15

REMOVING A NODE

Understanding Removing a Node

Decommissioning a Node

Putting a Node Back into Service

Removing a Dead Node

Understanding Removing a Node

A node may need to be removed from a cluster for a variety of reasons. For example, the capacity might no longer be needed, or a node may need hardware maintenance, such having more memory added, or a node might be down due to hardware failure.

Designed to be fault-tolerant, Cassandra handles node removal gracefully.

Depending on whether a node is planned for removal or has unexpectedly died, the `nodetool decommission` or `nodetool removenode` commands can be used to handle the node removal.

The `nodetool decommission` command is for a planned removal, whereas the `nodetool removenode` command is for a dead node.

Decommissioning a Node

Decommissioning a node is when you choose to take a node out of service.

To decommission a node, the `nodetool decommission` command can be used.

```
nodetool -h 192.168.159.103 -p 7199 decommission
```

The decommission command assigns the token ranges that the node was responsible for to other nodes, and then streams the data from the node being decommissioned to the other nodes.

Decommissioning a node does not remove data from the decommissioned node. It simply copies data to the nodes that are now responsible for it.

Exercise 1: Decommission a Node

In this exercise, you decommission a node.

1. In vm**1**, in the nodetool status window, see that all three nodes are currently up:

```
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address          Load       Owns   Host ID
UN  192.168.159.102  9.81 MB    33.9%  75151546-2217-49
UN  192.168.159.103  9.2 MB     31.6%  a6a90230-35e1-4c
UN  192.168.159.101  10.03 MB   34.5%  bcc09b1b-51e1-40
```

2. In another terminal window, in the directory where Cassandra is installed, enter the following to decommission the vm3 node, using your vm3 IP address:

```
bin/nodetool -h 192.168.159.103 -p 7199 decommission
```

3. See that the node is leaving, as indicated by **UL** in the nodetool status window:

```
Status=Up/Down
|/ State=Normal/Leavi
--  Address
UN  192.168.159.102
UL  192.168.159.103
UN  192.168.159.101
```

4. After a while, see that the node is gone and that its load has been assigned to the remaining nodes:

```
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address          Load       Owns
UN  192.168.159.102  14.5 MB    50.4%
UN  192.168.159.101  14.3 MB    49.6%
```

Putting a Node Back into Service

Since data is not removed from a node when it is decommissioned (the data is copied to the other nodes, but not removed from the decommissioned node), it is best to clear the data from the decommissioned node, if the node has been down for any length of time, before putting the node back into service.

In general, it is faster to have the node join as a clean one (with no data), rather than have it join with old data that then needs to be repaired.

Once the data has been deleted from the decommissioned node, the node can join as a new node.

Clearing Data From a Node

To completely remove the data on a Cassandra node, the data, commitlog, and saved_caches directories need to be cleared. This can be done on the command line using the `rm` command.

```
vmx@vm3:~$ cd /var/lib/cassandra
vmx@vm3:/var/lib/cassandra$ ls
commitlog  data  saved_caches
vmx@vm3:/var/lib/cassandra$ rm -r commitlog data saved_caches
```

Exercise 2: Put a Node Back into Service

In this exercise, you put a decommissioned node back into service.

1. In vm**3**, in the terminal window where Cassandra is running, press **Ctrl-C** to stop Cassandra.
2. Enter `ps aux | grep cass` to see that Cassandra is no longer running:

```
vmx@vm3:~/cassandra/apache-cassandra-2.0.7$ ps aux | grep cass
vmx       9999  0.0  0.0  13596   940 pts/1    S+   08:38   0:0
0 grep --color=auto cass
vmx@vm3:~/cassandra/apache-cassandra-2.0.7$
```

3. In another terminal window in vm3, enter `cd /var/lib/cassandra` to navigate to the directory where the node's Cassandra data is stored.
4. Enter `ls` to see the contents of the directory.
5. **Notice** the commitlog, data, and saved_caches directories:

```
vmx@vm3:~$ cd /var/lib/cassandra
vmx@vm3:/var/lib/cassandra$ ls
commitlog  data  saved_caches
```

6. Enter `rm -r commitlog data saved_caches` to delete all three directories, to clear all of Cassandra data stored on this node.
7. Enter `ls` to confirm that the directories have been deleted.
8. Back in the previous terminal window, in the directory where Cassandra is installed, enter `bin/cassandra -f` to start Cassandra running on this node.
9. In vm1, in the nodetool status window, see the vm3 node joining the cluster:

```
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address          Load       Owns   Host ID
UN  192.168.159.102  14.51 MB   50.4%  75151546-
UJ  192.168.159.103  14.17 KB   ?      a7feb8c0-
UN  192.168.159.101  14.3 MB    49.6%  bcc09b1b-
```

10. After awhile, see that the vm3 node is now the owner of approximately a third of the data:

```
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address          Load       Owns   Host ID
UN  192.168.159.102  14.52 MB   33.8%  75151546-
UN  192.168.159.103  9.05 MB    31.2%  a7feb8c0-
UN  192.168.159.101  14.3 MB    34.9%  bcc09b1b-
```

11. Notice that the vm1 and vm2 nodes still have approximately the same Load values as they did before. Realize this is because, although data was copied to the vm3 node for the ranges it owns, that no data was deleted from the vm1 or vm2 nodes.

12. In another terminal window in vm3 (or any node in your cluster), in the directory where Cassandra is installed, enter `bin/nodetool -h 192.168.159.101 -p 7199 cleanup` to cleanup the vm1 node.

13. After awhile, in the nodetool status window, notice that the vm1 node has been cleaned up:

```
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address          Load       Owns   Host ID
UN  192.168.159.102  14.52 MB   33.8%  75151546-
UN  192.168.159.103  9.05 MB    31.2%  a7feb8c0-
UN  192.168.159.101  10.15 MB   34.9%  bcc09b1b-
```

14. Enter `bin/nodetool -h 192.168.159.102 -p 7199 cleanup` to cleanup the vm2 node.

15. After awhile, in the nodetool status window, notice that the vm2 node has been cleaned up:

```
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address          Load       Owns   Host ID
UN  192.168.159.102  9.79 MB    33.8%  75151546-
UN  192.168.159.103  9.05 MB    31.2%  a7feb8c0-
UN  192.168.159.101  10.15 MB   34.9%  bcc09b1b-
```

Removing a Dead Node

Removing a dead node from the cluster is done to reassign the token ranges that the dead node was responsible for to other nodes in the cluster and to populate other nodes with the data that the dead node had been responsible for.

To remove a dead node from the cluster, and reassign its token ranges and data, the nodetool removenode command can be used.

```
bin/nodetool removenode 1c978113-0fbd-425c-83df-353389044bba
```

Note: the host id can be copied from a nodetool status window.

With a dead node being dead, the data that it was responsible for needs to come from other nodes in the cluster, which happens when the `nodetool removenode` command is run.

Watching removenode Happen

The `nodetool removenode status` command can be used to watch removenode happen.

```
Every 2.0s: bin/nodetool ...  Tue Jun  3 11:56:34 2014

RemovalStatus: Removing token (-9129580477490204828).
Waiting for replication confirmation from [/192.168.15
9.102,/192.168.159.101].
```

Exercise 3: Remove a Dead Node

In this exercise, you remove a dead node.

1. In vm**1**, in the nodetool status window, see that all three nodes are currently up:

```
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address          Load       Owns   Host ID
UN  192.168.159.102  9.77 MB    33.8%  75151546-2217-49
UN  192.168.159.103  9.08 MB    31.2%  a7feb8c0-ff78-4a
UN  192.168.159.101  10.17 MB   34.9%  bcc09b1b-51e1-40
```

2. In vm**3**, in the terminal window where Cassandra is running, press **Ctrl-C** to stop Cassandra.

3. In vm**1**, in the window with nodetool status, see that vm3 is now down:

```
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address          Load       Owns   Host ID
UN  192.168.159.102  9.77 MB    33.8%  75151546-2217-49
DN  192.168.159.103  9.08 MB    31.2%  a7feb8c0-ff78-4a
UN  192.168.159.101  10.17 MB   34.9%  bcc09b1b-51e1-40
```

4. In another terminal window on vm1, in the directory where Cassandra is installed, set up a watch for removenode status:

```
$ watch -n 2 bin/nodetool removenode status
```

5. See that the watch command is running every 2 seconds:

```
Every 2.0s: bin/nodetool removenode status

RemovalStatus: No token removals in process.
```

6. In the nodetool status window, highlight and copy the host id of the vm3 node (e.g. a7feb8c0-ff78-4ac1-b043-1517e532b6b9).

7. In yet another terminal window on vm1 (or vm2), in the directory where Cassandra is installed, enter the removenode command below, **substituting in** the host id for your vm3, so that Cassandra will remove the vm3 node from the cluster, assign the token ranges to the other nodes, and replicate the data from the other nodes to the other nodes:

```
bin/nodetool removenode a7feb8c0-ff78-4ac1-b043-1517e532b6b9
```

8. In the remove status watch window, watch the tokens previously assigned to vm3 being assigned to other nodes:

```
Every 2.0s: bin/nodetool ...  Tue Jun  3 11:56:34 2014

RemovalStatus: Removing token (-9129580477490204828).
Waiting for replication confirmation from [/192.168.15
9.102,/192.168.159.101].
```

9. After awhile, in the regular nodetool status window, see that the vm3 node has been removed from the cluster:

```
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address          Load       Owns   Host ID
UN  192.168.159.102  9.77 MB    50.4%  75151546-2217-49
UN  192.168.159.101  10.2 MB    49.6%  bcc09b1b-51e1-40
```

10. Notice that the vm1 and vm2 nodes have taken on the ownership of the token ranges that had been owned by the vm3 node.

11. Know that, because the replication factor of the vehicle_tracker and home_security keyspaces is 2, that the data that was on the vm3 node was able to be regenerated from the remaining nodes.

12. Realize that approximately one third of the data in the Keyspace1 keyspace is missing from the cluster, because, with the replication factor for that keyspace only being 1, and the node dying unexpectedly (rather than being decommissioned), there are no replicas of the data that was on the vm3 node to get from the remaining nodes.

Summary

The focus of this chapter was on removing a node:

- Understanding Removing Nodes
- Decommissioning a Dead Node
- Putting a Node Back into Service
- Removing a Dead Node

Unit Review Questions

1) Which command is for when a node has unexpectedly died?

a. nodetool restore

b. nodetool decommission

c. nodetool live

d. nodetool removenode

2) Which command is for removing a node due to a decreased need for capacity?

a. nodetool restore

b. nodetool decommission

c. nodetool live

d. nodetool removenode

3) It is generally best to delete a node's stored data before putting the node back into service.

a. True

b. False

4) What could potentially be done to get back the Keyspace1 data that was lost in the last exercise?

a. Nothing can be done

b. Run removenode again

c. Delete the stored data on the vm3 node, start the node back up, and run removenode again

d. Start the node back up and, to prevent the problem from happening in the future, change the replication factor for Keyspace1 to 2, and run repair for the keyspace, to generate the second replicas

Lab: Put a Node Back into Service

In this exercise, you remove the stored data on vm3 and put the node back into service. (We don't need the data from Keyspace1, generated by the cassandra-stress tool.)

1. In vm**3**, in a terminal window, enter `cd /var/lib/cassandra` to navigate to the directory where the node's Cassandra data is stored.
2. Enter `ls` to see the contents of the directory.
3. **Notice** the commitlog, data, and saved_caches directories.
4. Enter `rm -r commitlog data saved_caches` to delete all three directories, to clear all of Cassandra data stored on this node.
5. Enter `ls` to confirm that the directories have been deleted.
6. In a terminal window, in the directory where Cassandra was installed, enter `bin/cassandra -f` to start Cassandra running on this node.
7. In vm1, in the nodetool status window, see the vm3 node joining the cluster:

```
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address          Load       Owns    Host ID
UN  192.168.159.102  9.77 MB    50.4%   75151546-2217-49
UJ  192.168.159.103  14.17 KB   ?       a9910b7d-075f-47
UN  192.168.159.101  10.2 MB    49.6%   bcc09b1b-51e1-40
```

8. See that the vm3 node is now responsible for approximately a third of the data:

```
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address          Load       Owns    Host ID
UN  192.168.159.102  9.77 MB    33.6%   75151546-2217-49
UN  192.168.159.103  4.55 MB    31.3%   a9910b7d-075f-47
UN  192.168.159.101  10.2 MB    35.0%   bcc09b1b-51e1-40
```

9. Notice that the load values for the vm1 and vm2 nodes did not go down.

10. In another terminal window (in any of the vms in your cluster), in the directory where Cassandra is installed, enter `bin/nodetool -h 192.168.159.101 -p 7199 cleanup` to cleanup the vm1 node.

11. Enter `bin/nodetool -h 192.168.159.102 -p 7199 cleanup` to cleanup the vm2 node.

12. After awhile, in the nodetool status window, see that the vm1 and vm2 nodes have been cleaned up:

```
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address          Load       Owns   Host ID
UN  192.168.159.102  7.47 MB    33.6%  75151546-2217-49
UN  192.168.159.103  4.55 MB    31.3%  a9910b7d-075f-47
UN  192.168.159.101  8.08 MB    35.0%  bcc09b1b-51e1-40
```

Alternate Lab Steps: Put a Node Back into Service

In this exercise, you remove the stored data on vm3 and put the node back into service. (We don't need the data from Keyspace1, generated by the cassandra-stress tool.)

1. In vm3, clear the data, commitlog, and saved_caches directories.
2. In vm3, start Cassandra.
3. In vm1, watch the vm3 node join the cluster.
4. Clean up vm1 and vm2.
5. See that vm3 is part of the cluster, with approximately one third of the ring ownership and approximately one third of the data load:

```
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address          Load       Owns   Host ID
UN  192.168.159.102  7.47 MB    33.6%  75151546-2217-49
UN  192.168.159.103  4.55 MB    31.3%  a9910b7d-075f-47
UN  192.168.159.101  8.08 MB    35.0%  bcc09b1b-51e1-40
```