Deploying Hadoop Cluster on AWS using EC2 instances

Installation and configuration of Infrastructure to deploy our projects often consume a lot of our time which can be used in learning instead. With the help of this post I attempt to make the process easier and save some of your time. Wasting no more of your precious time lets dive right into the subject. While on the article look out for some tips and tricks mentioned as notes to help you out!

In this post I will walk you through the process of configuring and deploying a Hadoop cluster using Amazon EC2 instances in 10 Simple steps.

STEP 1) Create AWS account

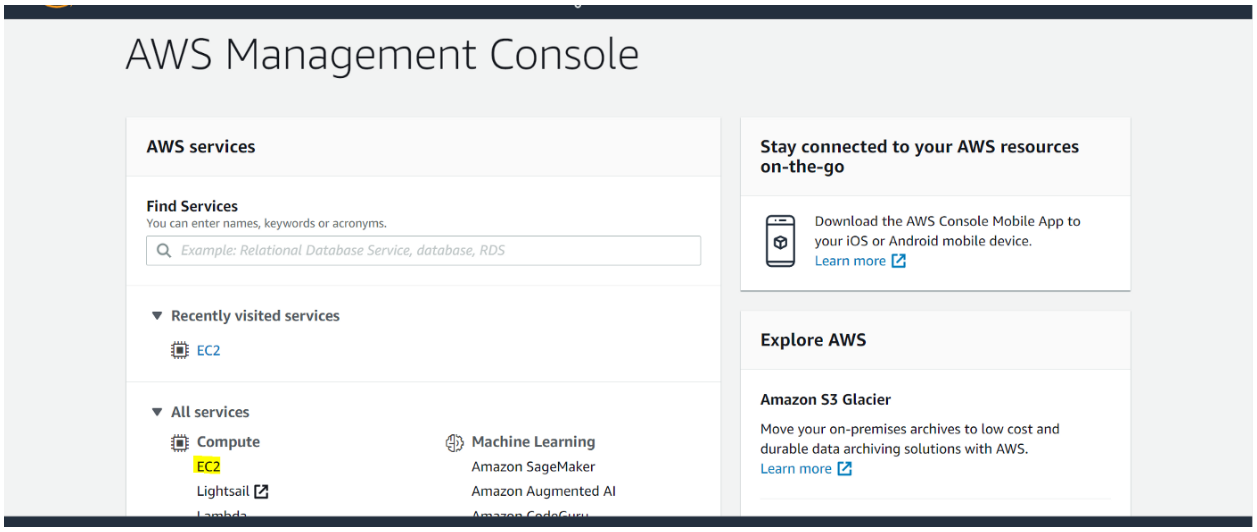
For those of you who are new to AWS, AWS(Amazon Web Services) is a cloud computing platform provided by Amazon in a pay-as-you-go model.

AWS offers certain services in it's free tier so as to give customers a look and feel of their platform. Luckily for you, the resources used in this post will cost you only as much as you use them and some fall right into the free tier category. Anyone can create an AWS Account with a valid email id and access free tier resources(for first 12 months).

STEP 2) Create EC2 Instance

Once you have created your AWS account and have reached the landing page, select the EC2 option under Compute Services in the AWS Management Console.

*Note : Amazon **EC2** (Elastic Compute Cloud) is Infrastructure as a Service offering from AWS that provides compute resources in AWS cloud. You will be charged on how much time you use the instance and keep it running.*



AWS Management Console

AWS services

Find Services
You can enter names, keywords or acronyms.
Example: Relational Database Service, database, RDS

Recently visited services
EC2

All services
Compute
EC2
Lightsail
Machine Learning
Amazon SageMaker
Amazon Augmented AI

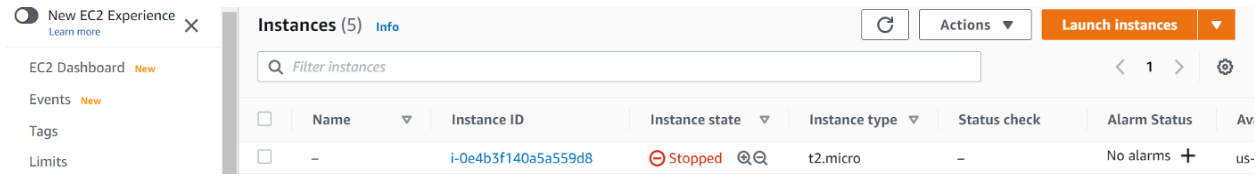
Stay connected to your AWS resources on-the-go

Download the AWS Console Mobile App to your iOS or Android mobile device.
Learn more

Explore AWS

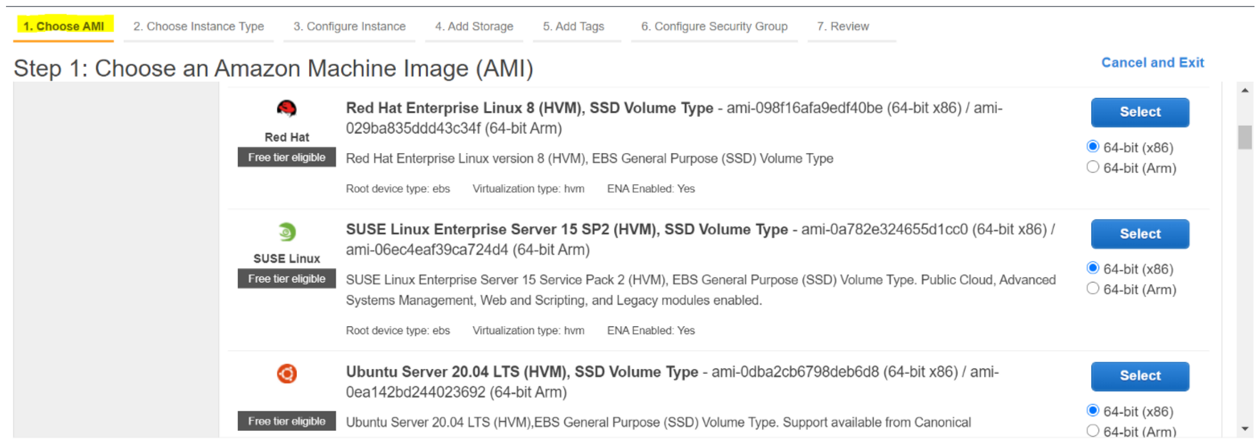
Amazon S3 Glacier
Move your on-premises archives to low cost and durable data archiving solutions with AWS.
Learn more

Once on the EC2 dashboard, click on the “Launch Instances” option.



Name	Instance ID	Instance state	Instance type	Status check	Alarm Status	Av
–	i-0e4b3f140a5a559d8	Stopped	t2.micro	–	No alarms	us-

This will lead to a 7-step procedure to launch your required configuration. The **Choose an Amazon Machine Image (AMI)** page displays a list of Instances with basic configurations called Amazon Machine Images (AMIs) that serve as templates for your instance.



Step 1: Choose an Amazon Machine Image (AMI)

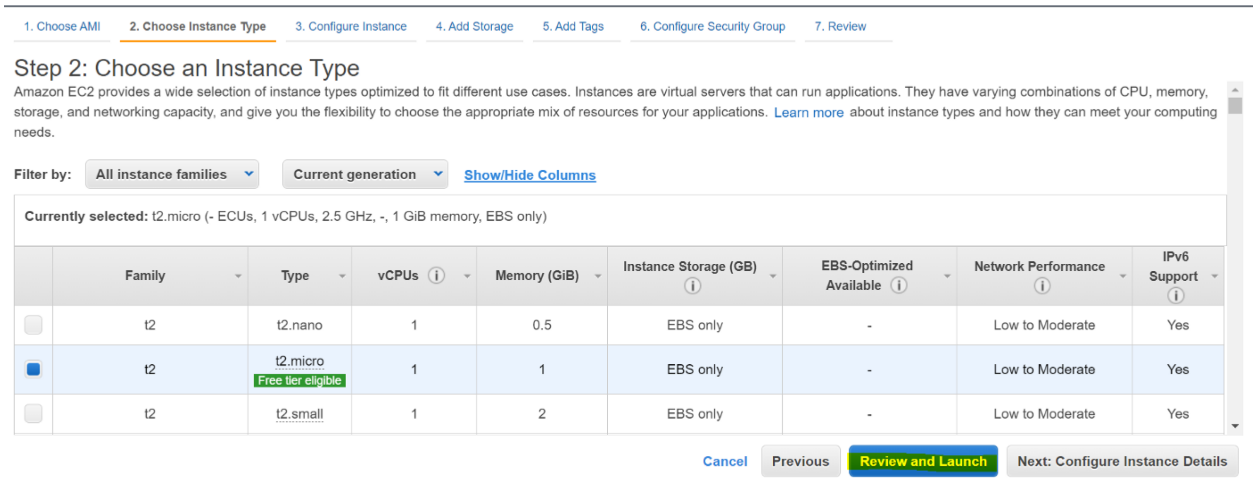
Red Hat Enterprise Linux 8 (HVM), SSD Volume Type - ami-098f16afa9edf40be (64-bit x86) / ami-029ba835ddd43c34f (64-bit Arm)
Red Hat Enterprise Linux version 8 (HVM), EBS General Purpose (SSD) Volume Type
Root device type: ebs Virtualization type: hvm ENA Enabled: Yes
[✓ 64-bit (x86)] 64-bit (Arm)

SUSE Linux Enterprise Server 15 SP2 (HVM), SSD Volume Type - ami-0a782e324655d1cc0 (64-bit x86) / ami-06ec4eaf39ca724d4 (64-bit Arm)
SUSE Linux Enterprise Server 15 Service Pack 2 (HVM), EBS General Purpose (SSD) Volume Type. Public Cloud, Advanced Systems Management, Web and Scripting, and Legacy modules enabled.
Root device type: ebs Virtualization type: hvm ENA Enabled: Yes
[✓ 64-bit (x86)] 64-bit (Arm)

Ubuntu Server 20.04 LTS (HVM), SSD Volume Type - ami-0dba2cb6798deb6d8 (64-bit x86) / ami-0ea142bd244023692 (64-bit Arm)
Ubuntu Server 20.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical
[✓ 64-bit (x86)] 64-bit (Arm)

Notice the “Free Tier Eligible” mark on the AMIs and select the any one you are familiar with. For my project I choose the ubuntu AMI.

Once done you can skip through the remaining configurations leaving them to be in their default settings. You can go through them if you wish to.



Step 2: Choose an Instance Type

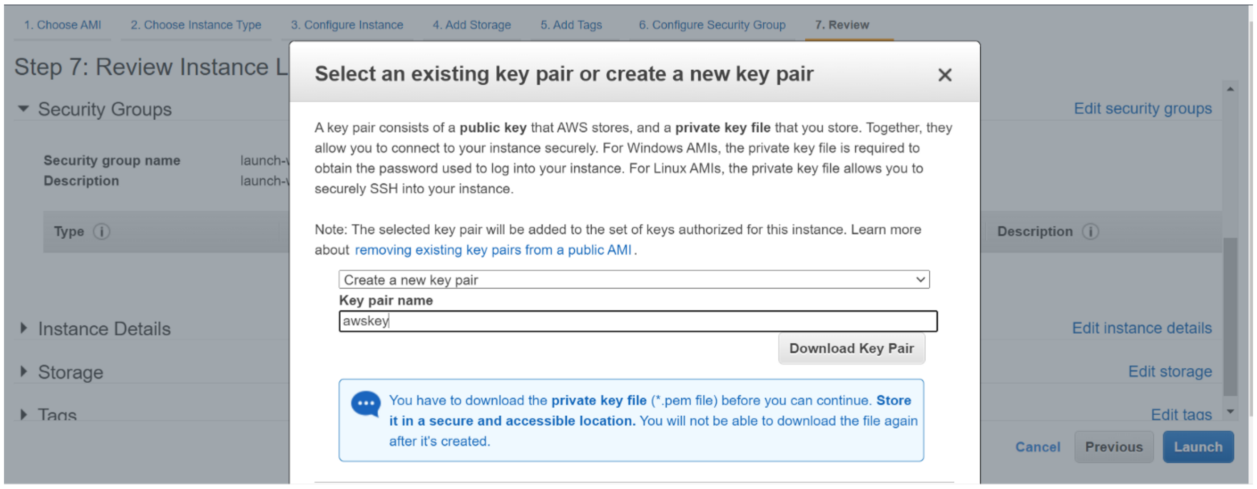
Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.

Filter by: All instance families | Current generation | Show/Hide Columns

Currently selected: t2.micro (- ECUs, 1 vCPUs, 2.5 GHz, -, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
☐	t2	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
☑	t2	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
☐	t2	t2.small	1	2	EBS only	-	Low to Moderate	Yes

Click on the “Review and Launch” button. You will be prompted to create or use a private key for connecting to the instance. If this is your first time creating your instance, choose “Create a new key pair” and enter the key name you would like to save it as. You can use an existing pair if you already have a key.



Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about removing existing key pairs from a public AMI.

Create a new key pair

Key pair name

awskey

Download Key Pair

You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

Once you have downloaded your key pair, click on “Launch Instance” and congratulations! You have successfully launched your first EC2 instance on AWS. :)

Note : Before you proceed, I suggest to use ELASTIC IP Address provided by AWS and assign them to your instance. This is because amazon charges you if you keep your instances running. If you don’t have any application running, it is best to shut down your instances to not get charged for them.

When you shut down your instance, the public address assigned to it, will be released back into the pool of available public IP addresses. So the next time you start your instance, you will have to SSH with the new public IP. This would cause an issue when you configure your instance's IP address in the config files for this setup as you will save "known hosts" to connect the nodes to each other.

STEP 3) Allocate Elastic IP addresses to your instances.

- For allocating Elastic IP address , just click on "Elastic IPs" under the network and security tab of your EC2 dashboard.
- Click on "Allocate Elastic IP Address"
- Further click on "Allocate" in the next window.
- The new Elastic IP is now allocated to you AWS account. You need to allocate this IP to your EC2 instance so that the IP never changes even after you shut down your Instance. Select the newly created IP and select "Associate Elastic IP" from the Actions menu.

- Associate the IP to your instance available in the drop down menu and proceed.

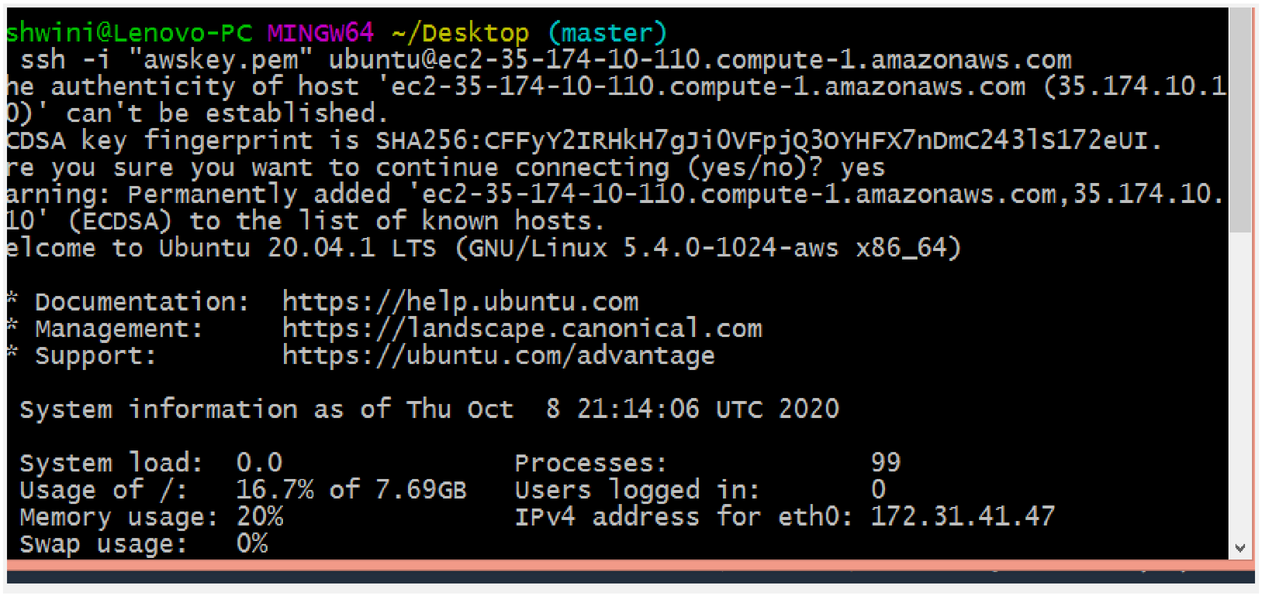
Now SSH with your newly created Elastic IP and check if the connection is working.

STEP 3) Install Java

Once your instance is launched and Elastic IP is allocated, connect to your instance from the actions menu. There are a number of ways to connect to it. The easiest and what I prefer is to SSH in your instance.

The instance you created will be assigned a public Elastic IP address that you can use to connect to it. If you are using a windows machine you can install bash or can used in ssh service in windows as well. I have Git Bash installed so I am going to use that.

```
ssh -i "yourkeyname.pem" DNSaddressofEC2
```



```
shwini@Lenovo-PC MINGW64 ~/Desktop (master)
 ssh -i "awskey.pem" ubuntu@ec2-35-174-10-110.compute-1.amazonaws.com
he authenticity of host 'ec2-35-174-10-110.compute-1.amazonaws.com (35.174.10.1
0)' can't be established.
CDSA key fingerprint is SHA256:CFFyY2IRHkH7gJi0VFpjQ3OYHFX7nDmC243lS172eUI.
re you sure you want to continue connecting (yes/no)? yes
arning: Permanently added 'ec2-35-174-10-110.compute-1.amazonaws.com,35.174.10.
10' (ECDSA) to the list of known hosts.
elcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-1024-aws x86_64)

* Documentation:  https://help.ubuntu.com
* Management:     https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage

 System information as of Thu Oct  8 21:14:06 UTC 2020

 System load:  0.0               Processes:             99
 Usage of /:   16.7% of 7.69GB   Users logged in:       0
 Memory usage: 20%               IPv4 address for eth0: 172.31.41.47
 Swap usage:   0%
```

Post that, run the following commands to keep your system up to date

```
sudo apt-get update
```

```
sudo apt-get dist-upgrade
```

Install open-jdk

```
sudo apt-get install openjdk-8-jdk
```

To verify if java is successfully installed, run the following command.

```
java -version
```

STEP 5) Install Hadoop

- Once Java is installed, we will go ahead and install Hadoop. You can get the latest Hadoop version here. Use the following command to download hadoop to the downloads folder.

```
wget
http://apache.mirrors.tds.net/hadoop/common/hadoop-2.10.0/hadoop
-2.10.0.tar.gz -P ~/Downloads
```

- Extract the contents to the /usr/local directory and rename the folder to hadoop for easy access.

```
sudo tar zxvf ~/Downloads/hadoop-* -C /usr/local
```

```
sudo mv /usr/local/hadoop-* /usr/local/hadoop
```

STEP 6) Declare Environment variables in /.bashrc

We need to declare environment variables so that applications can locate Hadoop and java.

To know where java is installed, you can use the following command.

```
readlink -f $(which java)
```

Open .bashrc file in your home directory with an editor. Include the below lines.

```
#JAVA_HOME

export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

export PATH=$PATH:$JAVA_HOME/bin


#HADOOP_HOME


export HADOOP_HOME=/usr/local/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
```

```
#HADOOP_CONF_DIR
```

```
export HADOOP_CONF_DIF=/usr/local/hadoop/etc/hadoop
```

Close the editor after the above changes and exit the editor. Executing the following command to reflect the changes without restarting.

```
source ~/.bashrc
```

Check whether the environmental variables are available or not.

STEP 7) Edit Hadoop Config files

We need to edit a few configuration files for Hadoop. The instance we created in this tutorial, we will use it as our Namenode for our Hadoop Cluster.

- In your local pc, open Git Bash and open the ~/.ssh/config file using the command below

```
vi ~/.ssh/config
```

- Add the following lines in the the config file

```
Host Namenode

       HostName <public_DNS_Address_of_your_instance>

       User Ubuntu

       IdentityFile ~/.ssh/awskey.pem
```

- Save and Exit the editor. Now you can just type `sss Namenode` to SSH into your Namenode.
- SSH into your Namenode instance and navigate to Hadoop configuration directory that you have stored in `$HADOOP_CONF_DIR` environment variable.

```
cd $HADOOP_CONF_DIR
```

Configure core-site.xml

- When editing config files, you'll need root access so remember to use `$ sudo`.

```
<configuration>

 <property>

   <name>fs.defaultFS</name>

   <value>hdfs://<your namenode public dns name>:9000</value>

 </property>

</configuration>
```

- `fs.defaultFS` will allow dfs commands to run without providing full site name in the command. Running **hdfs dfs -ls /** instead of **hdfs dfs -ls hdfs://hdfs/** . This is used to specify the default file system and defaults to your local file system that's why it needs be set to a HDFS address.

- Here, `hdfs://` will tell each node that which node is the Namenode and where to send it's heartbeat (port 9000).

Configure yarn-site.xml

- Configure your yarn-site.xml to set the properties of your resource manager.

```
<configuration>



<!— Site specific YARN configuration properties -->



 <property>

   <name>yarn.nodemanager.aux-services</name>
   <value>mapreduce_shuffle</value>

 </property>

 <property>

   <name>yarn.resourcemanager.hostname</name>

   <value><your namenode public dns name></value>

 </property>

</configuration>
```

- An AuxiliaryService is a generic service that is started by the NodeManager in Hadoop 2. The services is defined by the Yarn configuration “yarn.nodemanager.aux-services”. The default value is mapreduce_shuffle, i.e., the ShuffleHandler for MR 2.

Configure mapred-site.xml

- Copy `mapred-site.xml` from `mapred-site.xml.template`

```
sudo cp mapred-site.xml.template mapred-site.xml
```

- Add the following lines to mapred-site.xml
- On each node, copy `mapred-site.xml` from `mapred-site.xml.template`

```
<configuration>

 <property>

   <name>mapreduce.jobtracker.address</name>

   <value><your namenode public dns name>:54311</value>

 </property>

 <property>

   <name>mapreduce.framework.name</name>
   <value>yarn</value>

 </property>

</configuration>
```

- In this file, `mapreduce.jobtracker.address` sets the value of the Namenode where the jobtracker of hadoop runs. `mapreduce.framework.name` sets the resource manager for our hadoop cluster

Configure Hadoop-env.sh

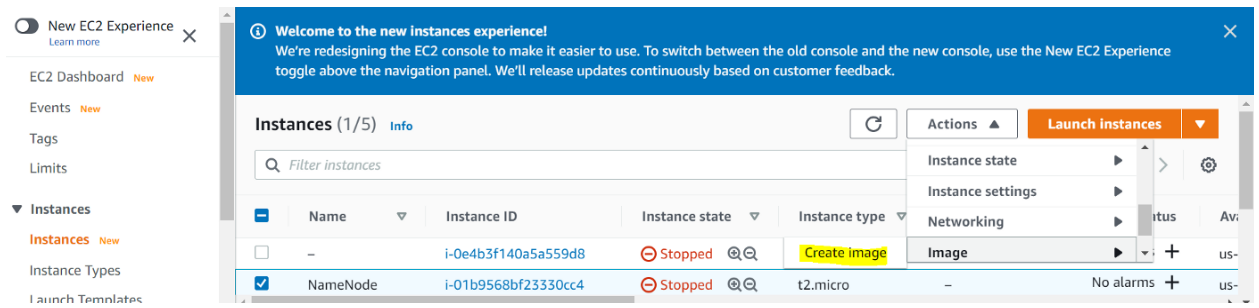
- In this file you just need to edit your JAVA_HOME variable and set it to the JAVA_HOME path in your environment same as how you set in /.bashrc file.

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

STEP 8) Create AMI Image and create 3 Datanodes based on our AMI

AWS provides you the option to create your own custom Amazon Machine Image based on your custom configuration along with providing standard AMIs. We create the AMI of out Namenode with current configuration to create 3 Datanodes for our cluster.

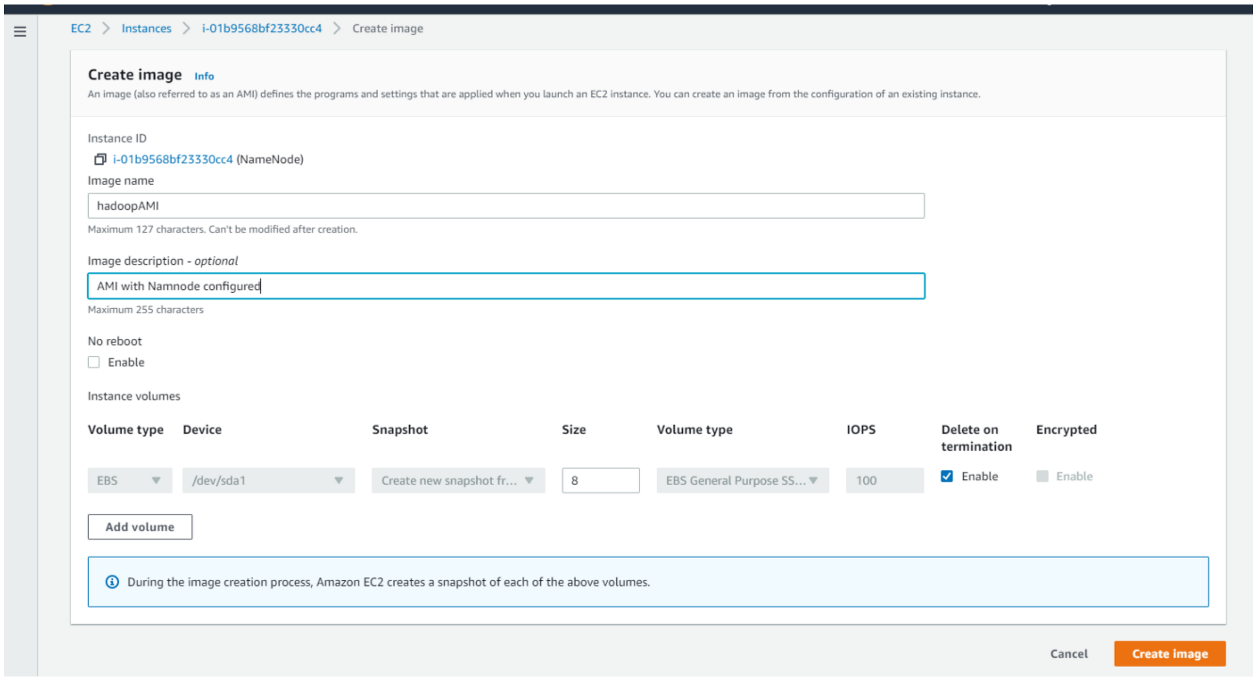
- Select the Instance, click on Actions Menu, Select Image and click on “Create Image” option.



Welcome to the new instances experience!
We’re redesigning the EC2 console to make it easier to use. To switch between the old console and the new console, use the New EC2 Experience toggle above the navigation panel. We’ll release updates continuously based on customer feedback.

	Name	Instance ID	Instance state	Instance type		atus	Av
☐	–	i-0e4b3f140a5a559d8	Stopped				us-
✓	NameNode	i-01b9568bf23330cc4	Stopped	t2.micro	–	No alarms	us-

- Provide a Name and Description for your AMI and click on Create Image Button.



EC2 > Instances > i-01b9568bf23330cc4 > Create image

Create image Info

An image (also referred to as an AMI) defines the programs and settings that are applied when you launch an EC2 instance. You can create an image from the configuration of an existing instance.

Instance ID: i-01b9568bf23330cc4 (NameNode)

Image name: hadoopAMI

Maximum 127 characters. Can't be modified after creation.

Image description - *optional*: AMI with Namnode configured

Maximum 255 characters

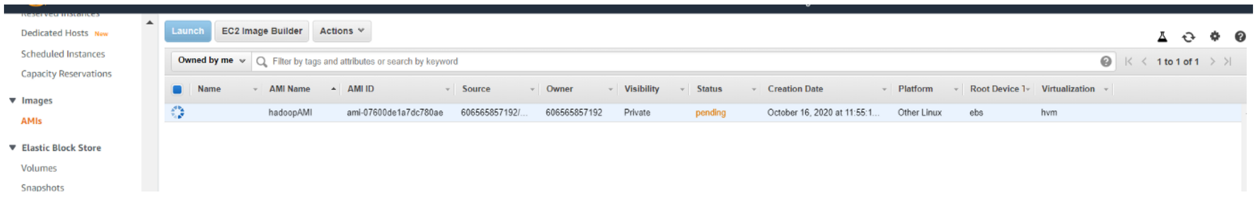
No reboot: ☐ Enable

Instance volumes

Volume type	Device	Snapshot	Size	Volume type	IOPS	Delete on termination	Encrypted
EBS	/dev/sda1	Create new snapshot fr...	8	EBS General Purpose SS...	100	✓ Enable	☐ Enable

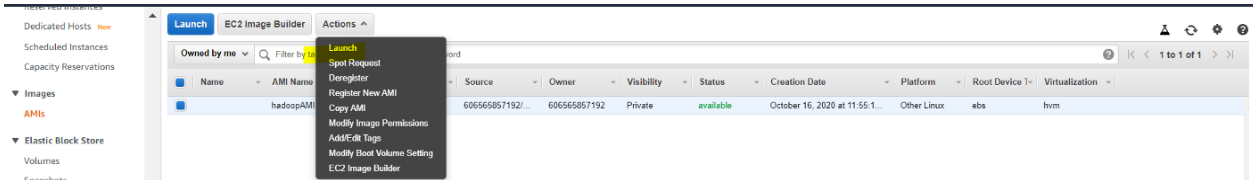
During the image creation process, Amazon EC2 creates a snapshot of each of the above volumes.

- Navigate to the AMIs option on the navigation menu. Your new AMI machine will be seen in the list of AMIs. It will take a while for AMI to be created and will be seen in pending state until then.



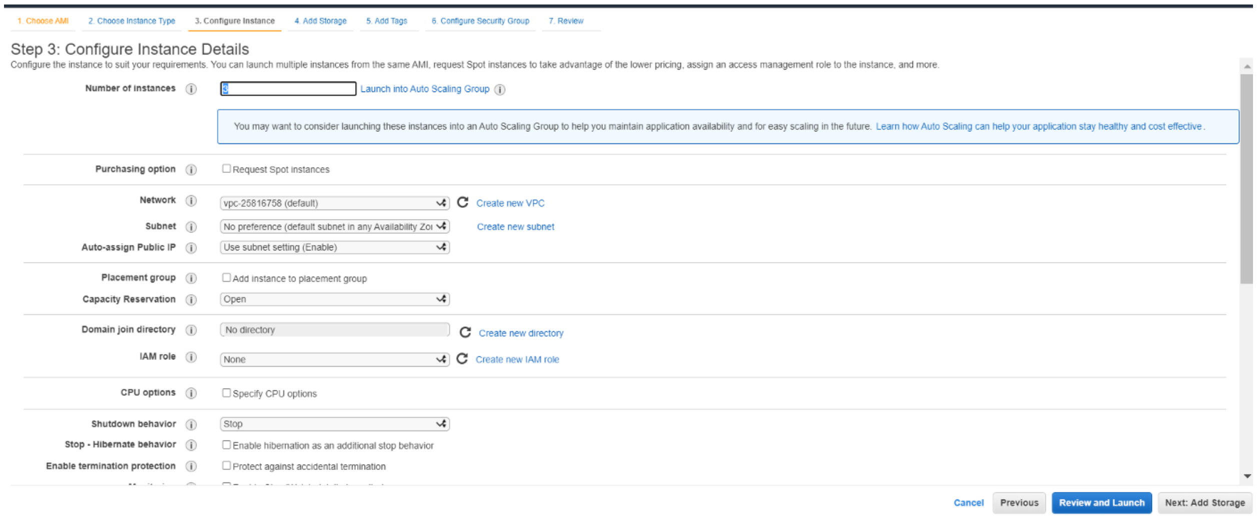
Name	AMI Name	AMI ID	Source	Owner	Visibility	Status	Creation Date	Platform	Root Device 1	Virtualization
	hadoopAMI	ami-07600de1a7dc780ae	606565857192/...	606565857192	Private	pending	October 16, 2020 at 11:55:1...	Other Linux	ebs	hvm

- Once your AMI is ready, select the AMI and click on the Actions Menu. Select the “Launch” option from the menu.



Name	AMI Name	Source	Owner	Visibility	Status	Creation Date	Platform	Root Device Type	Virtualization
	hadoopAMI	606565857192/...	606565857192	Private	available	October 16, 2020 at 11:55:1...	Other Linux	ebs	hvm

- In the “Configure Instance” tab, change the number of Instances from 1 to 3.

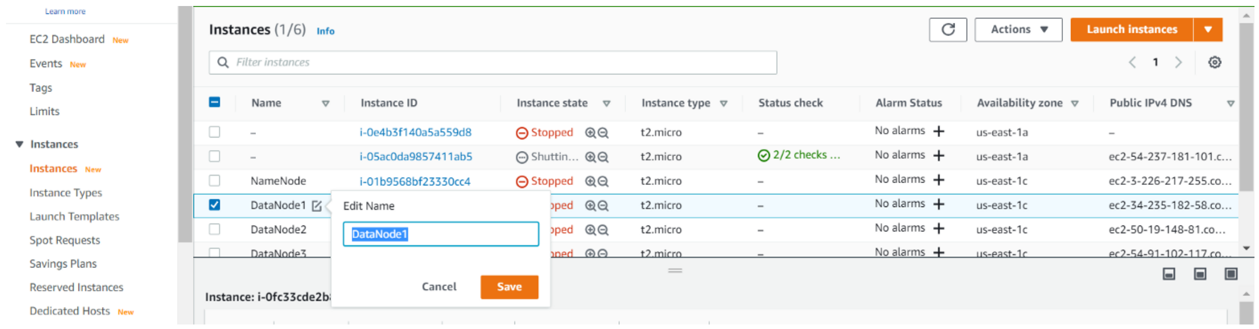


Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances	3 — Launch into Auto Scaling Group
	You may want to consider launching these instances into an Auto Scaling Group to help you maintain application availability and for easy scaling in the future. Learn how Auto Scaling can help your application stay healthy and cost effective.
Purchasing option	☐ Request Spot instances
Network	vpc-25816758 (default) — Create new VPC
Subnet	No preference (default subnet in any Availability Zone) — Create new subnet
Auto-assign Public IP	Use subnet setting (Enable)
Placement group	☐ Add instance to placement group
Capacity Reservation	Open
Domain join directory	No directory — Create new directory
IAM role	None — Create new IAM role
CPU options	☐ Specify CPU options
Shutdown behavior	Stop
Stop - Hibernate behavior	☐ Enable hibernation as an additional stop behavior
Enable termination protection	☐ Protect against accidental termination

- Review and Launch your EC2 instances(Datanodes)
- SSH into your instances and check if your configured hadoop files are available with your custom configurations.
- Assign identifiers to your nodes. These have no configuration significance but is just there for easily identifying.



Instances (1/6)

Name	Instance ID	Instance state	Instance type	Status check	Alarm Status	Availability zone	Public IPv4 DNS
–	i-0e4b3f140a5a559d8	Stopped	t2.micro	–	No alarms	us-east-1a	–
–	i-05ac0da9857411ab5	Shuttin...	t2.micro	2/2 checks ...	No alarms	us-east-1a	ec2-54-237-181-101.c...
NameNode	i-01b9568bf23330cc4	Stopped	t2.micro	–	No alarms	us-east-1c	ec2-3-226-217-255.co...
DataNode1		pped	t2.micro	–	No alarms	us-east-1c	ec2-34-235-182-58.co...
DataNode2		pped	t2.micro	–	No alarms	us-east-1c	ec2-50-19-148-81.co...
DataNode3		pped	t2.micro	–	No alarms	us-east-1c	ec2-54-91-102-117.co...

DataNode1

Instance: i-0fc33cde2b

- **Assign Elastic IP addresses to each of your DataNodes as we did before for the Namenode**

STEP 9) Enable Password-Less Authentication between nodes.

Your Hadoop nodes need to be able to connect with each other to execute their respective tasks. For this purpose, you need to enable password-less authentication between them.

- SSH into your Namenode
- Execute the following command to generate a public and private key pair. Press enter and set no passphrase for keys.

```
ssh-keygen
```

- You will find two keys created in your ~/.ssh folder. ***id_rsa*** is your private key and ***id_rsa.pub*** is your public key. You can rename these keys if you wish to. I am going to keep them same. We need to place the ***id_rsa.pub*** in the Datanode so as to enable it to recognise your private key in your namenode.
- Download the public key to your local pc.

```
scp Namenode:~/.ssh/id_rsa.pub /C/Users/Ashwini/Desktop/
```

- execute the "exit" command in your bash shell to exit from the Namenode and return to your local Environment. Now ssh into each Datanode (1,2 and 3) and execute the following commands one by one to your Datanode.

```
scp /C/Users/Ashwini/Desktop/id_rsa.pub DataNode1:~/.ssh
```

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

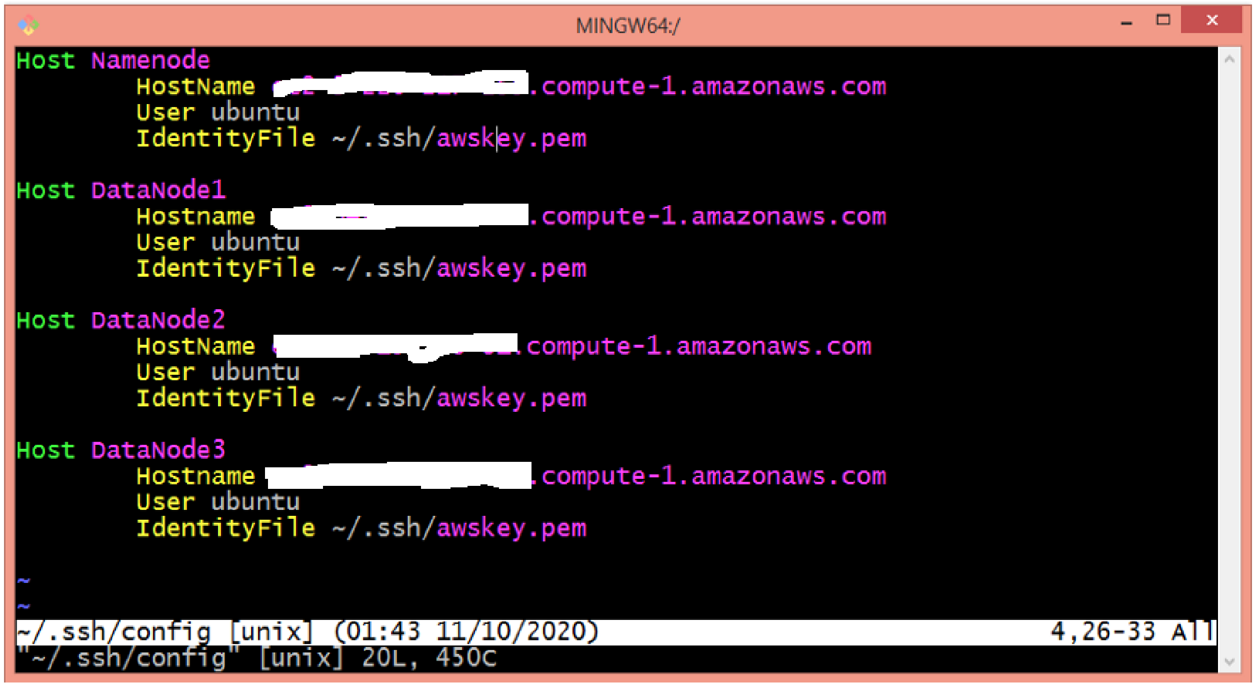
```
scp /C/Users/Ashwini/Desktop/id_rsa.pub DataNode2:~/.ssh
```

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

```
scp /C/Users/Ashwini/Desktop/id_rsa.pub DataNode3:~/.ssh
```

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

- The cat command will append your new public key to the authorized_keys file to enable authentication. Delete the public key from your local machine once you are done.
- Now ssh into your Namenode once again and check connectivity to each datanode by using ssh “public DNS address of DataNode”.
- Modify your ~/.ssh/config file in your local machine and add the following lines as shown below. Replace the machine addresses with your instances addresses.



```
Host Namenode
        HostName [redacted].compute-1.amazonaws.com
        User ubuntu
        IdentityFile ~/.ssh/awskey.pem

Host DataNode1
        Hostname [redacted].compute-1.amazonaws.com
        User ubuntu
        IdentityFile ~/.ssh/awskey.pem

Host DataNode2
        HostName [redacted]compute-1.amazonaws.com
        User ubuntu
        IdentityFile ~/.ssh/awskey.pem

Host DataNode3
        Hostname [redacted].compute-1.amazonaws.com
        User ubuntu
        IdentityFile ~/.ssh/awskey.pem

~
~
~/.ssh/config [unix] (01:43 11/10/2020)                    4,26-33 All
"~/.ssh/config" [unix] 20L, 450C
```

- Copy the config file to your namenode

```
scp ~/.ssh/config namenode:~/.ssh
```

- Don't forget to replace the "awskey.pem" of your local machine to the "id_rsa" key of your namenode machine. We will use this config file on Namenode to ssh into Datanodes.

Now you are ready with all your nodes.

STEP 10) NameNode and DataNode Specific Configurations

NAMENODE:

- Add the DataNode hostnames to `/etc/hosts`. You can get the hostname for each DataNode by entering `$ hostname`, or `$ echo $(hostname)` on each DataNode.

```
<namenode_IP> namenode_hostname

<datanode1_IP> datanode1_hostname

<datanode2_IP> datanode2_hostname

<datanode3_IP> datanode3_hostname

127.0.0.1 localhost
```

- Create Namenode directory for storing Namenode Metadata

```
sudo mkdir -p $HADOOP_HOME/data/hdfs/namenode
```

- Edit `hdfs-site.xml` file on NameNode :

```xml
<configuration>

 <property>

   <name>dfs.replication</name>

   <value>3</value>

 </property>

 <property>

   <name>dfs.namenode.name.dir</name>

   <value>file:///usr/local/hadoop/data/hdfs/namenode</value>

 </property>

</configuration>
```

- `dfs.replication` sets the amount of times each data block is replicated across the cluster. `dfs.namenode.name.dir` sets the directory for storing NameNode metadata

- Edit the `slaves` file in `HADOOP_CONF_DIR`with the hostnames of each datanode . Remove the line with `localhost`otherwise the NameNode would run as a DataNode too.

```
datanode1_hostname

datanode2_hostname

datanode3_hostname
```

- Edit the `masters` file and add the Namenode's hostname to the file. Typically we use the masters file to set up Secondary Namenode which usually runs on a different Namenode but for my project I am running it on the same instance.
- Finally on the NameNode, change the owner of `HADOOP_HOME` to `ubuntu`

```
sudo chown -R ubuntu $HADOOP_HOME
```

- This is because you will login as ubuntu user and you need Hadoop to follow your commands.

DATANODES:

- Create Datanode Directory for storing DataNode Metadata.

```
sudo mkdir -p $HADOOP_HOME/data/hdfs/datanode
```

- Edit `hdfs-site.xml` on each DataNode:

```
<configuration>

 <property>

   <name>dfs.replication</name>

   <value>3</value>

 </property>

 <property>

   <name>dfs.datanode.data.dir</name>
   <value>file:///usr/local/hadoop/data/hdfs/datanode</value>
```

```
 </property>
```

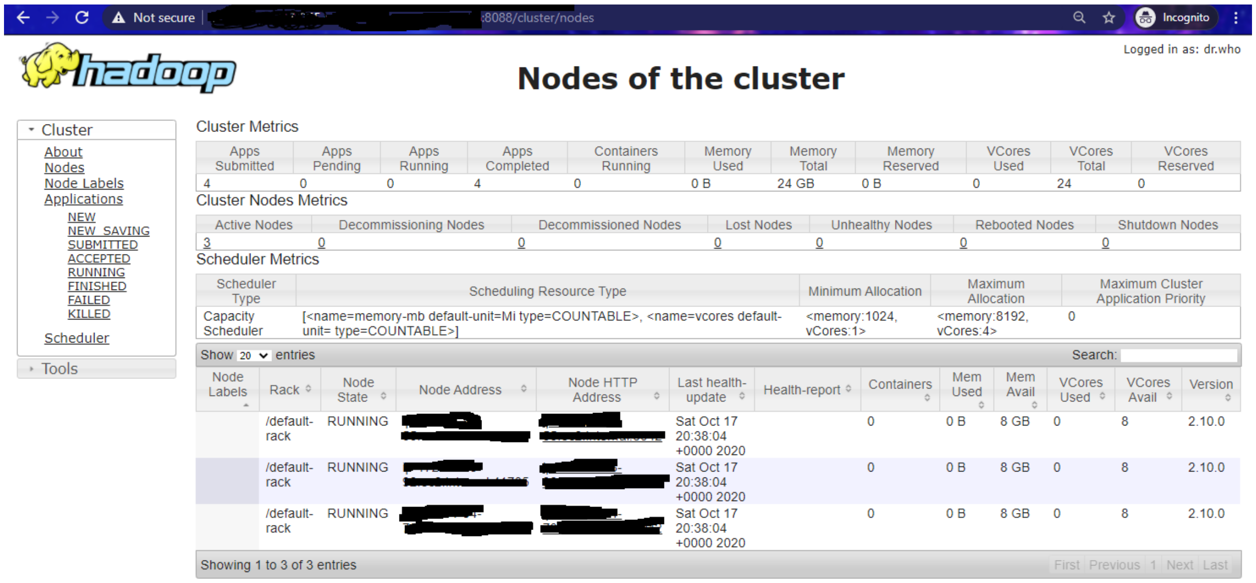
```
</configuration>
```

- Again, this sets the directory where the data is stored on the DataNodes. And again, create the directory on each DataNode. Also change the owner of the Hadoop directory.

```
sudo chown -R ubuntu $HADOOP_HOME
```

And Voila!! You have your Hadoop cluster ready.

You can navigate to`<namenode public DNS>:8088` and you should see the Hadoop WebUI.



Logged in as: dr.who

Nodes of the cluster

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved
4	0	0	4	0	0 B	24 GB	0 B	0	24	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Shutdown Nodes
3	0	0	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Cluster Application Priority
Capacity Scheduler	[<name=memory-mb default-unit=Mi type=COUNTABLE>, <name=vcores default-unit= type=COUNTABLE>]	<memory:1024, vCores:1>	<memory:8192, vCores:4>	0

Show 20 entries Search:

Node Labels	Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Mem Used	Mem Avail	VCores Used	VCores Avail	Version
	/default-rack	RUNNING	[redacted]	[redacted]	Sat Oct 17 20:38:04 +0000 2020		0	0 B	8 GB	0	8	2.10.0
	/default-rack	RUNNING	[redacted]	[redacted]	Sat Oct 17 20:38:04 +0000 2020		0	0 B	8 GB	0	8	2.10.0
	/default-rack	RUNNING	[redacted]	[redacted]	Sat Oct 17 20:38:04 +0000 2020		0	0 B	8 GB	0	8	2.10.0

Showing 1 to 3 of 3 entries