# Table of Contents

[Revision and Signoff Sheet 2](#_Toc51685885)

[Change Record 2](#_Toc51685886)

[Reviewers 2](#_Toc51685887)

[Table of Contents 3](#_Toc51685888)

[Purpose 4](#_Toc51685889)

[Description 4](#_Toc51685890)

[Deployment Steps 4](#_Toc51685891)

[Details 4](#_Toc51685892)

[Attach Secondary ENI to cluster Nodes 5](#_Toc51685893)

[Conclusion : 20](#_Toc51685894)

# Purpose

Best Practices for configuring a three node Casandra cluster using DataStax and ensuring cluster resilience with minimal effort.

# Description

[Apache Cassandra](http://cassandra.apache.org/) is a commonly used, high performance NoSQL database. AWS customers that currently maintain Cassandra on-premises may want to take advantage of the scalability, reliability, security, and economic benefits of running Cassandra on [Amazon EC2](https://aws.amazon.com/ec2/).

In this post, we will outline the deployment steps of 3 node Cassandra cluster using DataStax and also ensure cluster is able to recover from planned /un planned outages with minimal effort.

# Deployment Steps

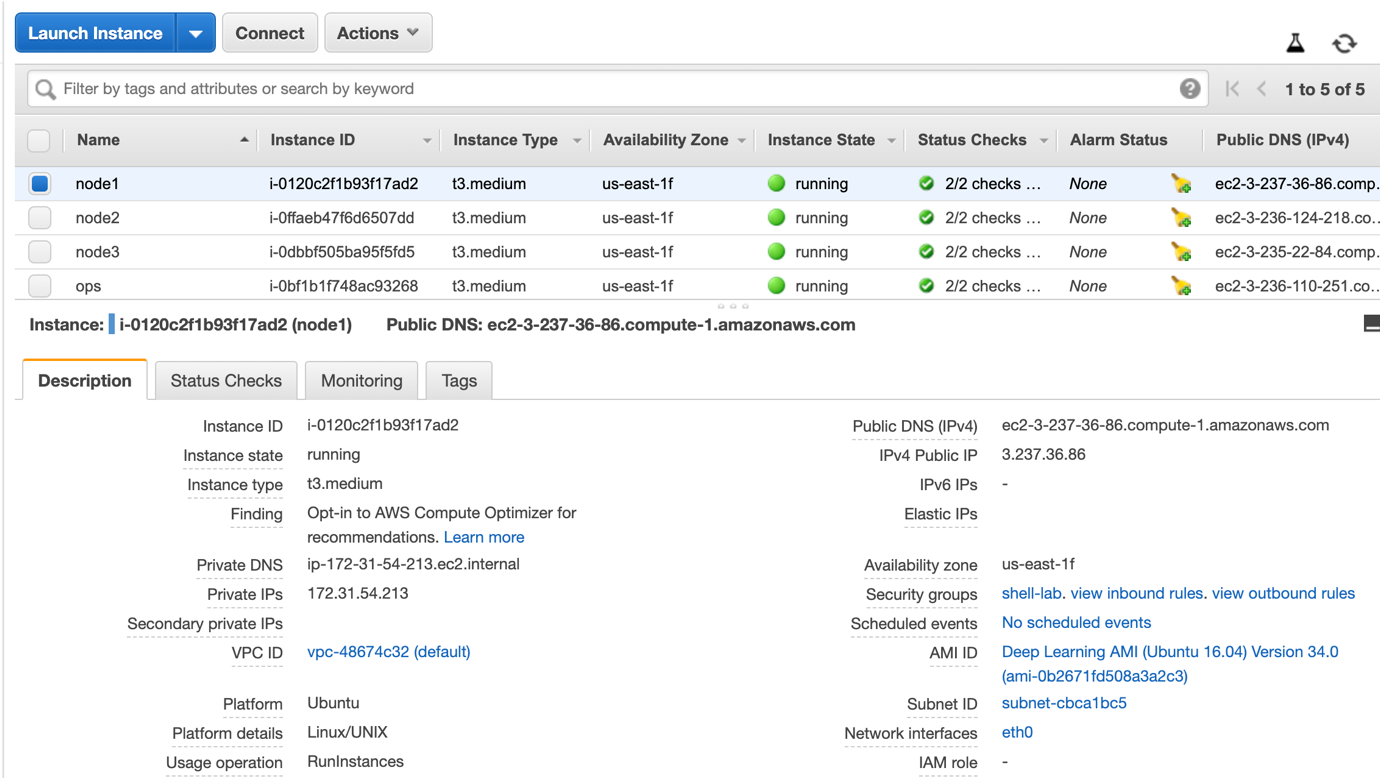
**Outline**

* Launch EC2 nodes to configure Casandra Cluster
* Ensure Ssecurity group allows cluster communication to happen
* Add secondary ENI to each node and configure IP manually to each ENI
* Configure IP and routes in each nodes to allow communication to happen via secondary ENI
* Install and configure Casandra cluster using DataStax
* Create AMI for cluster node
* Incase of node failure , launch new instance from AMI and attach the secondary ENI to new node to recover from Failure.

## Details

Cluster Deployment using DataStax ( DataStax is a hybrid database-as-a-service built on Apache Cassandra™) DataStax Enterprise Server version : 5.11.1

1. Launch 4 **Ubuntu 16.04** nodes. 3 for cluster and 4th one will act as a cluster management node carrying DSE OpsCenter 6.1.



1. Ensure nodes can communicate to each other over required Database ports.
2. Install DSE OpsCenter 6.1 from the Debian package on 4th nodes using this [Link](https://docs.datastax.com/en/opscenter/6.1/opsc/install/opscInstallDeb_t.html)

Caveat:-

* Casandra cluster seed node IP address change cause whole ring has to be rebalanced.

**Casandra Networking** : Cassandra generates a universal unique identifier (UUID) for each node based on IP address for the instance. This UUID is used for distributing vnodes on the ring.In the case of an AWS deployment, IP addresses are assigned automatically to the instance when an EC2 instance is created. With the new IP address, the data distribution changes and the whole ring has to be rebalanced. This is not desirable.

Workaround:-

* Use primary nic with manual ip configuration during instance launch
* Use secondary nic with static private ip

### Attach Secondary ENI to cluster Nodes

Use a [secondary elastic network interface](http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-eni.html#scenarios-enis) with a fixed IP address. Before swapping an EC2 instance with a new one, detach the secondary network interface from the old instance and attach it to the new one.

This way, the UUID remains same and there is no change in the way that data is distributed in the cluster. But we might have one more challenge as Asymmetric Routing before we moving forward.

Steps to configure a secondary ENI with static IP

Asymmetric Routing

If you attach the second ENI and bring it up (with ifup) in Linux after adding to /etc/network/interfaces, your network will very likely be performing asymmetric routing. Try and ping the Elastic IP of eth1, you get no response. This is because the response packets leaving the instance do not get sent out via the correct gateway.

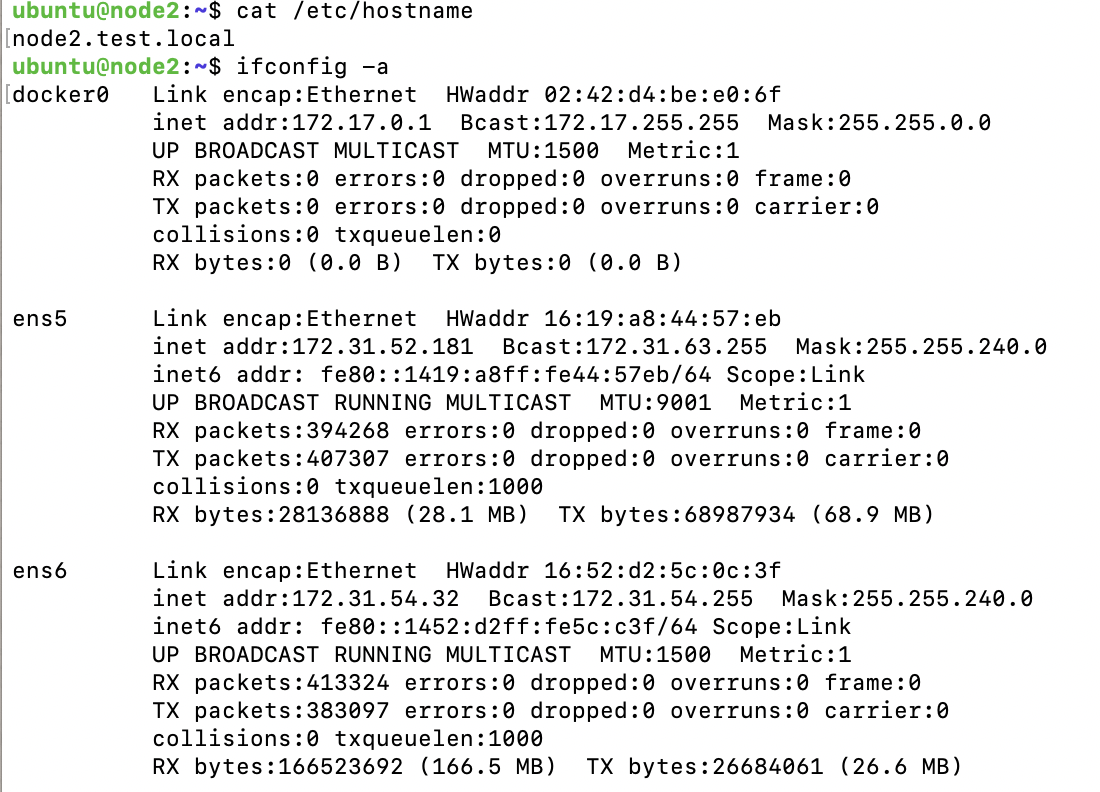
Asymmetric routing is explained in depth in this article <http://www.linuxjournal.com/article/7291>

Route configuration with additional ENIs

The fix is to add additional routes for the new ENIs. This guide assumes that so far you have followed this documentation for adding a second ENI

<http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-eni.html#attach_eni_launch>

We’re assuming the instance has an interface eth0 with the private address 172.31.52.181 from a 172.31.48.0/20 subnet and we want to add an ENI using same subnet 172.31.48.0/20 with an IP address of 172.31.54.32



The /etc/network/interfaces file should look like this after adding routes and eth1 (ens6 in this case):

**Then bring up eth1 interface:**

#ubuntu@node2:~$ sudo ifup eth1

**Let’s check the route:**

ubuntu@node2:~$ ip route show  
default via 172.31.48.1 dev ens5   
172.17.0.0/16 dev docker0  proto kernel  scope link  src 172.17.0.1 linkdown   
172.31.48.0/20 dev ens5  proto kernel  scope link  src 172.31.55.56

There is one default gateway at 172.31.48.1 (which is bound to VPC the internet gateway) and will route any traffic from eth0 ( ens5 in our case). However any traffic from eth1 ( ens6 in our case ) with a destination outside of 172.31.48.0/20 will be dropped, so we need to re-configure the routing to the default gateway for the 172.31.48.0/20 subnet.

**Firstly, add an entry “2 eth1\_rt” to the route table:**

#ubuntu@node2:~$ sudo bash -c "echo '2 eth1\_rt' >> /etc/iproute2/rt\_tables"

**Next we need to add a default route to the gateway for eth1:**

#ubuntu@node2:~$ sudo ip route add default via 172.31.54.32 dev ens5 table eth1\_rt

**Verify that the route is added:**

#ubuntu@node2:~$ ip route show table eth1\_rt  
default via 172.31.54.32 dev ens6

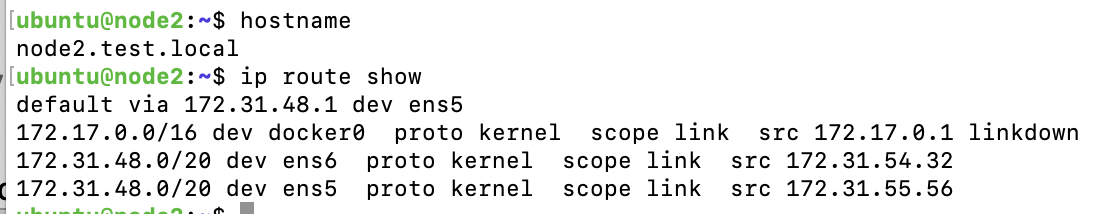
**Finally we need to add a rule which will tell the route table to route traffic with a source of 10.0.2.0/24 via the rt\_eth1 table:**

ubuntu@node2:~$ sudo ip rule add from 172.31.48.0/20 lookup eth1\_rt prio 1000

**Verify that the rule is added:**

ubuntu@node2:~$ ip rule show  
0: from all lookup local   
1000: from 172.31.54.32/24 lookup eth1\_rt   
32766: from all lookup main   
32767: from all lookup default

**Check the route again:**



Now from your machine, try and ping the private IP associated with eth1 (end6 in our case) and it should now work, asymmetrical routing has been fixed !

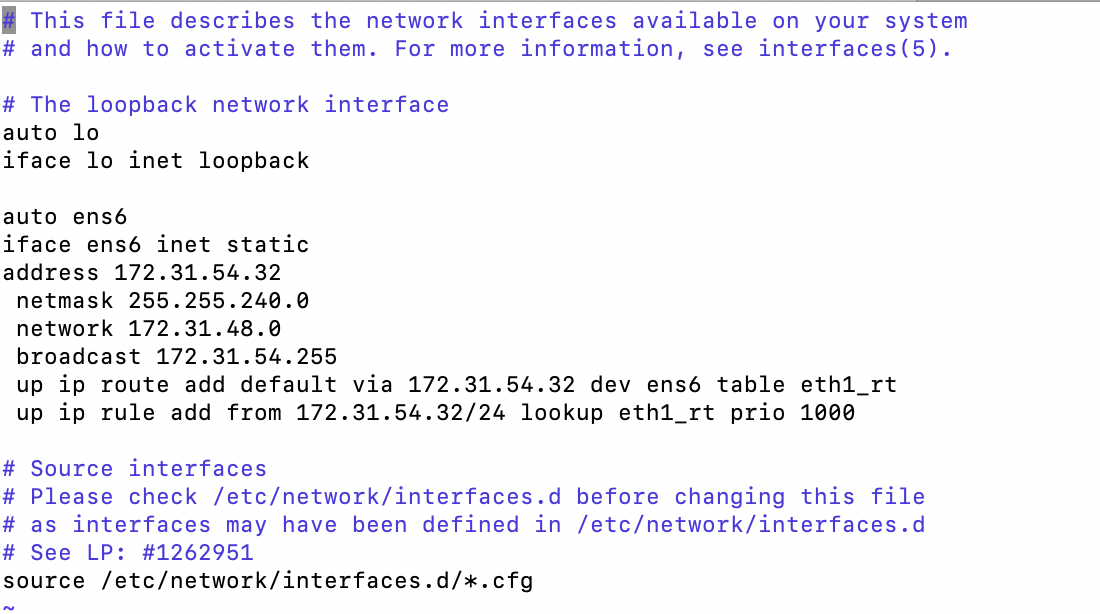
If you wish to associate an private IP from different subnet to eth1 (end6 in our case), just replace the gateway and subnet values accordingly.

To make the route changes permanent so that they can survive a reboot, add them to the interfaces file:

**Save**

Shar

ubuntu@node2:~$ sudo vi /etc/network/interfaces

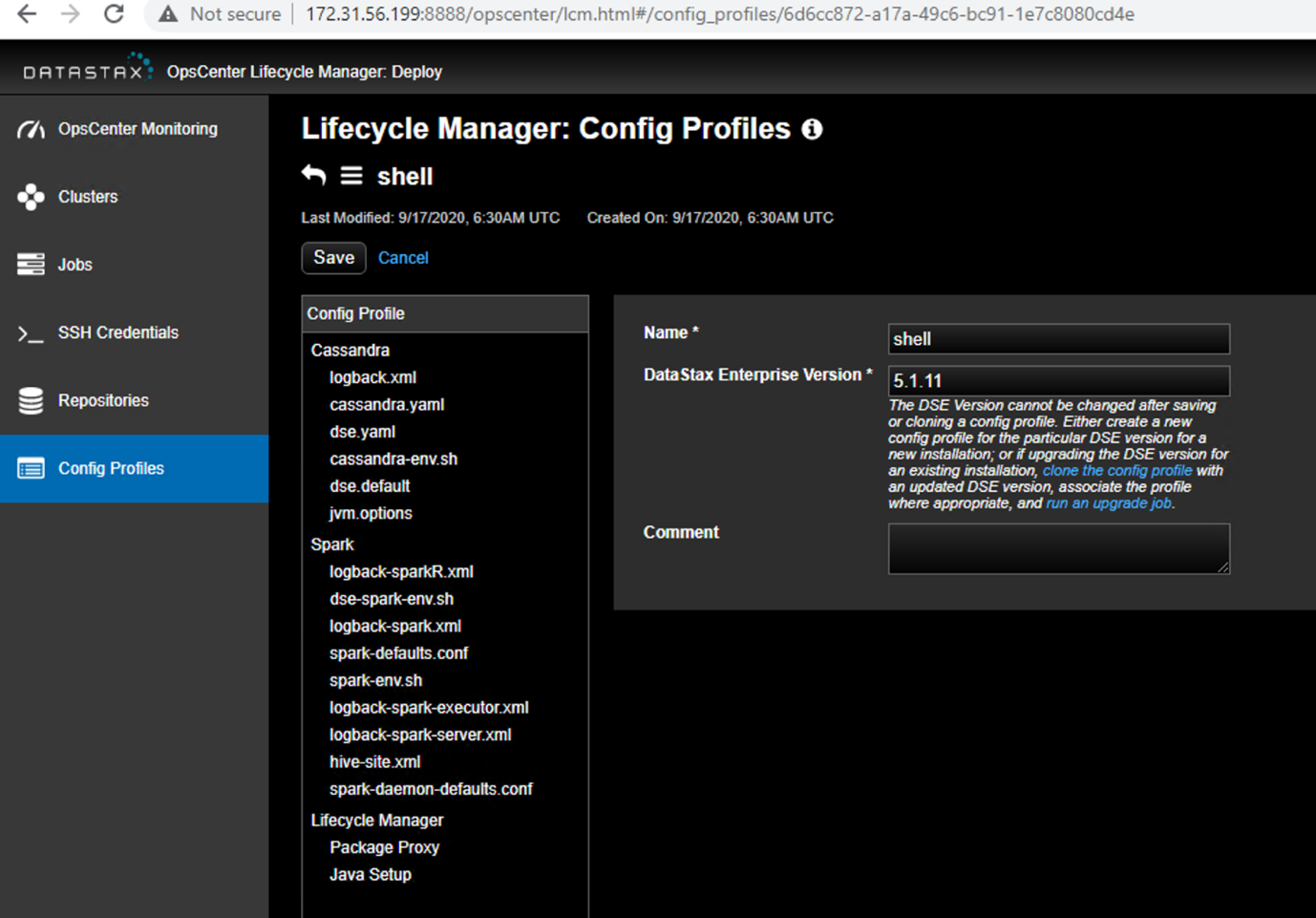


Also configure the DNS records pointing to IP of secondary ENI and update the hostname in the nodes accordingly.

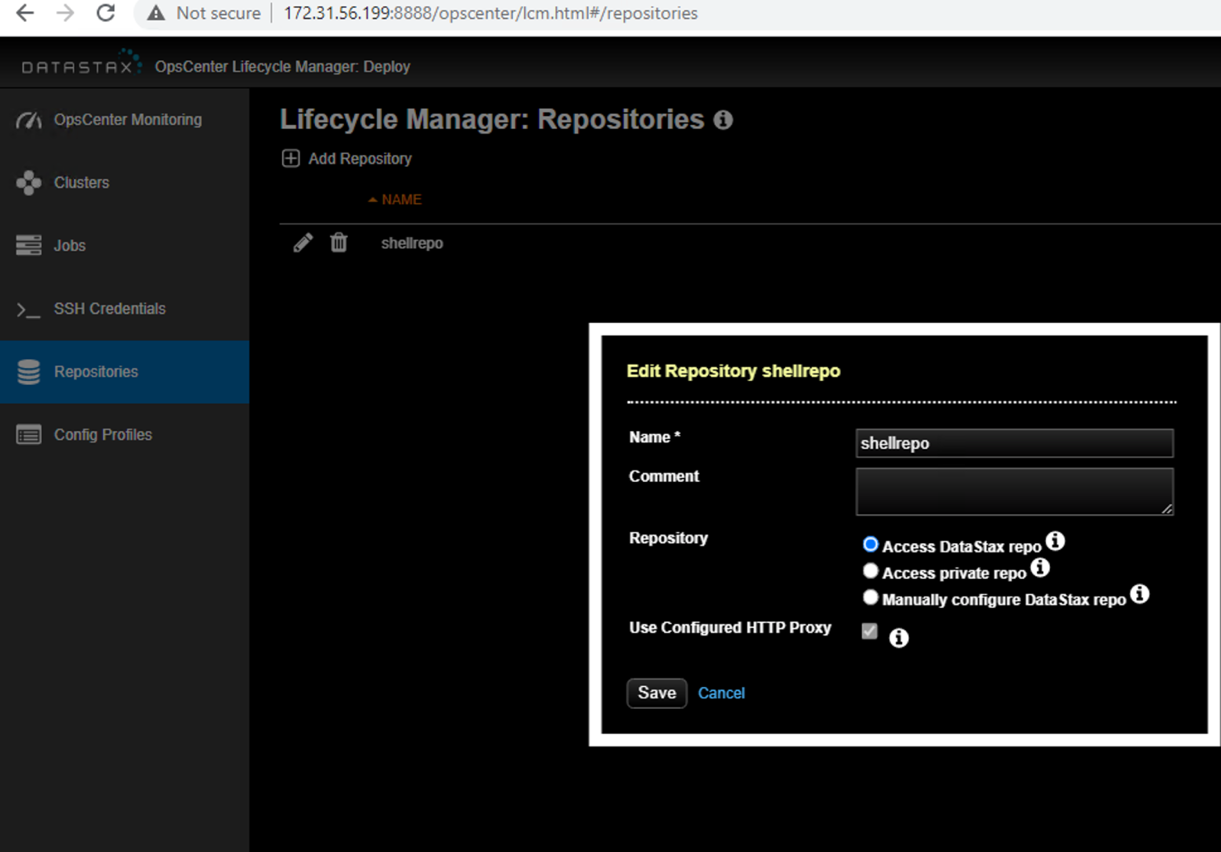


Now let’s go back to configuring Cassandra cluster nodes.

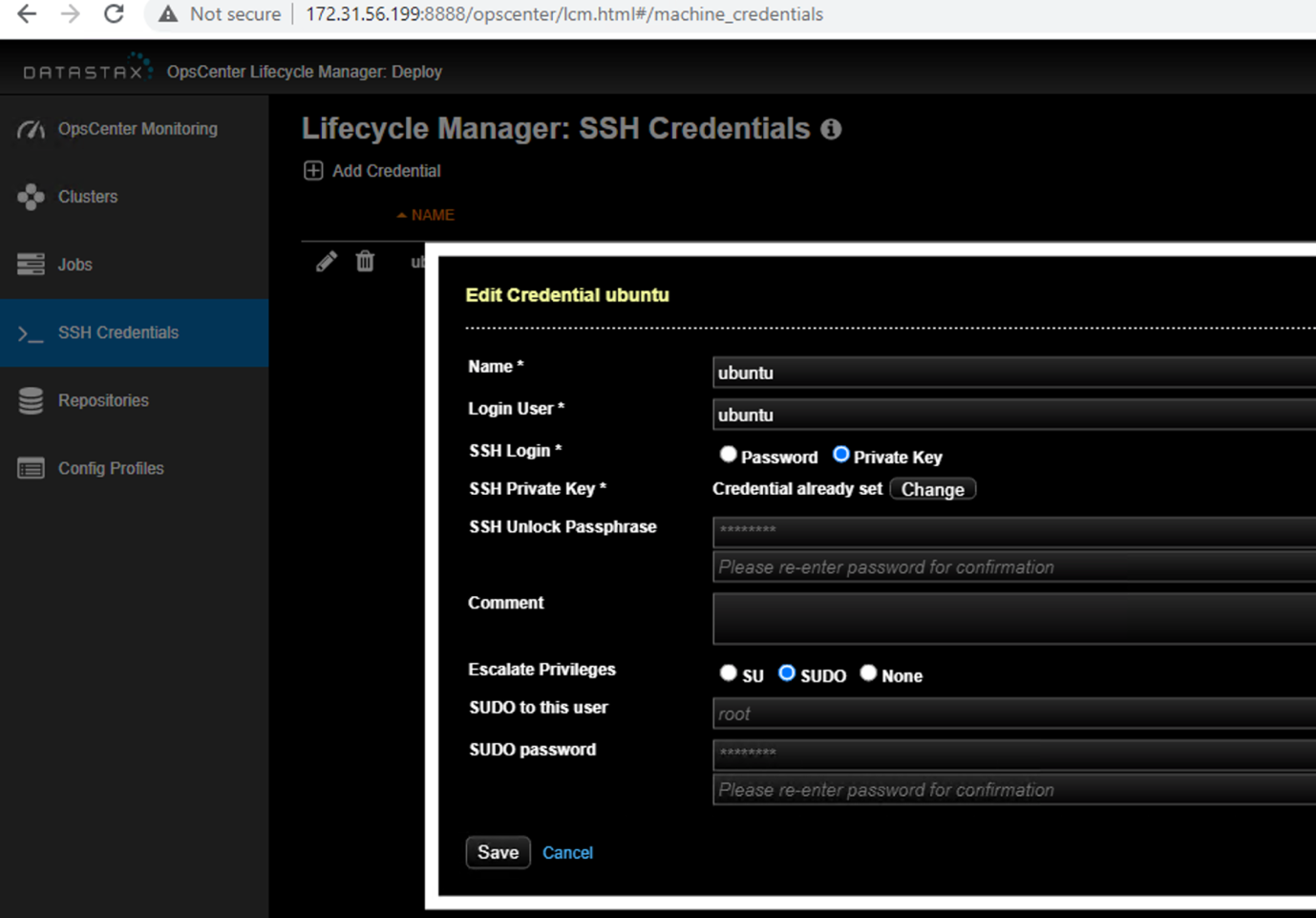
1. Open opscenter using [http://***opscenter-host***:8888/](http://opscenter-host:8888/)
2. Configure Profile



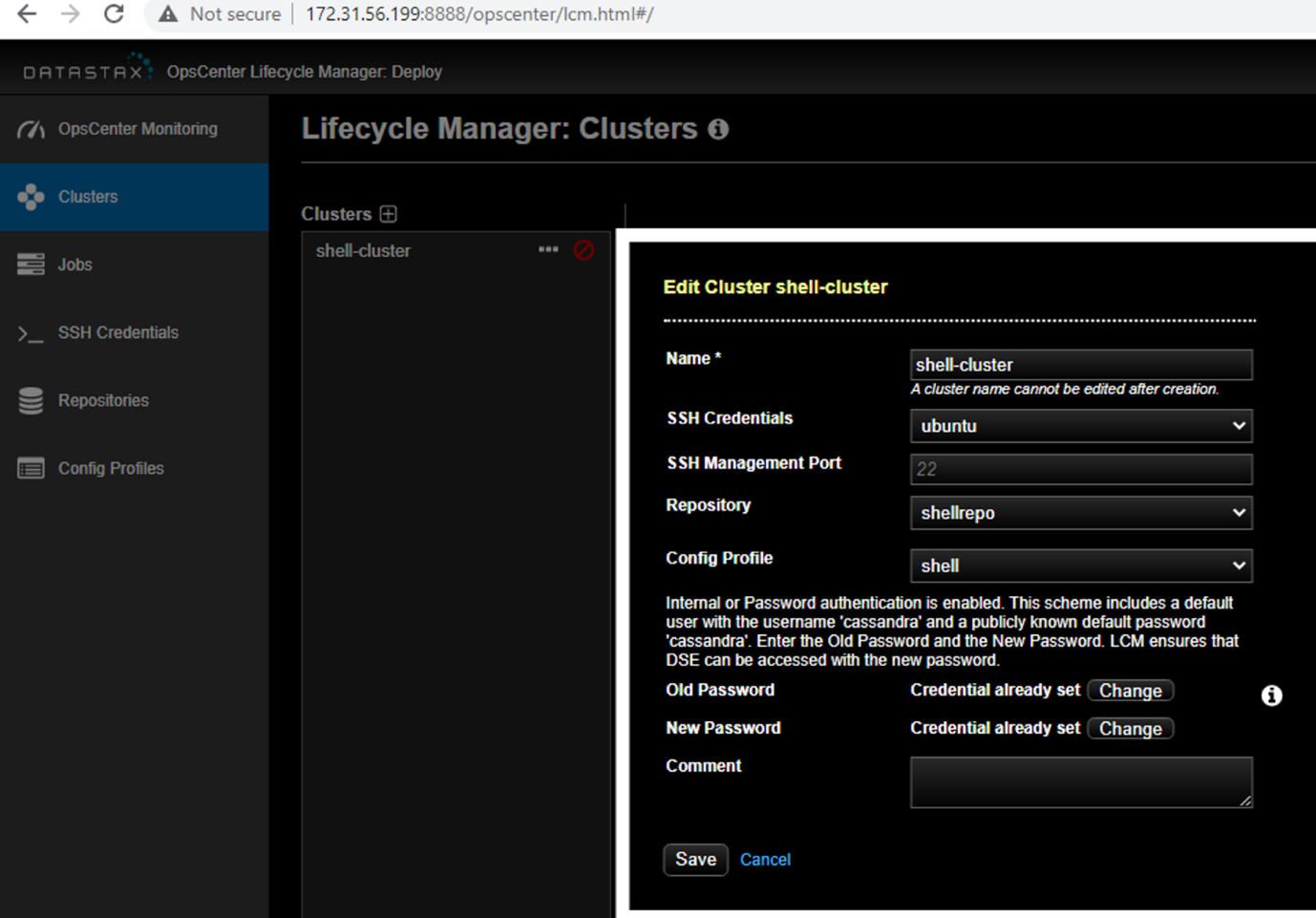
1. Configure Repositories



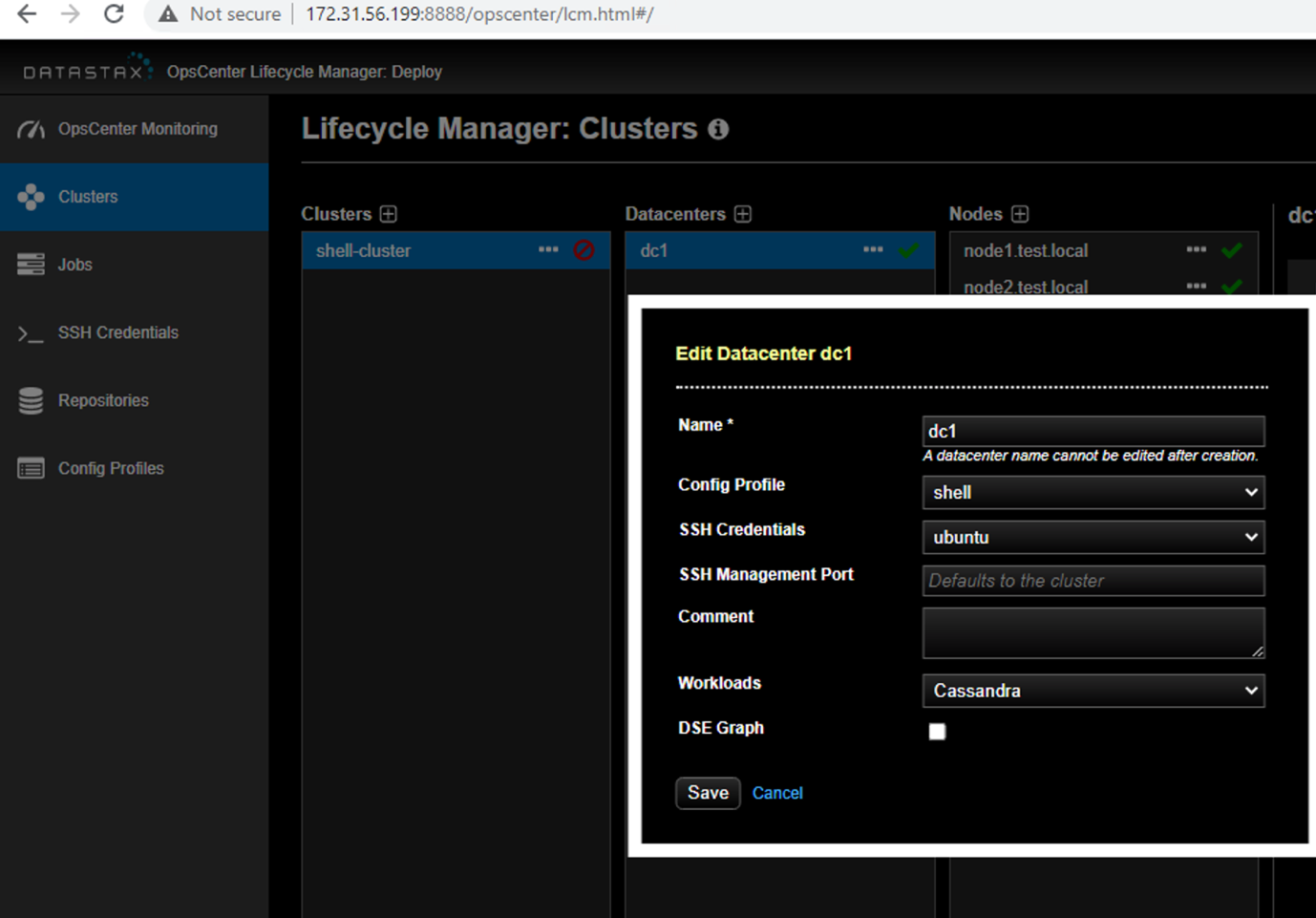
1. Configure SSH credentials



1. Create Cluster



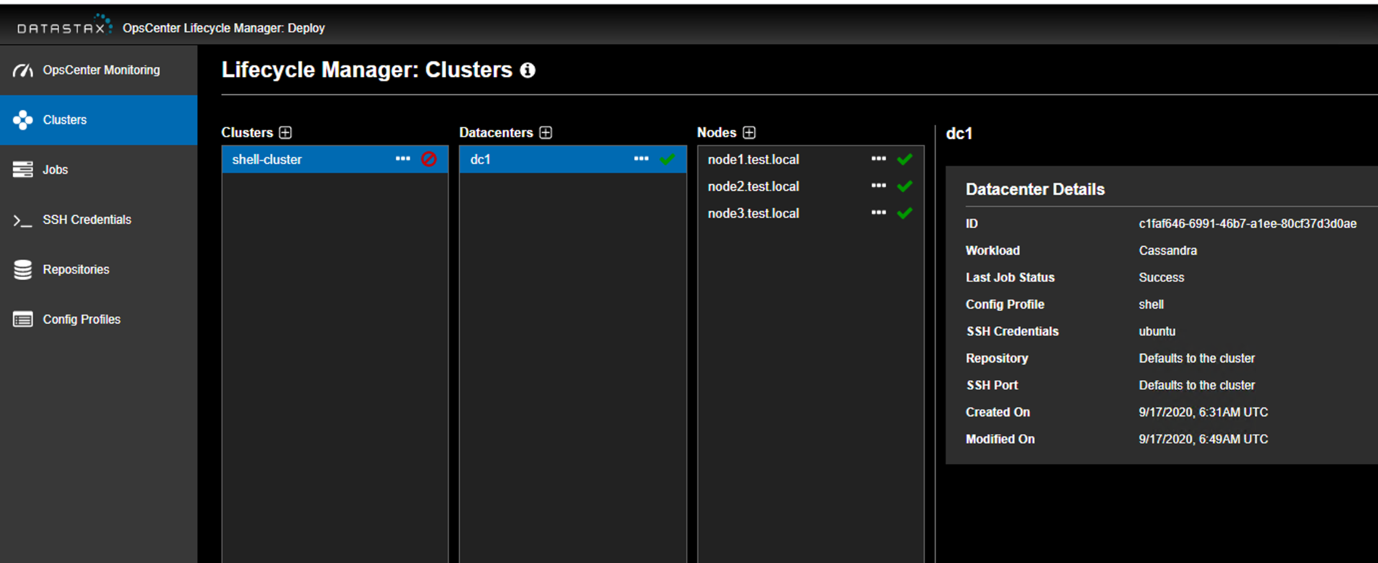
1. Followed by Datacenter



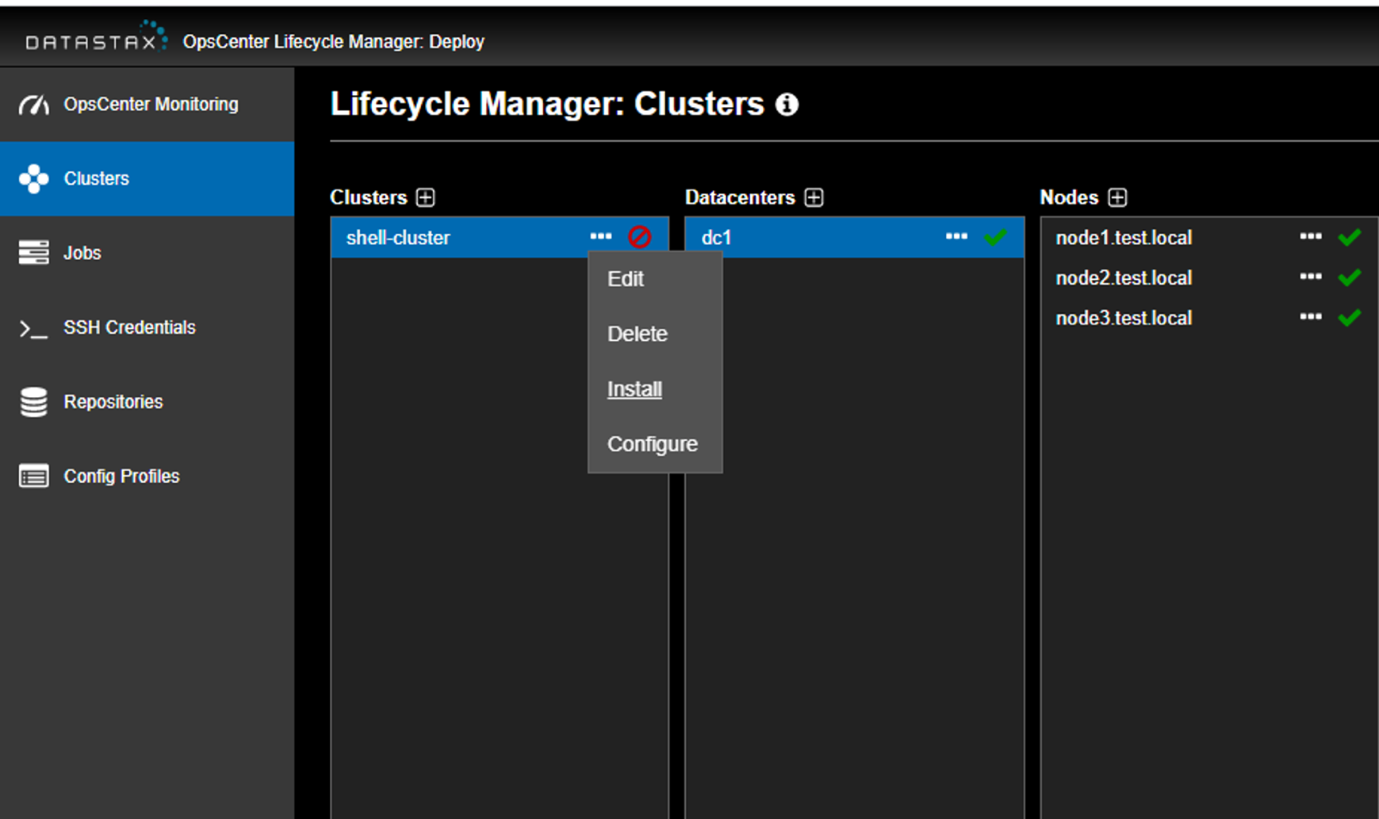
1. Finally add nodes to Datacenter



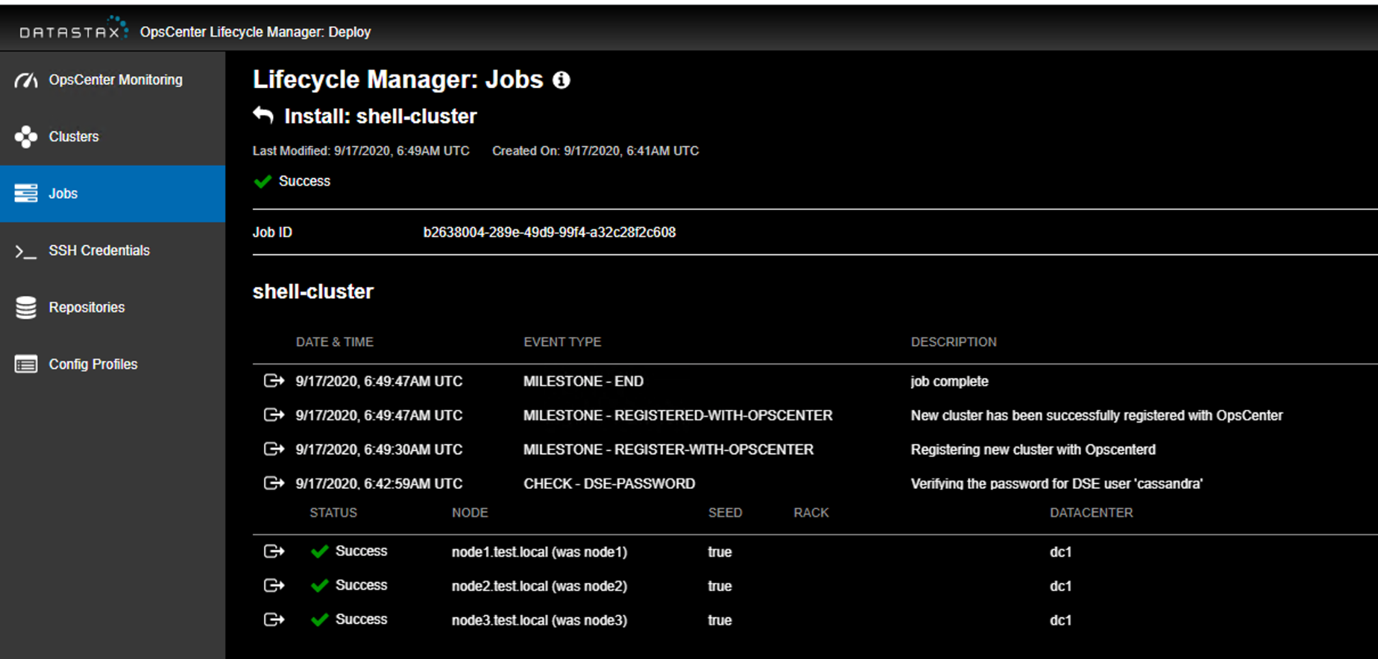
1. Ensure nodes are added



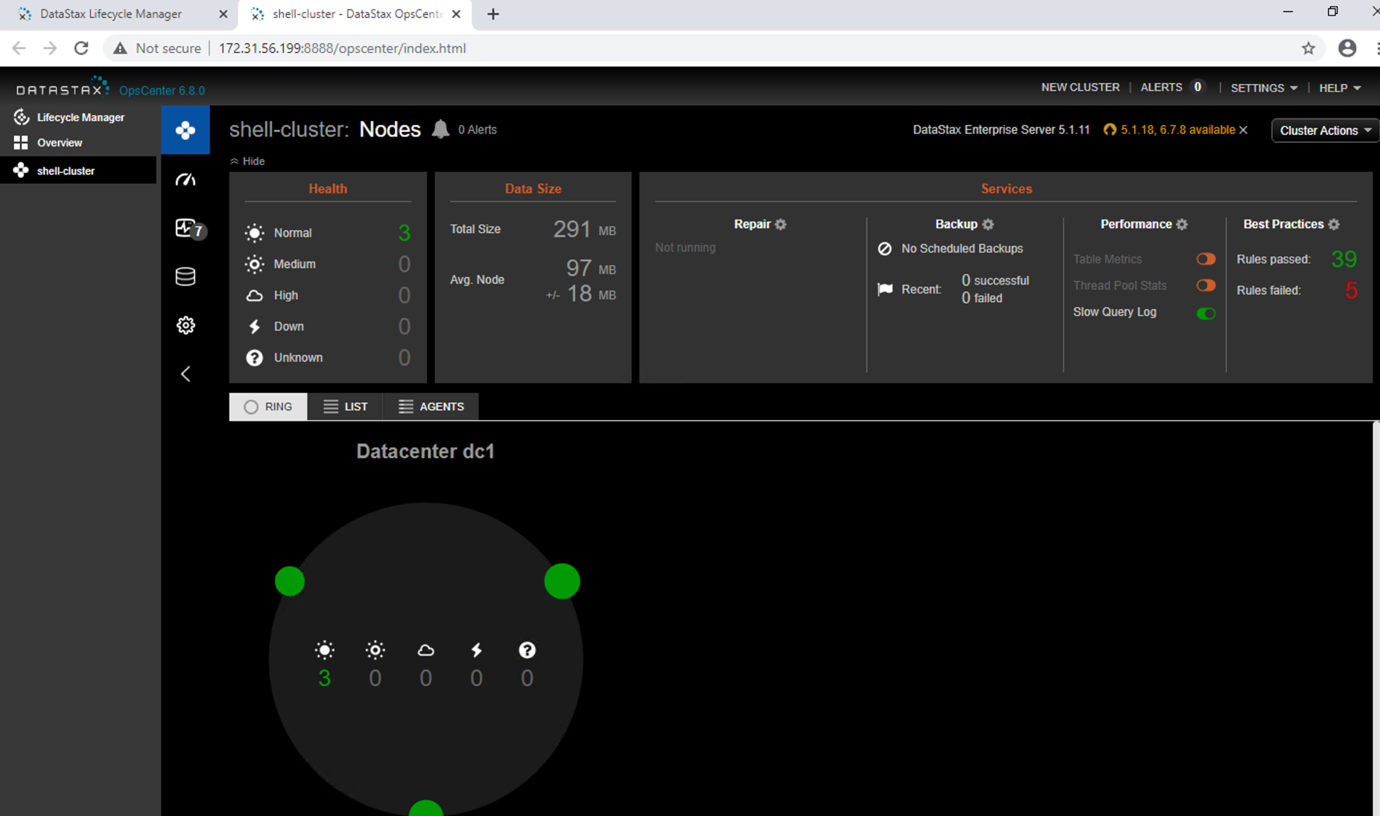
1. Post cluster , Datacenter, Nodes, we will install Cassandra package on the nodes

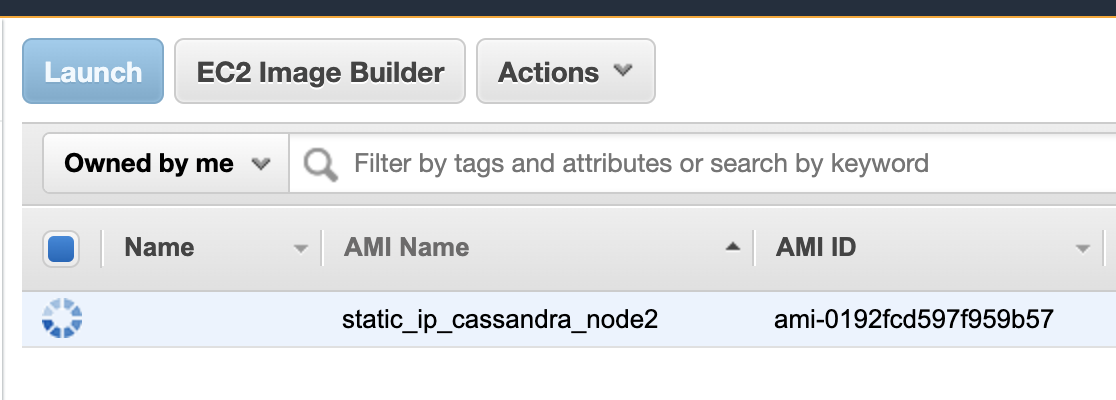


1. Check the job status from Jobs section to ensure binaries are installed and cluster is configured

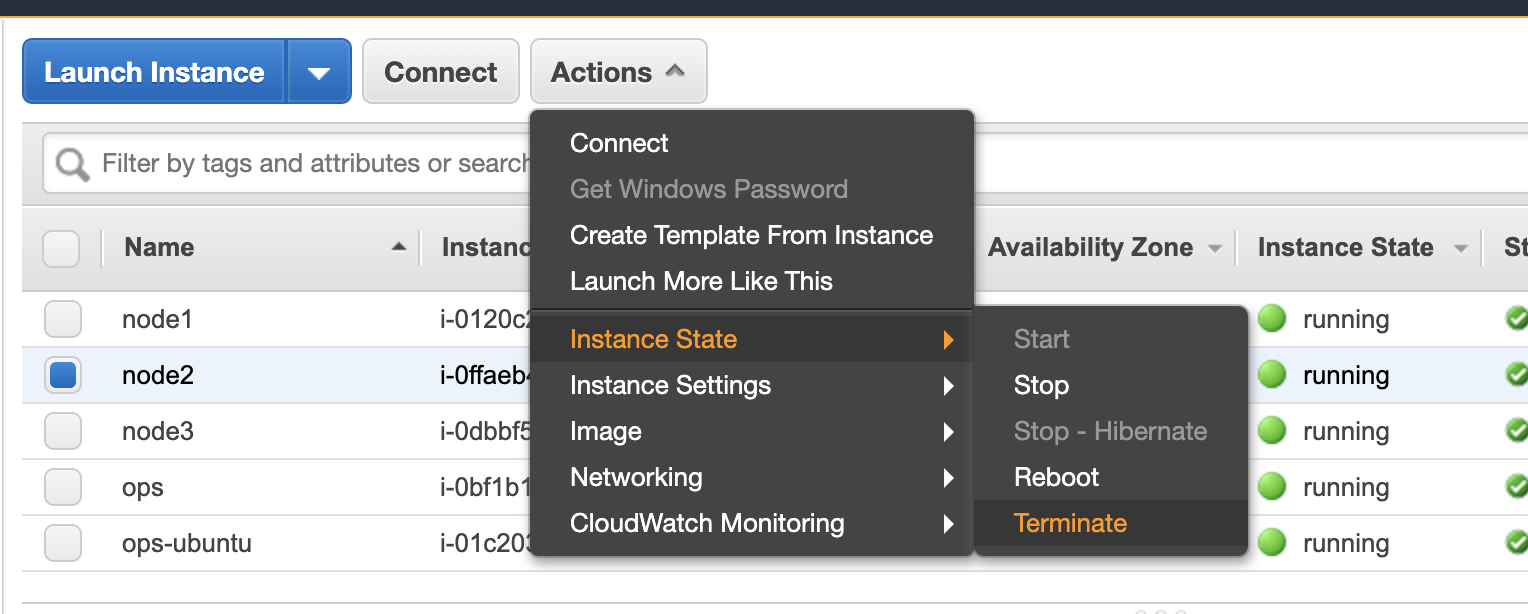


1. Once cluster is configured and all 3 nodes seem healthy , Create AMI from the healthy instance.

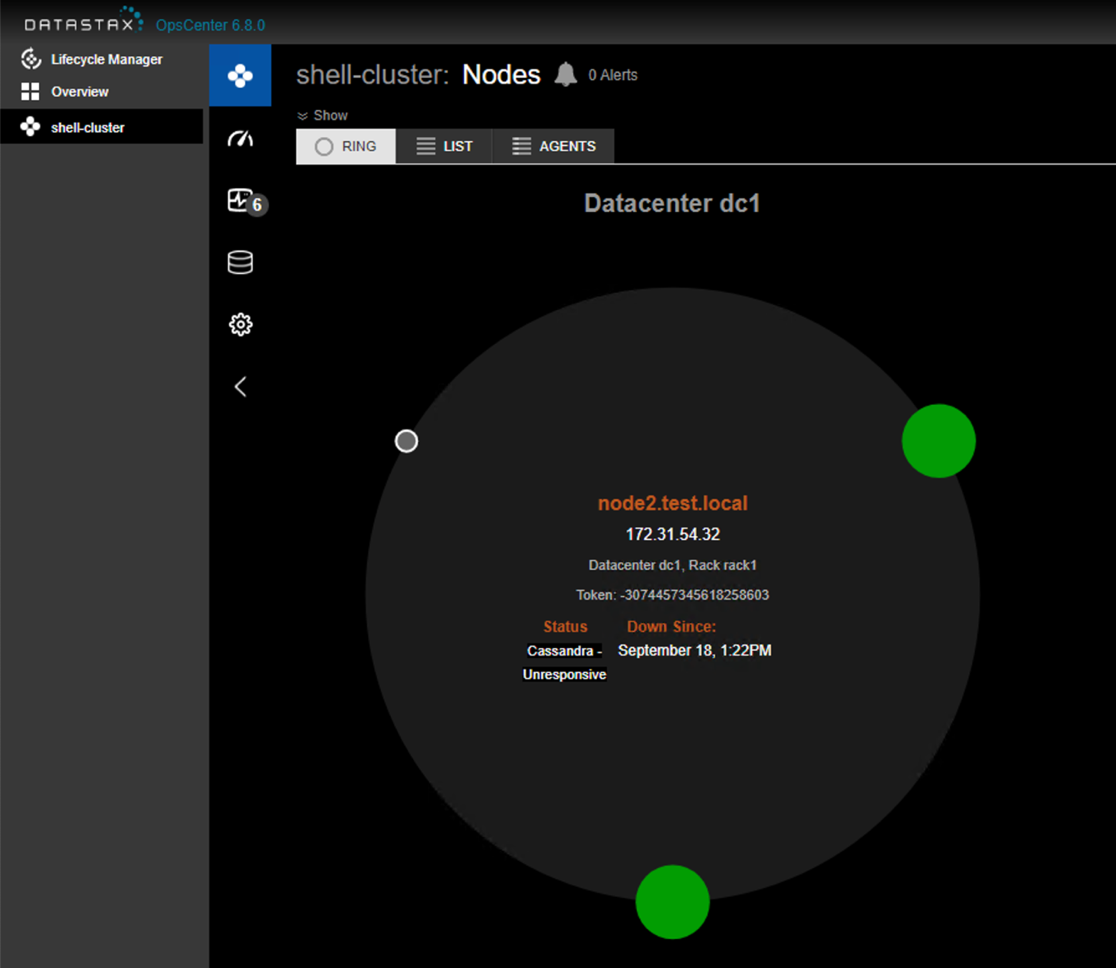




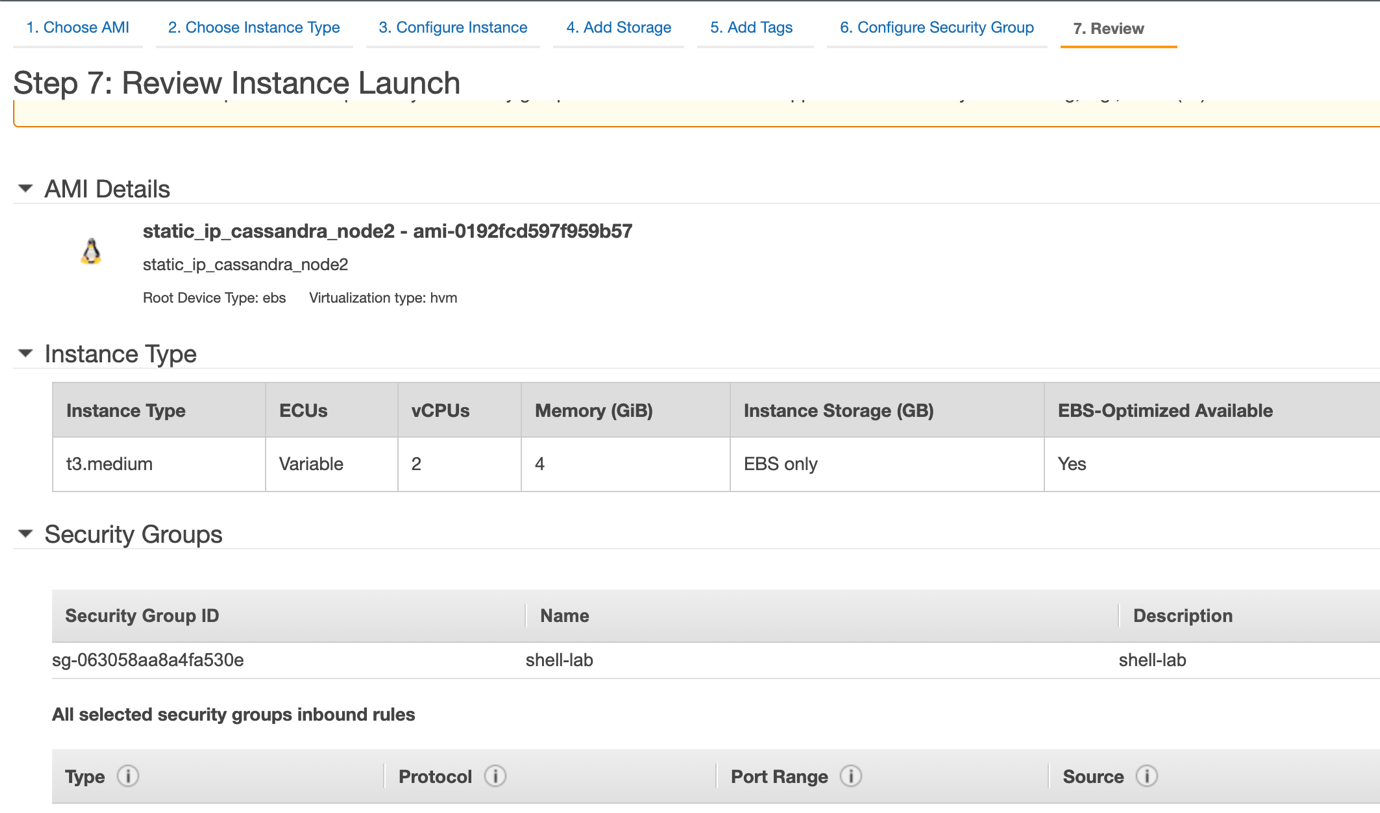
1. Now to simulate failure of cluster node, Lets terminate one of the node ( node 2 in our case)



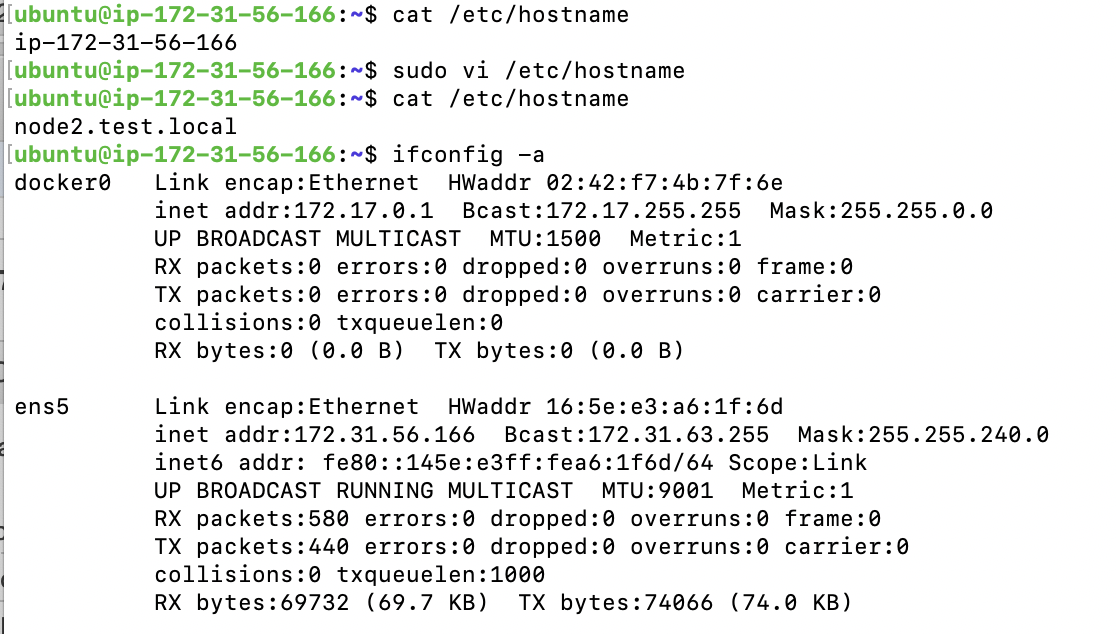
1. Cluster will immediately discover the failed node ,but will continue to function.



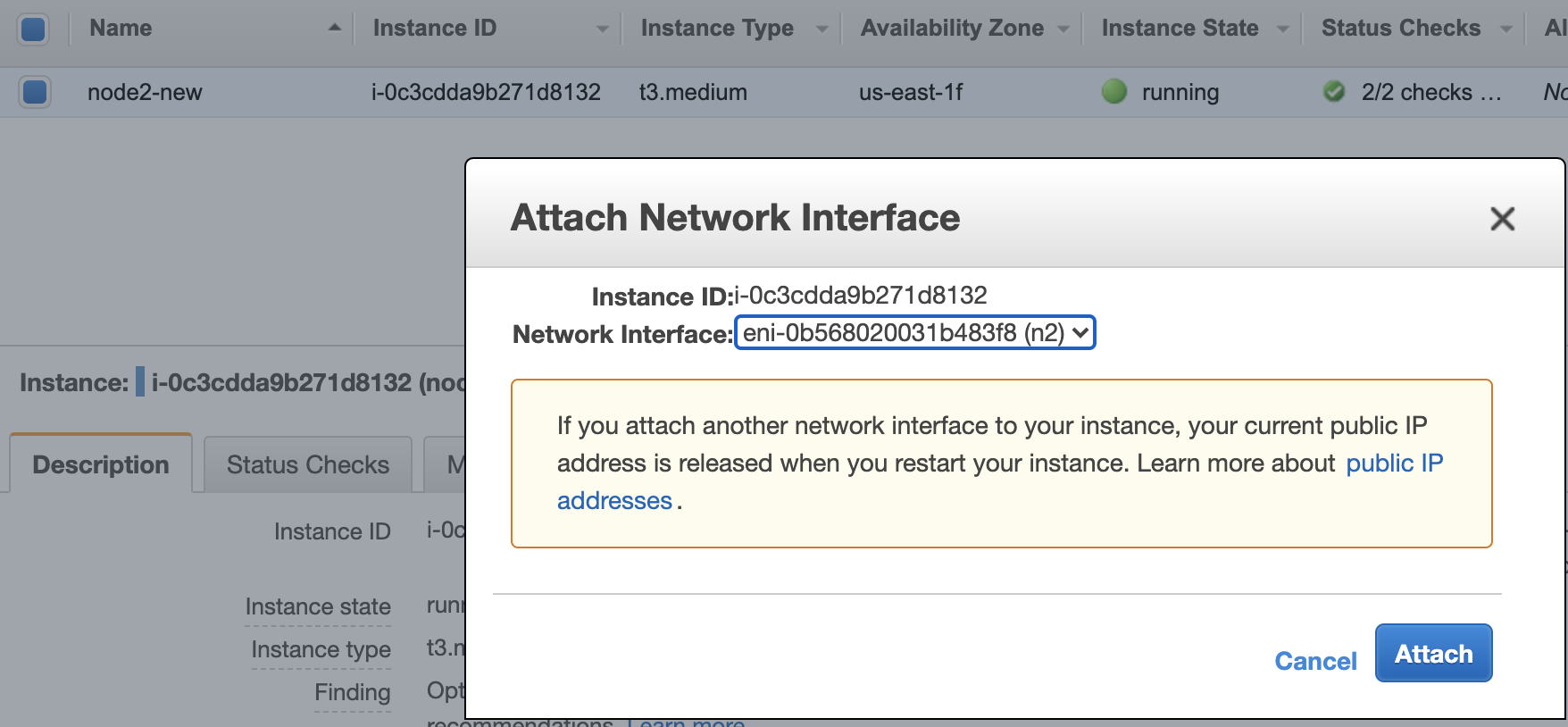
1. To recover from the failure Let’s Launch a new node from previously created AMI

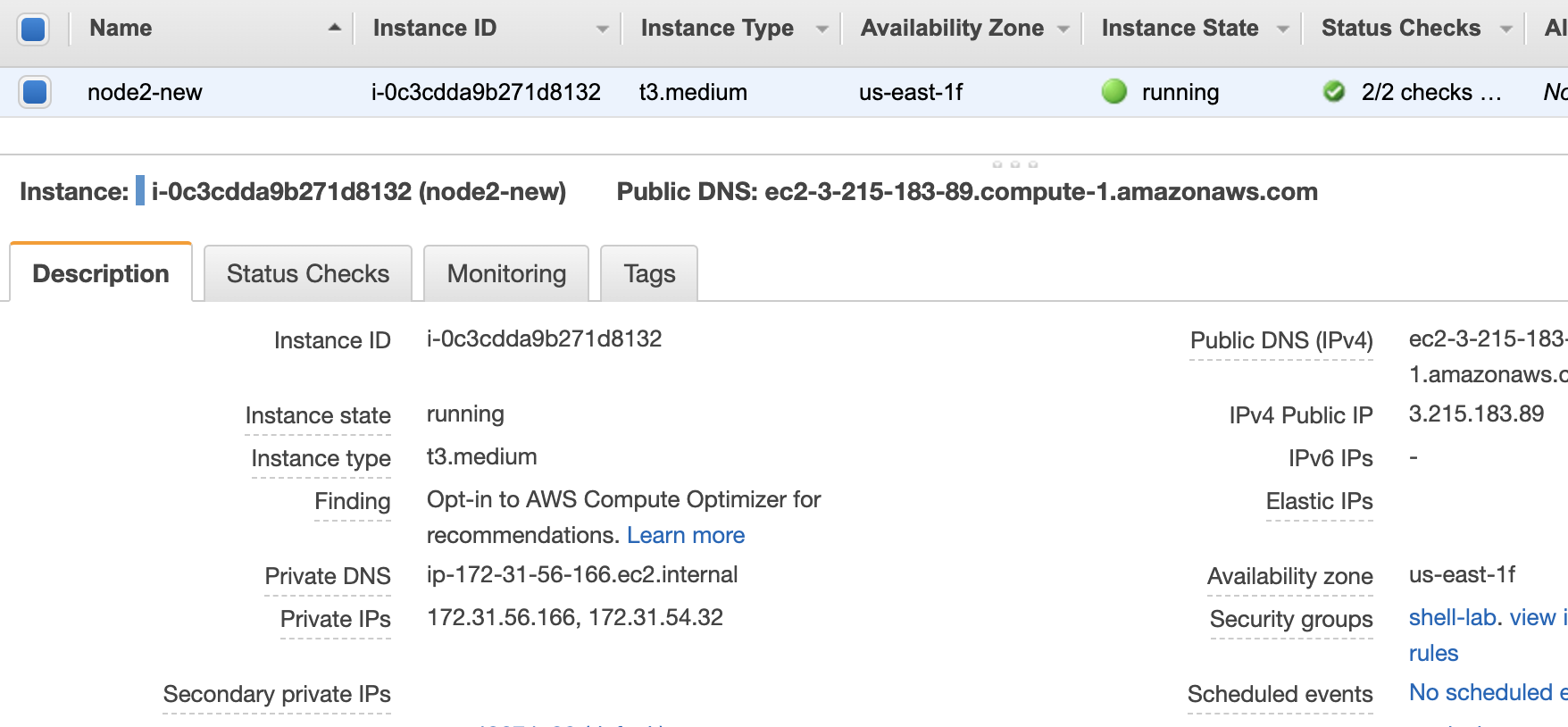


1. Login to new node and update hostname

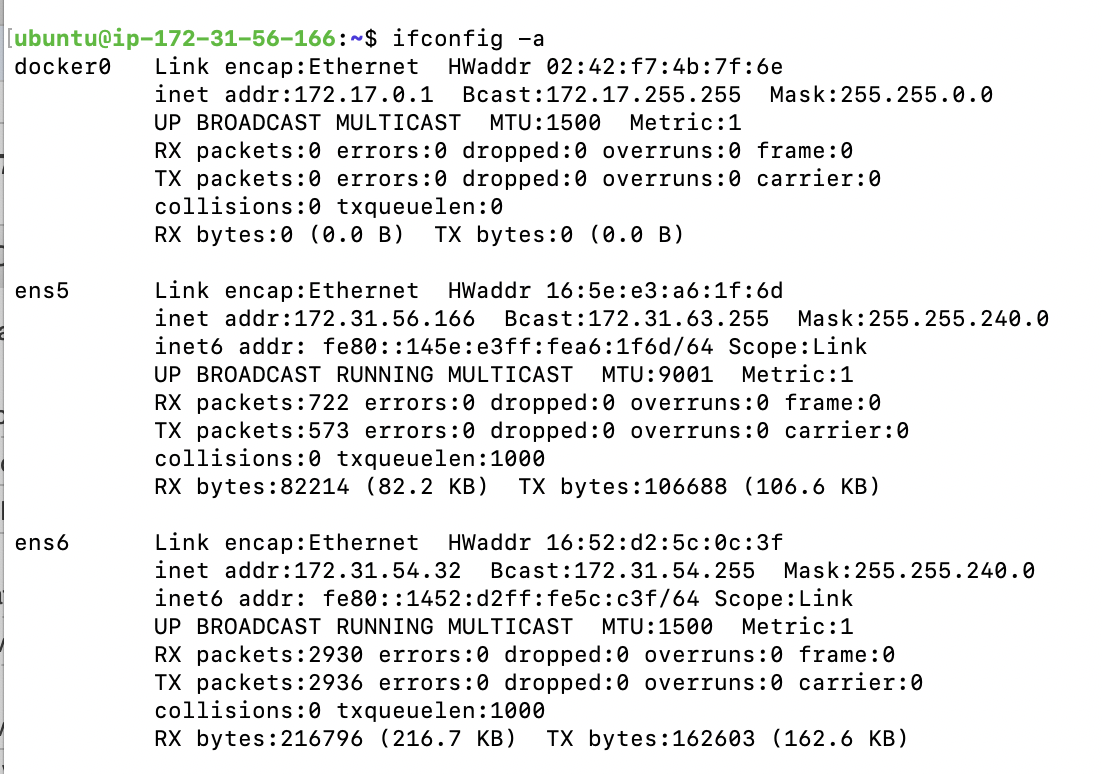


1. Attach secondary interface previously attached to failed node (with old static IP)

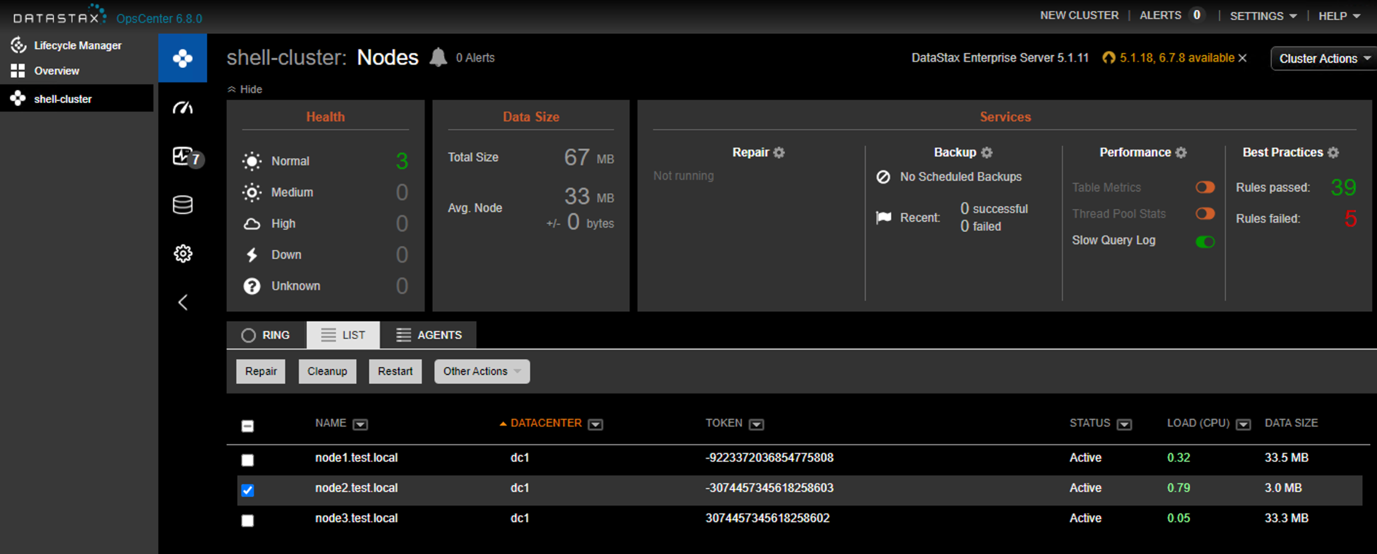


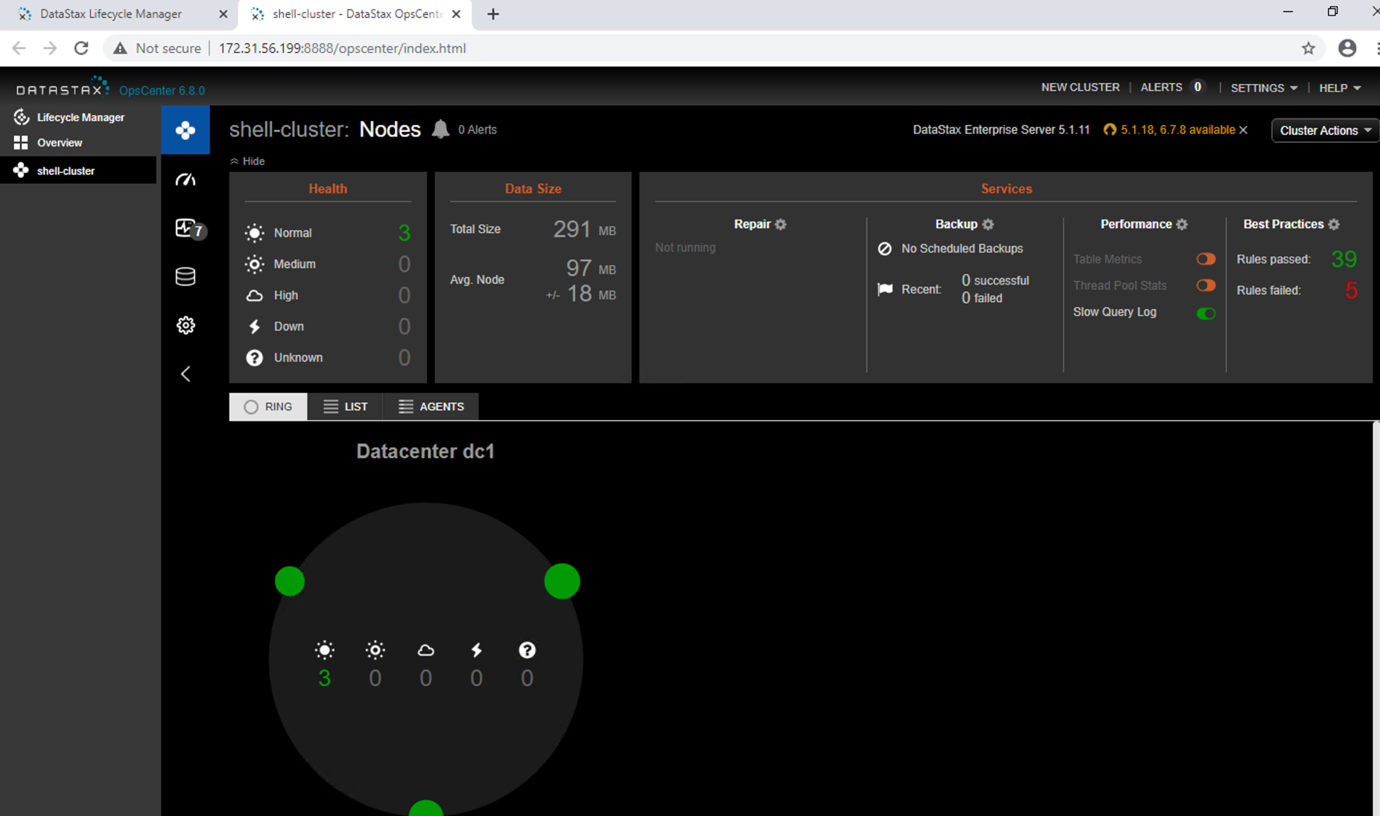


1. Ensure the secondary Eni is up and reachable



1. Cluster should automatically detect the replacement node ( because of old IP) and without the need of rebalance, the cluster is back to original shape.





# Conclusion :

We have successfully recovered from node failure ,without causing cluster rebalance to happen.

Appendix