DATASTAX ACADEMY

Exercise 04.04: Add and Remove a Node to and from the Cluster

In this exercise, you will accomplish the following:

- Add a node to your existing single node cluster
- Observe the effects of the additional node on cluster performance
- Remove the node from the cluster

Background

In the previous exercise, you stress-tested your single node cluster and monitored performance characteristics. You suspect that the node is CPU constrained, so you want to add more CPU. One method of doing this is to add an additional node to the cluster.

Step 1: Configure the 3rd Node

1. Confirm there are still two (2) terminal windows from the previous exercise. One is connected to the instance running the Cassandra node, and the other is connected to the instance running the test.
    a. Remember, we refer to the terminal window connected to the Cassandra node as the DSE-node1 window, and the window connected to the instance that runs the stress test as the DSE-node2 window.
2. In the DSE-node1 window, verify your cluster is up and running.

```
nodetool status
```

3. In the DSE-node2 window, observe the output from the previous cassandra-stress run. Make note of the mean I/O times. What will adding an additional node do to the load?
4. Open an additional terminal window, which we will call the DSE-node3 window. Connect to a third instance using the built-in IDE terminal

NOTE: After connecting to the node, verify this is DSE-node3 by using the hostname command.

5. Configure DSE on DSE-node3 just like we did on node1.
6. In the DSE-node3 window, configure the node by editing the *cassandra.yaml* file. This node will be in the same cluster as the first node, i.e. DSE-node1. Modify the

cassandra.yaml file to be similar to that which was done in the first exercise, except the `- seeds` entry will need to point at the first (1st) node.

   a. Remember that the *cassandra.yaml* file is in */home/ubuntu/dse/conf/cassandra.yaml*.

7. The entries needing modification include the following:

```
...
cluster_name: 'KillrVideoCluster'
...
listen_address: <THIRD_NODE'S_INTERNAL_IP_ADDRESS_GOES_HERE>
...
- seeds: "<FIRST_NODE'S_INTERNAL_IP_ADDRESS_GOES_HERE>"
...
num_tokens: 8
...
# initial_token:
...
endpoint_snitch: GossipingPropertyFileSnitch
...
start_native_transport: true
...
native_transport_port: 9042
...
rpc_address: 0.0.0.0
...
rpc_port: 9160
...
broadcast_rpc_address: <INTERNAL_IP_ADDRESS>
...
rpc_keepalive: true
...
```

8. Edit the */home/ubuntu/dse/conf/cassandra-rackdc.properties* file. Make sure the datacenter and rack values are as shown:

```
# These properties are used with GossipingPropertyFileSnitch and will
# indicate the rack and dc for this node
dc=dc1
rack=rack1
```

9. Save your changes to the *cassandra-rackdc.properties* file and exit the text editor.

10. We are now ready to start the other node and let it bootstrap into the cluster. In the DSE-node3 window, start the node with the following command:

```
/home/ubuntu/dse/bin/cassandra
```

Step 2: Confirm Connectivity of Nodes in Cluster

1. In the DSE-node3 window, check the status of the cluster:

```
nodetool status
```

The IP address displayed here should be that of the seed node or the DSE-node1 internal IP address.

2. In the DSE-node1 window, look at the load on the node using `nodetool info`. This command provides a wealth of information. Load refers to the amount of space used by SSTables. Take note of this number.

```
nodetool info
```

3. Run the `nodetool status` command and confirm that both nodes; DSE-node1 and DSE-node3 are present in the datacenter.
4. In the DSE-node1 window, perform a cleanup operation. Note that this operation requires writing to the commit log:

```
nodetool cleanup
```

5. In the DSE-node1 window, take note the load on the node after the cleanup:

```
nodetool info
```

6. Notice that the cleanup has reduced the load. This would have happened over time anyway due to compaction, but if we need to free up space immediately after adding a node, we can do that by using `nodetool cleanup`.

Step 3: Stress the Cluster

1. In the DSE-node2 window, re-run the `cassandra-stress` command and wait for it to complete.

```
cassandra-stress user profile=/home/ubuntu/labwork/TestProfile.yaml
ops\(insert=1,user_by_email=3\) -node DSE-node1
```

2. Investigate the summary output from the test and compare these numbers with the output of the test you ran before you added the node.
    a. Notice that the mean I/O times are reduced. The drop in the mean I/O times is a result of the additional CPU from the additional node.

Step 4: Remove the New Node from the Cluster

1. Remove the new node (node3) from the cluster. In the DSE-node1 window, check the status of the cluster:

```
nodetool status
```

2. Notice that both nodes in the cluster are Up and Normal (i.e., UN). Continue to monitor the status of the cluster by re-running this command while we remove the additional node from the cluster.
3. In the DSE-node3 window, find the process ID of the Cassandra node using:

```
ps -ef | grep cassandra
```

4. In the DSE-node3 window, decommission the additional node:

```
nodetool decommission --force
```

NOTE: The '--force' flag is necessary in because nodetool checks to see if there are enough replicas to satisfy the replication factors of the keyspaces. The 'system_distributed' keyspace has a replication factor of 3, which would cause the decommission to fail.

5. As this command runs, continue to monitor the status of the cluster from the DSE-node1 window. You may also be interested in running `nodetool netstats` in the DSE-node1 window to see the effects of streaming.
6. Once the decommission operation completes, the node will disappear from the cluster in the `nodetool status` command.
    a. Notice that, although the node has disappeared from the cluster, the node's process is still running. In the DSE-node3 window, look at the cassandra process:

```
ps -ef | grep cassandra
```

7. The node is running; however, it is no longer communicating with other nodes in the cluster. Leaving the node running would allow investigation of the JVM using JMX.
8. To terminate the node, in the DSE-node3 window, run:

```
nodetool flush
kill <cassandra_pid>
```

9. Back in the DSE-node1 window, look at the results of migrating the tokens back to the first node:

```
nodetool ring
```

Step 5: Re-introduce the New Node to the Cluster

1. Bring the additional node (DSE-node3) back online in preparation for the next exercise. Since the node's IP address is not changing, we will just delete the data and restart the node. In the DSE-node3 window, remove the contents of the *data/* directory:

```
rm -rf /home/ubuntu/dse/data/*
```

2. Bring the node back online and let it join the cluster. In the DSE-node3 window, start the node:

```
/home/ubuntu/dse/bin/cassandra
```

3. Back in the DSE-node1 window, check the status of the cluster as the additional node comes online. Repeat the following command:

```
nodetool status
```

4. Note the load on the new additional node. Has it changed from before?
5. Once the additional node is Up and Normal (i.e., UN), make certain the node has all the data by running repair. In the DSE-node3 window, run:

```
nodetool repair
```

6. Once again in the DSE-node1 window, check the status of the cluster:

```
nodetool status
```

7. Note how the load on the new node has changed due to the repair operation.
8. To remove the excess data from the original node, run cleanup. In the DSE-node1 window, run:

```
nodetool cleanup
```

9. Finally, check the status of the cluster by running (in either window):

```
nodetool status
```

10. Note the load on the original node has decreased due to the cleanup operation. Both nodes of the cluster should be up and running normally.

END OF EXERCISE