Exercise 05.02: Stand Up a Second Datacenter

In this exercise, you will accomplish the following:

- Modify the Cassandra cluster to add a second datacenter
- Modify the existing schema to take advantage of multiple datacenters

Step 1: Stand Up Node2

1. At this point you should have three (3) SSH sessions running; one session should be connected to the first node you created in the first exercise. We refer to this window as DSE-node1. A second session is connected to the instance that has been running `cassandra-stress` and we refer to this window as DSE-node2. A third session is connected to the second node added to the cluster. We refer to this window as DSE-node3.
2. In this exercise, we will configure the DSE-node2 instance to be its own datacenter.
3. In the DSE-node2 window, use your preferred text editor to edit the *cassandra.yaml* file. Change the settings as you did when you added a node to the cluster. To find this node's IP address, use the command `hostname -i`. The '- seeds' setting should be the IP address of the first node configured in the first exercise. After making the updates, save all changes and exit the editor.

```
...
cluster_name: 'KillrVideoCluster'
...
listen_address: <SECOND_NODE'S_IP_ADDRESS_GOES_HERE>
...
- seeds: "<FIRST_NODE'S_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
```

4. In the DSE-node2 window, use a text editor to modify the */home/ubuntu/DSE/conf/cassandra-rackdc.properties* file. This time change the dc setting to dc2. This is how we create a second datacenter. Make sure the datacenter and rack values are as shown, then save the changes and exit the editor:

```
# These properties are used with GossipingPropertyFileSnitch and will
# indicate the rack and dc for this node
dc=dc2
rack=rack1
```

5. Back in the DSE-node1 window, reference the cluster status:

```
nodetool status
```

6. Notice at this point the cluster only knows about dc1. Make sure both nodes in dc1 are up and normal: `UN`.
7. Again, in the DSE-node1 window, examine the distribution of tokens:

```
nodetool ring
```

8. Assuming the 'num_tokens' setting to set to 8 for each node, there should now be 16 ring entries.

Step 2: Start Node2 as a Second Datacenter

1. In the DSE-node2 window, start node2 as part of the second datacenter.

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

2. While the node is joining the cluster, proceed to the next two steps.
3. Back in the DSE-node1 and DSE-node3 windows, monitor the progress of the joining node. In the DSE-node1 window, observe the cluster status:

```
nodetool status
```

4. The results of this command show the new node in dc2. Notice that the status will be UJ (Up and Joining) until the node has completed receiving the streamed data.
5. Now, in the DSE-node1 window, observe the cluster ring:

```
nodetool ring
```

6. Notice there are now two (2) datacenters, and each has its own ring. However, the keyspace is not aware of the two (2) datacenters. The keyspace still treats the two (2) datacenters as if they were one (1) big ring. We need to alter the keyspace to make the keyspace aware of the topology.
7. In the DSE-node1 window using `cqlsh`, alter the keyspace so that there are two (2) replicas in dc1 and one (1) replica in dc2.

```
ALTER KEYSPACE killr_video WITH replication = {'class':
'NetworkTopologyStrategy', 'dc1':2, 'dc2':1 };
```

8. The keyspace should now be aware of how to replicate the data, but the data has not yet moved. The consequences of the data not moving is that if we lost the node in dc2 we would lose data. Force the data to move to the right nodes by running `nodetool repair` on each of the two dc1 nodes, i.e., DSE-node1 and DSE-node3:

```
nodetool repair -pr
```

9. One final step that may be helpful to complete the migration of the second datacenter, cleanup the nodes. Remember that repair replicates data to the correct nodes but does not delete the unneeded data. Run the command `nodetool cleanup` on each of the nodes in the cluster to reclaim the extra space. Finally, run the command `nodetool ring` again to see the difference.

```
nodetool cleanup
nodetool ring
```

10. In order to demonstrate the second datacenter provides better availability, run the following commands. Using the DSE-node2 window and by running `cqlsh` within that node, insert a record into the 'user_by_email' table as follows:

```
INSERT INTO killr_video.user_by_email (email, password, user_id) VALUES
('FredFlintstone@gmail.com', 'Yabadabado!', 00000000-0000-0000-0000-
000123456789);
```

Step 3: Shutdown Node2 and Retrieve Data

1. Use the DSE-node2 window to shut down the node in dc2.

```
ps -ef | grep cassandra
kill <cassandra_pid>
```

2. Perform a query from `cqlsh` in the DSE-node1 window:

```
SELECT * FROM killr_video.user_by_email WHERE
email='FredFlintstone@gmail.com';
```

3. Observe that even though an entire datacenter (dc2) is down, it is still possible to retrieve the data (from dc1).
4. Restart the node in the second datacenter (DSE-node2). From the DSE-node2 window, execute the following:

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

5. Perform a final status check and confirm both datacenters and all three (3) nodes are evenly balanced.

```
nodetool status
```

6. The terminal should appear similar to the following. Note the Load on all three (3) nodes.

END OF EXERCISE