READY

NESTO.TV

```
%sh
STATUS="$(service cassandra status)"

if [[ $STATUS == *"is running"* ]]; then
    echo "Cassandra is running"
else
    echo " Cassandra not running .... Starting"
    service cassandra restart > /dev/null 2>&1 &
    echo " Started"
fi
```

READY

Exercise 5 – Node

READY

In this exercise, you will:

- Understand what Apache Cassandra™ nodes are.
- Understand core hardware/software requirements of a node.

Nodes are the building blocks of Apache Cassandra™'s clusters. Therefore, it is useful to understand the care and feeding of nodes. These exercises will do just that.

Execute nodetool with the `help` command to list all possible commands.

READY

```
%sh
nodetool help
```

READY

```
nodetool status
```

READY

The status command shows information about the entire cluster, particularly the state of each
node, and information about each of those nodes: IP address, data load, number of tokens,total percentage of data saved on each node, host ID, and datacenter and rack. We will discuss these in detail as the course progresses

READY

05_Node_Solution

```
nodetool info
```

READY

```
nodetool describecluster
```

READY

```
nodetool getlogginglevels
```

READY

```
nodetool setlogginglevel org.apache.cassandra TRACE
```

READY

The command setlogginglevel dynamically changes the logging level used by Apache Cassandra™ without the need for a restart. You can also look at the /var/log/cassandra/system.log afterwards to observe the changes.

READY

```
%sh
cat /var/log/cassandra/system.log
```

READY

```
nodetool settraceprobability 0.1
```

READY

The resultant value from the settraceprobability command represents a decimaldescribing the percentage of queries being saved, starting from 0 (0%) to 1 (100%).
Saved traces can then be viewed in the system_traces keyspace.

READY

READY

```
cassandra-stress write n=50000 no-warmup -rate threads=1
```

READY

Initially, we will see a long list of setting for the stress run. As Apache Cassandra™ stress executes, it logs several statistics to the terminal. Each line displays the statistics for the operations that occurred each second and shows number of partitions written, operations per
second, latency information, and more.

READY

```
nodetool flush
```

READY

The flush command commits all written (memtable, discussed later) data to disk. Unlike drain, flush allows further writes to occur. READY

READY

Check the new load on the node. We will now examine the data cassandra-stress wrote to our node. READY

Execute the following CQLSH **describe** command to view the current keyspaces:

```
nodetool status
```

READY

```
%cassandra

//Notice the presence of keyspace1 which cassandra-stress created.
DESCRIBE KEYSPACES;
```

READY

```
%cassandra

// Switch to that keyspace by executing the following:
USE keyspace1;
```

READY

```
%cassandra

// View the tables in keyspace1 by executing the following:
DESCRIBE TABLES;
```

READY

READY

Query the first five rows from **standard1** by executing the following query: READY

The data that was written is not very meaningful, since they are all arbitrary BLOB values.

05_Node_Solution

```
%cassandra

SELECT *
```

READY

```
FROM standard1
```

nodetool drain

READY

The drain command stops writes from occurring on the node and flushes all data to disk. Typically, this command may be run before stopping an Apache Cassandra™ node.

READY