Cassandra Operation

$ bin/nodetool status

- Use nodetool status to make sure all of the nodes are up and reporting normal status.
- Check the load on each node to make sure the cluster is well balanced. An uneven number of nodes per rack can lead to an imbalanced cluster.

Nodetool/status info

Address	The node's URL.
Rack	The rack or, in the case of Amazon EC2, the availability zone of the node.
Status	Up or Down Indicates whether the node is functioning or not.
State	N (normal), L (leaving), J (joining), M (moving)The state of the node in relation to the cluster.
Load	Updates every 90 seconds. The amount of file system data under the cassandra data directory after excluding all content in the snapshots subdirectories. Because all SSTable data files are included, any data that is not cleaned up, such as TTL-expired cell or tombstoned data) is counted.
Token	The end of the token range up to and including the value listed.
Own	The percentage of the data owned by the node per datacenter times the replication factor. For example, a node can own 33% of the ring, but show100% if the replication factor is 3.
Host ID	The network ID of the node

`$ bin/nodetool ring`

- Displays node status and information about the ring as determined by the node being queried.
- This information can give you an idea of the load balance and if any nodes are down.
- If your cluster is not properly configured, different nodes may show a different ring. Check that the node appears the same way in the ring.
- If you use virtual nodes (vnodes), use nodetool status for succinct output.

$ nodetool flush [ks [tbl]]

- To force Cassandra to write data from its memtables to SSTables on the filesystem
- If you check the server logs, you'll see a series of output statements similar to this, one per table stored on the node

```
DEBUG [RMI TCP Connection(297)-127.0.0.1] 2015-12-21
19:20:50,794 StorageService.java:2847 - Forcing flush on
keyspace hotel, CF reservations_by_hotel_date
```

`$ bin/nodetool drain`

- Performs a flush and then directs Cassandra to stop listening to commands from the client and other nodes.
- The drain command is typically used as part of an orderly shutdown of a Cassandra node
- Helps the node startup to run more quickly, as there is no commitlog to replay.

`$ bin/nodetool cleanup [ks [tbl]]`

- The cleanup command scans all of the data on a node and discards any data that is no longer owned by the node. Unwanted data appear at the node for:
  - If you change the replication factor or the replication strategy.
  - If you decrease the number of replicas for any data center
  - Reassigned the tokens ranges

- The normal compaction processes discard this data.
- However, you may wish to reclaim the disk space used by this excess data more quickly to reduce the strain on your cluster. To do this, you run cleanup command

$ bin/nodetool repair [ks [tbl]]

- Cassandra's tuneable consistency means that it is possible for nodes in a cluster to get out of sync over time.
- Repair command as known as anti-entropy repair or manual repair brings back the one or more nodes to in sync with its replicas

```
$ nodetool repair -pr -hosts 10.2.2.20 10.2.2.21
$ nodetool repair -dc DC1
```

- Indexes cannot be repair. Run the rebuild command instead.

```
$ bin/nodetool rebuild_index
```

Adding Node

- Use the same configuration values as other nodes in *cassandra.yaml* and *cassandra-env.sh*
  - cluster_name, dynamic_snitch, partitioner, listen_address and seeds
- Set autobootstrap = true
- Once the nodes are configured, start them and use nodetool status to determine when they are fully initialized
- Watch the progress of a bootstrap operation on a node by running the nodetool bootstrap
- After all new nodes are running, run a nodetool cleanup on each of the previously existing nodes

Adding new DC

- Configure each node as appropriate for gossip
- Set autobootstrap = false
- After all of the nodes in the new data center have been brought online, configure replication options for the NetworkTopologyStrategy for all keyspaces that you wish to replicate to the new data center
- Run the nodetool rebuild command on each node in the new data center
- Reconfigure the *cassandra.yaml* file for each node in the new data center to make autobootstrap: true

Repairing a Node

- If the node has been down for less than the hints delivery window specified by the max_hint_window_in_ms property, the hinted handoff mechanism should be able to recover the node.Restart the node and see whether it is able to recover. You can watch the node's logs or track its progress using nodetool status.
- If the node has been down for less than the repair window defined lowest value of gc_grace_seconds for any of its contained tables, then restart the node. If it comes up successfully, run a nodetool repair.
- If the node has been down for longer than the repair window, it should be replaced, in order to avoid tombstone resurrection.

Recover from disk failure

- If the is set to the default (stop), the node will stop gossiping, which will cause it to appear as a downed node in nodetool status. You can still connect to the node via JMX.
- If the policy is set to die, the JVM exits and the node will appear as a downed node in nodetool status.
- If the policy is set to ignore, there's no immediate way to detect the failure.
- If the policy is set to best_effort, Cassandra continues to operate using the other disks, but a WARN log entry is written, which can be detected if you are using a log aggregation tool. Alternatively, you can use a JMX monitoring tool to monitor the state of the org.apache.cassandra.db.BlacklistedDirectoriesMBean, which lists the directories for which the node has recorded failures.
- If a disk failure is detected on node startup and the policy is anything besides best_effort, the node writes an ERROR log entry and exits
- If you replace the disk, delete the contents of the *data/system* directory in the remaining disks so that when you restart the node, it comes up in a consistent state. When the node is up, run a repair.

Replace an existing node

- Removing and then adding nodes results in excess streaming of data, first to move data away from the old node and then to move it back to the new node.
- Edit the *cassandra-env.sh* file for the new node to add the following JVM option (where <address> is the IP address or hostname of the node that is being replaced):

```
JVM_OPTS="$JVM_OPTS -Dcassandra.replace_address=<address>”
```

- After the replacement node finishes bootstrapping, remove this option

Replace a Seed Node

- If the node you’re replacing is a seed node, select an existing non-seed node to promote to a seed node.You’ll need to add the promoted seed node to the seeds property in the *cassandra.yaml* file of existing nodes.
- Typically, these will be nodes in the same data center, assuming you follow the recommendation of using a different seed list per data center. In this way, the new node we create will be a non-seed node and can bootstrap normally.

Removing a node

- Three options in order of priority
  - Decommission
  - Remove
  - Assassin

Decommission

- Decommissioning a node means pulling it out of service. The decommissioning reassigns the token ranges and streams the data to those nodes. This is effectively the opposite of the bootstrapping operation. You can view the status as UL
- Data is not automatically removed from a decommissioned node. Manually delete its data first.

Removing a node

- If the node is down, use the nodetool removenode command instead of decommission

```
$ nodetool removenode  e78529c8-ee9f-46a4-8bc1-
3479f99a1860
```

Assassin a node

- If the nodetool removenode operation fails, run nodetool assassinate as a last resort.
- The assassinate command is similar to removenode, except that it does not re-replicate the removed node’s data. This leaves your cluster in a state where repair is needed.
- Another key difference from removenode is that the assassinate command takes the IP address of the node to assassinate, rather than the host ID

```
$ nodetool assassinate 127.0.0.3
```

Backup and Recovery

- A *full backup* includes the entire state of a database (or specific tables within a database) and are the most expensive to create.
- An *incremental backup* includes the changes made over a period of time, typically the period of time since the last incremental backup. Taken together, a series of incremental backups provides a differential backup.

Snapshot

- The purpose of a snapshot is to make a copy of some or all of the keyspaces and tables in a node and save it to what is essentially a separate database file.
- When you take a snapshot, Cassandra first performs a flush, and then makes a hard link for each SSTable file
- Cassandra also provides an *auto snapshot* capability that takes a snapshot on every DROP KEYSPACE, DROP TABLE, or TRUNCATE operation. This capability is enabled by default via the auto_snapshot property in the *cassandra.yaml* file to prevent against accidental data loss.
- Clear old snapshot after taking new snapshot

```
$ nodetool snapshot
$ nodetool listsnapshots
```

Enable Incremental Backup

- To enable backups across a restart of the node, set the incremental_backups property to true in the *cassandra.yaml* file.
- When incremental backups are enabled, Cassandra creates backups as part of the process of flushing SSTables to disk. The backup consists of a hard link to each data file Cassandra writes under a *backups* directory
- You can safely clear incremental backups after you perform a snapshot and save the snapshot to permanent storage.

Backup Schema

- Cassandra does not include the database schema as part of snapshots and backups. You will need to make sure that the schema is in place before doing any restore operations. Fortunately, this is easy to do using the cqlsh’s DESCRIBE TABLES operation, which can easily be scripted.

Restoring from Snapshot

- Restoring from snapshots or backups follows the same process.
- If your cluster topology is the same as when the snapshots were taken, there have been no changes to the token ranges for each node, and there are no changes to the replication factor for the tables in question, you can copy the SSTable data files into the data directory for each node.
- If the nodes are already running, running the nodetool refresh command will cause Cassandra to load the data.
- If there has been a change to the topology, token ranges, or replication, you'll need to use a tool called sstableloader to load the data.