Building an HPC Cluster in 10 minutes.

Brendan Bouffler
Scientific Computing (SciCo) @ AWS
bouffler@amazon.com
My slides on cfnCluster

http://boofla.io/u/cfnCluster

“hello world” tarball at http://boofla.io/u/q8
It's a real cluster

Virtual Private Cloud

Head Instance
- 2 or more cores (as needed)
- CentOS 6.x, 7.x, Amazon Linux
- OpenMPI, gcc etc...

Choice of scheduler:
- Torque, SGE, OpenLava, SLURM

Compute Instances
- 2 or more cores (as needed)
- CentOS 6.x, 7.x, Amazon Linux

Auto Scaling group driven by scheduler queue length.

Can start with 0 (zero) nodes and only scale when there are jobs.
cfnCluster - provision an HPC cluster in minutes

cfncluster is a sample code framework that deploys and maintains clusters on AWS. It is reasonably agnostic to what the cluster is for and can easily be extended to support different frameworks. The CLI is stateless, everything is done using CloudFormation or resources within AWS.

https://github.com/awslabs/cfncluster

10 minutes

Configuration is really simple ....

There’s not a great deal involved getting a cluster up and running.

The config file below will do it. We’ll spend the next ten minutes or so showing you how to assemble all the bits of data you need.

```plaintext
[aws]
aws_region_name = us-east-1

[cluster default]
vpc_settings = public
key_name = boof-cluster

[vpc public]
master_subnet_id = subnet-fe83e3c4
vpc_id = vpc-7cf12419

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

10 minutes
Establish an AWS account

cfnCluster uses a number of AWS services to provision an HPC cluster.

To get started, you’ll need an AWS login, with appropriate privileges, and a Key ID & Secret Key.

When you create a user and subsequently their access key, IAM returns the access key ID and a secret access key. You should save these in a secure location and give them to the user. To ensure the security of your AWS account, the secret access key is accessible only when you create the access key. If a secret key is lost, you can delete the access key for the associated user and then create a new key.

REMEMBER though: these keys control your AWS account and can run up a bill, so you need to keep them safe from anyone else.

```
export AWS_ACCESS_KEY_ID="AKIAJJB7CYTBGG2E0QTQ"
export AWS_SECRET_ACCESS_KEY="/5UY7g0&yyh8hc1r8xWtaFW3G$OBBGrfk6+mq1U9"
```
Pull the cfnCluster source from GitHub.

Pull the source using easy_install or pip (python) and install it.

When you run easy_install or pip to install cfnCluster for the first time it will also download a few dependencies (if you don't have them already), such as "boto" the python API for controlling AWS.

$ sudo easy_install cfncluster

$ sudo pip install cfncluster
You’ll need some ssh credentials

Connecting via ssh into most instances in AWS requires keys to be generate in advance. You can generate these keys in the AWS EC2 console and import them to your desktop, but you can also use your existing SSH keys and import them to the EC2 console.

More detail is available at:

You can import this into your SSH environment.

```
$ cat random.em
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAlHFWmApmDnGZvnt1+h8ETK7yxg8xMWYqxmYXpN6LzwhxanBxck7q/OsZrGl7VeC8nHYmpQm5a9x8Iim0yoz+FPrLrLb5169YJkALGdBD+e+Fr6Eebplpv8kprj767T68rHBT
2k6gUqJRLx/u/UmGR+XgusPJX8SkgYmgEAs31omNoSe3GbkQ/fbshwNpPQPsp4G6JHlw5YTr5Z
OcoeLx59SCHCrAKvQ2g6r+8pjJKthBBDHbCafjy+y8HvYiibbdYArZ+QY61KKrflRlpLp91ixg
2vsENNVy09jXMm/BnYrIbnb2Xk23LCYQy2p+VrPuoodpvq5J5iX1DAQAAbAoIBAbBYyQJLzB7
jwKp2zv3FpueWrJIPXnwN85pp+nHNO7/35xqelqdtmEurxqdc/GOFWSbMlKlxm3s+Vr79d9h
2eqOOGZtksnFm+WvB7bxsxS8560r31L9z+BbMyKdwhxLoQ8yfa3tx17YtymrcmzR25ShKJde
kW+i6bDShYgDQLa6Spv5QYp4evwGANnTY/7QpeoZAIhB3j8d9g91wiVLCdZ5cBq1FneIjp4
+AZ+eKGLJmdHkktv4V9fohjKXcvmn9z2/75JrYgW6aV7v5J+6JuYCI3yAgz5Xv4Ab1B56nMvYrG/
4YgNFB88mDsGo13PfPmCupwV7P0CGYeA/dIZgg3bKQOex4301+vhFRTfjeVH2QndK7XiJNn
1HFK3RM3EaIDSH282-----------
RSA PRIVATE KEY-----
```

Either output the resultant public key to it’s own file (eg ~/.ssh/myCluster.pem). In this case you’ll need to use `ssh -i /path/to/file.pem`.  
Or place it in ~/.ssh/id_rsa.pub in which case it will be your default key.

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAABAABQCUcVaYcmYocERme2X6HxMrv3LFPFFZ1rg2heksoyPCGTOcHYur8
6mmsxvL41Rv3r31C31m3Hw1ebTLPT4sastw3J3kposY2E3ME6s4W0cQ5uK/sy/51GpVhxODBcGqFC
ToRKCWctL7s9B8YH5cGwWdW3I3AkoQcz7fwiY2I5x6Ci19+92A2K290/0jQoKLL1MxN3Ma5cutL0c
I0KREBkargyCv4Q0Qrk2E3CwF1B9LPL6we9j0JrJl1gCVP58jctU0oF+VGKu1m3WeDm80x17P2Z2Mcyb80
disIFS3Y9d7bcsJgdRnV5s+6h12mDkmML
```

#cfncluster
Configuring cfnCluster

All of cfnCluster's configuration lives in the file `~/.cfncluster/config`

The “cfncluster configure” command will prompt you through some sensible defaults and create a config file for you. You can then edit this to suit your needs before you create any clusters.

Comments begin with a “#”, as normal.

```
~ [10] $ cfncluster configure
Cluster Template [default]:
AWS Access Key ID []:
AWS Secret Access Key ID []:
Acceptable Values for AWS Region ID:
    us-east-1
cn-north-1
    ap-northeast-1
eu-west-1
    ap-southeast-1
    ap-southeast-2
    us-west-2
    us-gov-west-1
    us-west-1
eu-central-1
    sa-east-1
AWS Region ID []: us-east-1
VPC Name [public]:
Acceptable Values for Key Name:
    boof-cluster
Key Name []: boof-cluster
Acceptable Values for VPC ID:
    vpc-7cf12419
VPC ID []: vpc-7cf12419
Acceptable Values for Master Subnet ID:
    subnet-fe83e3c4
    subnet-8f6f86c8
    subnet-12876ef5
    subnet-286bc100
Master Subnet ID []: subnet-fe83e3c4
```

The result (in `~/.cfncluster/config`)

```
[aws]
aws_region_name = us-east-1

[cluster default]
vpc_settings = public
key_name = boof-cluster

[vpc public]
master_subnet_id = subnet-fe83e3c4
vpc_id = vpc-7cf12419

[global]
update_check = true
sanity_check = true
cluster_template = default
```
Other config options to consider

There are a few other options to consider adding which will impact your usage, as well as your costs - especially whilst you experiment with cfnCluster to get a feel for how it works.

- **t2.micro instances are really cheap** (around 1c per hour, really) and are good to use whilst you’re playing around with cfnCluster’s functionality.

- You can change to higher spec’d instances later by editing the config and issuing “cfncluster update”.

- **Queue size options**: yes, 0 instances is really ok. The cluster will grow on demand when you put jobs in the queue.

- **Scheduler**: choose from “sge”, “torque” or “openlava” (SLURM is coming soon).

There are also options to fool around with the **spot market**.

The complete set of configurables are documented in the comments fields of the master config file which you’ll find in:

```
/Library/Python/2.7/site-packages/cfncluster/examples/config
```

(or similar, depending on where your python distribution installed your cfnCluster files).

```python
[aws]
aws_region_name = us-east-1

[cluster default]
vpc_settings = public
key_name = boof-cluster

[vpc public]
master_subnet_id = subnet-fe83e3c4
vpc_id = vpc-7cf12419

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

# (defaults to t2.micro for default template)
compute_instance_type = t2.micro
# Master Server EC2 instance type
# (defaults to t2.micro for default template
master_instance_type = t2.micro

# Initial number of EC2 instances to launch as compute nodes in the cluster.
# (defaults to 2 for default template)
initial_queue_size = 0

# Maximum number of EC2 instances that can be launched in the cluster.
# (defaults to 10 for the default template)
max_queue_size = 10

# Boolean flag to set autoscaling group to maintain initial size and scale back
# (defaults to false for the default template)
maintain_initial_size = true

# Cluster scheduler
# (defaults to sge for the default template)
scheduler = sge
```

The complete set of configurables are documented in the comments fields of the master config file which you’ll find in:

```
/Library/Python/2.7/site-packages/cfncluster/examples/config
```

(or similar, depending on where your python distribution installed your cfnCluster files).

#cfncluster
OS Choices

Your choice of Operating System Distro

- base_os = centos6  # Centos 6.x latest
- base_os = centos6  # Centos 7.x latest
- base_os = alinux   # Amazon Linux (default)

Logging into the head node

$ ssh centos@1.2.3.4  
Centos

$ ssh ec2-user@1.2.3.4  
Amazon Linux

[aws]
aws_region_name = us-east-1

[cluster default]
vpc_settings = public
key_name = boof-cluster
base_os = centos7

[vpc public]
master_subnet_id = subnet-7c4fe83e3c4
vpc_id = vpc-7c4fe83e3c4

[global]
update_check = true
sanity_check = true
cluster_template = default
Build your first HPC Cluster in the cloud

The creation process might take a few minutes (maybe up to 5 mins or so, depending on how you configured it.

Because the API to Cloud Formation (the service that does all the orchestration) is asynchronous, we can kill the terminal session if we wanted to and watch the whole show from the AWS console (where you’ll find it all under the “Cloud Formation” dashboard in the events tab for this stack.

```
$ cfnCluster create boof-cluster
Starting: boof-cluster
Status: cfncluster-boof-cluster - CREATE_COMPLETE
Output:"MasterPrivateIP"="10.0.0.17"
Output:"MasterPublicIP"="54.66.174.113"
Output:"GangliaPrivateURL"="http://10.0.0.17/ganglia/"
Output:"GangliaPublicURL"="http://54.66.174.113/ganglia/"
```
It's a real cluster

Virtual Private Cloud

Head Instance
- 2 or more cores (as needed)
- CentOS 6.x
- OpenMPI, gcc etc...

Choice of scheduler: Torque, SGE, OpenLava, Slurm (coming soon)

Compute Instances
- 2 or more cores (as needed)
- CentOS 6.x

Auto Scaling group driven by scheduler queue length.

Can start with 0 (zero) nodes and only scale when there are jobs.
Your first cluster job

arthur ~ [26] $ cfnCluster create boof-cluster
Starting: boof-cluster
Status: cfncluster

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

You can customize quite a lot via the .cfncluster/config file - check out the comments.

#cfncluster
Hello, MPI_COMM_WORLD :)

```bash
#!/bin/bash
#
#$ -cwd
#$ -j y
#$ -pe mpi 16
#$ -S /bin/bash
#
module load openmpi-x86_64
mpirun -np $NSLOTS /shared/demo/hello-world
```

```
Your job 4 ("hello-world.qsub") has been submitted
[ec2-user@ip-172-31-34-119 ~]$ qstat
Your job 4 ("hello-world.qsub") has been submitted
```

```
Hello World from rank 7
Hello World from rank 1
Hello World from rank 2
Hello World from rank 3
Hello World from rank 5
Hello World from rank 0
Numprocs == 16
** All your processor are belong to us. **
Hello World from rank 4
Hello World from rank 6
Hello World from rank 8
Hello World from rank 9
Hello World from rank 10
Hello World from rank 11
Hello World from rank 15
Hello World from rank 12
Hello World from rank 14
Hello World from rank 13
```
Your cluster is ephemeral.
Yes, that’s right, you’ve created a disposable cluster. But it’s 100% recyclable.
It’s worth noting that anything you put into this cluster will vaporize when you issue the command

```
$ cfnccluster delete <your cluster name>
```
… which might not be what you first expect.
It’s easy to save your data tho, and pick up from where you left off later.
Before you delete your cluster, take a snapshot of the EBS (block storage) volume that you used for your /shared filesystem using the AWS EC2 console (see the pic on the right).
The EBC volume you care most about is the one attached to the headnode instance (hint: it’s probably the largest one).
To create a new cluster (or recreate your old one) from the EBS snapshot, modify the "[cluster default]" section of your config file (near the top) to tell it you have custom settings for your EBS volume.

Then add that custom section to the bottom of your config file like we have here, referring to the snapshot ID of the snapshot you created earlier.

If you want, you could create dozens of different clusters all using the same snapshot as their starting point. Or a different one for each cluster.

The sky's the limit. Go Crazy.

Just remember: they’re snapshots that are used at the moment in time when the file system is created. They don’t get updated magically, in fact: they don’t get updated at all. To create a new snapshot, you need to create a new snapshot.
System-wide Upgrade from Ivy Bridge to Haswell

Yes, really :-) 

```
$ ed ~/.cfncluster/config
/compute_instance_type/
compute_instance_type = c3.8xLarge
s/c3/c4/p
compute_instance_type = c4.8xLarge
w
949
$ cfnccluster update boof-cluster
```

Downgrading is just as easy. Honest.
Many options, but the most interesting ones immediately are:

```
# (defaults to t2.micro for default template)
compute_instance_type = c4.2xlarge
# Master Server EC2 instance type
# (defaults to t2.micro for default template)
#master_instance_type = c4.4xlarge
# Initial number of EC2 instances to launch as compute nodes in the cluster.
# (defaults to 2 for default template)
initial_queue_size = 0
# Maximum number of EC2 instances that can be launched in the cluster.
# (defaults to 10 for the default template)
max_queue_size = 10
# Boolean flag to set autoscaling group to maintain initial size and scale back
# (defaults to false for the default template)
maintain_initial_size = true
# Cluster scheduler
# (defaults to sge for the default template)
scheduler = sge
# Type of cluster to launch i.e. (defaults to ondemand for the default template)
cluster_type = spot
# Spot price for the ComputeFleet
spot_price = 0.50
# Cluster placement group. This placement group must already exist.
# (defaults to NONE for the default template)
placement_group = NONE
```

- Min & Max size of your cluster.
- Whether to fall back when things get quiet.
- Also can use ‘openlava’ or ‘torque’.
- Explore the SPOT market if you want to save money :)
- A placement group will provision your instances very close to each other on the network.
AWS Spot Market

Spot Market

Our ultimate space filler.

Spot Instances allow you to name your own price for spare AWS computing capacity.

Great for workloads that aren’t time sensitive, and especially popular in research (hint: it’s really cheap).
Spot Market - playing the [good] odds

Spot Bid Advisor

The Spot Bid Advisor analyzes Spot price history to help you determine a bid price that suits your needs.

You should weigh your application’s tolerance for interruption and your cost saving goals when selecting a Spot instance and bid price.

The lower your frequency of being outbid, the longer your Spot instances are likely to run without interruption.

Bid Price & Savings

Your bid price affects your ranking when it comes to acquiring resources in the SPOT market, and is the maximum price you will pay.

But frequently you’ll pay a lot less.

Spot Market - playing the [good] odds

bash-3.2$ python get_spot_duration.py \
--region us-east-1 \
--product-description 'Linux/UNIX' \
--bids c3.large:0.05,c3.xlarge:0.105,c3.2xlarge:0.21,c3.4xlarge:0.42,c3.8xlarge:0.84

<table>
<thead>
<tr>
<th>Duration</th>
<th>Instance Type</th>
<th>Availability Zone</th>
</tr>
</thead>
<tbody>
<tr>
<td>168.0</td>
<td>c3.8xlarge</td>
<td>us-east-1a</td>
</tr>
<tr>
<td>168.0</td>
<td>c3.8xlarge</td>
<td>us-east-1d</td>
</tr>
<tr>
<td>168.0</td>
<td>c3.8xlarge</td>
<td>us-east-1e</td>
</tr>
<tr>
<td>168.0</td>
<td>c3.4xlarge</td>
<td>us-east-1b</td>
</tr>
<tr>
<td>168.0</td>
<td>c3.4xlarge</td>
<td>us-east-1d</td>
</tr>
<tr>
<td>168.0</td>
<td>c3.4xlarge</td>
<td>us-east-1e</td>
</tr>
<tr>
<td>168.0</td>
<td>c3.xlarge</td>
<td>us-east-1d</td>
</tr>
<tr>
<td>168.0</td>
<td>c3.xlarge</td>
<td>us-east-1e</td>
</tr>
<tr>
<td>168.0</td>
<td>c3.large</td>
<td>us-east-1b</td>
</tr>
<tr>
<td>168.0</td>
<td>c3.large</td>
<td>us-east-1d</td>
</tr>
<tr>
<td>168.0</td>
<td>c3.large</td>
<td>us-east-1e</td>
</tr>
<tr>
<td>168.0</td>
<td>c3.2xlarge</td>
<td>us-east-1b</td>
</tr>
<tr>
<td>168.0</td>
<td>c3.2xlarge</td>
<td>us-east-1e</td>
</tr>
<tr>
<td>117.7</td>
<td>c3.large</td>
<td>us-east-1a</td>
</tr>
<tr>
<td>36.1</td>
<td>c3.2xlarge</td>
<td>us-east-1d</td>
</tr>
<tr>
<td>34.5</td>
<td>c3.4xlarge</td>
<td>us-east-1a</td>
</tr>
<tr>
<td>23.0</td>
<td>c3.xlarge</td>
<td>us-east-1b</td>
</tr>
<tr>
<td>21.9</td>
<td>c3.2xlarge</td>
<td>us-east-1a</td>
</tr>
<tr>
<td>17.3</td>
<td>c3.8xlarge</td>
<td>us-east-1b</td>
</tr>
<tr>
<td>0.1</td>
<td>c3.xlarge</td>
<td>us-east-1a</td>
</tr>
</tbody>
</table>

Spot Bid Advisor

As usual with AWS, anything you can do with the web console, you can do with an API or command line.

https://github.com/awslabs/aws-spot-labs
My slides on cfnCluster

http://boofla.io/u/cfnCluster

"hello world" tarball at http://boofla.io/u/q8
Thanks