

Deploying an Elastic HPC Cluster

Access on-demand, scalable resources for your High-
Performance Computing (HPC) workloads

September 2016



Notices

This document is provided for informational purposes only. It represents AWS's current product offerings and practices as of the date of issue of this document, which are subject to change without notice. Customers are responsible for making their own independent assessment of the information in this document and any use of AWS's products or services, each of which is provided "as is" without warranty of any kind, whether express or implied. This document does not create any warranties, representations, contractual commitments, conditions or assurances from AWS, its affiliates, suppliers or licensors. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

Contents

Introduction	1
Step 1: Set Up Prerequisites for Your CfnCluster	2
Sign Up for AWS	2
Create an Amazon EC2 Key Pair	2
Set Up the AWS CLI	2
Install Python & pip on the Launch Computer	2
Step 2: Install CfnCluster	3
Step 3: Configure and Launch CfnCluster	3
Create the Base CfnCluster Configure File	3
Customize the CfnCluster Config File	6
Launch CfnCluster	6
Log onto Your CfnCluster	8
Step 4: Submit and Run a Simple Parallel MPI Job	8
Create the mpi_hello_world Executable	9
Create the Job Submittal File	10
Launch the Job	10
Step 5: Create an EBS Volume Snapshot for Cluster Reusability	12
Step 6: Delete and Clean Up the Cluster	14

Introduction

Imagine a High Performance Computing cluster that delivers the capabilities of a supercomputer and can be deployed in less than 15 minutes. CfnCluster is an open source tool published by AWS that provides just that: a fully-elastic HPC cluster in the cloud. CfnCluster constructs an HPC environment with the “look and feel” of 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; and 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 an 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 CfnCluster, 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 this tutorial, it is expected to cost less than one dollar.

CfnCluster is most often installed on a local computer running OS X or Linux. From the local computer, CfnCluster launches an HPC cluster comprised of a master instance and on-demand compute nodes running on Amazon EC2. If a local Mac or Linux computer is not available, then CfnCluster can be run from an EC2 instance running Linux.

More information on CfnCluster can be found at: <http://aws.amazon.com/hpc/cfncluster/>

Step 1: Set Up Prerequisites for Your CfnCluster

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; detailed instructions for installing these can be found by following the AWS CLI link. Additionally, you will need to have Python and pip installed on the launch computer. If you meet these prerequisites you can skip this step.

Sign Up for AWS

If you already have an AWS account, you can skip this prerequisite and use your existing account. To sign up for an AWS account if you do not already one:

1. Open <http://aws.amazon.com/>.
2. Choose **Create an AWS Account**.
3. Follow the online instructions.

Create an Amazon EC2 Key Pair

You must have an Amazon Elastic Compute Cloud (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

Different installation steps are needed, depending on your operating system and environment. Available installation methods include an *MSI* installer, a *bundled* installer, or *pip*. The following sections will help you decide which option to use. Please 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. Use the information at 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](#)

Install Python & pip on the Launch Computer

CfnCluster is written in python and is easily installed with pip (the python installation package). If python is not already installed on the launch computer, install it from here: python.org. Pip is

installed by default when using Python 2 >=2.7.9 or Python 3>=3.4. For more information, visit:
<https://pypi.python.org/pypi/pip/>

1. The existence and version of python can be checked from the command line prompt by typing:

```
$ python --version
```

2. The existence and version of pip can be checked from the command line prompt by typing:

```
$ pip -version
```

Proceed to the next step.

Step 2: Install CfnCluster

Installing CfnCluster is quick and easy. From the launch computer command-line prompt, type:

```
$ sudo pip install --upgrade cfncluster
```

CfnCluster has now been installed.

Step 3: Configure and Launch CfnCluster

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

Create the Base CfnCluster Configure File

CfnCluster customization is specified with the CfnCluster “config” file. The config file is a simple text file with keyword entries. The config file is created with a CfnCluster configuration tool which creates a baseline config file for the user.

1. At the prompt, type:

```
$ cfncluster configure
```

2. Accept the defaults for the first three entries by typing “enter” after each of the following lines:

```
Cluster Template [default]:
```

```
AWS Access Key ID []:
```

AWS Secret Access Key ID []:

Note: It is only necessary to enter your credentials if you have not already configured the AWS CLI or if you would like to override the credentials used by the AWS CLI or the environment.

3. The next prompt requires the AWS region:

Acceptable Values for AWS Region ID:

us-east-1

us-west-1

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

AWS Region ID [us-west-2]:

4. Enter the desired AWS region from the examples provided. A copy and paste of the regions offered ensures correct syntax. This prompt cannot be left blank.

5. At the prompt: “VPC Name [public]:”, a name is requested for use in the config file. This name is used to head the VPC section of the config file and is not used anywhere else on AWS or the cluster. Type enter to accept the default.
6. The next prompt asks for an SSH Key Name and offers acceptable values. Copy and paste the name of the Amazon EC2 keypair created earlier. Note that you should not include “.pem” suffix.
7. Select a VPC ID from the VPCs listed. Possible VPC IDs are offered for user selection. Copy and paste the desired VPC ID you want to use.
8. Select a VPC Subnet ID. Possible Subnet ID’s are offered for user selection. Copy and paste the desired VPC Subnet ID.
9. With this information, the CfnCluster “configure” command creates the ~/.cfncluster directory and then writes a simple config file to that directory. This configuration file provides the specifications for the cfncluster to be launched.
10. Review your new config file if you would like to verify that everything is correct:

```
$ cat ~/.cfncluster/config
```

The config file will look like something like this:

```
[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
```

Note: Information on configuration settings are available in the [CfnCluster documentation](#).

Customize the CfnCluster Config File

Using your favorite editor, add the following two keyword lines to the bottom of the cluster default section. These lines can go right after the “key_name= mysshkey” keyword entry.

From your editor type:

```
initial_queue_size = 3

max_queue_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 CfnCluster config file. For example, the **placement_group** keyword is used to create an EC2 [placement group](#) which can result in better performance for applications which require the lowest latency possible. For the example in this tutorial, a placement group is not required. More information on the CfnCluster config file is found [here](#):

<http://cfncluster.readthedocs.io/en/latest/working.html>

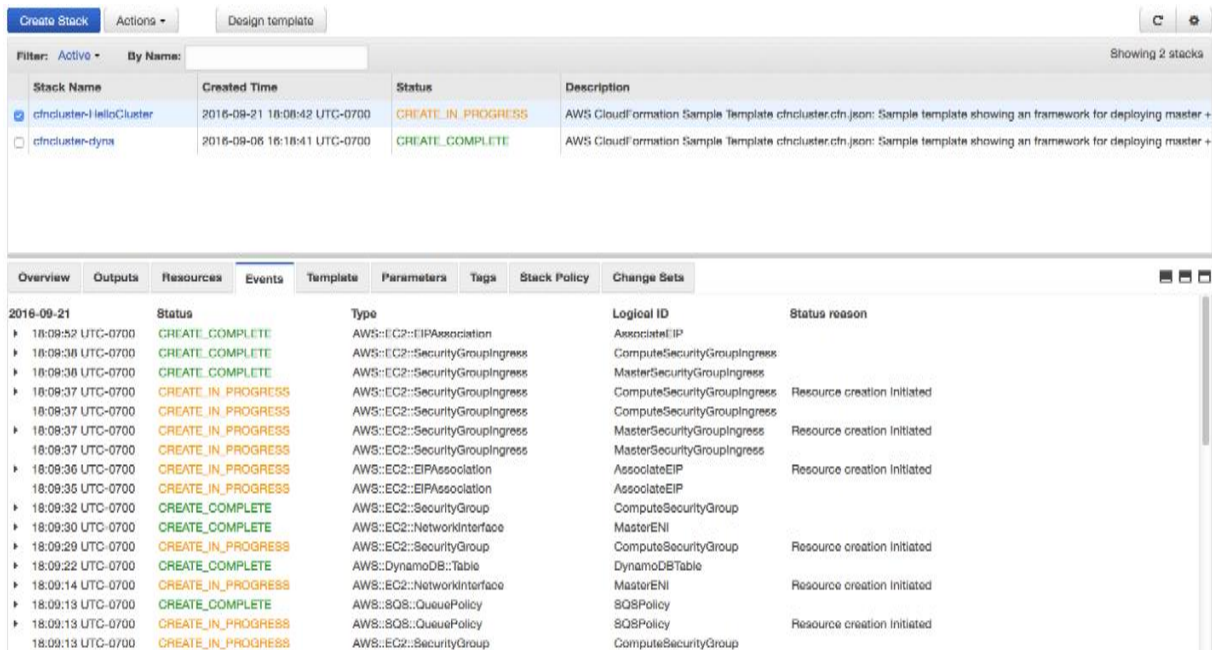
Launch CfnCluster

1. To launch your CfnCluster, enter the following at the command line prompt:

```
cfncluster create HelloCluster
```

Note: In this example, the cluster name is “HelloCluster”.

Full deployment of the CfnCluster, “HelloCluster”, will take about 20 minutes depending on the instance types chosen and the size of the EBS volumes selected. Deployment progress can be monitored from the [AWS CloudFormation console](#) events tab, as shown in Figure 1.



Stack Name	Created Time	Status	Description
cfncluster-HelloCluster	2016-09-21 18:08:42 UTC-0700	CREATE_IN_PROGRESS	AWS CloudFormation Sample Template cfncluster.cfn.json: Sample template showing an framework for deploying master +
cfncluster-dyna	2016-09-06 16:18:41 UTC-0700	CREATE_COMPLETE	AWS CloudFormation Sample Template cfncluster.cfn.json: Sample template showing an framework for deploying master +

Overview	Outputs	Resources	Events	Template	Parameters	Tags	Stack Policy	Change Sets
2016-09-21		Status	Type	Logical ID	Status reason			
18:09:52 UTC-0700		CREATE_COMPLETE	AWS::EC2::EIPAssociation	AssociateEIP				
18:09:30 UTC-0700		CREATE_COMPLETE	AWS::EC2::SecurityGroupIngress	ComputeSecurityGroupIngress				
18:09:30 UTC-0700		CREATE_COMPLETE	AWS::EC2::SecurityGroupIngress	MasterSecurityGroupIngress				
18:09:37 UTC-0700		CREATE_IN_PROGRESS	AWS::EC2::SecurityGroupIngress	ComputeSecurityGroupIngress	Resource creation Initiated			
18:09:37 UTC-0700		CREATE_IN_PROGRESS	AWS::EC2::SecurityGroupIngress	ComputeSecurityGroupIngress				
18:09:37 UTC-0700		CREATE_IN_PROGRESS	AWS::EC2::SecurityGroupIngress	MasterSecurityGroupIngress	Resource creation Initiated			
18:09:37 UTC-0700		CREATE_IN_PROGRESS	AWS::EC2::SecurityGroupIngress	MasterSecurityGroupIngress				
18:09:36 UTC-0700		CREATE_IN_PROGRESS	AWS::EC2::EIPAssociation	AssociateEIP	Resource creation Initiated			
18:09:35 UTC-0700		CREATE_IN_PROGRESS	AWS::EC2::EIPAssociation	AssociateEIP				
18:09:32 UTC-0700		CREATE_COMPLETE	AWS::EC2::SecurityGroup	ComputeSecurityGroup				
18:09:30 UTC-0700		CREATE_COMPLETE	AWS::EC2::NetworkInterface	MasterENI				
18:09:29 UTC-0700		CREATE_IN_PROGRESS	AWS::EC2::SecurityGroup	ComputeSecurityGroup	Resource creation Initiated			
18:09:22 UTC-0700		CREATE_COMPLETE	AWS::DynamoDB::Table	DynamoDBTable				
18:09:14 UTC-0700		CREATE_IN_PROGRESS	AWS::EC2::NetworkInterface	MasterENI	Resource creation Initiated			
18:09:13 UTC-0700		CREATE_COMPLETE	AWS::SQS::QueuePolicy	SQSPolicy				
18:09:13 UTC-0700		CREATE_IN_PROGRESS	AWS::SQS::QueuePolicy	SQSPolicy	Resource creation Initiated			
18:09:13 UTC-0700		CREATE_IN_PROGRESS	AWS::EC2::SecurityGroup	ComputeSecurityGroup				

Figure 1 Example of the AWS CloudFormation console

2. Verify successful completion of the cluster using the following:

Status: cfncluster-HelloCluster - CREATE_COMPLETE

Output: "MasterPublicIP"="52.43.23.96"

Output: "MasterPrivateIP"="172.31.1.45"

Output: "GangliaPublicURL"="http://52.43.22.95/ganglia/"

Output: "GangliaPrivateURL"=<http://172.31.1.45/ganglia/>

Note: Both the public and private IP address for your cluster are provided. Additionally, a Ganglia monitoring portal is created by default on the master node and can be used to monitor the cluster.

3. Copy and paste the line the GangliaPublicURL in a browser to see a console view of the cluster, similar to the one shown in Figure 2.

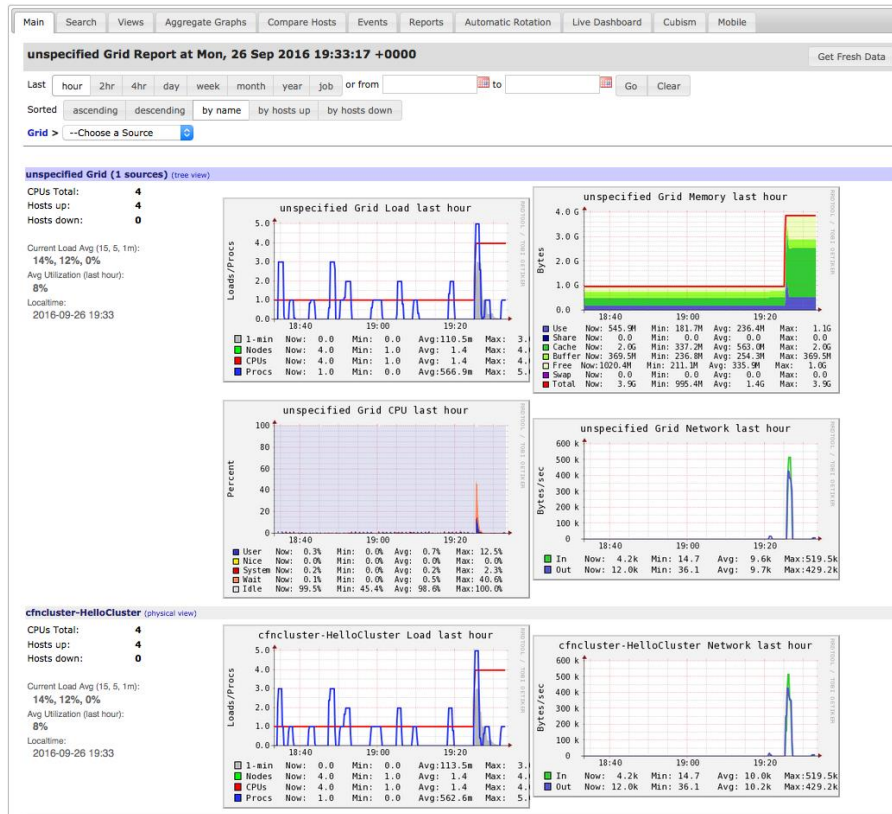


Figure 2 CfnCluster Monitor

Log onto Your CfnCluster

1. Use the public IP address and the ssh key to log into your cluster. For example:

```
ssh -i ~/.ssh/mysshkey.pem ec2-user@52.43.23.96 (but using your public IP address and key name) .
```

2. Verify the current state of your cluster by typing the following command

```
type "qstat -f"
```

Proceed to the next step.

Step 4: Submit and Run a Simple Parallel MPI Job

In this exercise you will submit and run a simple parallel MPI job. First, you create an executable, and then create the job submittal file. Lastly, you launch the job.

Create the mpi_hello_world Executable

1. A shared NFS mount is created for you and is available at `/shared`. The share is an EBS volume mounted and shared using NFS. Change to this directory:

```
$ cd /shared
```

2. Create a directory called “hw_work”:

```
$ mkdir hw_work
```

3. Change to the directory you created:

```
$ cd hw_work
```

Using your favorite editor, copy and paste this text into a 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();
```

```
}
```

4. Compile the code:

```
/usr/lib64/openmpi/bin/mpicxx mpi_hello_world.c -o hw.x
```

You have created the `mpi_hello_world` executable.

Create the Job Submittal File

1. 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
```

```
/usr/lib64/openmpi/bin/mpirun ./hw.x > hello_all.out
```

Launch the Job

2. Submit the job:

```
$ qsub hw.job
```

3. If you are quick, you can check the status of the job with

```
$ qstat
```

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

4. Display `hello_all.out` to the screen:

```
$ cat hello_all.out
```

The output file should look something like this:

```
[ec2-user@ip-172-31-2-164 hw_work]$ 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
```

5. Display hello_world.o1 to the screen:

```
$ cat hello_world.o1
```

The output will look like this:

```
[ec2-user@ip-172-31-2-164 hw_work]$ cat hello_world.o2
```

Fri Sep 2 04:43:00 UTC 2016

6. Log off the master instance.

Proceed to the next step.

Step 5: Create an 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 (Figure 3), you can deploy the same pre-configured software on future clusters. The following example is designed to walk you through this process.

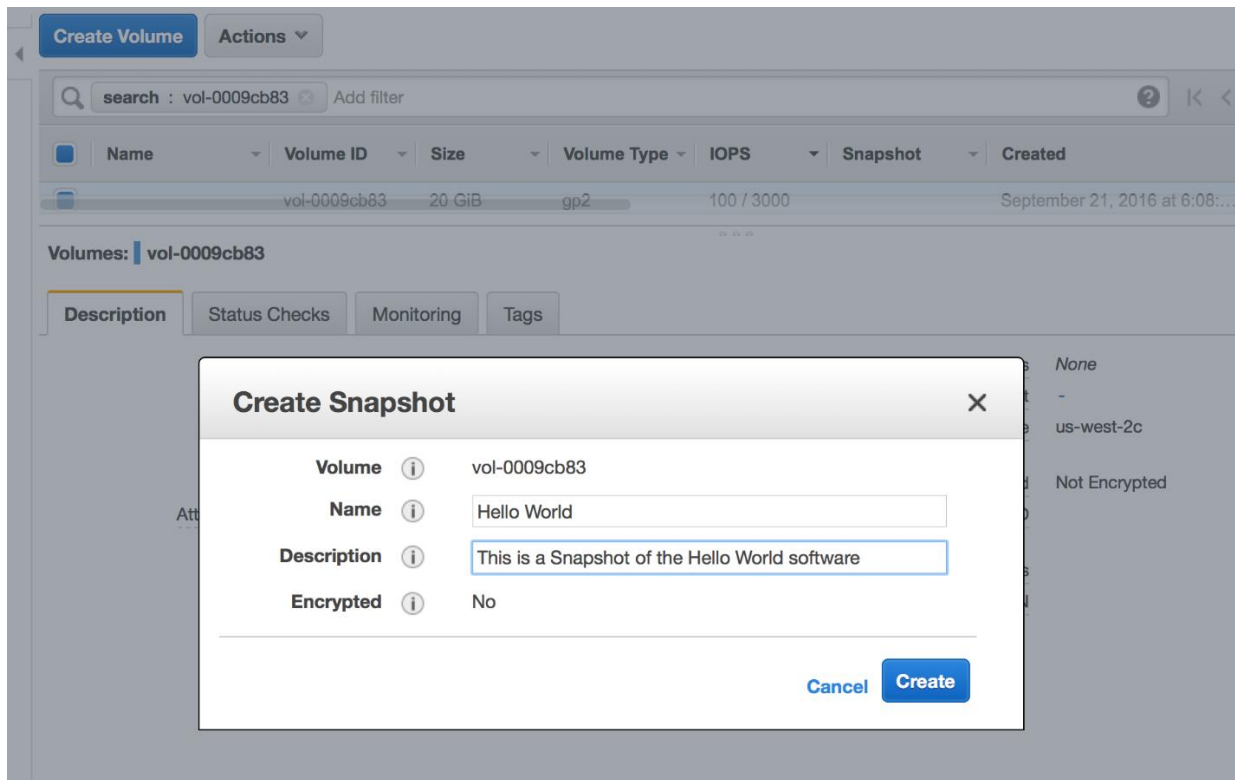


Figure 3 Example of console for creating an EBS snapshot

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 click on `/dev/sdb`. A panel displaying the details for this block device will appear.

3. Click on the volume id (i.e. vol-xxxxxxx) to bring up the volume dashboard.
4. Using the “Action” pull-down menu, select “Create Snapshot”.
5. Enter the name “Hello World” and the description “This is a Snapshot of the Hello World software”.
6. Select create and note the snapshot id number. For example: [snap-0896bea72d42813f3](#), copy the ID number into the copy and paste buffer. You may find it convenient to move the ID number to a text file for use in step 8 below.

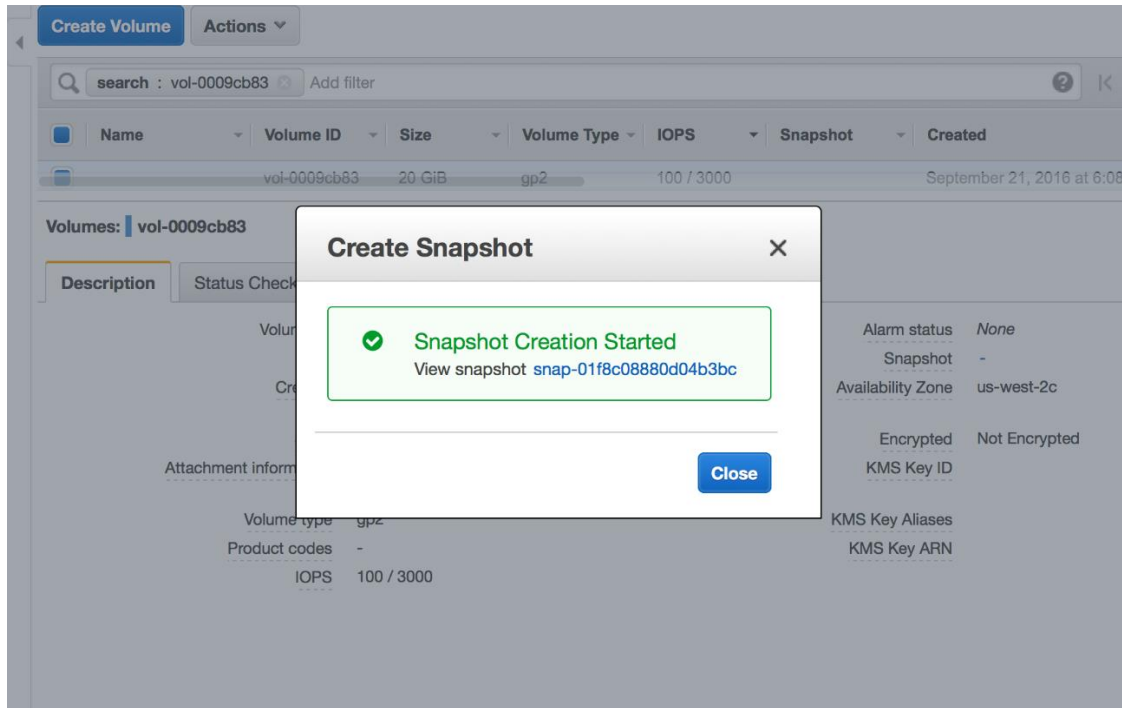


Figure 4 Screenshot showing snapshot ID number

7. Edit the cfncluster config file (`~/cfncluster/config`) and add the following keyword line below `max_queue_size=3`:

```
ebs_settings = helloebs
```

8. Add the following two lines at the end of the very end of the config file:

```
[ebs helloebs]
```

```
ebs_snapshot_id = snap-XXXXXXXXXXXXXXXXX (using your EBS snapshot ID)
```

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

Proceed to the next step.

Step 6: Delete and Clean Up the Cluster

CfnCluster setups are easy to setup and tear down.

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

```
$ cfncluster list
```

The recently created HelloCluster will be listed:

```
[ec2-user@ip-172-31-21-239 .cfncluster]$ cfncluster list
```

```
HelloCluster
```

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

```
$ cfncluster delete HelloCluster
```

3. Verify the cluster is removed by waiting for the following to be displayed:

```
Deleting: HelloCluster
```

```
Status: DynamoDBTable - DELETE_COMPLETE
```

```
Stack with id cfncluster-HelloCluster does not exist
```

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

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

```
$cfncluster 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 EC2 dashboard.