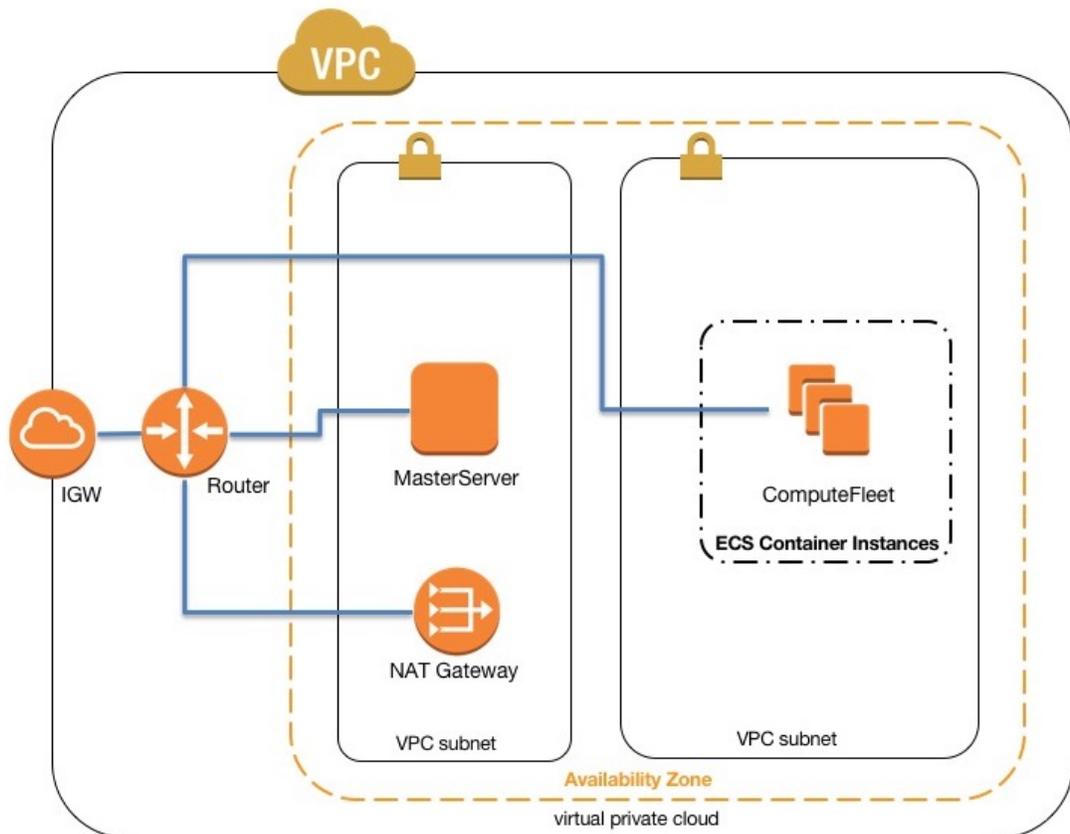
- The `compute_subnet` uses a NAT Gateway to access the internet. (We recommended this approach.)
- Instances launched in the `compute_subnet` have public IP addresses and can reach the internet through an Internet Gateway.

Additionally, if you are 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 does not 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 is configured to use a NAT Gateway.

This leaves us with the only option, to configure a NAT Gateway in order to enable the cluster to execute multi-node parallel jobs.



For more information, see the following AWS documents:

- [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 during cluster creation. In most cases, this code is stored in Amazon Simple Storage Service (Amazon S3) and accessed through an HTTPS connection. The code is executed as root and can be in any script language that is 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 serve 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 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 executed before any of the boot_as_* scripts are run
# (defaults to NONE)
pre_install = NONE
# Arguments to be passed to preinstall script
# (defaults to NONE)
pre_install_args = NONE
# URL to a postinstall script. This is executed after any of the boot_as_* scripts are run
# (defaults to NONE)
post_install = NONE
# Arguments to be passed to postinstall script
# (defaults to NONE)
post_install_args = NONE
```

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 are not appropriate, use either [s3_read_resource \(p. 84\)](#) or [s3_read_write_resource \(p. 85\)](#) parameters to grant access. For more information, see [Working with Amazon S3 \(p. 37\)](#).

```
$ 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

You can access Amazon S3 from within AWS ParallelCluster. You control the access to Amazon S3 through the [s3_read_resource](#) (p. 84) and [s3_read_write_resource](#) (p. 85) parameters in the AWS ParallelCluster configuration.

```
# Specify Amazon S3 resource which AWS ParallelCluster nodes will be granted read-only
access
# (defaults to NONE)
s3_read_resource = NONE
# Specify Amazon S3 resource which AWS ParallelCluster nodes will be granted read-write
access
# (defaults to NONE)
s3_read_write_resource = NONE
```

Both parameters accept either * or a valid Amazon S3 ARN. For details 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](#) (p. 74) = spot. The primary feature of Spot Instances is they are available for less than the cost of On-Demand Instances, but it is possible that they might be interrupted. The effect of the interruption varies depending on the scheduler used. It may 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 for Linux Instances*. The following sections describe three scenarios in which Spot Instances can be interrupted.

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](#) (p. 79) setting. If AWS ParallelCluster is unable to 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 is terminated and given a state code of `NODE_FAIL`. The compute instance is removed from the scheduler queue.

SGE

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_ODEL_IF_UNKNOWN`
- `ENABLE_RESCHEDULE_KILL=1`

Torque

The job is removed from the system and the node is removed from the scheduler. The job is not rerun. If multiple jobs are running on the instance when it is interrupted, Torque may time out during node removal. An error may display in the [sqswatcher](#) (p. 114) log file. This does not 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 is terminated and given a state code of `NODE_FAIL`. The compute instance is removed from the scheduler queue. Other nodes that were running the terminated jobs may be scaled down after the configured [scaledown_idletime](#) (p. 108) time has passed.

SGE

The job is not 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 (`ghost`), although this does not affect AWS ParallelCluster. To remove the host from the list, you could run the following command after replacing the instance.

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

Torque

The job is removed from the system and the node is removed from the scheduler. The job is not rerun. If multiple jobs are running on the instance when it is interrupted, Torque may time out during node removal. An error may display in the [sqswatcher \(p. 114\)](#) log file. This does not 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 for Linux Instances*.

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 \(p. 117\)](#) section.

Defaults

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

Using an existing IAM role for Amazon EC2

You can use an existing IAM role for Amazon EC2 when creating a cluster, but you must first define the IAM policy and role before attempting to launch the cluster. Typically, you choose an existing IAM role for Amazon EC2 to reduce the permissions that are granted to users as they launch clusters. The following example shows an IAM policy and role for both Amazon EC2 and the AWS ParallelCluster. You must create both policies and roles as individual policies in IAM and then attach the roles and policies to the appropriate resources. In the policies, replace *<REGION>*, *<AWS ACCOUNT ID>*, and similar strings with the appropriate values.

AWS ParallelCluster example instance and user policies

The following example policies include Amazon Resource Names (ARNs) for the resources. If you are working in the AWS GovCloud (US) or AWS China partitions, the ARNs 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 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*.

Topics

- [ParallelClusterInstancePolicy using SGE, Slurm, or Torque \(p. 40\)](#)
- [ParallelClusterInstancePolicy using awsbatch \(p. 42\)](#)
- [ParallelClusterUserPolicy using SGE, Slurm, or Torque \(p. 44\)](#)
- [ParallelClusterUserPolicy using awsbatch \(p. 50\)](#)
- [ParallelClusterUserPolicy for users \(p. 55\)](#)

ParallelClusterInstancePolicy using SGE, Slurm, or Torque

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

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ec2:DescribeVolumes",
        "ec2:AttachVolume",
        "ec2:DescribeInstanceAttribute",
        "ec2:DescribeInstanceStatus",
        "ec2:DescribeInstances",
        "ec2:DescribeRegions",
        "ec2:RunInstances",
        "ec2:TerminateInstances",
        "ec2:DescribeLaunchTemplates",
        "ec2:CreateTags"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow",
      "Sid": "EC2"
    },
    {
      "Action": [
        "dynamodb:ListTables"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow",
      "Sid": "DynamoDBList"
    },
    {
      "Action": [
        "sqs:SendMessage",
        "sqs:ReceiveMessage",
        "sqs:ChangeMessageVisibility",
        "sqs>DeleteMessage",
        "sqs:GetQueueUrl"
      ],
      "Resource": [
        "arn:aws:sqs:<REGION>:<AWS ACCOUNT ID>:parallelcluster-*"
      ],
      "Effect": "Allow",
      "Sid": "SQSQueue"
    },
    {

```

```

    "Action": [
      "autoscaling:DescribeAutoScalingGroups",
      "autoscaling:TerminateInstanceInAutoScalingGroup",
      "autoscaling:SetDesiredCapacity",
      "autoscaling:UpdateAutoScalingGroup",
      "autoscaling:DescribeTags",
      "autoscaling:SetInstanceHealth"
    ],
    "Resource": [
      "*"
    ],
    "Effect": "Allow",
    "Sid": "Autoscaling"
  },
  {
    "Action": [
      "cloudformation:DescribeStacks",
      "cloudformation:DescribeStackResource",
      "cloudformation:SignalResource"
    ],
    "Resource": [
      "arn:aws:cloudformation:<REGION>:<AWS ACCOUNT ID>:stack/parallelcluster-*/
*"
    ],
    "Effect": "Allow",
    "Sid": "CloudFormation"
  },
  {
    "Action": [
      "dynamodb:PutItem",
      "dynamodb:Query",
      "dynamodb:GetItem",
      "dynamodb:BatchWriteItem",
      "dynamodb>DeleteItem",
      "dynamodb:DescribeTable"
    ],
    "Resource": [
      "arn:aws:dynamodb:<REGION>:<AWS ACCOUNT ID>:table/parallelcluster-*"
    ],
    "Effect": "Allow",
    "Sid": "DynamoDBTable"
  },
  {
    "Action": [
      "s3:GetObject"
    ],
    "Resource": [
      "arn:aws:s3:::<REGION>-aws-parallelcluster/*"
    ],
    "Effect": "Allow",
    "Sid": "S3GetObj"
  },
  {
    "Action": [
      "sqs:ListQueues"
    ],
    "Resource": [
      "*"
    ],
    "Effect": "Allow",
    "Sid": "SQSList"
  },
  {
    "Action": [
      "iam:PassRole"
    ],

```

```
    "Resource": [
      "*"
    ],
    "Effect": "Allow",
    "Sid": "IAMPassRole"
  },
  {
    "Action": [
      "s3:GetObject"
    ],
    "Resource": [
      "arn:aws:s3:::dcv-license.<REGION>/*"
    ],
    "Effect": "Allow",
    "Sid": "DcvLicense"
  },
  {
    "Action": [
      "s3:GetObject",
      "s3:GetObjectVersion"
    ],
    "Resource": [
      "arn:aws:s3:::parallelcluster-*/*"
    ],
    "Effect": "Allow",
    "Sid": "GetClusterConfig"
  },
  {
    "Action": [
      "fsx:DescribeFileSystems"
    ],
    "Resource": [
      "*"
    ],
    "Effect": "Allow",
    "Sid": "FSx"
  },
  {
    "Action": [
      "logs:CreateLogStream"
    ],
    "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 AWS CloudFormation nested stack. The BatchUserRole ARN is provided as a stack output. The following example is an overview of the required permissions:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "batch:SubmitJob",
        "batch:RegisterJobDefinition",
        "cloudformation:DescribeStacks",
        "ecs:ListContainerInstances",
        "ecs:DescribeContainerInstances",
        "logs:GetLogEvents",
        "logs:FilterLogEvents",
        "s3:PutObject",
        "s3:Get*",
        "s3>DeleteObject",
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:batch:<REGION>:<AWS ACCOUNT ID>:job-definition/<AWS_BATCH_STACK -
        JOB_DEFINITION_SERIAL_NAME>:1",
        "arn:aws:batch:<REGION>:<AWS ACCOUNT ID>:job-definition/<AWS_BATCH_STACK -
        JOB_DEFINITION_MNP_NAME>*",
        "arn:aws:batch:<REGION>:<AWS ACCOUNT ID>:job-queue/<AWS_BATCH_STACK -
        JOB_QUEUE_NAME>",
        "arn:aws:cloudformation:<REGION>:<AWS ACCOUNT ID>:stack/<STACK NAME>/*",
        "arn:aws:s3:::<RESOURCES S3 BUCKET>/batch/*",
        "arn:aws:iam::<AWS ACCOUNT ID>:role/<AWS_BATCH_STACK - JOB_ROLE>",
        "arn:aws:ecs:<REGION>:<AWS ACCOUNT ID>:cluster/<ECS COMPUTE ENVIRONMENT>",
        "arn:aws:ecs:<REGION>:<AWS ACCOUNT ID>:container-instance/*",
        "arn:aws:logs:<REGION>:<AWS ACCOUNT ID>:log-group:/aws/batch/job:log-
stream:*"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "s3:List*"
      ],
      "Resource": [
        "arn:aws:s3:::<RESOURCES S3 BUCKET>"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "batch:DescribeJobQueues",
        "batch:TerminateJob",
        "batch:DescribeJobs",
        "batch:CancelJob",
        "batch:DescribeJobDefinitions",
        "batch:ListJobs",
        "batch:DescribeComputeEnvironments"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "ec2:DescribeInstances",
        "ec2:AttachVolume",
        "ec2:DescribeVolumes",
        "ec2:DescribeInstanceAttribute"
      ]
    }
  ]
}
```

```

    ],
    "Resource": "*",
    "Effect": "Allow",
    "Sid": "EC2"
  },
  {
    "Action": [
      "cloudformation:DescribeStackResource",
      "cloudformation:SignalResource"
    ],
    "Resource": "*",
    "Effect": "Allow",
    "Sid": "CloudFormation"
  },
  {
    "Action": [
      "fsx:DescribeFileSystems"
    ],
    "Resource": [
      "*"
    ],
    "Effect": "Allow",
    "Sid": "FSx"
  },
  {
    "Action": [
      "logs:CreateLogGroup",
      "logs:CreateLogStream"
    ],
    "Resource": [
      "*"
    ],
    "Effect": "Allow",
    "Sid": "CWLogs"
  }
]
}

```

ParallelClusterUserPolicy using SGE, Slurm, or Torque

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

Note

If you use a custom role, `ec2_iam_role` (p. 77) = `<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>"

```

{
  "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: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: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": [
      "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:::parallelcluster-*"
    ],
  },
```

```

    "Effect": "Allow",
    "Sid": "S3ResourcesBucket"
  },
  {
    "Action": [
      "s3:Get*",
      "s3:List*"
    ],
    "Resource": [
      "arn:aws:s3:::<REGION>-aws-parallelcluster*"
    ],
    "Effect": "Allow",
    "Sid": "S3ParallelClusterReadOnly"
  },
  {
    "Action": [
      "s3:DeleteBucket",
      "s3:DeleteObject",
      "s3:DeleteObjectVersion"
    ],
    "Resource": [
      "arn:aws:s3:::parallelcluster-*"
    ],
    "Effect": "Allow",
    "Sid": "S3Delete"
  },
  {
    "Action": [
      "iam:PassRole",
      "iam:CreateRole",
      "iam:CreateServiceLinkedRole",
      "iam:DeleteRole",
      "iam:GetRole",
      "iam:TagRole",
      "iam:SimulatePrincipalPolicy"
    ],
    "Resource": [
      "arn:aws:iam::<AWS ACCOUNT ID>:role/<PARALLELCLUSTER EC2 ROLE NAME>",
      "arn:aws:iam::<AWS ACCOUNT ID>:role/parallelcluster-*",
      "arn:aws:iam::<AWS ACCOUNT ID>:role/aws-service-role/*"
    ],
    "Effect": "Allow",
    "Sid": "IAMModify"
  },
  {
    "Action": [
      "iam:CreateInstanceProfile",
      "iam:DeleteInstanceProfile"
    ],
    "Resource": "arn:aws:iam::<AWS ACCOUNT ID>:instance-profile/*",
    "Effect": "Allow",
    "Sid": "IAMCreateInstanceProfile"
  },
  {
    "Action": [
      "iam:AddRoleToInstanceProfile",
      "iam:RemoveRoleFromInstanceProfile",
      "iam:GetRolePolicy",
      "iam:GetPolicy",
      "iam:AttachRolePolicy",
      "iam:DetachRolePolicy",
      "iam:PutRolePolicy",
      "iam>DeleteRolePolicy"
    ],
    "Resource": "*",
    "Effect": "Allow",
  }

```

```

    "Sid": "IAMInstanceProfile"
  },
  {
    "Action": [
      "elasticfilesystem:DescribeMountTargets",
      "elasticfilesystem:DescribeMountTargetSecurityGroups",
      "ec2:DescribeNetworkInterfaceAttribute"
    ],
    "Resource": "*",
    "Effect": "Allow",
    "Sid": "EFSDescribe"
  },
  {
    "Action": [
      "ssm:GetParametersByPath"
    ],
    "Resource": "*",
    "Effect": "Allow",
    "Sid": "SSMDescribe"
  },
  {
    "Action": [
      "fsx:*"
    ],
    "Resource": "*",
    "Effect": "Allow",
    "Sid": "FSx"
  },
  {
    "Action": [
      "elasticfilesystem:*"
    ],
    "Resource": "*",
    "Effect": "Allow",
    "Sid": "EFS"
  },
  {
    "Action": [
      "logs:DeleteLogGroup",
      "logs:PutRetentionPolicy",
      "logs:DescribeLogGroups",
      "logs:CreateLogGroup"
    ],
    "Resource": "*",
    "Effect": "Allow",
    "Sid": "CloudWatchLogs"
  },
  {
    "Action": [
      "lambda:CreateFunction",
      "lambda>DeleteFunction",
      "lambda:GetFunctionConfiguration",
      "lambda:GetFunction",
      "lambda:InvokeFunction",
      "lambda:AddPermission",
      "lambda:RemovePermission"
    ],
    "Resource": [
      "arn:aws:lambda:<REGION>:<AWS ACCOUNT ID>:function:parallelcluster-*",
      "arn:aws:lambda:<REGION>:<AWS ACCOUNT ID>:function:pcluster-*"
    ],
    "Effect": "Allow",
    "Sid": "Lambda"
  }
]

```

```
}
```

ParallelClusterUserPolicy using awsbatch

The following example sets the ParallelClusterUserPolicy using awsbatch as the scheduler:

```
{
  "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: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:PutItem",
      "dynamodb:Query",
      "dynamodb:TagResource"
    ],
    "Resource": "arn:aws:dynamodb:<REGION>:<AWS ACCOUNT ID>:table/parallelcluster-
  },
  {
    "Action": [
      "cloudformation:DescribeStackEvents",
      "cloudformation:DescribeStackResource",
      "cloudformation:DescribeStackResources",
      "cloudformation:DescribeStacks",
      "cloudformation:ListStacks",
      "cloudformation:GetTemplate",
      "cloudformation:CreateStack",
      "cloudformation>DeleteStack",
      "cloudformation:UpdateStack"
    ],
    "Resource": "arn:aws:cloudformation:<REGION>:<AWS ACCOUNT ID>:stack/
parallelcluster-*",
    "Effect": "Allow",
    "Sid": "CloudFormation"
  },
  {
    "Action": [
      "route53:ChangeResourceRecordSets",
      "route53:ChangeTagsForResource",
      "route53:CreateHostedZone",

```

```
        "route53:DeleteHostedZone",
        "route53:GetChange",
        "route53:GetHostedZone",
        "route53:ListResourceRecordSets"
    ],
    "Resource": "arn:aws:route53::hostedzone/*",
    "Effect": "Allow",
    "Sid": "Route53HostedZones"
},
{
    "Action": [
        "sqs:GetQueueAttributes",
        "sqs:CreateQueue",
        "sqs:SetQueueAttributes",
        "sqs>DeleteQueue",
        "sqs:TagQueue"
    ],
    "Resource": "*",
    "Effect": "Allow",
    "Sid": "SQS"
},
{
    "Action": [
        "sqs:SendMessage",
        "sqs:ReceiveMessage",
        "sqs:ChangeMessageVisibility",
        "sqs>DeleteMessage",
        "sqs:GetQueueUrl"
    ],
    "Resource": "arn:aws:sqs:<REGION>:<AWS ACCOUNT ID>:parallelcluster-*",
    "Effect": "Allow",
    "Sid": "SQSQueue"
},
{
    "Action": [
        "sns:ListTopics",
        "sns:GetTopicAttributes",
        "sns:CreateTopic",
        "sns:Subscribe",
        "sns:Unsubscribe",
        "sns>DeleteTopic"
    ],
    "Resource": "*",
    "Effect": "Allow",
    "Sid": "SNS"
},
{
    "Action": [
        "iam:PassRole",
        "iam:CreateRole",
        "iam>DeleteRole",
        "iam:GetRole",
        "iam:TagRole",
        "iam:SimulatePrincipalPolicy"
    ],
    "Resource": [
        "arn:aws:iam::<AWS ACCOUNT ID>:role/parallelcluster-*",
        "arn:aws:iam::<AWS ACCOUNT ID>:role/<PARALLELCLUSTER EC2 ROLE NAME>"
    ],
    "Effect": "Allow",
    "Sid": "IAMRole"
},
{
    "Action": [
        "iam:CreateInstanceProfile",
        "iam>DeleteInstanceProfile",
```

```

    "iam:GetInstanceProfile",
    "iam:PassRole"
  ],
  "Resource": "arn:aws:iam::<AWS ACCOUNT ID>:instance-profile/*",
  "Effect": "Allow",
  "Sid": "IAMInstanceProfile"
},
{
  "Action": [
    "iam:AddRoleToInstanceProfile",
    "iam:RemoveRoleFromInstanceProfile",
    "iam:GetRolePolicy",
    "iam:PutRolePolicy",
    "iam>DeleteRolePolicy",
    "iam:GetPolicy",
    "iam:AttachRolePolicy",
    "iam:DetachRolePolicy"
  ],
  "Resource": "*",
  "Effect": "Allow",
  "Sid": "IAM"
},
{
  "Action": [
    "s3:*"
  ],
  "Resource": [
    "arn:aws:s3::parallelcluster-*"
  ],
  "Effect": "Allow",
  "Sid": "S3ResourcesBucket"
},
{
  "Action": [
    "s3:Get*",
    "s3:List*"
  ],
  "Resource": [
    "arn:aws:s3::<REGION>-aws-parallelcluster/*"
  ],
  "Effect": "Allow",
  "Sid": "S3ParallelClusterReadOnly"
},
{
  "Action": [
    "s3:DeleteBucket",
    "s3:DeleteObject",
    "s3:DeleteObjectVersion"
  ],
  "Resource": [
    "arn:aws:s3::parallelcluster-*"
  ],
  "Effect": "Allow",
  "Sid": "S3Delete"
},
{
  "Action": [
    "lambda:CreateFunction",
    "lambda>DeleteFunction",
    "lambda:GetFunction",
    "lambda:GetFunctionConfiguration",
    "lambda:InvokeFunction",
    "lambda:AddPermission",
    "lambda:RemovePermission"
  ],
  "Resource": [

```

```

    "arn:aws:lambda:<REGION>:<AWS ACCOUNT ID>:function:parallelcluster-*",
    "arn:aws:lambda:<REGION>:<AWS ACCOUNT ID>:function:pcluster-*"
  ],
  "Effect": "Allow",
  "Sid": "Lambda"
},
{
  "Action": [
    "logs:*"
  ],
  "Resource": "arn:aws:logs:<REGION>:<AWS ACCOUNT ID>:*",
  "Effect": "Allow",
  "Sid": "Logs"
},
{
  "Action": [
    "codebuild:*"
  ],
  "Resource": "arn:aws:codebuild:<REGION>:<AWS ACCOUNT ID>:project/parallelcluster-*",
  "Effect": "Allow",
  "Sid": "CodeBuild"
},
{
  "Action": [
    "ecr:*"
  ],
  "Resource": "*",
  "Effect": "Allow",
  "Sid": "ECR"
},
{
  "Action": [
    "batch:*"
  ],
  "Resource": "*",
  "Effect": "Allow",
  "Sid": "Batch"
},
{
  "Action": [
    "events:*"
  ],
  "Effect": "Allow",
  "Resource": "*",
  "Sid": "AmazonCloudWatchEvents"
},
{
  "Action": [
    "ecs:DescribeContainerInstances",
    "ecs:ListContainerInstances"
  ],
  "Resource": "*",
  "Effect": "Allow",
  "Sid": "ECS"
},
{
  "Action": [
    "elasticfilesystem:CreateFileSystem",
    "elasticfilesystem:CreateMountTarget",
    "elasticfilesystem>DeleteFileSystem",
    "elasticfilesystem>DeleteMountTarget",
    "elasticfilesystem:DescribeFileSystems",
    "elasticfilesystem:DescribeMountTargets"
  ],
  "Resource": "*",

```

```

    "Effect": "Allow",
    "Sid": "EFS"
  },
  {
    "Action": [
      "fsx:*"
    ],
    "Resource": "*",
    "Effect": "Allow",
    "Sid": "FSx"
  }
]
}

```

ParallelClusterUserPolicy for users

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

- [pcluster dcw \(p. 23\)](#)
- [pcluster instances \(p. 25\)](#)
- [pcluster list \(p. 25\)](#)
- [pcluster ssh \(p. 26\)](#)
- [pcluster start \(p. 27\)](#)
- [pcluster status \(p. 27\)](#)
- [pcluster stop \(p. 28\)](#)
- [pcluster version \(p. 30\)](#)

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "MinimumModify",
      "Action": [
        "autoscaling:UpdateAutoScalingGroup",
        "batch:UpdateComputeEnvironment",
        "cloudformation:DescribeStackEvents",
        "cloudformation:DescribeStackResources",
        "cloudformation:GetTemplate",
        "dynamodb:GetItem",
        "dynamodb:PutItem"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:autoscaling:<REGION>:<AWS ACCOUNT ID>:autoScalingGroup:*:autoScalingGroupName/parallelcluster-*",
        "arn:aws:batch:<REGION>:<AWS ACCOUNT ID>:compute-environment/*",
        "arn:aws:cloudformation:<REGION>:<AWS ACCOUNT ID>:stack/<CLUSTERNAME>/*",
        "arn:aws:dynamodb:<REGION>:<AWS ACCOUNT ID>:table/<CLUSTERNAME>"
      ]
    },
    {
      "Sid": "Describe",
      "Action": [
        "cloudformation:DescribeStacks",
        "ec2:DescribeInstances",
        "ec2:DescribeInstanceStatus"
      ],
      "Effect": "Allow",
    }
  ]
}

```

```
    "Resource": "*"
  }
]
}
```

Schedulers supported by AWS ParallelCluster

AWS ParallelCluster supports several schedulers, set using the [scheduler](#) (p. 85) setting.

Warning

Starting on December 31, 2021, AWS will no longer include SGE and Torque support for all released versions of AWS ParallelCluster. Previous versions of AWS ParallelCluster that support SGE and Torque will still be available for download and use. However, these versions will not be eligible for future updates or troubleshooting support from AWS service and customer support teams. Moreover, future releases of AWS ParallelCluster made before and after December 31, 2021 will not include support for either SGE or Torque.

Topics

- [Son of Grid Engine \(sge\)](#) (p. 56)
- [Slurm Workload Manager \(slurm\)](#) (p. 56)
- [Torque Resource Manager \(torque\)](#) (p. 56)
- [AWS Batch \(awsbatch\)](#) (p. 57)

Son of Grid Engine (sge)

Warning

Starting on December 31, 2021, AWS will no longer include SGE support for all released versions of AWS ParallelCluster. Previous versions of AWS ParallelCluster that support SGE will still be available for download and use. However, these versions will not be eligible for future updates or troubleshooting support from AWS service and customer support teams. Moreover, future releases of AWS ParallelCluster made before and after December 31, 2021 will not include support for SGE.

AWS ParallelCluster uses Son of Grid Engine 8.1.9. For information about this scheduler, see <https://arc.liv.ac.uk/trac/SGE>. For downloads, see <https://arc.liv.ac.uk/downloads/SGE/releases/8.1.9/>. For the source code, see <https://arc.liv.ac.uk/trac/SGE/browser/sge>.

Slurm Workload Manager (slurm)

AWS ParallelCluster uses Slurm 20.02.4. For information about Slurm, see <https://slurm.schedmd.com/>. For downloads, see <https://www.schedmd.com/downloads.php>. For the source code, see <https://github.com/SchedMD/slurm>.

AWS ParallelCluster versions between 2.6 and 2.8.1 use [Slurm 19.05.5](#). AWS ParallelCluster versions 2.5.0 and 2.5.1 use [Slurm 19.05.3-2](#). AWS ParallelCluster versions between 2.3.1 and 2.4.1 use [Slurm 18.08.6-2](#). AWS ParallelCluster versions prior to 2.3.1 use Slurm 16.05.3-1, which is no longer available for download.

Torque Resource Manager (torque)

Warning

Starting on December 31, 2021, AWS will no longer include Torque support for all released versions of AWS ParallelCluster. Previous versions of AWS ParallelCluster that support Torque will still be available for download and use. However, these versions will not be eligible for

future updates or troubleshooting support from AWS service and customer support teams. Moreover, future releases of AWS ParallelCluster made before and after December 31, 2021 will not include support for Torque.

AWS ParallelCluster uses 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

Important

When using AWS Batch, GPU jobs are not supported.

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.

Topics

- [awsbsub \(p. 57\)](#)
- [awsbstat \(p. 59\)](#)
- [awsbout \(p. 60\)](#)
- [awsbkill \(p. 60\)](#)
- [awsbqueues \(p. 61\)](#)
- [awsbhosts \(p. 61\)](#)

awsbsub

Submits jobs to the cluster's job queue.

```
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 ...]
```

Positional Arguments

command

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

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 alphanumeric. The job name can contain up to 128 characters. Letters (both uppercase and lowercase), numbers, hyphens, and underscores are allowed.

-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 is not specified, the job is executed 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 is not 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 per 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 killed.

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 will not 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 RUNNABLE that 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 has not 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 are not supported when the `cluster_type` (p. 74) 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=<string>,type=<string>;...".

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

awskill

Cancels or terminates jobs submitted in the cluster.

```
awskill [ - 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

Multiple queue mode

AWS ParallelCluster version 2.9.0 introduced multiple queue mode. Multiple queue mode is supported when [scheduler \(p. 85\)](#) is set to `slurm` and the [queue_settings \(p. 84\)](#) setting is defined. This mode allows different instance types to coexist in the compute nodes, and the compute resources that contain the different instance types can scale up or scale down as needed. In queue mode, up to five (5) queues are supported, and each [\[queue\] section \(p. 103\)](#) can refer to up to three (3) [\[compute_resource\] sections \(p. 88\)](#). Each of these [\[queue\] sections \(p. 103\)](#) is a partition in Slurm Workload Manager.

Each [\[compute_resource\] section \(p. 88\)](#) 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 \(p. 89\)](#). Dynamic nodes for each [\[compute_resource\]](#) are numbered from one (1) to ([max_count \(p. 89\)](#) - [min_count](#)). For example, if [min_count](#) is 2 and [max_count](#) is 10, the dynamic nodes for that [\[compute_resource\]](#) would be 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:

```
$HOSTNAME=$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 \(p. 88\)](#) setting.
- `$NODENUM` is the number of the node. `$NODENUM` is between one (1) and the value of [min_count \(p. 89\)](#) for static nodes and between one (1) and ([max_count \(p. 89\)](#) - [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 \(p. 71\)](#) used for the cluster.

To convert your configuration to a queue mode, use the [pcluster-config convert \(p. 30\)](#) command. It will write an updated configuration with a single [\[queue\] section \(p. 103\)](#) named [\[queue compute\]](#). That queue will contain a single [\[compute_resource\] section \(p. 88\)](#) named [\[compute_resource default\]](#). The [\[queue compute\]](#) and [\[compute_resource default\]](#) will have settings migrated from the specified [\[cluster\] section \(p. 71\)](#).

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 \(p. 89\)](#) and the [cw_log_settings \(p. 75\)](#) 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.

This list contains the path of the logs and the *logIdentifier* used for those logs.

- `/opt/sge/default/spool/qmaster/messages` (sge-qmaster)
- `/var/log/cfn-init.log` (cfn-init)
- `/var/log/chef-client.log` (chef-client)
- `/var/log/cloud-init.log` (cloud-init)
- `/var/log/cloud-init-output.log` (cloud-init-output)
- `/var/log/dcv/agent.*.log` (dcv-agent)
- `/var/log/dcv/dcv-xsession.*.log` (dcv-xsession)
- `/var/log/dcv/server.log` (dcv-server)
- `/var/log/dcv/sessionlauncher.log` (dcv-session-launcher)
- `/var/log/dcv/Xdcv.*.log` (Xdcv)
- `/var/log/jobwatcher` (jobwatcher)
- `/var/log/messages` (system-messages)
- `/var/log/nodewatcher` (nodewatcher)
- `/var/log/parallelcluster/clustermgtd` (clustermgtd)
- `/var/log/parallelcluster/computemgtd` (computemgtd)
- `/var/log/parallelcluster/pcluster_dcv_authenticator.log` (dcv-authenticator)
- `/var/log/parallelcluster/pcluster_dcv_connect.log` (dcv-ext-authenticator)
- `/var/log/parallelcluster/slurm_resume.log` (slurm_resume)
- `/var/log/parallelcluster/slurm_suspend.log` (slurm_suspend)
- `/var/log/slurmctld.log` (slurmctld)
- `/var/log/slurmd.log` (slurmd)
- `/var/log/sqswatcher` (sqswatcher)
- `/var/log/supervisord.log` (supervisord)
- `/var/log/syslog` (syslog)
- `/var/spool/sge/*/messages` (sge-exec-daemon)
- `/var/spool/torque/client_logs/*` (torque-client)
- `/var/spool/torque/server_logs/*` (torque-server)

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 = compute` to the `[cluster]` section (p. 71). EFA is supported by specific instance types (`compute_instance_type` (p. 75)) are to one of `c5n.18xlarge`, `c5n.metal`, `g4dn.metal`, `i3en.24xlarge`, `i3en.metal`, `m5dn.24xlarge`, `m5n.24xlarge`, `r5dn.24xlarge`, `r5n.24xlarge`, and `p3dn.24xlarge` on specific operating systems (`base_os` (p. 73)) is `alinux`, `alinux2`, `centos7`, `ubuntu1604`, or `ubuntu1804`). For more information about the `enable_efa` setting, see `enable_efa` (p. 77). A cluster placement group should be used to minimize latencies between instances. For more information, see `placement` (p. 81) and `placement_group` (p. 82).

For more information, see [Elastic Fabric Adapter](#) in the *Amazon EC2 User Guide for Linux Instances* 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 for Linux Instances*.

Enable Intel MPI

Intel MPI is available on the AWS ParallelCluster AMIs for `alinux`, `alinux2`, `centos7`, `ubuntu1604`, and `ubuntu1804` values for the `base_os` (p. 73) setting. 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 modulelet. 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
```

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

```
$ 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 \(p. 73\)](#) = `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 \(p. 75\)](#) = `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 \(p. 78\)](#) = `true`).
- Each compute node must have at least 80 GB of storage ([compute_root_volume_size \(p. 75\)](#) = `80`).

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

Connect to the head node through NICE DCV

NICE 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 [NICE DCV](#).

The NICE DCV software is automatically installed on the head node when using [base_os \(p. 73\)](#) = `alinux2`, [base_os \(p. 73\)](#) = `centos7`, or [base_os \(p. 73\)](#) = `ubuntu1804`.

To enable NICE DCV on the head node, [dcv_settings](#) (p. 76) must contain the name of a [\[dcv\]](#) [section](#) (p. 90) that has [enable](#) (p. 91) = `master` and [base_os](#) (p. 73) must be set to `alinux2`, `centos7` or `ubuntu1804`.

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

For more information about NICE DCV configuration parameters, see [dcv_settings](#) (p. 76). To connect to the NICE DCV session, use the `pcluster dcv` (p. 23) command.

Note

Support for NICE DCV on AWS Graviton-based instances was added in AWS ParallelCluster version 2.9.0. Support for NICE DCV on `alinux2` and `ubuntu1804` was added in AWS ParallelCluster version 2.6.0. Support for NICE DCV on `centos7` was added in AWS ParallelCluster version 2.5.0.

Note

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

NICE DCV HTTPS certificate

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

To replace the default self-signed NICE 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` (p. 23) command.

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

Licensing NICE DCV

The NICE DCV server doesn't require a license server when running on Amazon EC2 instances. However, the NICE 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 [NICE DCV on Amazon EC2](#) in the *NICE DCV Administrator Guide*.

For troubleshooting tips, see [NICE DCV troubleshooting](#) (p. 146).

Using `pcluster update`

Starting with AWS ParallelCluster version 2.8.0, `pcluster update` (p. 29) 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](#) (p. 75) 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` (p. 29).

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` (p. 24)), and then a new cluster created (using `pcluster create` (p. 21)) 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` (p. 29).

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` (p. 28)), updated (using `pcluster update` (p. 29)), and then a new compute fleet created (using `pcluster start` (p. 27)).

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` (p. 24)), and create a new cluster (using `pcluster create` (p. 21)).

This example demonstrates a `pcluster update` (p. 29) 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.
```

Configuration

Topics

- [Layout](#) (p. 69)
- [\[global\]](#) section (p. 70)
- [\[aws\]](#) section (p. 70)
- [\[aliases\]](#) section (p. 71)
- [\[cluster\]](#) section (p. 71)
- [\[compute_resource\]](#) section (p. 88)
- [\[cw_log\]](#) section (p. 89)
- [\[dcv\]](#) section (p. 90)
- [\[ebs\]](#) section (p. 92)
- [\[efs\]](#) section (p. 94)
- [\[fsx\]](#) section (p. 97)
- [\[queue\]](#) section (p. 103)
- [\[raid\]](#) section (p. 105)
- [\[scaling\]](#) section (p. 107)
- [\[vpc\]](#) section (p. 108)
- [Example](#) (p. 37)

By default, AWS ParallelCluster uses the file `~/.parallelcluster/config` 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/v2.9.1/cli/pcluster/examples/config>.

Layout

An AWS ParallelCluster configuration is defined in multiple sections.

The following sections are required: [\[global\]](#) section (p. 70) and [\[aws\]](#) section (p. 70).

You also must include at least one [\[cluster\]](#) section (p. 71) and one [\[vpc\]](#) section (p. 108).

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

Topics

- [cluster_template](#) (p. 70)
- [update_check](#) (p. 70)
- [sanity_check](#) (p. 70)

Specifies global configuration options related to `pcluster`.

```
[global]
```

cluster_template

Defines the name of the `cluster` section that is used by default for the cluster. For additional information about `cluster` sections, see [\[cluster\] section](#) (p. 71). 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. (p. 67)

update_check

Checks for updates to `pcluster`.

The default value is `true`.

```
update_check = true
```

Update policy: This setting is not analyzed during an update. (p. 67)

sanity_check

Attempts to validate the existence of the resources that are defined in the cluster parameters.

The default value is `true`.

```
sanity_check = true
```

Note

Prior to AWS ParallelCluster version 2.5.0, [sanity_check](#) (p. 70) defaulted to `false`.

Update policy: This setting is not analyzed during an update. (p. 67)

[aws] section

Specifies AWS Region information.

These settings apply to all clusters and are required.

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]
# Defaults to us-east-1 if not defined in environment or below
aws_region_name = #region
```

Update policy: This setting is not analyzed during an update. (p. 67)

[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. (p. 67)

[cluster] section

Topics

- [additional_cfn_template](#) (p. 73)
- [additional_iam_policies](#) (p. 73)
- [base_os](#) (p. 73)
- [cluster_type](#) (p. 74)
- [compute_instance_type](#) (p. 75)
- [compute_root_volume_size](#) (p. 75)
- [custom_ami](#) (p. 75)
- [cw_log_settings](#) (p. 75)
- [dcv_settings](#) (p. 76)
- [desired_vcpus](#) (p. 76)
- [disable_hyperthreading](#) (p. 76)
- [ebs_settings](#) (p. 77)
- [ec2_iam_role](#) (p. 77)
- [efs_settings](#) (p. 77)
- [enable_efa](#) (p. 77)

- [enable_intel_hpc_platform](#) (p. 78)
- [encrypted_ephemeral](#) (p. 78)
- [ephemeral_dir](#) (p. 78)
- [extra_json](#) (p. 79)
- [fsx_settings](#) (p. 79)
- [initial_queue_size](#) (p. 79)
- [key_name](#) (p. 80)
- [maintain_initial_size](#) (p. 80)
- [master_instance_type](#) (p. 80)
- [master_root_volume_size](#) (p. 80)
- [max_queue_size](#) (p. 81)
- [max_vcpus](#) (p. 81)
- [min_vcpus](#) (p. 81)
- [placement](#) (p. 81)
- [placement_group](#) (p. 82)
- [post_install](#) (p. 82)
- [post_install_args](#) (p. 83)
- [pre_install](#) (p. 83)
- [pre_install_args](#) (p. 83)
- [proxy_server](#) (p. 83)
- [queue_settings](#) (p. 84)
- [raid_settings](#) (p. 84)
- [s3_read_resource](#) (p. 84)
- [s3_read_write_resource](#) (p. 85)
- [scaling_settings](#) (p. 85)
- [scheduler](#) (p. 85)
- [shared_dir](#) (p. 86)
- [spot_bid_percentage](#) (p. 86)
- [spot_price](#) (p. 86)
- [tags](#) (p. 87)
- [template_url](#) (p. 87)
- [vpc_settings](#) (p. 87)

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 (p. 71) named by the `cluster_template` (p. 70) setting in the [global] section (p. 70) is used by default, but can be overridden on the `pcluster` (p. 19) 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]
```

additional_cfn_template

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.

When set to a value other than `NONE`, the value must be an HTTP URL to a public template, with all parameters provided.

The default value is `NONE`.

```
additional_cfn_template = NONE
```

Update policy: If this setting is changed, the update is not allowed. (p. 67)

additional_iam_policies

Specifies a comma-separated 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. An IAM policy name and its ARN are different. Names cannot be used as an argument to `additional_iam_policies` (p. 73). `additional_iam_policies` (p. 73) should be used instead of the `ec2_iam_role` (p. 77). This is because `additional_iam_policies` (p. 73) are added to the permissions that AWS ParallelCluster requires, and the `ec2_iam_role` (p. 77) must include all permissions required. The permissions required often change from release to release as features are added.

The default value is `NONE`.

```
additional_iam_policies = arn:aws:iam::aws:policy/AdministratorAccess
```

Note

Support for `additional_iam_policies` (p. 73) was added in AWS ParallelCluster version 2.5.0.

Update policy: This setting can be changed during an update. (p. 67)

base_os

(Required) Specifies which OS type is used in the cluster.

Available options are:

- `alinux`
- `alinux2`
- `centos6`
- `centos7`
- `ubuntu1604`
- `ubuntu1804`

Note

For AWS Graviton-based instances, only `alinux2` or `ubuntu1804` are supported.

Note

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 Regions mentioned in the following table that do not support `centos6` and `centos7`. All other AWS commercial Regions support all of the following operating systems.

Partition (Regions)	alinux and alinux2	centos6 and centos7	ubuntu1604 and ubuntu1804
Commercial (All Regions not specifically mentioned)	True	True	True
AWS GovCloud (US-East) (<code>us-gov-east-1</code>)	True	False	True
AWS GovCloud (US-West) (<code>us-gov-west-1</code>)	True	False	True
China (Beijing) (<code>cn-north-1</code>)	True	False	True
China (Ningxia) (<code>cn-northwest-1</code>)	True	False	True

Note

The `base_os` (p. 73) parameter also determines the user name that is used to log into the cluster.

- `centos6` and `centos7`: `centos`
- `ubuntu1604` and `ubuntu1804`: `ubuntu`
- `alinux` and `alinux2`: `ec2-user`

Note

Before AWS ParallelCluster version 2.7.0, the `base_os` (p. 73) parameter was optional, and the default was `alinux`. Starting with AWS ParallelCluster version 2.7.0, the `base_os` (p. 73) parameter is required.

Note

If the `scheduler` (p. 85) parameter is `awsbatch`, either `alinux` or `alinux2` is supported.

```
base_os = alinux
```

Update policy: If this setting is changed, the update is not allowed. (p. 67)

cluster_type

Defines the type of cluster to launch. If the `queue_settings` (p. 84) setting is defined, then this setting must be replaced by the `compute_type` (p. 104) settings in the `[queue]` sections (p. 103).

Valid options are: `ondemand`, and `spot`.

The default value is `ondemand`.

For more information about Spot Instances, see [Working with Spot Instances](#) (p. 37).

```
cluster_type = ondemand
```

Update policy: The compute fleet must be stopped for this setting to be changed for an update. (p. 67)

compute_instance_type

Defines the Amazon EC2 instance type that is 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](#) (p. 80) setting. If the [queue_settings](#) (p. 84) setting is defined, then this setting must be replaced by the [???](#) (p. 88) settings in the [\[compute_resource\]](#) sections (p. 88).

If you are 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 (such as A1 and C6g) 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. (p. 67)

compute_root_volume_size

Specifies the ComputeFleet root volume size in GB. The AMI must support growroot.

The default value is 25.

Note

Prior to AWS ParallelCluster version 2.5.0, the default was 20.

```
compute_root_volume_size = 20
```

Update policy: The compute fleet must be stopped for this setting to be changed for an update. (p. 67)

custom_ami

Specifies the ID of a custom AMI to use for the head and compute nodes instead of the default [published AMIs](#).

The default value is `NONE`.

```
custom_ami = NONE
```

Update policy: If this setting is changed, the update is not allowed. (p. 67)

cw_log_settings

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 (p. 89) and [Integration with Amazon CloudWatch Logs](#) (p. 63).

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` (p. 75) was added in AWS ParallelCluster version 2.6.0.

Update policy: If this setting is changed, the update is not allowed. (p. 67)

dcv_settings

Identifies the [`dcv`] section with the NICE 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 (p. 90).

For example, the following setting specifies that the section that starts [`dcv custom-dcv`] is used for the NICE DCV configuration.

```
dcv_settings = custom-dcv
```

Note

On AWS Graviton-based instances, NICE DCV is only supported on `linux2`.

Note

Support for `dcv_settings` (p. 76) was added in AWS ParallelCluster version 2.5.0.

Update policy: If this setting is changed, the update is not allowed. (p. 67)

desired_vcpus

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. (p. 67)

disable_hyperthreading

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 per CPU core per instance type](#) in the *Amazon EC2 User Guide for Linux Instances*.

If the `queue_settings` (p. 84) setting is defined, either this setting can be defined, or the `disable_hyperthreading` (p. 104) settings in the [`queue`] sections (p. 103) can be defined.

The default value is `false`.

```
disable_hyperthreading = true
```

Note

Support for `disable_hyperthreading` (p. 76) was added in AWS ParallelCluster version 2.5.0.

Update policy: If this setting is changed, the update is not allowed. (p. 67)

ebs_settings

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 as a comma-separated list. 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 (p. 92).

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. (p. 67)

ec2_iam_role

Defines the name of an existing IAM role for Amazon EC2 that is attached to all instances in the cluster. An IAM role name and its Amazon Resource Name (ARN) are different. ARNs cannot be used as an argument to `ec2_iam_role` (p. 77). If this option is specified, the `additional_iam_policies` (p. 73) setting is ignored. AWS recommends using `additional_iam_policies` (p. 73) rather than the `ec2_iam_role` (p. 77), because features added to AWS ParallelCluster often require new permissions.

The default value is NONE.

```
ec2_iam_role = NONE
```

Update policy: This setting can be changed during an update. (p. 67)

efs_settings

Specifies settings related to the Amazon EFS filesystem. 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 (p. 94).

For example, the following setting specifies that the section that starts [efs customfs] is used for the Amazon EFS filesystem configuration.

```
efs_settings = customfs
```

Update policy: If this setting is changed, the update is not allowed. (p. 67)

enable_efa

If present, specifies that Elastic Fabric Adapter (EFA) is enabled for the compute nodes. EFA is supported by specific instance types (ec5n.18xlarge, c5n.metal, i3en.24xlarge, m5dn.24xlarge,

m5n.24xlarge, r5dn.24xlarge, r5n.24xlarge, and p3dn.24xlarge) on specific operating systems ([base_os](#) (p. 73) is alinux, alinux2, centos7, ubuntu1604, or ubuntu1804). For more information, see [Elastic Fabric Adapter](#) (p. 64). If the [queue_settings](#) (p. 84) setting is defined, either this setting can be defined, or the [enable_efa](#) (p. 104) settings in the [\[queue\]](#) sections (p. 103) can be defined.

```
enable_efa = compute
```

Update policy: If this setting is changed, the update is not allowed. (p. 67)

enable_intel_hpc_platform

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](#) (p. 78) setting is only supported on CentOS 7 ([base_os](#) (p. 73) = centos7).

The default value is `false`.

```
enable_intel_hpc_platform = true
```

Note

The [enable_intel_hpc_platform](#) (p. 78) parameter is not compatible with AWS Graviton-based instances.

Note

Support for [enable_intel_hpc_platform](#) (p. 78) was added in AWS ParallelCluster version 2.5.0.

Update policy: If this setting is changed, the update is not allowed. (p. 67)

encrypted_ephemeral

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 = false
```

Update policy: If this setting is changed, the update is not allowed. (p. 67)

ephemeral_dir

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. (p. 67)

extra_json

Defines the extra JSON that is merged into the Chef `dna.json`. For more information, see [Building a Custom AWS ParallelCluster AMI \(p. 126\)](#).

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 start up times, add `"skip_install_recipes" : "no"` to the `cluster` key in the [extra_json \(p. 79\)](#) 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. \(p. 67\)](#)

fsx_settings

Specifies the section that defines the Amazon 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 \(p. 97\)](#).

For example, the following setting specifies that the section that starts `[fsx fs]` is used for the Amazon FSx for Lustre configuration.

```
fsx_settings = fs
```

Update policy: [If this setting is changed, the update is not allowed. \(p. 67\)](#)

initial_queue_size

Sets the initial number of Amazon EC2 instances to launch as compute nodes in the cluster. If the [queue_settings \(p. 84\)](#) setting is defined then this setting must be removed and replaced by the [initial_count \(p. 88\)](#) settings in the [\[compute_resource \] sections \(p. 88\)](#).

This setting is applicable only for traditional schedulers (SGE, Slurm, and Torque). If the [maintain_initial_size \(p. 80\)](#) setting is `true`, then the [initial_queue_size \(p. 79\)](#) setting must be at least 1.

If the scheduler is `awsbatch`, use [min_vcpus \(p. 81\)](#) instead.

Defaults to 2.

```
initial_queue_size = 2
```

Update policy: [This setting can be changed during an update. \(p. 67\)](#)

key_name

Names an existing Amazon EC2 key pair with which to enable SSH access to the instances.

```
key_name = mykey
```

Update policy: If this setting is changed, the update is not allowed. (p. 67)

maintain_initial_size

Maintains the initial size of the Auto Scaling group for traditional schedulers (SGE, Slurm, and Torque).

If the scheduler is `awsbatch`, use `desired_vcpus` (p. 76) 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` (p. 79), and the value of `initial_queue_size` (p. 79) must be 1 or greater. The cluster can still scale up to the value of `max_queue_size` (p. 81). If `cluster_type = spot` then the Auto Scaling group can have instances interrupted and the size can drop below `initial_queue_size` (p. 79).

If set to `false`, the Auto Scaling group can scale down to zero (0) members to prevent resources from sitting idle when they are not needed.

If the `queue_settings` (p. 84) setting is defined then this setting must be removed and replaced by the `initial_count` (p. 88) and `min_count` (p. 89) settings in the `[compute_resource]` sections (p. 88).

Defaults to `false`.

```
maintain_initial_size = false
```

Update policy: This setting can be changed during an update. (p. 67)

master_instance_type

Defines the Amazon EC2 instance type that is used for the head node. The architecture of the instance type must be the same as the architecture used for the `compute_instance_type` (p. 75) setting.

Defaults to `t2.micro`.

```
master_instance_type = t2.micro
```

Note

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. (p. 67)

master_root_volume_size

Specifies the head node root volume size in GB. The AMI must support `growroot`.

The default value is 25.

Note

Prior to AWS ParallelCluster version 2.5.0, the default was 20.

```
master_root_volume_size = 25
```

Update policy: If this setting is changed, the update is not allowed. (p. 67)

max_queue_size

Sets the maximum number of Amazon EC2 instances that can be launched in the cluster. If the [queue_settings](#) (p. 84) setting is defined, then this setting must be removed and replaced by the [max_count](#) (p. 89) settings in the [\[compute_resource\]](#) sections (p. 88).

This setting is applicable only for traditional schedulers (SGE, Slurm, and Torque).

If the scheduler is `awsbatch`, use [max_vcpus](#) (p. 81) instead.

Defaults to 10.

```
max_queue_size = 10
```

Update policy: This setting can be changed during an update. (p. 67)

max_vcpus

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. (p. 67)

min_vcpus

Maintains the initial size of the Auto Scaling group for the `awsbatch` scheduler.

If the scheduler is SGE, Slurm, or Torque, use [maintain_initial_size](#) (p. 80) instead.

The compute environment never has fewer members than the value of [min_vcpus](#) (p. 81).

Defaults to 0.

```
min_vcpus = 0
```

Update policy: This setting can be changed during an update. (p. 67)

placement

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](#) (p. 84) setting is defined, then this setting should be removed and replaced with [placement_group](#) (p. 104) settings for each of the [\[queue\]](#) sections (p. 103). If the same placement group is used for different instance types, it's much more likely that the request will fail

with an insufficient capacity error. For more information, see [Insufficient instance capacity](#) in the *Amazon EC2 User Guide for Linux Instances*. Multiple queues can only share a placement group if it's created in advance and configured in the [placement_group](#) (p. 104) setting for each queue. If each [\[queue\] sections](#) (p. 103) defines a [placement_group](#) (p. 104) setting, then the head node cannot be in the placement group for a queue.

Valid options are `cluster` or `compute`.

This parameter is not 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. (p. 67)

placement_group

Defines the cluster placement group. If the [queue_settings](#) (p. 84) setting is defined, then this setting should be removed and replaced by the [placement_group](#) (p. 104) settings in the [\[queue\] sections](#) (p. 103).

Valid options are:

- NONE
- 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 is not used when the scheduler is `awsbatch`.

For more information about placement groups, see [Placement groups](#) in the *Amazon EC2 User Guide for Linux Instances*.

The default value is `NONE`.

Not all instance types support cluster placement groups. For example, the default instance type of `t2.micro` does not 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 for Linux Instances*. See [Placement groups and instance launch issues](#) (p. 145) for tips when working with placement groups.

```
placement_group = NONE
```

Update policy: If this setting is changed, the update is not allowed. (p. 67)

post_install

Specifies the URL of a postinstall script that is executed after all of the `boot_as_*` scripts are run.

When using `awsbatch` as the scheduler, the postinstall script is executed only on the head node.

The parameter format can be either `http://hostname/path/to/script.sh` or `s3://bucketname/path/to/script.sh`.

The default value is NONE.

```
post_install = NONE
```

Update policy: The compute fleet must be stopped for this setting to be changed for an update. (p. 67)

post_install_args

Specifies a quoted list of arguments to pass to the postinstall script.

The default value is NONE.

```
post_install_args = "NONE"
```

Update policy: The compute fleet must be stopped for this setting to be changed for an update. (p. 67)

pre_install

Specifies the URL of a preinstall script that is executed before any of the `boot_as_*` scripts are run.

When using `awsbatch` as the scheduler, the preinstall script is executed only on the head node.

The parameter format can be either `http://hostname/path/to/script.sh` or `s3://bucketname/path/to/script.sh`.

The default value is NONE.

```
pre_install = NONE
```

Update policy: The compute fleet must be stopped for this setting to be changed for an update. (p. 67)

pre_install_args

Specifies a quoted list of arguments to pass to the preinstall script.

The default value is NONE.

```
pre_install_args = "NONE"
```

Update policy: The compute fleet must be stopped for this setting to be changed for an update. (p. 67)

proxy_server

Defines an HTTP or HTTPS proxy server, typically `http://x.x.x.x:8080`.

The default value is NONE.

```
proxy_server = NONE
```

Update policy: If this setting is changed, the update is not allowed. (p. 67)

queue_settings

Specifies that the cluster uses queues instead of a homogenous compute fleet, and which [queue] sections (p. 103) are used. The first [queue] section (p. 103) 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` (p. 84) is only supported when `scheduler` (p. 85) is set to `slurm`. The `cluster_type` (p. 74), `compute_instance_type` (p. 75), `initial_queue_size` (p. 79), `maintain_initial_size` (p. 80), `max_queue_size` (p. 81), `placement` (p. 81), `placement_group` (p. 82), and `spot_price` (p. 86) settings must not be specified. The `disable_hyperthreading` (p. 76) and `enable_efa` (p. 77) settings can either be specified in the [cluster] section (p. 71) or the [queue] sections (p. 103), but not both.

Up to five (5) [queue] sections (p. 103) are supported.

For more information, see the [queue] section (p. 103).

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` (p. 84) was added in AWS ParallelCluster version 2.9.0.

Update policy: If this setting is changed, the update is not allowed. (p. 67)

raid_settings

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 (p. 105).

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. (p. 67)

s3_read_resource

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 all objects in the `my_corporate_bucket` bucket.

See [working with Amazon S3 \(p. 37\)](#) for details on format.

The default value is NONE.

```
s3_read_resource = NONE
```

Update policy: This setting can be changed during an update. (p. 67)

s3_read_write_resource

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](#) (p. 37) for details on format.

The default value is NONE.

```
s3_read_write_resource = NONE
```

Update policy: This setting can be changed during an update. (p. 67)

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](#) (p. 107).

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. (p. 67)

scheduler

(Required) Defines the cluster scheduler.

Valid options are:

`awsbatch`

AWS Batch

`sge`

Son of Grid Engine (SGE)

`slurm`

Slurm Workload Manager (Slurm)

`torque`

Torque Resource Manager (Torque)

Warning

Starting on December 31, 2021, AWS will no longer include SGE and Torque support for all released versions of AWS ParallelCluster. Previous versions of AWS ParallelCluster that support SGE and Torque will still be available for download and use. However, these versions will not be eligible for future updates or troubleshooting support from AWS service and customer support teams. Moreover, future releases of AWS ParallelCluster made before and after December 31, 2021 will not include support for either SGE or Torque.

For more information about the `awsbatch` scheduler, see [networking setup \(p. 33\)](#).

Note

Before AWS ParallelCluster version 2.7.0, the `scheduler` parameter was optional, and the default was `sge`. 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. (p. 67)

shared_dir

Defines the path where the shared Amazon EBS volume is mounted.

Do not use this option with multiple Amazon EBS volumes. Instead, provide [shared_dir \(p. 86\)](#) values under each [\[ebs\] section \(p. 92\)](#).

See the [\[ebs\] section \(p. 92\)](#) 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. (p. 67)

spot_bid_percentage

Optionally 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. (p. 67)

spot_price

Optionally sets the maximum Spot price for the ComputeFleet on traditional schedulers (SGE, Slurm, and Torque). Used only when the [cluster_type \(p. 74\)](#) setting is set to `spot`. If you do not specify a value, you are charged the Spot price, capped at the On-Demand price. If the [queue_settings \(p. 84\)](#) setting is defined, then this setting must be removed and replaced by the [spot_price \(p. 89\)](#) settings in the [\[compute_resource\] sections \(p. 88\)](#).

If the scheduler is `awsbatch`, use [spot_bid_percentage \(p. 86\)](#) 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` (p. 86) is not 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. (p. 67)

tags

Defines tags to be used by AWS 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. Do not use quotes outside of the curly braces.

For more information, see [AWS 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. (p. 67)

template_url

Defines the path to the AWS CloudFormation template that is 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`.

```
template_url = https://us-east-1-aws-parallelcluster.s3.amazonaws.com/templates/aws-parallelcluster-2.9.1.cfn.json
```

Update policy: This setting is not analyzed during an update. (p. 67)

vpc_settings

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](#) (p. 108).

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. (p. 67)

[compute_resource] section

Topics

- [initial_count](#) (p. 88)
- [instance_type](#) (p. 88)
- [max_count](#) (p. 89)
- [min_count](#) (p. 89)
- [spot_price](#) (p. 89)

Defines configuration settings for a compute resource. [compute_resource] sections (p. 88) are referenced by the [compute_resource_settings](#) (p. 103) setting in the [queue] section (p. 103). [compute_resource] sections (p. 88) are only supported when [scheduler](#) (p. 85) 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 (p. 88) was added in AWS ParallelCluster version 2.9.0.

initial_count

(Optional) Sets the initial number of Amazon EC2 instances to launch for this compute resource. Cluster creation will not complete until at least this many nodes have been launched into the compute resource. If the [compute_type](#) (p. 104) setting for the queue is `spot` and there are not enough Spot instances available, the cluster creation may time out and fail. Any count larger than the [min_count](#) (p. 89) setting is dynamic capacity subject to the [scaledown_idletime](#) (p. 108) setting. This setting replaces the [initial_queue_size](#) (p. 79) setting.

Defaults to 0.

```
initial_count = 2
```

Update policy: The compute fleet must be stopped for this setting to be changed for an update. (p. 67)

instance_type

(Required) Defines the Amazon EC2 instance type that is used for this compute resource. The architecture of the instance type must be the same as the architecture used for the [master_instance_type](#) (p. 80) setting. The `instance_type` setting must be unique for each

[[compute_resource](#)] section (p. 88) referenced by a [[queue](#)] section (p. 103). This setting replaces the [compute_instance_type](#) (p. 75) setting.

```
instance_type = t2.micro
```

Update policy: The compute fleet must be stopped for this setting to be changed for an update. (p. 67)

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](#) (p. 88) setting is started in a power down mode. This setting replaces the [max_queue_size](#) (p. 81) setting.

Defaults to 10.

```
max_count = 10
```

Update policy: The compute fleet must be stopped for this setting to be changed for an update. (p. 67)

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 will not complete until at least this many nodes have been launched into the compute resource.

Defaults to 0.

```
min_count = 1
```

Update policy: The compute fleet must be stopped for this setting to be changed for an update. (p. 67)

spot_price

(Optional) Sets the maximum Spot price for this compute resource. Used only when the [compute_type](#) (p. 104) setting for the queue containing this compute resources is set to spot. This setting replaces the [spot_price](#) (p. 86) setting.

If you do not specify a value, you are 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. (p. 67)

[[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 \(p. 63\)](#).

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. (p. 67)

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. (p. 67)

[dcv] section

Defines configuration settings for the NICE DCV server running on the head node.

To create and configure a NICE DCV server, specify a [dcv_settings \(p. 76\)](#) with the name of your section, with [enable \(p. 91\)](#) set to master, and a [base_os \(p. 73\)](#) set 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
```

Important

NICE DCV is not supported on AWS Graviton-based instances.

Important

By default the NICE DCV port setup by AWS ParallelCluster is open to all IPv4 addresses. However, users can connect to a NICE DCV port only if they have the URL for the NICE DCV session and connect to the NICE DCV session within 30 seconds of when the URL is returned from `pcluster dcv connect`. Use the [access_from \(p. 91\)](#) setting to further restrict access to the NICE DCV port with a CIDR-formatted IP range, and use the [port \(p. 91\)](#) setting to set a nonstandard port.

Note

Support for the [\[dcv \] section \(p. 90\)](#) on `alinux2` and `ubuntu1804` was added in AWS ParallelCluster version 2.6.0. Support for the [\[dcv \] section \(p. 90\)](#) on `centos7` was added in AWS ParallelCluster version 2.5.0.

access_from

(Optional, Recommended) Specifies the CIDR-formatted IP range for connections to NICE 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. (p. 67)

enable

(Required) Indicates whether NICE DCV is enabled on the head node. To enable NICE DCV on the head node and configure the required security group rule, set the `enable` setting to `master`.

The default value is `NONE`.

The following example enables NICE DCV on the head node.

```
enable = master
```

Note

NICE DCV automatically generates a self-signed certificate that is used to secure traffic between the NICE DCV client and NICE DCV server running on the head node. To configure your own certificate, see [NICE DCV HTTPS certificate \(p. 66\)](#).

Update policy: If this setting is changed, the update is not allowed. (p. 67)

port

(Optional) Specifies the port for NICE DCV.

The default value is `8443`.

```
port = 8443
```

Update policy: If this setting is changed, the update is not allowed. (p. 67)

[ebs] section

Topics

- [shared_dir](#) (p. 92)
- [ebs_snapshot_id](#) (p. 92)
- [volume_type](#) (p. 93)
- [volume_size](#) (p. 93)
- [volume_iops](#) (p. 93)
- [encrypted](#) (p. 93)
- [ebs_kms_key_id](#) (p. 93)
- [ebs_volume_id](#) (p. 94)

Defines Amazon EBS volume configuration settings for volumes that are mounted on the head node and shared via NFS to the compute nodes.

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-xxxxxx
volume_type = io1
volume_iops = 200
...

[ebs custom2]
shared_dir = vol2
...

...
```

shared_dir

Specifies the path where the shared Amazon EBS volume is mounted.

This parameter is required when using multiple Amazon EBS volumes.

When using one (1) Amazon EBS volume, this option overwrites the [shared_dir](#) (p. 86) that is specified under the [\[cluster \]](#) section (p. 71). In the following example, the volume mounts to /vol1.

```
shared_dir = vol1
```

Update policy: If this setting is changed, the update is not allowed. (p. 67)

ebs_snapshot_id

Defines the Amazon EBS snapshot Id, if you are using a snapshot as the source for the volume.

The default value is NONE.

```
ebs_snapshot_id = snap-xxxxxx
```

Update policy: If this setting is changed, the update is not allowed. (p. 67)

volume_type

Specifies the [Amazon EBS volume type](#) of the volume that you want to launch.

Valid options are:

- gp2
- io1
- st1
- sc1

The default value is gp2.

```
volume_type = io1
```

Update policy: If this setting is changed, the update is not allowed. (p. 67)

volume_size

Specifies the size of the volume to be created, in GiB (if not using a snapshot).

The default value is 20.

```
volume_size = 20
```

Update policy: If this setting is changed, the update is not allowed. (p. 67)

volume_iops

Defines the number of IOPS for io1-type volumes.

The default value is 100.

```
volume_iops = 200
```

Update policy: This setting can be changed during an update. (p. 67)

encrypted

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. (p. 67)

ebs_kms_key_id

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](#) (p. 77).

For more information, see [Disk encryption with a custom KMS Key](#) (p. 134).

```
ebs_kms_key_id = xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

Update policy: If this setting is changed, the update is not allowed. (p. 67)

ebs_volume_id

Defines the volume Id of an existing Amazon EBS volume to attach to the head node.

The default value is `NONE`.

```
ebs_volume_id = vol-xxxxxx
```

Update policy: If this setting is changed, the update is not allowed. (p. 67)

[efs] section

Topics

- [efs_fs_id](#) (p. 94)
- [efs_kms_key_id](#) (p. 95)
- [encrypted](#) (p. 95)
- [performance_mode](#) (p. 96)
- [provisioned_throughput](#) (p. 96)
- [shared_dir](#) (p. 96)
- [throughput_mode](#) (p. 97)

Defines configuration settings for the Amazon EFS that is mounted on the head and compute nodes. For more information, see [CreateFileSystem](#) in the Amazon EFS documentation.

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
```

efs_fs_id

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](#) (p. 86).

If you set this option to `config_sanity`, it only supports file systems:

- That do not have a mount target in the stack's Availability Zone

OR

- That do have an existing mount target in the stack's Availability Zone, with inbound and outbound NFS traffic allowed from 0.0.0.0/0.

The sanity check for validating [efs_fs_id](#) (p. 94) 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 recommends that you *not* create a mount target in the stack's Availability Zone, and instead let AWS handle this step. If you must have a mount target in the stack's Availability Zone, consider using a custom security group by providing a [vpc_security_group_id](#) (p. 110) option under the `[vpc]` section (p. 108). Then add that security group to the mount target, and turn off config sanity to create the cluster.

The default value is `NONE`.

```
efs_fs_id = fs-12345
```

Update policy: If this setting is changed, the update is not allowed. (p. 67)

efs_kms_key_id

(Optional) Identifies the AWS Key Management Service (AWS KMS) customer managed key (CMK) to be used to protect the encrypted file system. If this is set, the [encrypted](#) (p. 95) setting must be set to `true`. This corresponds to the `KmsKeyId` parameter in the *Amazon EFS API Reference*.

The default value is `NONE`.

```
efs_kms_key_id = 1234abcd-12ab-34cd-56ef-1234567890ab
```

Update policy: If this setting is changed, the update is not allowed. (p. 67)

encrypted

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. (p. 67)

performance_mode

Defines the performance mode of the file system. This corresponds to the [PerformanceMode](#) parameter in the *Amazon EFS API Reference*.

Valid choices are:

- `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 is a trade-off of slightly higher latencies for most file operations.

This parameter cannot be changed after the file system has been created.

The default value is `generalPurpose`.

```
performance_mode = generalPurpose
```

Update policy: If this setting is changed, the update is not allowed. (p. 67)

provisioned_throughput

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](#) (p. 97) to `provisioned`.

The limit on throughput is 1024 MiB/s. To request a limit increase, contact AWS Support.

The minimum value is 0.0 MiB/s.

```
provisioned_throughput = 1024
```

Update policy: This setting can be changed during an update. (p. 67)

shared_dir

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](#) (p. 86) is specified.

Do not 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. (p. 67)

throughput_mode

Defines the throughput mode of the file system. This corresponds to the [ThroughputMode](#) parameter in the *Amazon EFS API Reference*.

Valid options are:

- `bursting`
- `provisioned`

The default value is `bursting`.

```
throughput_mode = provisioned
```

Update policy: This setting can be changed during an update. (p. 67)

[fsx] section

Topics

- [automatic_backup_retention_days](#) (p. 98)
- [copy_tags_to_backups](#) (p. 98)
- [daily_automatic_backup_start_time](#) (p. 99)
- [deployment_type](#) (p. 99)
- [export_path](#) (p. 100)
- [fsx_backup_id](#) (p. 100)
- [fsx_fs_id](#) (p. 100)
- [fsx_kms_key_id](#) (p. 101)
- [import_path](#) (p. 101)
- [imported_file_chunk_size](#) (p. 101)
- [per_unit_storage_throughput](#) (p. 102)
- [shared_dir](#) (p. 102)
- [storage_capacity](#) (p. 102)
- [weekly_maintenance_start_time](#) (p. 103)

Defines configuration settings for an attached Amazon FSx for Lustre file system. For more information about Amazon FSx for Lustre, see [Amazon FSx CreateFileSystem](#).

Amazon FSx for Lustre is supported if the [base_os](#) (p. 73) is `alinux`, `alinux2`, `centos7`, `ubuntu1604`, or `ubuntu1804`.

When using Amazon Linux, the kernel must be `>= 4.14.104-78.84.amzn1.x86_64`. For detailed instructions, see [Installing the lustre client](#) in the *Amazon FSx for Lustre User Guide*.

Note

Amazon FSx for Lustre is not currently supported when using `awsbatch` as a scheduler.

Note

Support for Amazon FSx for Lustre on `alinux2`, `ubuntu1604`, and `ubuntu1804` was added in AWS ParallelCluster version 2.6.0. Support for Amazon 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 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 more specific CIDR ranges, IP addresses, or security group IDs. This is done automatically when not using `vpc_security_group_id` (p. 110).

To use an existing Amazon FSx file system, specify `fsx_fs_id` (p. 100).

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
```

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` (p. 98) parameter is specified, the `export_path` (p. 100), `import_path` (p. 101), and `imported_file_chunk_size` (p. 101) 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` (p. 98) was added in AWS ParallelCluster version 2.8.0.

Update policy: This setting can be changed during an update. (p. 67)

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` (p. 98) parameter is specified, the `automatic_backup_retention_days` (p. 98) must be specified with a value greater than 0, and the `export_path` (p. 100), `import_path` (p. 101), and `imported_file_chunk_size` (p. 101) 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` (p. 98) was added in AWS ParallelCluster version 2.8.0.

Update policy: If this setting is changed, the update is not allowed. (p. 67)

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` (p. 99) parameter is specified, the `automatic_backup_retention_days` (p. 98) must be specified with a value greater than 0, and the `export_path` (p. 100), `import_path` (p. 101), and `imported_file_chunk_size` (p. 101) 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 would be:

```
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` (p. 99) was added in AWS ParallelCluster version 2.8.0.

Update policy: This setting can be changed during an update. (p. 67)

deployment_type

(Optional) Specifies the Amazon FSx for Lustre deployment type. This corresponds to the `DeploymentType` property. For more information, see [Amazon 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, and offers higher burst throughput over baseline throughput and also in-transit encryption of data.

The valid values are `SCRATCH_1`, `SCRATCH_2`, and `PERSISTENT_1`.

SCRATCH_1

The default deployment type for Amazon FSx for Lustre. With this deployment type, the `storage_capacity` (p. 102) 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 that supports up to six times the baseline throughput for spiky workloads, and supports in-transit encryption of data for supported instance types in supported regions. For more information, see [Encrypting data in transit](#) in the *Amazon FSx for Lustre User Guide*. With this deployment type, the `storage_capacity` (p. 102) 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 (AZ), and supports in-transit encryption of data for

supported instance types. With this deployment type, the [storage_capacity](#) (p. 102) 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](#) (p. 99) was added in AWS ParallelCluster version 2.6.0.

Update policy: If this setting is changed, the update is not allowed. (p. 67)

export_path

(Optional) Specifies the Amazon S3 path where the root of your file system is exported. The path **must** be in the same Amazon S3 bucket as the [import_path](#) (p. 101) parameter. When the [export_path](#) (p. 100) parameter is specified, the [automatic_backup_retention_days](#) (p. 98), [copy_tags_to_backups](#) (p. 98), [daily_automatic_backup_start_time](#) (p. 99), and [fsx_backup_id](#) (p. 100) parameters must not be specified. This corresponds to the `ExportPath` property.

The default value is `s3://import-bucket/FSxLustre[creation-timestamp]`, where `import-bucket` is the bucket provided in the [import_path](#) (p. 101) parameter.

```
export_path = s3://bucket/folder
```

Update policy: If this setting is changed, the update is not allowed. (p. 67)

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](#) (p. 100) parameter is specified, the [deployment_type](#) (p. 99), [export_path](#) (p. 100), [fsx_kms_key_id](#) (p. 101), [import_path](#) (p. 101), [imported_file_chunk_size](#) (p. 101), [storage_capacity](#) (p. 102), and [per_unit_storage_throughput](#) (p. 102) parameters must not be specified. These parameters are read from the backup. Additionally, the [export_path](#) (p. 100), [import_path](#) (p. 101), and [imported_file_chunk_size](#) (p. 101) parameters must not be specified.

This corresponds to the `BackupId` property.

```
fsx_backup_id = backup-fedcba98
```

Note

Support for [fsx_backup_id](#) (p. 100) was added in AWS ParallelCluster version 2.8.0.

Update policy: If this setting is changed, the update is not allowed. (p. 67)

fsx_fs_id

(Optional) Attaches an existing Amazon FSx for Lustre file system.

If this option is specified, only the [shared_dir](#) (p. 102) and [fsx_fs_id](#) (p. 100) settings in the `[fsx]` section (p. 97) are used and any other settings in the `[fsx]` section (p. 97) are ignored.

```
fsx_fs_id = fs-073c3803dca3e28a6
```

Update policy: If this setting is changed, the update is not allowed. (p. 67)

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](#) (p. 77). For more information, see [Disk encryption with a custom KMS Key](#) (p. 134). This corresponds to the `KmsKeyId` parameter in the *Amazon FSx API Reference*.

```
fsx_kms_key_id = xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

Note

Support for [fsx_kms_key_id](#) (p. 101) was added in AWS ParallelCluster version 2.6.0.

Update policy: If this setting is changed, the update is not allowed. (p. 67)

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](#) (p. 100). If you specify the [import_path](#) (p. 101) parameter, the [automatic_backup_retention_days](#) (p. 98), [copy_tags_to_backups](#) (p. 98), [daily_automatic_backup_start_time](#) (p. 99), and [fsx_backup_id](#) (p. 100) 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 Amazon S3 bucket](#) in the *Amazon FSx for Lustre User Guide*.

If a value is not provided, the file system is empty.

```
import_path = s3://bucket
```

Update policy: If this setting is changed, the update is not allowed. (p. 67)

imported_file_chunk_size

(Optional) Determines the stripe count and the maximum amount of data per file (in MiB) stored on a single physical disk for files that are imported from a data repository (using [import_path](#) (p. 101)). 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](#) (p. 101) parameter is specified, the [automatic_backup_retention_days](#) (p. 98), [copy_tags_to_backups](#) (p. 98), [daily_automatic_backup_start_time](#) (p. 99), and [fsx_backup_id](#) (p. 100) 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. (p. 67)

per_unit_storage_throughput

(Required for PERSISTENT_1 deployment types) For the [deployment_type](#) (p. 99) = 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](#) (p. 102) (MB/s/TiB). For a 2.4 TiB file system, provisioning 50 MB/s/TiB of [per_unit_storage_throughput](#) (p. 102) 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 are 50, 100, 200.

```
per_unit_storage_throughput = 200
```

Note

Support for [per_unit_storage_throughput](#) (p. 102) was added in AWS ParallelCluster version 2.6.0.

Update policy: If this setting is changed, the update is not allowed. (p. 67)

shared_dir

(Required) Defines the mount point for the Amazon FSx for Lustre file system on the head and compute nodes.

Do not 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. (p. 67)

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](#) (p. 99) setting.

SCRATCH_1

The possible values are 1200, 2400, and any multiple of 3600.

SCRATCH_2 and PERSISTENT_1

The possible values are 1200 and any multiple of 2400.

```
storage_capacity = 7200
```

Note

For AWS ParallelCluster version 2.5.0 and 2.5.1, [storage_capacity](#) (p. 102) supported possible values of 1200, 2400, and any multiple of 3600. For versions earlier than AWS ParallelCluster version 2.5.0, [storage_capacity](#) (p. 102) had a minimum size of 3600.

Update policy: If this setting is changed, the update is not allowed. (p. 67)

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:

```
weekly_maintenance_start_time = 1:00:00
```

Update policy: This setting can be changed during an update. (p. 67)

[queue] section

Topics

- [compute_resource_settings](#) (p. 103)
- [compute_type](#) (p. 104)
- [disable_hyperthreading](#) (p. 104)
- [enable_efa](#) (p. 104)
- [placement_group](#) (p. 104)

Defines configuration settings for a single queue. [[queue](#)] sections (p. 103) are only supported when [scheduler](#) (p. 85) 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 (p. 103) was added in AWS ParallelCluster version 2.9.0.

compute_resource_settings

Identifies the [[compute_resource](#)] sections (p. 88) 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 (p. 88) are supported per [[queue](#)] section (p. 103).

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. (p. 67)

compute_type

Defines the type of instances to launch for this queue. This setting replaces the [cluster_type](#) (p. 74) setting.

Valid options are: `ondemand`, and `spot`.

The default value is `ondemand`.

For more information about Spot Instances, see [Working with Spot Instances](#) (p. 37).

The following example uses Spot instances 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. (p. 67)

disable_hyperthreading

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 per CPU core per instance type](#) in the *Amazon EC2 User Guide for Linux Instances*. If the [disable_hyperthreading](#) (p. 76) setting in the `[cluster]` section (p. 71) 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. (p. 67)

enable_efa

If present, specifies that Elastic Fabric Adapter (EFA) is enabled for the nodes in this queue. EFA is supported by the following instance types (`c5n.18xlarge`, `c5n.metal`, `g4dn.metal`, `i3en.24xlarge`, `i3en.metal`, `m5dn.24xlarge`, `m5n.24xlarge`, `r5dn.24xlarge`, `r5n.24xlarge`, and `p3dn.24xlarge`) on these operating systems (`alinux`, `alinux2`, `centos7`, `ubuntu1604`, or `ubuntu1804`). For more information, see [Elastic Fabric Adapter](#) (p. 64). If the [enable_efa](#) (p. 77) setting in the `[cluster]` section (p. 71) is defined, then this setting cannot be defined.

```
enable_efa = true
```

Update policy: The compute fleet must be stopped for this setting to be changed for an update. (p. 67)

placement_group

(Optional) If present, defines the placement group for this queue. This setting replaces the [placement_group](#) (p. 82) setting.

Valid options are:

- `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 for Linux Instances*. If the same placement group is used for different instance types, it's much more likely that the request will fail with an insufficient capacity error. For more information, see [Insufficient instance capacity](#) in the *Amazon EC2 User Guide for Linux Instances*.

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 for Linux Instances*. See [Placement groups and instance launch issues \(p. 145\)](#) 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. (p. 67)

[raid] section

Topics

- [shared_dir](#) (p. 105)
- [raid_type](#) (p. 106)
- [num_of_raid_volumes](#) (p. 106)
- [volume_type](#) (p. 106)
- [volume_size](#) (p. 107)
- [volume_iops](#) (p. 107)
- [encrypted](#) (p. 107)
- [ebs_kms_key_id](#) (p. 107)

Defines configuration settings for a RAID array that is built from a number of identical Amazon EBS volumes. The RAID drive is mounted on the head node and is exported to compute nodes via 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
```

shared_dir

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.

Do not 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. (p. 67)

raid_type

Defines the RAID type for the RAID array.

The RAID drive is created only if this parameter is specified.

Valid options are:

- 0
- 1

For more information on RAID types, see [RAID info](#) in the *Amazon EC2 User Guide for Linux Instances*.

The following example creates a RAID 0 array:

```
raid_type = 0
```

Update policy: If this setting is changed, the update is not allowed. (p. 67)

num_of_raid_volumes

Defines the number of Amazon EBS volumes to assemble the RAID array from.

Minimum number of volumes = 2.

Maximum number of volumes = 5.

The default value is 2.

```
num_of_raid_volumes = 2
```

Update policy: If this setting is changed, the update is not allowed. (p. 67)

volume_type

Defines the type of volume to build.

Valid options are:

- gp2
- io1
- st1
- sc1

For more information, see [Amazon EBS volume types](#) in the *Amazon EC2 User Guide for Linux Instances*.

The default value is gp2.

```
volume_type = io1
```

Update policy: If this setting is changed, the update is not allowed. (p. 67)

volume_size

Defines the size of the volume to be created, in GiB.

The default value is 20.

```
volume_size = 20
```

Update policy: If this setting is changed, the update is not allowed. (p. 67)

volume_iops

Defines the number of IOPS for io1 type volumes.

The default value is 100.

```
volume_iops = 500
```

Update policy: This setting can be changed during an update. (p. 67)

encrypted

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. (p. 67)

ebs_kms_key_id

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` (p. 77).

For more information, see [Disk encryption with a custom KMS Key](#) (p. 134).

```
ebs_kms_key_id = xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

Update policy: If this setting is changed, the update is not allowed. (p. 67)

[scaling] section

Topics

- [scaledown_idletime](#) (p. 108)

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

Specifies the amount of time in minutes without a job, after which the compute node terminates.

This parameter is not 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. (p. 67)

[vpc] section

Topics

- [additional_sg](#) (p. 108)
- [compute_subnet_cidr](#) (p. 109)
- [compute_subnet_id](#) (p. 109)
- [master_subnet_id](#) (p. 109)
- [ssh_from](#) (p. 109)
- [use_public_ips](#) (p. 109)
- [vpc_id](#) (p. 110)
- [vpc_security_group_id](#) (p. 110)

Specifies Amazon VPC configuration settings.

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-xxxxxxx
```

additional_sg

Provides an additional Amazon VPC security group Id for all instances.

The default value is `NONE`.

```
additional_sg = sg-xxxxxxx
```

compute_subnet_cidr

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. (p. 67)

compute_subnet_id

Specifies the ID of an existing subnet in which to provision the compute nodes.

If not specified, `compute_subnet_id` (p. 109) uses the value of `master_subnet_id` (p. 109).

If the subnet is private, you must set up NAT for web access.

```
compute_subnet_id = subnet-xxxxxxx
```

Update policy: The compute fleet must be stopped for this setting to be changed for an update. (p. 67)

master_subnet_id

Specifies the ID of an existing subnet in which to provision the head node.

```
master_subnet_id = subnet-xxxxxxx
```

Update policy: If this setting is changed, the update is not allowed. (p. 67)

ssh_from

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. (p. 67)

use_public_ips

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](#) (p. 31).

The default value is `true`.

```
use_public_ips = true
```

Important

By default, all AWS accounts are limited to five (5) Elastic IP addresses per Region. For more information, see [Elastic IP address limit](#) in *Amazon EC2 User Guide for Linux Instances*.

Update policy: The compute fleet must be stopped for this setting to be changed for an update. (p. 67)

vpc_id

Specifies the ID of the Amazon VPC in which to provision the cluster.

```
vpc_id = vpc-xxxxxxx
```

Update policy: If this setting is changed, the update is not allowed. (p. 67)

vpc_security_group_id

Specifies the use of an existing security group for all instances.

The default value is `NONE`.

```
vpc_security_group_id = sg-xxxxxxx
```

The security group created by AWS ParallelCluster allows SSH access using port 22 from the addresses specified in the [ssh_from](#) (p. 109) setting, or all IPv4 addresses (0.0.0.0/0) if the [ssh_from](#) (p. 109) setting isn't specified. If NICE DCV is enabled, then the security group allows access to NICE DCV using port 8443 (or whatever the [port](#) (p. 91) setting specifies) from the addresses specified in the [access_from](#) (p. 91) setting, or all IPv4 addresses (0.0.0.0/0) if the [access_from](#) (p. 91) setting isn't specified.

Update policy: This setting can be changed during an update. (p. 67)

Example

The following example launches a cluster with the `awsbatch` scheduler. It is set to pick the optimal 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.

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

[aws]
aws_region_name = [your_aws_region]
```

```
[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 = alinux # optional, defaults to alinux, controls the base_os of the
  head node and the docker image for the compute fleet
key_name = [your_ec2_keypair]
vpc_settings = public

[vpc public]
master_subnet_id = [your_subnet]
vpc_id = [your_vpc]
```

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 \(p. 112\)](#)
- [AWS services used in AWS ParallelCluster \(p. 117\)](#)
- [AWS ParallelCluster Auto Scaling \(p. 121\)](#)

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 the compute node provisioning and removal by interacting with both the Auto Scaling group and the underlying job scheduler.

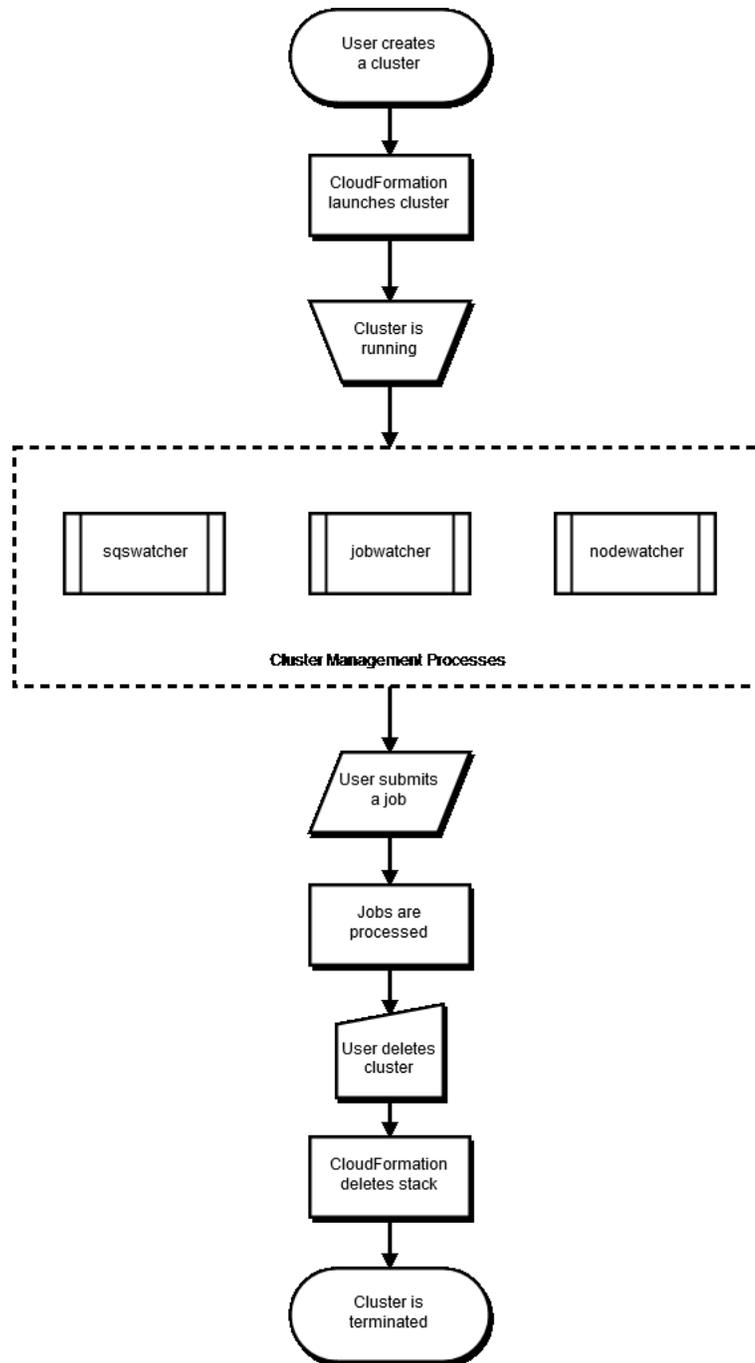
For HPC clusters that are based on AWS Batch, AWS ParallelCluster relies on the capabilities provided by the AWS Batch for the compute node management.

Topics

- [General overview \(p. 112\)](#)
- [jobwatcher \(p. 113\)](#)
- [sqswatcher \(p. 114\)](#)
- [nodewatcher \(p. 115\)](#)
- [clustermgtd \(p. 116\)](#)
- [computemgtd \(p. 117\)](#)

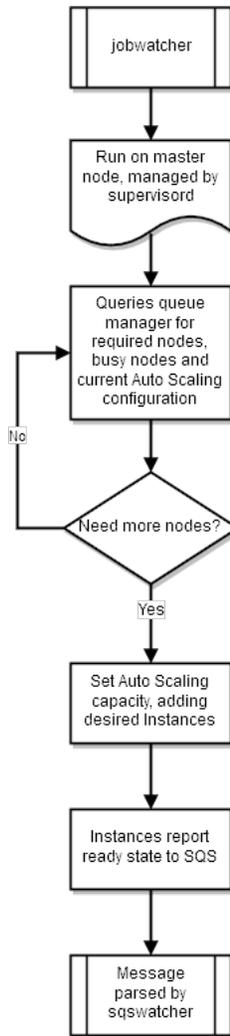
General overview

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.



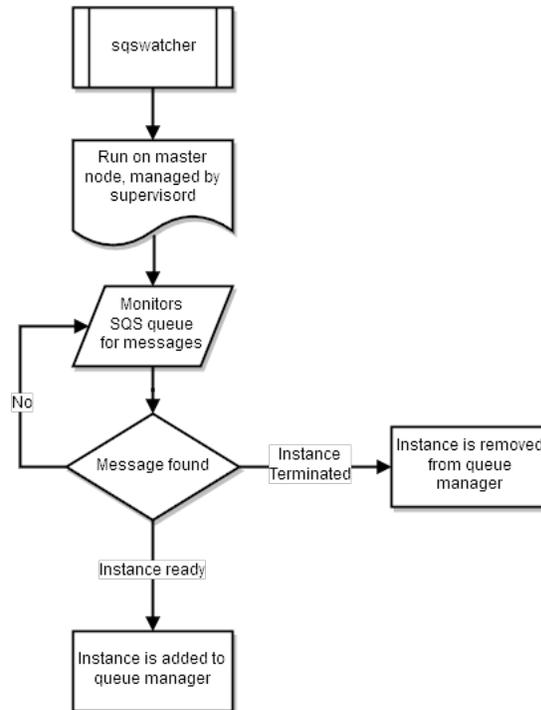
jobwatcher

When a cluster is running, a process owned by the root user monitors the configured scheduler (SGE, Slurm, or Torque) and each minute, it evaluates the queue in order to decide when to scale up.



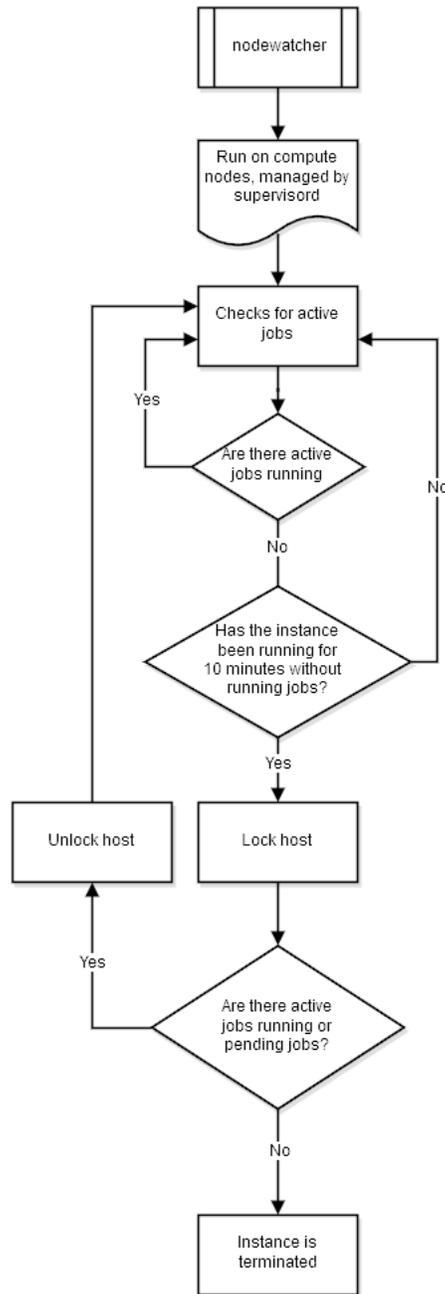
sqswatcher

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_idle_time` period, as defined by the user, the instance is terminated.



clustermgtd

Clusters that run in heterogeneous mode (indicated by specifying a [queue_settings](#) (p. 84) 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 \(p. 84\)](#) 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 in AWS ParallelCluster

The following Amazon Web Services (AWS) services are used by AWS ParallelCluster.

Topics

- [AWS Auto Scaling \(p. 117\)](#)
- [AWS Batch \(p. 118\)](#)
- [AWS CloudFormation \(p. 118\)](#)
- [Amazon CloudWatch \(p. 118\)](#)
- [Amazon CloudWatch Logs \(p. 118\)](#)
- [AWS CodeBuild \(p. 119\)](#)
- [Amazon DynamoDB \(p. 119\)](#)
- [Amazon Elastic Block Store \(p. 119\)](#)
- [Amazon Elastic Compute Cloud \(p. 119\)](#)
- [Amazon Elastic Container Registry \(p. 119\)](#)
- [Amazon EFS \(p. 119\)](#)
- [Amazon FSx for Lustre \(p. 120\)](#)
- [AWS Identity and Access Management \(p. 120\)](#)
- [AWS Lambda \(p. 120\)](#)
- [NICE DCV \(p. 120\)](#)
- [Amazon Route 53 \(p. 120\)](#)
- [Amazon Simple Notification Service \(p. 121\)](#)
- [Amazon Simple Queue Service \(p. 121\)](#)
- [Amazon Simple Storage Service \(p. 121\)](#)
- [Amazon VPC \(p. 121\)](#)

AWS Auto Scaling

AWS Auto Scaling is a service that monitors your applications and automatically adjusts the capacity of your applications based on your specific and changing service requirements. This service manages

your ComputeFleet instances as an Auto Scaling group, which 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/>.

AWS CloudFormation

AWS CloudFormation is an infrastructure-as-code service that provides a common language for you 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 AWS CloudFormation template. In most cases, AWS ParallelCluster CLI commands directly correspond to AWS CloudFormation stack commands, such as create, update, and delete. Instances that are launched within a cluster make HTTPS calls to the AWS CloudFormation endpoint in the Region where the cluster is launched.

For more information about AWS 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 in particular to monitor and log Docker image build steps as well as the output of the AWS Batch jobs.

CloudWatch is 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 \(p. 63\)](#).

AWS CodeBuild

AWS CodeBuild (CodeBuild) is an AWS managed continuous integration service that compiles 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 minimal state 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, and 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, it stores the Docker images 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](#) (p. 77) setting is specified and refers to an [\[efs\] section](#) (p. 94). 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

Amazon FSx for Lustre provides a high-performance file system that uses the open-source Lustre file system. Amazon FSx for Lustre is used when the [fsx_settings](#) (p. 79) setting is specified and refers to an [\[fsx\] section](#) (p. 97). Support for Amazon FSx for Lustre was added in AWS ParallelCluster version 2.2.1.

For more information about Amazon 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 and 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.

Lambda is used only with AWS Batch clusters.

For more information about Lambda, see <https://aws.amazon.com/lambda/> and <https://docs.aws.amazon.com/lambda/>.

NICE DCV

NICE 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. NICE DCV is used when the [dcv_settings](#) (p. 76) setting is specified and refers to an [\[dcv\] section](#) (p. 90). Support for NICE DCV was added in AWS ParallelCluster version 2.5.0.

For more information about NICE 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

Amazon Simple Notification Service (Amazon SNS) receives notifications from Auto Scaling. These events are called life cycle 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

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 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 from creating a Docker image and scripts 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 (p. 108).

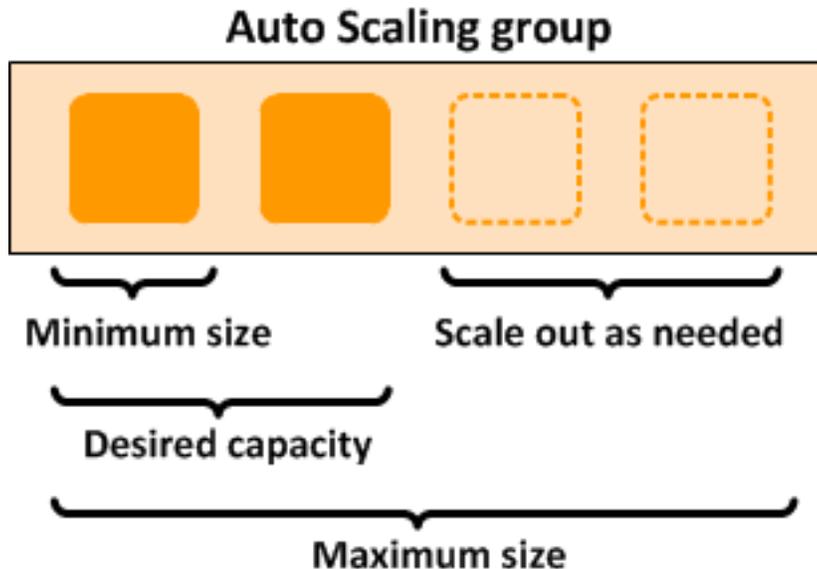
For more information about Amazon VPC, see <https://aws.amazon.com/vpc/> and <https://docs.aws.amazon.com/vpc/>.

AWS ParallelCluster Auto Scaling

The auto scaling strategy described here applies to HPC clusters that are deployed with one of the supported traditional job schedulers, either Son of Grid Engine (SGE), Slurm Workload Manager (Slurm), 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](#) (p. 79) specifies the minimum size value of the ComputeFleet Auto Scaling group, and also the desired capacity value. Setting the [max_queue_size](#) (p. 81) 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](#) (p. 81).

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 is specified in the cluster configuration.

With Slurm and Torque schedulers, each job might require both a number of nodes and a number of slots per 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 does not consider partially loaded busy nodes. i.e. A node that is 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` (p. 108) (the default setting is 10 minutes)
- There are no pending jobs in the cluster

To terminate an instance, `nodewatcher` calls the `TerminateInstanceInAutoScalingGroup` API call, 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` (p. 79) and `max_queue_size` (p. 81) parameters to the exact size of cluster that is required, and then you set the `maintain_initial_size` (p. 80) 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 \(p. 124\)](#)
- [Building a Custom AWS ParallelCluster AMI \(p. 126\)](#)
- [Running an MPI job with AWS ParallelCluster and awsbatch scheduler \(p. 128\)](#)
- [Disk encryption with a custom KMS Key \(p. 134\)](#)

Running your first job on AWS ParallelCluster

This tutorial walks you through running your first Hello World job on AWS ParallelCluster.

If you haven't yet completed installation of AWS ParallelCluster, and configured your CLI, follow the instructions in the [getting started \(p. 2\)](#) guide before continuing with this tutorial.

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  SWAPTO
SWAPUS
-----
global                  -              -    -    -     -     -     -       -       -
-
ip-192-168-1-125        lx-amd64       2    1    2     2     0.15  3.7G   130.8M  1024.0M
0.0
ip-192-168-1-126        lx-amd64       2    1    2     2     0.15  3.7G   130.8M  1024.0M
0.0
```

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

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.

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 \(p. 35\)](#) 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 is not 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 \(p. 79\)](#) setting. For example:

```
extra_json = { "cluster" : { "skip_install_recipes" : "no" } }
```

How to Customize the AWS ParallelCluster AMI

There are three ways to use a custom AWS ParallelCluster AMI. Two of these three methods require you to build a new AMI that is available under your AWS account. The third method (Use a Custom AMI at Runtime) doesn't require that you build anything in advance. Choose the method that best fits your needs.

Customization Methods

- [Modify an AWS ParallelCluster AMI \(p. 127\)](#)
- [Build a Custom AWS ParallelCluster AMI \(p. 127\)](#)
- [Use a Custom AMI at Runtime \(p. 128\)](#)

Modify an AWS ParallelCluster 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 is installed and configured. You can start with this as the base.

1. In the AMI list, find the AMI that corresponds to the specific 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 2.9.1, go to <https://github.com/aws/aws-parallelcluster/blob/v2.9.1/amis.txt>. To select another version, use the same link, choose the **Tag: 2.9.1** button, select the **Tags** tab, and then select the appropriate version.
2. Within the Amazon EC2 console, choose **Launch Instance**.
3. Navigate to **Community AMIs**, and enter the AMI id for your Region into the search box.
4. Select the AMI, choose your instance type and properties, and launch your instance.
5. Log into your instance using the OS user and your SSH key.
6. Customize your instance as required.
7. Run the following command to prepare your instance for AMI creation:

```
sudo /usr/local/sbin/ami_cleanup.sh
```

8. Stop the instance.
9. Create a new AMI from the instance.
10. Enter the AMI id in the [custom_ami](#) (p. 75) field within 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. 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 OS AMI>
```

Important

You shouldn't use an AWS ParallelCluster AMI as `<BASE AMI>` for the `createami` command. Otherwise, the command fails.

For other parameters, consult the command help:

```
pcluster createami -h
```

4. The command in Step 4 executes 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` (p. 75) field within your cluster configuration.

Note

The instance type used to build a custom AWS ParallelCluster AMI is `t2.xlarge`. This instance type does not 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

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 is 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` (p. 75) 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.

If you haven't yet installed AWS ParallelCluster and configured your CLI, follow the instructions in the [getting started](#) (p. 2) guide before continuing with this tutorial. Also, make sure to read through the [awsbatch networking setup](#) (p. 33) documentation before moving to the next step.

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 Master 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 \(p. 57\)](#) 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` ([p. 81](#)) 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-edbfeccab5d2 (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-edbfeccab5d2  hello        RUNNING    2018-11-12 09:41:29  -
-
```

The output provides detailed information for the job.

```
$ awsbstat 6efe6c7c-4943-4c1a-baf5-edbfeccab5d2
jobId                : 6efe6c7c-4943-4c1a-baf5-edbfeccab5d2
jobName              : hello
createdAt            : 2018-11-12 09:41:21
startedAt            : 2018-11-12 09:41:29
stoppedAt            : -
status               : RUNNING
statusReason         : -
jobDefinition        : parallelcluster-myBatch:1
jobQueue             : parallelcluster-myBatch
command              : /bin/bash -c 'aws s3 --region us-east-1 cp s3://parallelcluster-
mybatch-lui1ftboklhpns95/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-myBatch/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-myBatch/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 `awsbstat` again.

```
$ awsbstat
```

jobId	exitCode	jobName	status	startedAt	stoppedAt
-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----

Now you can see that the job is in the SUCCEEDED status.

```
$ awsbstat -s SUCCEEDED
jobId                jobName    status    startedAt
stoppedAt            exitCode
-----
6efe6c7c-4943-4c1a-baf5-edbfeccab5d2  hello      SUCCEEDED 2018-11-12 09:41:29
2018-11-12 09:42:00      0
```

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://parallelcluster-mybatch-lu1lftboklhpn95/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 \(p. 57\)](#). 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 $HOSTNAME! I let the main node orchestrate the mpi
execution!"
    # Since mpi orchestration happens on the main node, we need to make sure the containers
representing the compute
    # nodes are not terminated. A simple trick is to wait for a file containing the status
code to be created.
    # All compute nodes are terminated by Batch if the main node exits abruptly.
    while [ ! -f "${_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 awsbatch -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 ~]$ awsbatch -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://parallelcluster-awsbatch-tutorial-iwyl4458saiwgwvg/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 ~]$ awsbatch -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://parallelcluster-awsbatch-tutorial-iwyl4458saiwgwvg/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 ~]$ awsbatch -s ALL
```

jobId	stoppedAt	exitCode	jobName	status	startedAt
5b4d50f8-1060-4ebf-ba2d-1ae868bbd92d	2018-11-27 15:51:26	-	submit_mpi_sh	SUCCEEDED	2018-11-27 15:50:10

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 Amazon 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`.

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 \(p. 39\)](#). 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-xxxxxxxxxxxx
```

The following is an example with the Amazon 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-xxxxxxxxxxxx
```

Similar configurations apply to Amazon EBS and Amazon FSx based file systems.

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 \(p. 136\)](#)
- [Data protection in AWS ParallelCluster \(p. 137\)](#)
- [Identity and Access Management for AWS ParallelCluster \(p. 138\)](#)
- [Compliance validation for AWS ParallelCluster \(p. 139\)](#)
- [Enforcing a Minimum Version of TLS 1.2 \(p. 139\)](#)

Security information for services used by AWS ParallelCluster

- [Security in Amazon EC2](#)
- [Security in AWS Batch](#)
- [Security in AWS 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 Amazon FSx for Lustre](#)
- [Security in AWS Identity and Access Management \(IAM\)](#)
- [Security in AWS Lambda](#)

- [Security in Amazon SNS](#)
- [Security in Amazon SQS](#)
- [Security in Amazon S3](#)
- [Security in Amazon VPC](#)

Data protection in AWS ParallelCluster

AWS ParallelCluster conforms to the AWS [shared responsibility model](#), which includes regulations and guidelines for data protection. AWS is responsible for protecting the global infrastructure that runs all the AWS services. AWS maintains control over data hosted on this infrastructure, including the security configuration controls for handling customer content and personal data. AWS customers and APN partners, acting either as data controllers or data processors, are responsible for any personal data that they put in the AWS Cloud.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS Identity and Access Management (IAM), so that 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.
- Set up API and user activity logging with AWS CloudTrail.
- 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 personal data that is stored in Amazon S3.

We strongly recommend that you never put sensitive identifying information, such as your customers' account numbers, into free-form fields such as a **Name** field. This includes when you work with AWS ParallelCluster or other AWS services using the console, API, or AWS SDKs. Any data that you enter into AWS ParallelCluster or other services might get picked up for inclusion in diagnostic logs. When you provide a URL to an external server, don't include credentials information in the URL to validate your request to that server.

For more information about data protection, see the [AWS shared responsibility model and GDPR](#) blog post on the *AWS Security Blog*.

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 [encrypted](#) (p. 93) and [ebs_kms_key_id](#) (p. 93) settings in the [\[ebs \] section](#) (p. 92). For more information, see [Amazon EBS encryption](#) in the Amazon EC2 User Guide for Linux Instances. For Amazon EFS volumes, encryption is configured using the [encrypted](#) (p. 95) and [efs_kms_key_id](#) (p. 95) settings in the [\[efs \] section](#) (p. 94). For more information, see [How encryption at rest works](#) in the *Amazon Elastic File System User Guide*. For Amazon 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 for Linux Instances*.

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 & 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 for Linux Instances*.

See also

- [Data protection in Amazon EC2](#)
- [Data protection in AWS CloudFormation](#)
- [Data protection in Amazon S3](#)
- [Data protection in Amazon 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 \(p. 39\)](#).

The only major difference is how you authenticate when using a standard IAM user and long-term credentials. Although an IAM user requires a password to access an AWS service's console, that same IAM 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 `$HOME/.aws/credentials` file stores long-term credentials required to access your AWS resources. These include your access key ID and secret access key.
- Short-term credentials, such as those for roles that you assume, or that are for AWS Single Sign-On services, are also stored in the `$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 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](#) – 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 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"
```

```
OPENSSSL_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- $\$$ OPENSSSL_VERSION.tar.gz"
tar -xzf "openssl- $\$$ OPENSSSL_VERSION.tar.gz"
cd openssl- $\$$ OPENSSSL_VERSION
./config --prefix= $\$$ OPENSSSL_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= $\$$ OPENSSSL_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 \(p. 139\)](#) using the newly installed Python version (that is, `/opt/python-with-min-tls1_2/bin/python3`).

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 \(p. 35\)](#) 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 \(p. 142\)](#)
- [Setting up a custom AWS ParallelCluster node package \(p. 143\)](#)

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 \(p. 35\)](#) 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 (p. 71).

```
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` (p. 79) 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](#) (p. 35) 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 (p. 71).

```
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` (p. 79) setting.

AWS ParallelCluster troubleshooting

The AWS ParallelCluster community maintains a Wiki with many troubleshooting tips at the [aws-parallelcluster wiki](#).

Failure submitting AWS Batch multi-node parallel jobs

If you have problems submitting multi-node parallel jobs when using AWS Batch as the job scheduler, we recommend that you upgrade to AWS ParallelCluster version 2.5.0. If that is not feasible, you can use a workaround. For information about this workaround, see [Self patch a cluster used for submitting multi node parallel jobs through AWS Batch](#).

Placement groups and instance launch issues

To get the lowest inter-node latency, we recommend that you use a *placement group*. A placement group guarantees that your instances are on the same networking backbone. If not enough instances are available when the request is made, an `InsufficientInstanceCapacity` error is returned. To reduce the possibility of receiving an `InsufficientInstanceCapacity` error when using cluster placement groups, set the `placement_group` (p. 82) parameter to `DYNAMIC` and set the `placement` (p. 81) parameter to `compute`.

If you need a high performance shared filesystem, consider using [Amazon FSx for Lustre](#).

If the head node must be in the placement group, use the same instance type and subnet for the head and all compute nodes. As such, the `compute_instance_type` (p. 75) parameter has the same value as the `master_instance_type` (p. 80) parameter, the `placement` (p. 81) parameter is set to `cluster`, and the `compute_subnet_id` (p. 109) parameter is not specified. With this configuration, the value of the `master_subnet_id` (p. 109) 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 for Linux Instances*

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`

This includes Son of Grid Engine and related files. (Conditional, only if `scheduler` (p. 85) = `sge`.)

`/opt/slurm`

This includes Slurm Workload Manager and related files. (Conditional, only if [scheduler \(p. 85\)](#) = `slurm`.)

`/opt/torque`

This includes Torque Resource Manager and related files. (Conditional, only if [scheduler \(p. 85\)](#) = `torque`.)

NICE DCV troubleshooting

The logs for NICE DCV are written to files in the `/var/log/dcv/` directory. Reviewing these logs can help to troubleshoot problems.

The instance type should have at least 1.7 GiB of RAM to run NICE DCV. Nano and micro instance types don't have enough memory to run NICE DCV.

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
AWS ParallelCluster version 2.9.0 released.	<p>AWS ParallelCluster version 2.9.0 released. Changes include:</p> <ul style="list-style-type: none"> • 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 (p. 62). • Added support for NICE DCV (p. 65) on 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 filesystems shared from the head node. • Removed dependency on cfn-init when bootstrapping compute nodes to avoid throttling by AWS CloudFormation when a large number of nodes join the cluster. • Elastic Fabric Adapter (p. 64) installer updated to 1.9.5: <ul style="list-style-type: none"> • EFA profile: <code>efa-profile-1.0.0</code> (new) • Kernel module: <code>efa-1.6.0</code> (no change) • RDMA core: <code>rdma-core-28.amzn0</code> (no change) • Libfabric: <code>libfabric-1.10.1amzn1.1</code> (no change) • Open MPI: <code>openmpi40-aws-4.0.3</code> (no change) • Upgraded Slurm to version 20.02.4 (from 19.05.5) • NICE DCV (p. 65) updated to NICE DCV 2020.1-9012. For more information, see DCV 2020.1-9012— August 24, 2020 Release Notes in the <i>NICE DCV Administrator Guide</i>. • When mounting shared NFS drives, use the head node private IP address instead of the hostname. • Added new log streams to CloudWatch Logs: <code>chef-client</code>, <code>clustermgtd</code>, <code>computemgtd</code>, <code>slurm_resume</code>, and <code>slurm_suspend</code>. • Added support for queue names in pre-install and post-install scripts. • In the AWS GovCloud (US) Regions, use the Amazon DynamoDB on-demand billing option. For more 	11 September 2020

Change	Description	Date
	<p>information, see On-Demand Mode in the <i>Amazon DynamoDB Developer Guide</i></p> <p>For more details of the changes, see the CHANGELOG files for the aws-parallelcluster, aws-parallelcluster-cookbook, and aws-parallelcluster-node packages on GitHub.</p>	
AWS ParallelCluster version 2.8.1 released.	<p>AWS ParallelCluster version 2.8.1 released. Changes include:</p> <ul style="list-style-type: none">• Disable screen lock for NICE DCV sessions to prevent users from being locked out.• Fix pcluster configure (p. 20) when including an AWS Graviton-based instance type. <p>For more details of the changes, see the CHANGELOG files for the aws-parallelcluster, aws-parallelcluster-cookbook, and aws-parallelcluster-node packages on GitHub.</p>	4 August 2020

Change	Description	Date
AWS ParallelCluster version 2.8.0 released.	<p data-bbox="659 275 1230 327">AWS ParallelCluster version 2.8.0 released. Changes include:</p> <ul data-bbox="659 352 1300 1188" style="list-style-type: none"> <li data-bbox="659 352 1284 415">• Added support for AWS Graviton-based instances (like the A1 and C6g). <li data-bbox="659 422 1300 594">• Added support for the automatic daily backup features of Amazon FSx for Lustre. For more information, see automatic_backup_retention_days (p. 98), copy_tags_to_backups (p. 98), daily_automatic_backup_start_time (p. 99), and fsx_backup_id (p. 100). <li data-bbox="659 600 1256 663">• Removed dependency on Berkshelf from pcluster createami (p. 22). <li data-bbox="659 669 1284 758">• Improved the robustness and user experience of pcluster update (p. 29). For more information, see Using pcluster update (p. 66). <li data-bbox="659 764 1295 1188">• Elastic Fabric Adapter (p. 64) installer updated to 1.9.4: <ul data-bbox="683 800 1276 1031" style="list-style-type: none"> <li data-bbox="683 800 1182 863">• Kernel module: <code>efa-1.6.0</code> (updated from <code>efa-1.5.1</code>) <li data-bbox="683 869 1276 932">• RDMA core: <code>rdma-core-28.amzn0</code> (updated from <code>rdma-core-25.0</code>) <li data-bbox="683 938 1263 1001">• Libfabric: <code>libfabric-1.10.1amzn1.1</code> (updated from <code>libfabric-aws-1.9.0amzn1.1</code>) <li data-bbox="683 1008 1247 1031">• Open MPI: <code>openmpi40-aws-4.0.3</code> (no change) <li data-bbox="659 1037 1230 1125">• Upgrade NVIDIA driver to Tesla version 440.95.01 on CentOS 6 and version 450.51.05 on all other distributions. <li data-bbox="659 1131 1170 1188">• Upgrade CUDA library to version 11.0 on all distributions other than CentOS 6. <p data-bbox="659 1236 1252 1350">For more details of the changes, see the CHANGELOG files for the aws-parallelcluster, aws-parallelcluster-cookbook, and aws-parallelcluster-node packages on GitHub.</p>	23 July 2020

Change	Description	Date
AWS ParallelCluster version 2.7.0 released.	<p>AWS ParallelCluster version 2.7.0 released. Changes include:</p> <ul style="list-style-type: none"> • base_os (p. 73) is now a required parameter. • scheduler (p. 85) is now a required parameter. • NICE DCV (p. 65) updated to NICE DCV 2020.0. For more information, see NICE DCV releases version 2020.0 with surround sound 7.1 and stylus support. <p>Intel MPI (p. 64) updated to Version 2019 Update 7 (updated from Version 2019 Update 6). For more information, see Intel® MPI Library 2019 Update 7.</p> <p>Elastic Fabric Adapter (p. 64) installer updated to 1.8.4:</p> <ul style="list-style-type: none"> • Kernel module: <code>efa-1.5.1</code> (no change) • RDMA core: <code>rdma-core-25.0</code> (no change) • Libfabric: <code>libfabric-aws-1.9.0amzn1.1</code> (no change) • Open MPI: <code>openmpi40-aws-4.0.3</code> (updated from <code>openmpi40-aws-4.0.2</code>) • Upgrade CentOS 7 AMI to version 7.8-2003 (updated from 7.7-1908). For more information, see CentOS-7 (2003) Release Notes	19 May 2020
AWS ParallelCluster version 2.6.1 released.	<p>AWS ParallelCluster version 2.6.1 released. Changes include:</p> <ul style="list-style-type: none"> • Removed <code>cfn-init-cmd</code> and <code>cfn-wire</code> from logs stored in Amazon CloudWatch Logs. For more information, see Integration with Amazon CloudWatch Logs (p. 63). 	17 April 2020
AWS ParallelCluster version 2.6.0 released.	<p>AWS ParallelCluster version 2.6.0 released. New features include:</p> <ul style="list-style-type: none"> • 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 (p. 63). • Added support for new Amazon FSx for Lustre deployment types <code>SCRATCH_2</code> and <code>PERSISTENT_1</code>. Support for Amazon FSx for Lustre on Ubuntu 18.04 and Ubuntu 16.04. For more information, see fsx (p. 97). • Added support for NICE DCV on Ubuntu 18.04. For more information, see Connect to the head node through NICE DCV (p. 65). 	27 February 2020
AWS ParallelCluster version 2.5.1 released.	<p>AWS ParallelCluster version 2.5.1 updates several drivers and fixes some issues. For more details, see AWS ParallelCluster version 2.5.1.</p>	13 December 2019

Change	Description	Date
AWS ParallelCluster version 2.5.0 released.	AWS ParallelCluster version 2.5.0 introduces support for Ubuntu 18.04, scheduling with GPU options in Slurm, and NICE DCV on CentOS 7. For more details on the other changes made for AWS ParallelCluster version 2.5.0, see AWS ParallelCluster version 2.5.0 .	18 November 2019
AWS ParallelCluster introduces support for Intel MPI.	AWS ParallelCluster version 2.4.1 introduces support for Intel MPI. For more information, see Enable Intel MPI (p. 64) . For more details on the other changes made for AWS ParallelCluster version 2.4.1, see AWS ParallelCluster version 2.4.1 .	29 July 2019
AWS ParallelCluster introduces support for EFA.	AWS ParallelCluster version 2.4.0 introduces support for EFA. For more information, see Elastic Fabric Adapter (p. 64) . For more details on the other changes made for AWS ParallelCluster version 2.4.0, see AWS ParallelCluster version 2.4.0 .	11 June 2019
AWS ParallelCluster documentation initial release on AWS Documentation website.	The AWS ParallelCluster documentation is now available in 10 languages and in both HTML and PDF formats.	11 June 2019