





Version 1.0   
6 September 2017

Unotech Software Pvt Ltd

**Index**

[**Overview:**](#_hwnfgq7f59a4) **6**

[Document Purpose:](#_ka8gg84kswzy) 6

[Brief Introduction :](#_ew6b75zap4xz) 6

[Architecture:](#_u1orq2xcz4uq) 6

[**Installation Guide:**](#_v932gfk17f5c) **8**

[Installation:](#_1zj1zyquhmps) 8

[Configuration:](#_92urwpg366et) 9

[Network and OS Level:](#_qh0v1a3e2q9x) 10

[Ports:](#_hnqc1wumm990) 10

[OS Level:](#_sq7ys7jai0qg) 10

[Starting Cassandra:](#_carmwfwu3o9a) 11

[Cqlsh & Test:](#_2j9rn2u0dnc2) 12

[Importing Schema:](#_r9csd3xxqmuw) 13

[Export Schema, Data](#_j47ukzdfvwbx) 14

[**Managing the Cluster & Data Center**](#_ugjihg2r9nk4) **16**

[Creating the New Data Center in same Cluster](#_ckpseh62hf94) 16

[Configuration](#_6db151cytbm0) 16

[Adding the new Node to the Datacenter/Cluster](#_1sflnn2co9gk) 16

[Configuration](#_1wr8dsi9fxk4) 16

[Adding the Data Center of different cluster](#_b71shd1bvzp0) 17

[Configuration](#_u21z1te47ykr) 17

[Removing the node from the cluster](#_b8z7o1z0xz5y) 17

[Case 1: nodetool decommission](#_eayy4lx71dtn) 17

[Case 2: nodetool removenode](#_xsoipfcqw6v8) 17

[Case 3: nodetool assassinate](#_aq1sx2dvnz0q) 17

[**Monitoring:**](#_lmmranoylcxz) **19**

[**Maintenance:**](#_4cqfdbtsslwc) **20**

[Good Practices:](#_ke64g6k9rnah) 20

[Maintenance Tools:](#_tojaobbo7l9k) 20

[nodetool rebuild](#_8qn9eyxg5o7d) 20

[nodetool repair](#_wtjvgx1ehiyz) 20

[Best Practice for Nodetool Repair](#_bn311zj8eg3c) 20

[**Security:**](#_9y7xfgc3ljo9) **21**

[Update system\_auth](#_cs8tvbpo1kxb) 21

[Configuration](#_tozcazrcf51p) 22

[Create Username and Password](#_huumxgt40mq0) 22

[Other Commands](#_8eqbjoh94kp8) 22

[**Performance:**](#_jtn8cpj8pkyt) **22**

# 

# 

# Overview:

## 1.1 Document Purpose:

This document is created with the purpose:

* Brief Introduction to the Cassandra
* Architecture
* Installation of the Cassandra-3.0.9 for distributed mode
* Configuring Cassandra for distributed mode
* Introduction to cqlsh
* Basic tutorial for creating the Keyspace and Column-family
* Inserting the Data and writing basic cqlsh query

## 1.2 Brief Introduction :

Apache Cassandra™ is a massively scalable NoSQL database. Cassandra’s technical roots can be found at companies recognized for their ability to effectively manage big data – Google, Amazon, and Facebook – with Facebook open sourcing Cassandra to the Apache Foundation in 2009.

Apache Cassandra is

* Free
* Distributed
* High Performance
* Massively Scalable
* Fault Tolerant

The massive scale, high performance, and never-go-down nature of these applications has forged a new set of technologies that have replaced the legacy RDBMS, with O’Reilly describing the situation in this way:

“Big data is data that exceeds the processing capacity of conventional database systems. The data is too big, moves too fast, or doesn’t fit the structures of your database architectures. To gain value from this data, you must choose an alternative way to process it.”

## 1.3 Architecture:

* Cassandra provides automatic data distribution across all nodes that participate in a “ring” or database cluster.
* There is nothing programmatic that a developer or administrator needs to do or code to distribute data across a cluster
* Data is transparently partitioned across all nodes in either a randomized or ordered fashion, with random being the default.

|  |  |
| --- | --- |
|  | * Cassandra was designed with the understanding that system/hardware failures can and do occur * Rather than using a legacy master-slave or a manual and difficult-to maintain sharded design, Cassandra has a peer-to-peer distributed architecture that is much more elegant, and easy to setup and maintain. * In Cassandra, all nodes are the same; there is no concept of a master node, with all nodes communicating with each other via a gossip protocol. * Cassandra’s built-for-scale architecture means that it is capable of handling petabytes of information and thousands of concurrent users/operations per second (across multiple data centers) as easily as it can manage much smaller amounts of data and user traffic |

Some of the application use cases that Cassandra excels in include:

• Real-time, big data workloads

• Time series data management

• High-velocity device data consumption and analysis

• Media streaming management (e.g., music, movies)

• Social media (i.e., unstructured data) input and analysis

• Online web retail (e.g., shopping carts, user transactions)

• Real-time data analytics

• Online gaming (e.g., real-time messaging)

• Software as a Service (SaaS) applications that utilize web services

• Online portals (e.g., healthcare provider/patient interactions)

• Most write-intensive systems

# 

# 

# 2. Installation Guide:

## 2.1 Installation:

**Install Java :** Install the latest version of 64 bit JVM (Oracle Java Platform or OpenJDK)

**Apache Cassandra 3.0.9 Installation:**

Create the directory (Recommended more disk space) for simplicity we considering /opt

###### cd /opt/damocles/

Download/copy the **apache-cassandra-3.9-bin.tar.gz**

###### wget http://www-eu.apache.org/dist/cassandra/3.9/apache-cassandra-3.9-bin.tar.gz

Extract the tar :

###### tar -zxvf apache-cassandra-3.9-bin.tar.gz

###### mv apache-cassandra-3.9 cassandra # Rename directory to cassandra

Setting the environment variables in bashrc

###### vim ~/.bashrc

###### #Cassandra Environment Variables Starts

###### export CASSANDRA\_HOME=/opt/cassandra/apache-cassandra-3.9

###### export PATH=$CASSANDRA\_HOME/bin:$PATH

###### #Cassandra Environment Variables ends

###### source ~/.bashrc

Create the folders - data, commitlog, saved\_caches

###### cd /opt/cassandra

###### mkdir data

###### mkdir commitlog

###### mkdir saved\_caches

--data folder → need more disk space, this is the actual place where the data is stored

--commitlog → need more disk space, before data flushed to disk data is written in commitlog, this is will replay during the restart of crash nodes

Its recommended to create these folders on different drives for faster writes/read

## 

## 

## 

## 2.2 Configuration:

Configuring the Cassandra for distributed mode

###### cd $CASSANDRA\_HOME/conf/

###### vim cassandra.yaml

Set the directories created

search for 'data\_file\_directories'

###### /data\_file\_directories

###### data\_file\_directories: /opt/cassandra/data

search for 'commitlog\_directory'

###### /commitlog\_directory

###### commitlog\_directory: /opt/cassandra/commitlog

search for 'saved\_caches\_directory'

###### /saved\_caches\_directory

###### saved\_caches\_directory: /opt/cassandra/saved\_caches

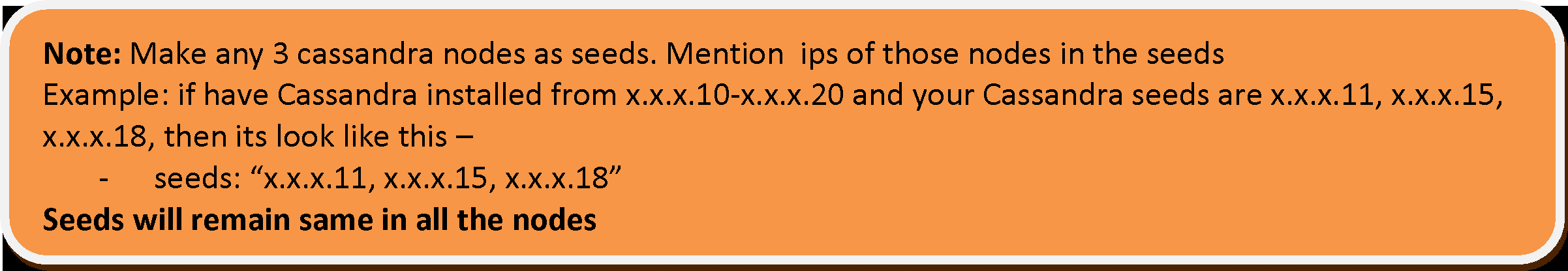
**JVM Options**

jvm.options file has the java parameters (details of this given in the Tuning Java section under Performance

Search seeds

###### /seeds

###### -seeds: "<ip-of-Cassandranode1>,<ip-of-Cassandranode2>,…,<ip-of-Cassandranoden>"

****

**Its recommended to give the odd number of seeds in configuration like 1, 3, 5, etc its not a good practice to make all the nodes as seeds. Seeds are required to share the cluster information**

Search for listen\_address

###### /listen\_address

###### listen\_address: <ip of the local machine>

Search for rpc\_address

###### /rpc\_address

###### rpc\_address: <ip of the local machine>

Search for rpc\_start

###### /start\_rpc

###### start\_rpc: true

Follow the same instruction for installing Cassandra in all the nodes

## 2.3 Network and OS Level:

### Ports:

Communication on the following ports must be enabled so the nodes can coordinate with each other. This will mean setting the correct **Selinux and Iptables permissions**.

|  |  |  |
| --- | --- | --- |
| Port Number | Internal to cassandra / External | Purpose |
| 7000 - 7001 | Internal | Internode communication (If multiple data centers are used then this will need to be opened between the data centers also.  7001 is used when TLS communication is enabled between nodes. |
| 7199 | External | JMX (if needed for remote debugging) |
| 9042 | External | If this port is default for clients to connect to cassandra. *This port is not used in the current deployment.* |
| 16632 | External | This is the port currently configured for communication between the cassandra cluster and external clients. |
| 9060 | External | Default port for thrift communication if thrift protocol is used. |
| 9142 | External | Default port for TLS communication with the nodes in case both TLS and non TLS communication need to be enabled. |

**Check communication of nodes internally and externally**

when the service is started, telnet to the ports between cassandra and client machines to make sure the ports are open between them.

### OS Level:

* Make sure all the binary, data and log folders have the correct ownership and permissions for the user who is going to be running cassandra. In this case will most probably be **toro**.
* Make sure this user has the following OS level params are not set very low otherwise cassandra may not start or may perform badly.
* **Make sure all the clocks of the machines are synchronized using NTP because cassandra uses timestamps for conflict resolution of the latest data. If the clocks are different it will have unexpected effects on data.**

in the /etc/sysctl.conf

Make sure the following settings are set:

###### net.core.rmem\_max = 16777216 net.core.wmem\_max = 16777216 net.core.rmem\_default = 16777216 net.core.wmem\_default = 16777216 net.core.optmem\_max = 40960 net.ipv4.tcp\_rmem = 4096 87380 16777216 net.ipv4.tcp\_wmem = 4096 65536 16777216

**Disable CPU frequency scaling if enabled, otherwise this will reduce throughput**

###### for CPUFREQ in /sys/devices/system/cpu/cpu\*/cpufreq/scaling\_governor do [ -f $CPUFREQ ] || continue echo -n performance > $CPUFREQ done

**Correct OS level settings :**

**check using :**

ulimit -a

Make sure they are at least set to the following:

toro - memlock unlimited  
toro - nofile 100000  
toro - nproc 32768  
toro - as unlimited

also make sure /etc/security/limits.conf has these same settings.

Reference: [Recommended Production Settings](https://docs.datastax.com/en/landing_page/doc/landing_page/recommendedSettings.html)

## 2.4 Starting Cassandra:

Once installation is done in all the nodes,

First start the Cassandra in seed nodes followed by other nodes,

###### cd $CASSANDRA\_HOME

###### ./bin/cassandra

After starting the Cassandra in all the nodes check the status

###### cd $CASSANDRA\_HOME

###### ./bin/nodetool status



## 2.5 Cqlsh & Test:

Once Cassandra is started in all the nodes, let's create the keyspace(Database in RDBMS) and column-family (table in RDBMS) in cassandra

**Start the cqlsh**

cqlsh <ip of the Cassandra node>

**Creating Keyspace and Column family:**

Creating Keyspace

cqlsh>CREATE KEYSPACE test WITH replication = {'class': SimpleStrategy', 'replication\_factor' : 1};



cqlsh>USE test;

Creating the Column-family

cqlsh:test>CREATE TABLE test1(emp\_id int PRIMARY KEY, emp\_name text, emp\_city text);

Insert the data

cqlsh:test>INSERT INTO test1 (emp\_id, emp\_city, emp\_name) VALUES ( 1, 'Rohan', 'Mumbai');

Do SELECT

cqlsh:test> SELECT \* FROM test1;

Output:

emp\_id | dmp\_name | emp\_city

--------+-----------+----------

 1 | Mumbai | Rohan

## 

## 

## 2.6 Importing Schema:

#### Script

Create the script for Creating the Keyspace, Column family and indexes in normal text file. All schema in the script should be in proper cqlsh syntax

**Example:**

CREATE KEYSPACE all\_trade WITH replication = {'class': 'NetworkTopologyStrategy', 'dc1': '4', 'dc2': '2', 'dc3': '2'} AND durable\_writes = true;

dc1, dc2, dc3 stands for the data center names across where this keyspace will be replicated. data\_center 1, 2, 3

durable\_writes = true Means if a crash or server failure occurs before the memtables are flushed to disk, the commit log is replayed on restart to recover any lost data.

###### USE all\_trade

CREATE TABLE print\_history\_detail (

print\_history\_detail\_pk uuid PRIMARY KEY,

create\_by text,

create\_datetime timestamp,

document\_no text,

edition text,

edition\_list set<text>,

int\_created\_date timestamp,

int\_updated\_date timestamp,

lucene text,

print\_result text,

print\_transaction\_type text,

transaction\_id text,

transaction\_type text,

update\_by text,

update\_datetime timestamp

);

save the script in normal text file

#### Import using Source or File

Start the cqlsh

###### cqlsh -p <password> -u <username> <host\_ip> <port> --request-timeout=9000

###### *# After login to the cqlsh session*

###### *SOURCE ‘`*/PATH\_OF\_THE\_SCRIPT/filename*’;*

alternate way is to pass the cql file via the -f or --file switch.

Whenever you plan to do schema level changes that need to replicated across the cluster please use a high session timeout like - 9000 (seconds) setting so you don’t get timeout errors. Even in situations where you get timeouts you can continue as normal because these changes will eventually flow through into the cluster.

Creating a large schema with a lot of indexes may sometimes take upto 15 - 30 mins or more.

helpful arguments to cqlsh

-e To pass a query to execute directly without going into the shell. Useful when you want to automate queries.

-f To run queries or schema creation scripts from a file directly.

-k Select a keyspace directly when logging on

### Export Schema, Data

Export the schema from the other cluster using cqlsh as below

**Schema:**

###### cqlsh -p <password>-u <username> <host\_ip> -e 'Describe <Keyspace\_name>' > '/tmp/filename.cql'

Copy the schema file into the cluster where you want to import and follow the steps given in ***Import using Source or File***

**Data:**Use node tool to take the backup the data as given below

###### nodetool -h <host\_ip> snapshot keyspace\_name

Result will be stored in:

###### data\_file\_directories/keyspace\_name/table\_name-UUID/snapshots/snapshot\_name

# 3. Managing the Cluster & Data Center

## 3.1 Adding the new Node to the Datacenter/Cluster

* Follow the installation steps for cassandra.
* Once the installation is complete don’t start the node, if started stop the node first.

### Configuration

###### vim cassandra.yaml

add the seed node ip details in the seeds option

###### 

###### vim cassandra-rackdc.properties

add the same datacenter and rack details as the other nodes in this cluster.

* Once the configuration is completed start the node.
* Verify using the **nodetool status**.
* After that verification run **nodetool cleanup** on each old nodes wait till it finish one node, this will clean up the keys which no longer belongs to it.

## 3.2 Creating the New Data Center for existing cluster

Follow the installation of Cassandra as above instruction for the new data center

To make the different data center in the same cluster do the below Configuration

Edit the configuration file cassandra-rackdc.properties in both new and old data center.

### Configuration

vim cassandra-rackdc.properties

Change the values below

dc=dc1 //Data Center Name

rack=rack1 //Data Center rack name

* Create at-least one seed in each data center.
* Then add the seeds of other datacenter into the seed nodes of current datacenter.
* And do the same with the old data center. No need to add the seed of the other data node to all the nodes of new data center.
* Once the configuration is set start the cassandra in new data center. Please start the seed node first and then other nodes.
* Verify the cluster by using **nodetool status**. After that run **nodetool cleanup** on each node of the old datacenter/datacenters wait till one node to finish and then run on the next node.

## 3.3 Replacing existing node (non seed)

In case an old node needs to be replaced with a new node it is not a good idea to first remove a node and then add another one this process is slow. Instead you can add a node as a replacement to an existing node. The original node will either be down by itself or you can take it down by using nodetool stopdaemon or stop the cassandra service on the node.

On the new node after installing cassandra, do not start the node.

change the following values in cassandra.yaml

auto\_bootstrap: true

In cassandra.env file add the following :

/cassandra-env.sh

JVM\_OPTS="$JVM\_OPTS -Dcassandra.replace\_address=address\_of\_dead\_node

Start the Cassandra with the below command  
***cassandra -Dcassandra.allow\_unsafe\_replace=true***

**To replace seed nodes -- follow the same process as above (Don’t add the nodes IP in the seeds list) Once the**

## 3.4 Removing a node from the cluster

### Case 1: nodetool decommission

If node is up and running use the above command, before that check the status of the node using the **nodetool status**

Monitor **nodetool netstats** to monitor the progress

### Case 2: nodetool removenode

If node is not running

### Case 3: nodetool assassinate

If node doesn’t stop streaming of data to other nodes

## 3.5 Node IP Change

To change the IP address of a node, simply change the IP of node and then restart Cassandra. If you change the IP address of a seed node, you must update the -seeds parameter in the seed\_provider list in each node's cassandra.yaml file.

# 4. Monitoring:

***nodetool version***

Gives the version of Cassandra

***nodetool status***

Give the status of the node in the datacenter U --> Up N --> Normal D --> Down

***nodetool describecluster***

prints the basic information of the cluster including the name, snitch and partitioner

***nodetool ring keyspace***

Displays node status and information about the ring as determined by the node being queried. This 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; this is a good way to check that every node views the ring the same way.

***nodetool info***

Outputs node information including the token, load info (on disk storage), generation number (times started), uptime in seconds, and heap memory usage. of that particular node

***nodetool gossipinfo***

Shows the gossip information for the cluster. Most useful to check the datasync

***nodetool describering keyspace***

Shows the token ranges for a given keyspace.

***nodetool statusgossip***

Gives the current status of the internode gossip

***nodetool statusthrift***

Gives the current status of the thrift server

# 

# 

# 5. Maintenance:

## 5.1 Good Practices:

**Check the disk space**

***nodetool status***Give the status of the node in the datacenter U --> Up N --> Normal D --> Down

***nodetool describecluster***prints the basic information of the cluster including the name, snitch and partitioner

***nodetool statusgossip***Gives the current status of the internode gossip

***nodetool cfhistograms all\_trade.local\_service\_requests***Give the read/write latency of the Column Family, include row size, column count useful for monitoring the column family

***nodetool netstats***Displays network information such as the status of data streaming operations (bootstrap, repair, move, and decommission) as well as the number of active, pending, and completed commands and responses.

***nodetool tablestats***Gives the information of the read and write latency and info at Keyspace and table level

## 5.2 Maintenance Tools:

###### 

**Nodetool :** [Nodetool Full Reference](http://docs.datastax.com/en/cassandra/latest/cassandra/tools/toolsNodetool.html)

###### nodetool rebuild

Rebuilds data by streaming from other nodes (similar to bootstrap). Use this command to bring up a new data center in an existing cluster.

###### nodetool repair

Begins an anti-entropy node repair operation. If the -pr option is specified, only the first range returned by the partitioner for a node is repaired. This allows you to repair each node in the cluster in succession without duplicating work. Without -pr, all replica ranges that the node is responsible for are repaired. Optionally takes a list of column family names.

##### Best Practice for Nodetool Repair

###### Repair Frequency

If you are using the read/write consistency level that don’t guarantee immediate consistency you will want to do more frequent repair

###### 

###### Repair Scheduling

Minimise the impact of repair by scheduling them off peak hours.

###### Operation Requiring the Repair

When snitch and replication of the keyspace is altered

###### nodetool repair keyspace column family

Begins an anti-entropy node repair operation. If the -pr option is specified, only the first range returned by the partitioner for a node is repaired. This allows you to repair each node in the cluster in succession without duplicating work.

###### nodetool flush

Flushes all memtables for a keyspace to disk, allowing the commit log to be cleared. Optionally takes a list of column family names.

###### nodetool drain

Flushes all memtables for a node and causes the node to stop accepting write operations. Read operations will continue to work. You typically use this command before upgrading a node to a new version of Cassandra or routinely before stopping a node to speed up the restart process. Because this operation writes the current memtables to disk, Cassandra does not need to read through the commit log when you restart the node. If you have durable writes set to false, which is unlikely, there is no commit log and you must drain the node before stopping it to prevent losing data.

###### nodetool cleanup keyspace columnfamily

Triggers the immediate cleanup of keys no longer belonging to this node. This has roughly the same effect on a node that a major compaction does in terms of a temporary increase in disk space usage and an increase in disk I/O. Optionally takes a list of column family names.

###### nodetool rebuild\_index

Cassandra repair mechanism aren’t helpful for keeping the secondary indexes up to date, Because secondary indexes cannot be repaired and there is simple way to check their validity. Rebuild\_index is used to rebuild the indexes from the scratch.

## 5.3 NodeTool Stats Commands

***nodetool cfhistograms all\_trade.local\_service\_requests***

Give the read/write latency of the Column Family, include row size, column count useful for monitoring the column family

all\_trade/local\_service\_requests histograms

Percentile SSTables Write Latency Read Latency Partition Size Cell Count

(micros) (micros) (bytes)

50% 0.00 0.00 0.00 1331 42

75% 0.00 0.00 0.00 6866 60

95% 0.00 0.00 0.00 29521 72

98% 0.00 0.00 0.00 35425 86

99% 0.00 0.00 0.00 61214 86

Min 0.00 0.00 0.00 43 0

Max 0.00 0.00 0.00 379022 86

***nodetool cfstats -H all\_trade.testsnapshot***

Gives the stats of the Keyspace and Column Family like size

Keyspace : all\_trade

Read Count: 0

Read Latency: NaN ms.

Write Count: 16190969

Write Latency: 0.11172297562919181 ms.

Pending Flushes: 0

Table: local\_service\_requests

SSTable count: 10

Space used (live): 1.06 GiB

Space used (total): 1.06 GiB

Space used by snapshots (total): 1.16 GiB

Off heap memory used (total): 1.74 MiB

SSTable Compression Ratio: 0.22971683945664217

Number of keys (estimate): 719181

Memtable cell count: 0

Memtable data size: 0 bytes

Memtable off heap memory used: 0 bytes

Memtable switch count: 24

Local read count: 0

Local read latency: NaN ms

Local write count: 422376

Local write latency: NaN ms

Pending flushes: 0

Percent repaired: 0.0

Bloom filter false positives: 0

Bloom filter false ratio: 0.00000

Bloom filter space used: 1023.16 KiB

Bloom filter off heap memory used: 1023.09 KiB

Index summary off heap memory used: 178.53 KiB

Compression metadata off heap memory used: 582.27 KiB

Compacted partition minimum bytes: 43

Compacted partition maximum bytes: 379022

Compacted partition mean bytes: 6328

Average live cells per slice (last five minutes): NaN

Maximum live cells per slice (last five minutes): 0

Average tombstones per slice (last five minutes): NaN

Maximum tombstones per slice (last five minutes): 0

Dropped Mutations: 0 bytes

----------------

***nodetool netstats***

Displays network information such as the status of data streaming operations (bootstrap, repair, move, and decommission) as well as the number of active, pending, and completed commands and responses.

Mode: NORMAL

Not sending any streams.

Read Repair Statistics:

Attempted: 0

Mismatch (Blocking): 0

Mismatch (Background): 1

Pool Name Active Pending Completed Dropped

Large messages n/a 0 881 5

Small messages n/a 0 7250424 55233

Gossip messages n/a 0 6443640 0

***nodetool tpstats***

To find statics on the thread pools.Top portions indicates how many operations are in what stage. Bottom portion indicates number of dropped messages, when more requests come the node defend itself.

Pool Name Active Pending Completed Blocked All time blocked

MutationStage 0 0 4484788 0 0

ViewMutationStage 0 0 0 0 0

ReadStage 0 0 102870 0 0

RequestResponseStage 0 0 3773718 0 0

ReadRepairStage 0 0 1891 0 0

CounterMutationStage 0 0 0 0 0

MiscStage 0 0 9 0 0

Message type Dropped

READ 1

RANGE\_SLICE 0

\_TRACE 0

HINT 14

MUTATION 2220

COUNTER\_MUTATION 0

BATCH\_STORE 0

BATCH\_REMOVE 0

REQUEST\_RESPONSE 1

PAGED\_RANGE 0

READ\_REPAIR 0

***nodetool tablestats***

Gives the information of the read and write latency and info at Keyspace and table level

Keyspace : all\_trade

Read Count: 0

Read Latency: NaN ms.

Write Count: 16190969

Write Latency: 0.11172297562919181 ms.

Pending Flushes: 0

Table: local\_service\_requests

SSTable count: 10

Space used (live): 1143244430

Space used (total): 1143244430

Space used by snapshots (total): 1241016117

Off heap memory used (total): 1826692

SSTable Compression Ratio: 0.22971683945664217

Number of keys (estimate): 719181

Memtable cell count: 0

Memtable data size: 0

Memtable off heap memory used: 0

Memtable switch count: 24

Local read count: 0

Local read latency: NaN ms

Local write count: 422376

Local write latency: NaN ms

Pending flushes: 0

Percent repaired: 0.0

Bloom filter false positives: 0

Bloom filter false ratio: 0.00000

Bloom filter space used: 1047720

Bloom filter off heap memory used: 1047640

Index summary off heap memory used: 182812

Compression metadata off heap memory used: 596240

Compacted partition minimum bytes: 43

Compacted partition maximum bytes: 379022

Compacted partition mean bytes: 6328

Average live cells per slice (last five minutes): NaN

Maximum live cells per slice (last five minutes): 0

Average tombstones per slice (last five minutes): NaN

Maximum tombstones per slice (last five minutes): 0

Dropped Mutations: 0

----------------

# 6. Security:

## 6.1 Encryption

Edit the cassandra.yaml and change the parameter internode\_encryption from none to dc | rack | all  
vim cassandra.yaml

### Inter-Node Encryption

***internode\_encryption: dc | rack | all***

--dc → will encrypt traffic between data center  
--rack → will encrypt traffic between rack  
--all → will encrypt traffic between racks and data center

### Client to Node Encryption

***client\_encryption\_options:  
enabled: false | true  
optional: false | true***  
--By default both parameters are set to false, Client connections are entirely unencrypted  
--enabled:true && optional:false → all client connection are secured   
--enabled:true && optional:true → both encrypted and unencrypted connections supported using same port, will automatically detected and handled by servers.

### Secure and Unsecure Connections

set → ***client\_encryption\_options:  
enabled: true  
optional: false***   
For will provide the secured connection  
set → ***native\_transport\_port\_ssl: 9142***  
uncomment the parameter, for unsecured connections

## 6.2 User Authentication

Cassandra uses default username cassandra and password cassandra. We can create customize username and password with below steps:

Alter system\_auth keyspace depending upon your Cluster if more than one data centers are configured.

###### ALTER KEYSPACE system\_auth WITH REPLICATION = {'class' : 'NetworkTopologyStrategy', 'dc1' : 3, 'dc2' : 2};

**Set Authenticator as PasswordAuthenticator**

Stop the cluster and edit the following parameters in cassandra.yaml

###### vim $CASSANDRA\_HOME/conf/cassandra.yaml

###### authenticator: PasswordAuthenticator

###### authorizer: CassandraAuthorizer

save and exit, Start Cassandra

## 6.2 Create users

cqlsh using default username and password

###### cqlsh -p cassandra -u cassandra <host-ip>

###### CREATE USER IF NOT EXISTS user\_name WITH PASSWORD 'password' NOSUPERUSER | SUPERUSER

***NOSUPERUSER →*** Don’t have access to CREATE, DELETE and INSERT

***SUPERUSER →*** Have all the access

**User Management Commands**

***LIST USER;***list all the users for the cassandra both NOSUPERUSER and SUPERUSER

***DROP USER;***Drop or delete the user only superuser can run this command

***ALTER USER <username> SUPERUSER;***Modify the user from NOSUPPER user to SUPERUSER

# 

# 7. Performance:

## 7.1 Java Tuning - Garbage Collection

2 options available for cassandra garbage collector are :

1. G1

* Self tuning
* Compacts the heap on the go and does not stop the application while collecting garbage.
* Preferred when the workload is variable and 14GB to 64GB heap space can be allocated.
* In Java 9 CMS will be deprecated.

2. CMS

* CMS is preferred when the heap size is less than 14GB.
* This GC needs to be manually tuned, this takes expertise and time.
* Allocating more memory results in worse performance.
* This is preferred for low latency workloads that do not change very much

The current deployment of cassandra uses G1 as the garbage collector

Setting G1 as garbage collector :

update the jvm.options file --

* Comment all the sections related to CMS
* Comment the line -Xmn800M
* Enable the line : -XX:+UseG1GC

Set the max heap size :

* Heap size is usually between ¼ and ½ of system memory.
* For a node using G1, the Cassandra community recommends a MAX\_HEAP\_SIZE as large as possible, up to 64 GB.
* Do not devote all memory to heap because it is also used for offheap cache and file system cache.

in jvm.options set the heap size as follows :

###### -Xms64G -Xmx64G

Reference : [Java Tuning](https://docs.datastax.com/en/cassandra/3.0/cassandra/operations/opsTuneJVM.html)

## 7.2 Commit log on a different drive

Commit log can be written on a different drive than the date drive. This will give a better throughput on writes. The size of commit log can be controlled via the cassandra.yaml file -- commitlog\_total\_space\_in\_mb the default value is about 8GB. A small commit log would mean more frequent flushes to disk and more IO usage.

## 7.3 Data Compression

Because data written on disk in Cassandra SSTables is immutable. Cassandra does not have a penalty when compression is used. Infact Compression gives performance benefits on all the actions : Data saving of upto 30%, upto 30% faster reads and upto 10% faster writes.

The 3 options available are LZ4, Snappy and Deflate -- LZ4 provides the fastest decompression and thus the fastest read speeds also. There is a direct tradeoff between compression and read speed. More the compression slower the read speed.

Compression can be set on a table level on creation.

###### CREATE TABLE DogTypes ( block\_id uuid, species text, alias text, population varint, PRIMARY KEY (block\_id) ) WITH compression = { 'class' : 'LZ4Compressor' };