08_Write_Path

READY


```
%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
```

FINISHED

```
Cassandra not running .... Starting
Started
```

Took 0 sec. Last updated by anonymous at July 15 2020, 4:52:42 PM.

Exercise 08 – Write Path

READY

In this exercise, you will:

- Understand the Apache Cassandra™ write path.

Apache Cassandra™ has an optimized write path. To understand how to use Apache Cassandra™,
it can be very helpful to understand the Apache Cassandra™ write path. In this exercise we will
see Apache Cassandra™ writing data to the file system.

Check the status of the running node nodetool status.

READY

```
%sh

nodetool status
```

READY

Investigate the data directory.

FINISHED

Took 2 sec. Last updated by anonymous at July 13 2020, 10:03:53 PM.

```
ls -lh /var/lib/cassandra/data/
```

ERROR

```
ls: cannot access '/var/lib/cassandra/data/': No such file or directory
```

```
ExitValue: 2
```

Took 1 sec. Last updated by anonymous at July 15 2020, 4:52:38 PM.

08_Write_Path

READY

We will now use the cassandra-stress tool to write several thousand records to our node. Execute the following command in your original terminal:

READY

```
cassandra-stress write no-warmup n=75000 -port native=9042 -rate threads=1
```

READY

FINISHED

Be sure your second terminal is also visible as cassandra-stress executes. cassandra-stress will write 75000 rows to your node.

There are a few things to watch out for while cassandra-stress inserts keys:

- The total size will continue to increase.
- The timestamp will change for the current segment being written.
- You may get additional commit log files as well.

Took 3 sec. Last updated by anonymous at July 15 2020, 4:46:20 PM.

READY

Execute the following nodetool command:

READY

```
nodetool cfstats keyspace1.standard1
```

READY

READY

cassandra-stress created the keyspace1.standard1 table and populated its data. `cfstats` gives you column family stats. Column family is a deprecated term for a table.

READY

```
nodetool cfstats
```

READY

READY

Notice the “Write Count” matches the number of rows we told cassandra-stress to insert. cfstats also reports the number of SSTables, space used, and bloom filter statistics.

Note the Memtable statistics.

```
Memtable cell count: 649
Memtable data size: 181071
Memtable off heap memory used: 0
Memtable switch count: 5
```

08_Write_Path

Execute the following nodetool command which will flush the memtable contents to disk. READY

```
nodetool flush
```
READY

Now check the table stats again by executing: READY

```
nodetool cfstats keyspace1.standard1
```
READY

Note the memtable statistics zeroed out because we flushed the previous memtable to disk. READY

```
Memtable cell count: 0
Memtable data size: 0
Memtable off heap memory used: 0
Memtable switch count: 6
```

If there were no commit log segments found during startup, no replay needs to be done. If Apache Cassandra™ finds commit log files, it will replay the mutations in those files into memtables and then flush the memtables to disk.

```
```
READY

```
```
READY