Deploying an Elastic HPC Cluster

Build an elastic HPC cluster in minutes using AWS ParallelCluster

June 2019

Notices

Contents

Abstract

AWS ParallelCluster is an AWS-supported open source cluster management tool built on the open source CfnCluster project that helps you to deploy and manage High Performance Computing (HPC) clusters in the AWS cloud. This getting started tutorial guides you through the setup, installation, configuration, and deployment steps for AWS ParallelCluster, as well as powering down the cluster when no longer needed.

Introduction

Imagine a high performance computing (HPC) cluster that delivers the capabilities of a supercomputer and can be deployed in less than 15 minutes. AWS ParallelCluster is an open source tool published by AWS that provides a fully-elastic HPC cluster in the cloud. AWS ParallelCluster constructs an HPC environment resembling conventional HPC clusters but with the added benefit of being scalable. Jobs are submitted to a queue. Nodes spin up as needed. Jobs are automatically launched. As nodes become idle, they are automatically shut down. Once created, the cloud-based master node provides access to standard HPC tools such as schedulers, shared storage, and a Message Passing Interface (MPI) environment. The master node maintains the scheduler and the running environment while the compute nodes spin up and down as jobs are submitted to the queue.

This tutorial provides the steps necessary to install and deploy AWS ParallelCluster, configure the environment, run a simple parallel *mpi_hello_world* job, and shut down the cluster. This tutorial can be completed in less than one hour and, if the defaults are used when running the tutorial, is expected to cost less than one dollar.

AWS ParallelCluster is most often installed on a local computer running OS X or Linux. From the local computer, AWS ParallelCluster launches an HPC cluster comprised of a master instance and on-demand compute nodes running on Amazon Elastic Compute Cloud (Amazon EC2). If a local Mac or Linux computer is not available, then AWS ParallelCluster can be run from an EC2 instance running Linux. Refer to AWS ParallelCluster for more information.

Step 1: Set Up Prerequisites for AWS ParallelCluster

To prepare for this exercise you will need an AWS account, an Amazon EC2 key pair, and have the AWS Command Line Interface (CLI) installed and configured. Additionally, you will need to have Python and pip installed on the launch computer. If you meet these prerequisites you can advance to Step 2 Install AWS ParallelCluster.

Sign into AWS

If you already have an AWS account, sign in and go to the next prerequisite topic **Create and Amazon EC2 Key Pair**. Otherwise, follow these steps to create a new AWS account:

1. Access https://aws.amazon.com/.
2. Choose **Create an AWS Account**.
3. Follow the instructions.

Create an Amazon EC2 Key Pair

You must have an Amazon EC2 key pair to connect to the nodes in your cluster over a secure channel using the Secure Shell (SSH) protocol. If you already have a key pair that you want to use, you can skip this step. If you don't have a key pair, follow these steps to create a key pair.

Set Up the AWS CLI

Installation procedures will vary depending on your operating system and environment. Options include an MSI installer, a bundled installer, and pip. Step through each of the following links to install and configure the AWS CLI:

1. Installing the AWS CLI
2. Configuring the AWS CLI
3. Using the AWS CLI

NOTE: The AWS CLI makes API calls to services over HTTPS. Outbound connections on TCP port 443 must be enabled in order to perform calls.

Install Python & pip on the Launch Computer

AWS ParallelCluster is written in Python and is installed with pip (the Python installation package). Pip is installed by default when using the following versions of Python:

- Python version 2 that is equal to or above v2.7.9
- Python version 3 that is equal to or above v3.4.

If Python is not already installed on the launch computer, go to the [Python Software Foundation]() site for instructions and installation package.

To check for Python (and the version) on your computer, enter the following in the command line prompt:

```
$ python --version
```

To check for pip (and the version), enter the following in the command line prompt:

```
$ pip --version
```

Step 2: Install AWS ParallelCluster

To install AWS ParallelCluster in the launch computer, enter the following in the command line prompt:

```
$ sudo pip install --upgrade aws-parallelcluster
```

Step 3: Configure AWS Credentials

To set up proper permissions, install the AWS CLI.

> **NOTE:** You do not need to install AWS CLI if you're running on an Amazon EC2 instance with an associated AWS Identity and Access Management (IAM) role.

```
$ sudo pip install awscli
```

Next, enter `aws configure` to set up your IAM credentials (as shown below along with the expected output). Refer to [AWS CLI]() documentation for help if necessary.

```
$ aws configure
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]:
wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Default region name [None]: us-west-2
Default output format [None]:
```

You can test whether the IAM credentials were installed correctly by entering the following command:

```
$ aws s3 ls
```

This command lists S3 buckets in your account. As long as it doesn't error, it's properly configured.

Step 4: Configure and Launch AWS ParallelCluster

The following steps are used to create, configure, and launch your AWS ParallelCluster.

Create the Base AWS ParallelCluster Configure File

AWS ParallelCluster customization is specified with the *ParallelCluster* configuration file (config file). The config file is a simple text file with keyword entries. The config file is created with a pcluster configuration tool which creates a baseline config file for the user.

1. To begin, enter the following from the command line prompt:

   ```
   $ pcluster configure
   ```

2. Accept the cluster template default option for the first entry by pressing **Enter**.
3. Enter the desired AWS Region identifier (ID) from the following list providing the acceptable values (you can copy the AWS region ID and paste into your command line prompt).

   NOTE: This prompt cannot be left blank.

   Acceptable values for AWS Region ID:

- us-east-1
- us-west-1
- us-west-2
- cn-north-1
- ap-northeast-1
- ap-southeast-2
- sa-east-1
- ap-southeast-1
- ap-northeast-2
- us-west-2
- us-gov-west-1
- ap-south-1
- eu-central-1
- eu-west-1

4. At the `VPC Name [public]` prompt, press **Enter** to accept the default option.
5. At the `SSH Key Name` prompt, copy and paste the name of the Amazon EC2 keypair created earlier. Do **NOT** include the `.pem` suffix.
6. Select a `VPC ID` from the list that is displayed.

   Acceptable VPC IDs are listed for use. Copy and paste the desired VPC ID.
7. Select a `VPC Subnet ID` from the list that is displayed.

   Acceptable Subnet IDs are listed for use. Copy and paste the desired VPC Subnet ID.
8. Optional: Enter the following command to review the new config file and verify that the information is correct:

```
$ cat ~/.parallelcluster/config
```

The following is an example config file:

```
[aws]
aws_region_name = us-west-2

[cluster default]
vpc_settings = public
key_name = mysshkey

[vpc public]
master_subnet_id = subnet-0b14ad52
vpc_id = vpc-442b5d21

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

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

NOTE: Information on configuration settings are available in the AWS ParallelCluster documentation.

Customize the AWS ParallelCluster Config File

1. Customize the AWS ParallelCluster config file:
2. Use your favorite text editor to open the config file.
3. Scroll through the file and locate the cluster default section, specifically, the `key_name= mysshkey` entry. Add the following entries after it:

```
initial_queue_size = 3
max_queue_size = 3
maintain_initial_size = 3
```

NOTE: These values will result in a cluster with three compute nodes on launch and a maximum of three nodes in the cluster.

A wide range of capabilities can be configured by modifying the AWS ParallelCluster config file. For example, the `placement_group` keyword is used to create an Amazon

EC2 placement group which can result in increased performance for applications which require the lowest latency possible. For the example in this tutorial, a placement group is not required. Refer to the AWS ParallelCluster User Guide for additional information.

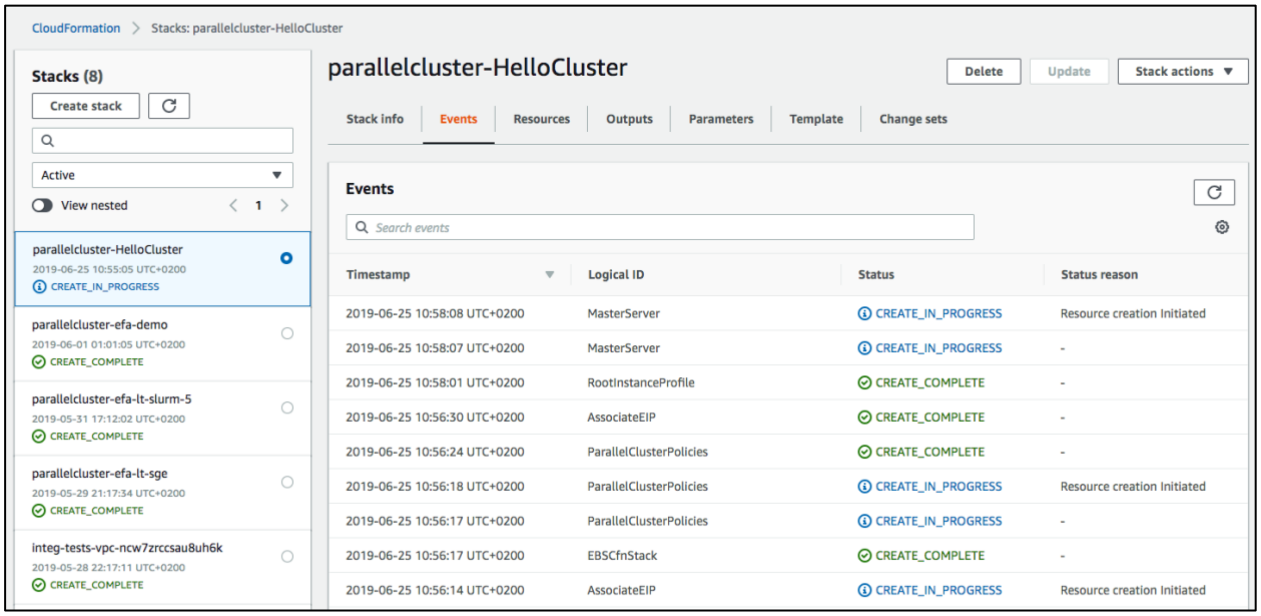
Launch the AWS ParallelCluster

To launch your cluster, enter the following at the command line prompt:

```
$ pcluster create HelloCluster
```

NOTE: In this example, the cluster name is *HelloCluster*.

Full deployment of *HelloCluster* will take about 20 minutes depending on the instance types chosen and the size of the Amazon Elastic Block Store (Amazon EBS) volumes selected. Deployment progress can be monitored from the AWS CloudFormation console events tab, as shown in Figure 1.



parallelcluster-HelloCluster
2019-06-25 10:55:05 UTC+0200
CREATE_IN_PROGRESS

parallelcluster-efa-demo
2019-06-01 01:01:05 UTC+0200
CREATE_COMPLETE

parallelcluster-efa-lt-slurm-5
2019-05-31 17:12:02 UTC+0200
CREATE_COMPLETE

parallelcluster-efa-lt-sge
2019-05-29 21:17:34 UTC+0200
CREATE_COMPLETE

integ-tests-vpc-ncw7zrccsau8uh6k
2019-05-28 22:17:11 UTC+0200
CREATE_COMPLETE

parallelcluster-HelloCluster

Events

Timestamp	Logical ID	Status	Status reason
2019-06-25 10:58:08 UTC+0200	MasterServer	CREATE_IN_PROGRESS	Resource creation Initiated
2019-06-25 10:58:07 UTC+0200	MasterServer	CREATE_IN_PROGRESS	-
2019-06-25 10:58:01 UTC+0200	RootInstanceProfile	CREATE_COMPLETE	-
2019-06-25 10:56:30 UTC+0200	AssociateEIP	CREATE_COMPLETE	-
2019-06-25 10:56:24 UTC+0200	ParallelClusterPolicies	CREATE_COMPLETE	-
2019-06-25 10:56:18 UTC+0200	ParallelClusterPolicies	CREATE_IN_PROGRESS	Resource creation Initiated
2019-06-25 10:56:17 UTC+0200	ParallelClusterPolicies	CREATE_IN_PROGRESS	-
2019-06-25 10:56:17 UTC+0200	EBSCfnStack	CREATE_COMPLETE	-
2019-06-25 10:56:14 UTC+0200	AssociateEIP	CREATE_IN_PROGRESS	Resource creation Initiated

Figure 1: Example of the AWS CloudFormation console

When the cluster has successfully launched, you should see a message similar to the following:

```
Status: CREATE_COMPLETE
MasterServer: RUNNING
MasterPublicIP: 3.212.159.171
ClusterUser: ec2-user
MasterPrivateIP: 172.31.29.209
```

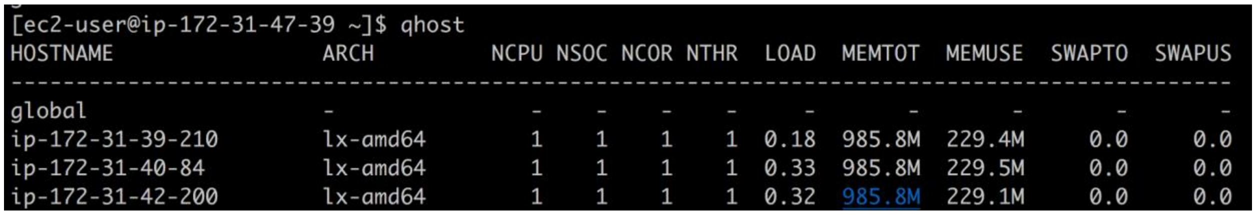
NOTE: Both the public and private IP addresses for your cluster are provided. If you wish to disable public IP address, set the `use_public_ips` flag to False. Refer to the aws-parallelcluster GitHub repo for more information.

Sign in to Your Cluster

1. Use the public IP address and the SSH key to sign in to your cluster. For example: `$ pcluster ssh HelloCluster –i ~/.ssh/mysshkey.pem`
2. Verify the current state of your cluster by entering the following command:

```
$ qhost
```

You'll see three hosts running, for example:



```
[ec2-user@ip-172-31-47-39 ~]$ qhost
HOSTNAME                ARCH         NCPU NSOC NCOR NTHR  LOAD  MEMTOT  MEMUSE  SWAPTO  SWAPUS
----------------------------------------------------------------------------------------------
global                  -               -    -    -    -     -       -       -       -       -
ip-172-31-39-210        lx-amd64        1    1    1    1  0.18  985.8M  229.4M     0.0     0.0
ip-172-31-40-84         lx-amd64        1    1    1    1  0.33  985.8M  229.5M     0.0     0.0
ip-172-31-42-200        lx-amd64        1    1    1    1  0.32  985.8M  229.1M     0.0     0.0
```

Step 5: Submit and Run a Simple Parallel MPI Job

In this exercise you will submit and run a simple parallel MPI job by taking the following actions: (1) create an executable, (2) create the job submittal file, and (3) launch the job.

Create the *mpi_hello_world* Executable

1. Change to the following directory:

```
$ cd /shared
```

NOTE: A shared Network File System (NFS) mount is created for you and is available at /shared. The share is an EBS volume mounted and shared using NFS.

2. Create the following directory:

```
$ mkdir hw_work
```

3. Change to the directory you created:

```
$ cd hw_work
```

4. Using your favorite editor, copy and paste the following text into the file named *mpi_hello_world.c*

```
/*A Parallel Hello World Program*/
#include <stdio.h>
#include <mpi.h>
main(int argc, char **argv)
{
int a, node;
MPI_Init(&argc,&argv);
MPI_Comm_rank(MPI_COMM_WORLD, &node);
for ( a = 1; a < 5; a=a+1){
printf("Hello World from Node %d\n",node);
}
MPI_Finalize();
}
```

5. Compile the code by entering the following:

```
$ /opt/amazon/efa/bin/mpicxx mpi_hello_world.c -o hw.x
```

You have created the *mpi_hello_world* executable.

Create the Job Submittal File

Create the job submittal file by copying and pasting the following text into a file called *hw.job*.

```
#!/bin/sh
#$ -cwd
#$ -N helloworld
#$ -pe mpi 3
#$ -j y
date
/opt/amazon/efa/bin/mpirun ./hw.x > hello_all.out
```

Launch the Job

1. At the command line prompt, enter the following to submit the job:

```
$ qsub hw.job
Your job 1 ("helloworld") has been submitted
```

NOTES:

Before the job completes, you can check its status by entering `qstat` at the command line prompt.

Two output files are created when the job is complete: (1) *hello_all.out* and (2) *helloworld.o1*

2. Display *hello_all.out* to the screen:

```
$ cat hello_all.out
```

Expected output:

```
$ cat hello_all.out
Hello World from Node 0
Hello World from Node 0
Hello World from Node 0
Hello World from Node 0
Hello World from Node 1
Hello World from Node 1
Hello World from Node 1
Hello World from Node 1
Hello World from Node 2
Hello World from Node 2
Hello World from Node 2
Hello World from Node 2
```

3. Display *hello_world.o1* to the screen:

```
$ cat helloworld.o1
```

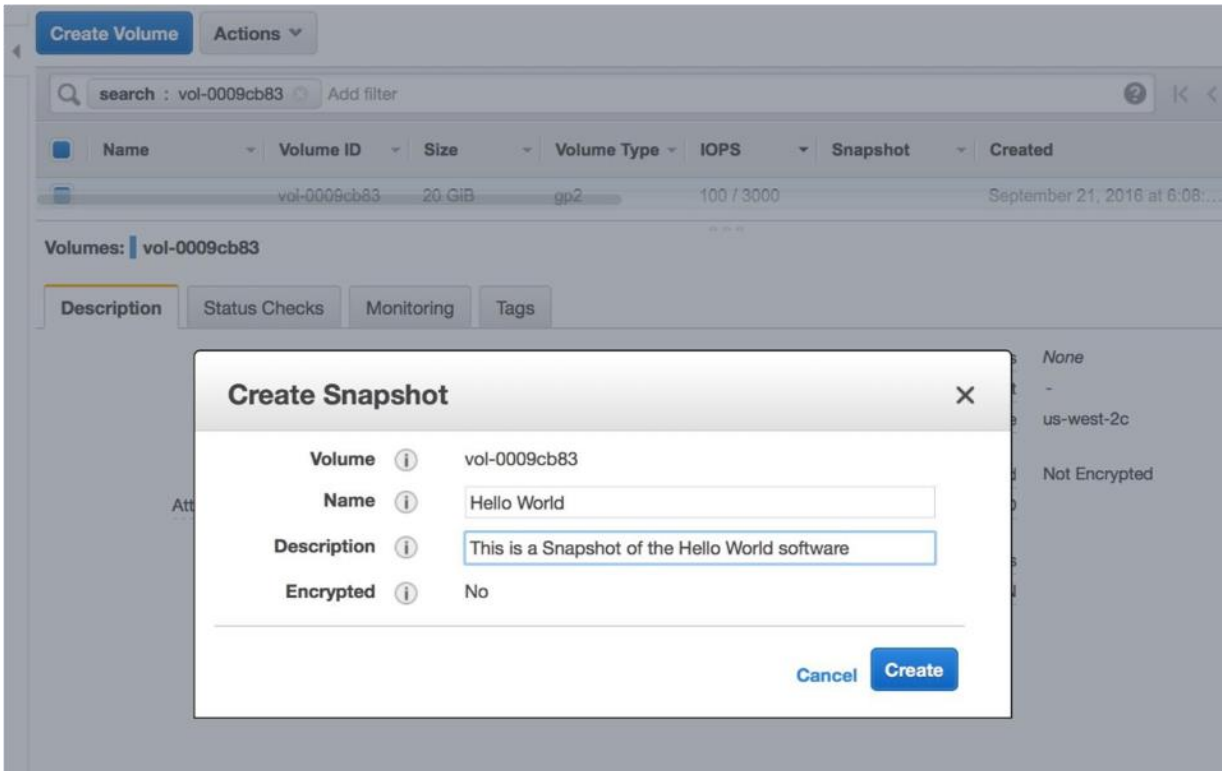
Expected output:

```
$ cat helloworld.o1
Fri Sep 2 04:43:00 UTC 2016
```

4. Log off the master instance by entering **exit** or pressing **Ctrl-d**.

Step 6: Create an Amazon EBS Volume Snapshot for Cluster Reusability

When setting up clusters, it is common to install large frequently-used HPC applications to the shared drive—*/shared*—that resides on an Amazon EBS volume. By creating a snapshot of this EBS volume (refer to Figure 2), you can deploy the same pre-configured software on future clusters. The following example is designed to walk you through this process.



search : vol-0009cb83 Add filter

Name	Volume ID	Size	Volume Type	IOPS	Snapshot	Created
	vol-0009cb83	20 GiB	gp2	100 / 3000		September 21, 2016 at 6:08:...

Volumes: vol-0009cb83

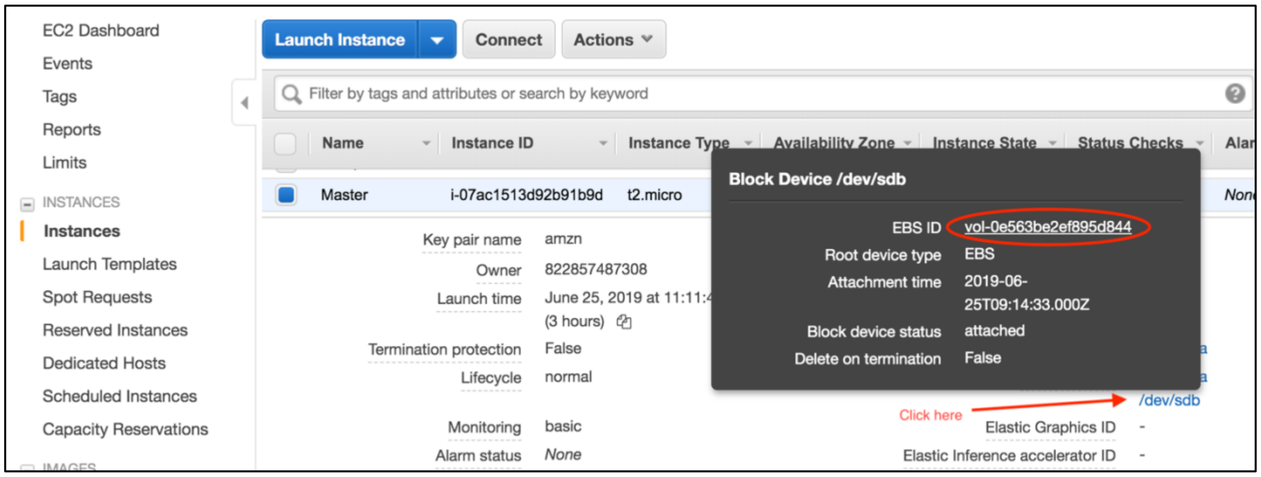
None
-
us-west-2c
Not Encrypted

Create Snapshot

Volume	vol-0009cb83
Name	Hello World
Description	This is a Snapshot of the Hello World software
Encrypted	No

Figure 2: Example of console for creating an EBS snapshot

1. Return to the Amazon EC2 console and select the master instance which is tagged with the name *Master*.
2. Scroll down to the block devices section of the instance description and select */dev/sdb*. A panel appears displaying the details for this block device.
3. Select the **volume ID** (for example, vol-xxxxxxxx) to access the volume dashboard.



Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alar
Master	i-07ac1513d92b91b9d	t2.micro				None

Key pair name	amzn
Owner	822857487308
Launch time	June 25, 2019 at 11:11:4 (3 hours)
Termination protection	False
Lifecycle	normal
Monitoring	basic
Alarm status	None

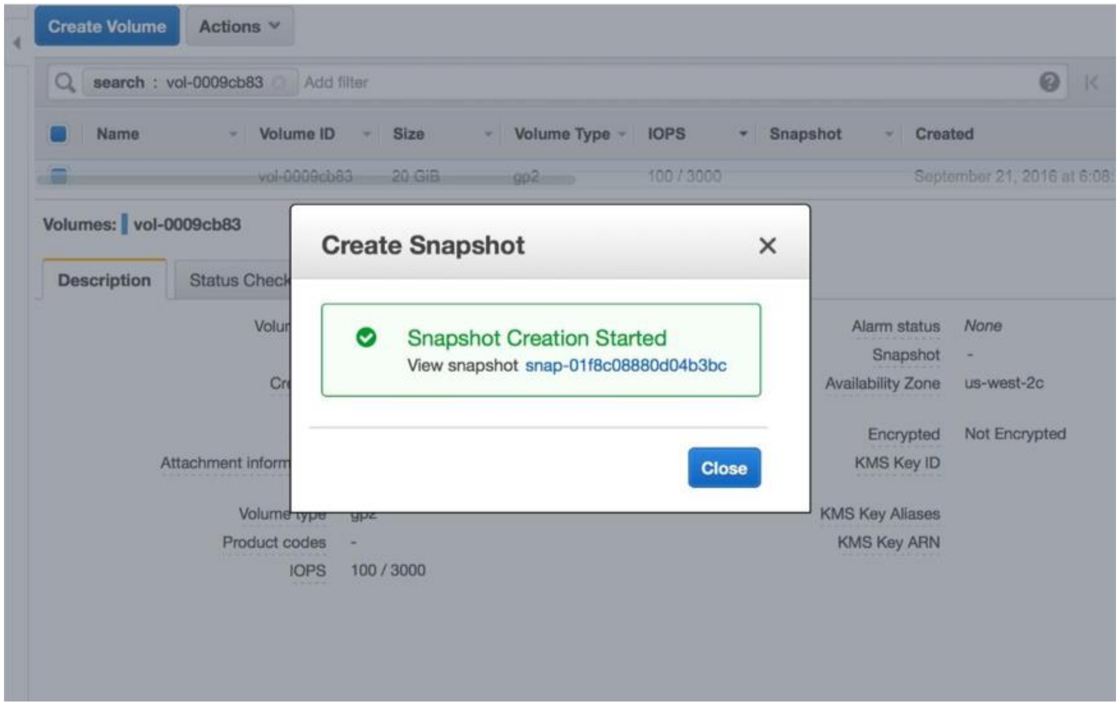
/dev/sdb

Elastic Graphics ID	-
Elastic Inference accelerator ID	-

Block Device /dev/sdb

EBS ID	vol-0e563be2ef895d844
Root device type	EBS
Attachment time	2019-06-25T09:14:33.000Z
Block device status	attached
Delete on termination	False

4. Open the **Action** drop-down menu and select **Create Snapshot**.
5. Enter the name **Hello World** and the following description: **This is a Snapshot of the Hello World software**.
6. Select **Create** and note the snapshot ID number.
   For this example, you can copy the following ID number into the copy and paste buffer: snap-0896bea72d42813f3. Consider copying the ID number to a text file for use in step 8 below.



search : vol-0009cb83

Name	Volume ID	Size	Volume Type	IOPS	Snapshot	Created
	vol-0009cb83	20 GiB	gp2	100 / 3000		September 21, 2016 at 6:08:

Volumes: vol-0009cb83

Create Snapshot

Snapshot Creation Started
View snapshot snap-01f8c08880d04b3bc

Alarm status	None
Snapshot	-
Availability Zone	us-west-2c
Encrypted	Not Encrypted
KMS Key ID	
KMS Key Aliases	
KMS Key ARN	
Volume type	gp2
Product codes	-
IOPS	100 / 3000

7. Edit the ParallelCluster config file (*~/.parallelcluster/config*) and add the following entries:

```
maintain_initial_size = 3
ebs_settings = helloebs
```

8. Add the following entries to the end of the config file:

```
[ebs helloebs]
ebs_snapshot_id = snap-XXXXXXXXXXXXXXXX (using your EBS snapshot
ID)
```

NOTE: This config file can be launched as a new cluster and the previously created volume and software will be automatically available on the shared drive: */shared*.

Step 7: Delete and Clean Up the Cluster

AWS ParallelCluster setups are easy to setup and tear down.

1. To list running clusters, enter the following from the launch computer:

```
$ pcluster list
HelloCluster  CREATE_COMPLETE   2.4.0
```

2. To clean up and remove the cluster enter the following command:

```
$ pcluster delete HelloCluster
```

   To verify that the cluster was successfully removed, look for the following message:

```
Deleting: HelloCluster
Status: DynamoDBTable - DELETE_COMPLETE
Stack with id  parallelcluster-HelloCluster does not exist
```

The cluster and all AWS cluster-related infrastructure is removed from AWS.

To restart the cluster when needed again, use the following command:

```
$ pcluster create HelloCluster2
```

The cluster is created with *mpi_hello_world*, which you already installed on the shared filesystem, */shared*.

Finally, when the snapshot created in step 5 is no longer needed, it can be deleted from the snapshot console found from the **EBS/Snapshot** tab on the left side of the Amazon EC2 dashboard.

Conclusion

This tutorial provided a low cost means to help you configure, install, deploy, and shut down AWS ParallelCluster, an AWS-supported open source cluster management tool built on the open source CfnCluster project. Keep in mind that AWS ParallelCluster is commonly installed on a local computer running OS X or Linux. Before you can get started, ensure you have an AWS account, an Amazon EC2 key pair, and AWS CLI is installed and configured. The launch computer should also have Python and pip installed.

By following the steps in this tutorial, you learned how to configure AWS credentials, launch and access AWS ParallelCluster, and submit a simple job; gaining practical experience in managing a fully-elastic HPC cluster in the cloud. Also, keep in mind that Amazon EBS volumes can be used to preserve applications between different clusters. As a result, it is not necessary to keep the cluster constantly running, which helps you to minimize costs.

Contributors

Contributors to this document include:

- Linda Hedges, Principal HPC Application Engineer, Amazon Web Services
- Sean Smith, Software Development Engineer, Amazon Web Services

Further Reading

- AWS ParallelCluster User Guide
- Amazon EC2
- Amazon EC2 Key Pairs
- AWS Command Line Interface User Guide
- Python Software Foundation
- Amazon Elastic Compute Cloud User Guide for Linux Instances: Placement Groups
- AWS CloudFormation

- aws-parallelcluster GitHub repo
- Amazon EBS

Document Revisions

Date	Description
June 2019	Updated references to the service name (changed from CfnCluster to AWS ParallelCluster) and corresponding instructions and diagrams.
September 2016	First publication