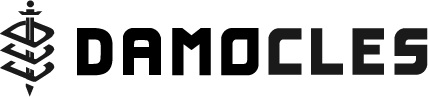
****

**DAMOCLES**

**O & M Guide**

# Contents

Contents

[**Contents 2**](#_Toc492037569)

[**1. Overview 4**](#_Toc492037570)

[**Architecture & Concept 4**](#_Toc492037571)

[**2. Installation 5**](#_Toc492037572)

[**Installation & Configuration 5**](#_Toc492037573)

[**Software path architecture 6**](#_Toc492037574)

[**3. Configuration 7**](#_Toc492037575)

[**Configuration file and description 7**](#_Toc492037576)

[**Configuration update 8**](#_Toc492037577)

[**Configuration compare and verification 9**](#_Toc492037578)

[**4. Connection check 10**](#_Toc492037579)

[**How to check network connection between component 10**](#_Toc492037580)

[**How to check network session 11**](#_Toc492037581)

[**How to check service port 12**](#_Toc492037582)

[**How to check software component and architecture 13**](#_Toc492037583)

[**5. Operation & Maintenance 14**](#_Toc492037584)

[**Operation command 14**](#_Toc492037585)

[**Operation tools 15**](#_Toc492037586)

[**How to check status of each component 16**](#_Toc492037587)

[**How to check database size 17**](#_Toc492037588)

[**How to check performance of each component (respond time , transaction per second etc) 17**](#_Toc492037589)

[**How to check data sync between data node in the cluster 19**](#_Toc492037590)

[**How to get and verify data in the database 20**](#_Toc492037591)

[**Log description and analyzing 21**](#_Toc492037592)

[**Stat description and analyzing 22**](#_Toc492037593)

[**Alarm description and analyzing 23**](#_Toc492037594)

[**Binary version check 24**](#_Toc492037595)

[**Configuration version check 25**](#_Toc492037596)

[**How to monitor software component workload 26**](#_Toc492037597)

[**How to recovery node in case IP change 26**](#_Toc492037598)

[**How to change node member name in cluster 27**](#_Toc492037599)

[**Data management on D11 28**](#_Toc492037600)

[**Log file rotation and log management. 28**](#_Toc492037601)

[**6. Software upgrade 29**](#_Toc492037602)

[**How to check package version 29**](#_Toc492037603)

[**Package version update 30**](#_Toc492037604)

[**How to check compatible between binary and configuration file 30**](#_Toc492037605)

[**7. Schema 31**](#_Toc492037606)

[**Online Update Schema 31**](#_Toc492037607)

[**Offline Update Schema 32**](#_Toc492037608)

[**Schema compare and verification 33**](#_Toc492037609)

[**8. Database 34**](#_Toc492037610)

[**Header provisioning and verification 34**](#_Toc492037611)

[**How to restore database from other node in the same cluster 34**](#_Toc492037612)

[**Database backup and restore 35**](#_Toc492037613)

[**Database Daily backup 36**](#_Toc492037614)

[**Direct access to database 36**](#_Toc492037615)

[**Test query data via D01/D02 37**](#_Toc492037616)

[**9. Expansion 38**](#_Toc492037617)

[**Add new cluster 38**](#_Toc492037618)

[**Add new component 38**](#_Toc492037619)

[**Delete cluster 39**](#_Toc492037620)

[**Delete component 40**](#_Toc492037621)

# Overview

## Architecture & Concept

|  |  |  |
| --- | --- | --- |
| * 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.  |  |  | | --- | --- | | https://lh6.googleusercontent.com/yt-t4y-tUAvPyXeKR89g5mhBypppHAOCegTQp_kTvdkh2-gEQNKvxTdnhiI02XperAbNeLx3XyNJH0YBPkbpcscSo_AvuEhNf7SPTqRQkfJ0oAc-G-hLV8HgAsCGZV4ugcNINY4ETvfxtCq5Wg | * 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 |

# Installation

## Installation & Configuration

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Hardware Requirement**  **The Damocles System are required the any CPU family that supported instruction set as below**   * **AMD64 is the 64 bit version of the x86 instruction set.** * **ASE-NI is a specification for the encryption of electronic data.** * **POPCNT is a population count instruction.**   **Software Requirement**   |  |  |  |  | | --- | --- | --- | --- | | **Software Name** | **Version** | **Description** | **M/O** | | **Linux kernel** | **2.6.32** | **Linux operation system (for el6)** | **M** |   Package Name : Cassandra 3.9  Prerequisites : java version "1.8.0\_121"  **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 |

**Software path architecture**

# Configuration

## Configuration file and description

**3.1 Configuration**

***Configuration path:* /opt/damocles/conf/**

**Default configuration:**

|  |
| --- |
| cassandra.yaml |

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Description** | **Example** |
| data\_file\_directories | Data directory, actual data is stored | data\_file\_directories: /opt/cassandra/data |
| commitlog\_directory | Node recover the data when restarted | commitlog\_directory: /opt/cassandra/commitlog |
| saved\_caches\_directory: | save the key cache | saved\_caches\_directory: /opt/cassandra/saved\_caches |
| -seeds: | Share cluster info | -seeds: ”10.138.32.76,10.138.32.75” |
| listen\_address | Ip of the local machine | listen\_address: 10.138.32.75 |
| rpc\_address | Ip of the local machine | rpc\_address: 10.138.32.76 |
| start\_rpc: | To start the thrift server | start\_rpc:true |

## Configuration update

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
| **Node has to be restarted** |

**Deployment Instruction**

|  |
| --- |
| * Stop the Cassandra before editing the configuration, you can kill the Cassandra pid * Change the configuration * Start the Cassandra. |

**Post Test**

|  |
| --- |
|  |

## Configuration compare and verification

**Objective:**

|  |
| --- |
| **1.** |

**Impact:**

|  |
| --- |
|  |

**Deployment Instruction:**

|  |
| --- |
|  |

**Post Test:**

|  |
| --- |
|  |

# Connection check

## How to check network connection between component

**Objective:**

|  |
| --- |
| **1.** |

**Impact:**

|  |
| --- |
|  |

**Deployment Instruction:**

|  |
| --- |
| Do the telnet from seeds nodes to all other nodes and check the connection between nodes. Please disable the iptables/firewall and set SELINUX off or configure the ip and port in iptables so that nodes can connect and gossip each other. |

**Post Test:**

|  |
| --- |
|  |

## How to check network session

**Objective:**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
|  |

**Post Test**

|  |
| --- |
|  |

## How to check service port

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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. | |

**Post Test**

|  |
| --- |
|  |

## How to check software component and architecture

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
| Run the below command  nodetool status  U --> Up N --> Normal D --> Down |

**Post Test**

|  |
| --- |
|  |

# Operation &Maintenance

## Operation command

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
| 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 removetoken status | force | token  Shows status of a current token removal, forces the the completion of a pending removal, or removes a specified token. This token’s range is assumed by another node and the data is streamed there from the remaining live replicas.    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. |

**Post Test**

|  |
| --- |
|  |

## Operation tools

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
| 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 |

**Post Test**

|  |
| --- |
|  |

## How to check status of each component

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
| 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. |

**Post Test**

|  |
| --- |
|  |

## How to check database size

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
| nodetool cfstats -H keyspace.column\_family  Gives the stats of the Keyspace and Column Family like size |

**Post Test**

|  |
| --- |
|  |

## How to check performance of each component (respond time , transaction per second etc)

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
| 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. |

**Post Test**

|  |
| --- |
|  |

## How to check data sync between data node in the cluster

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
| nodetool cfstats  you can verify that all of the nodes are getting a similar number of writes by looking at the write counts |

**Post Test**

|  |
| --- |
|  |

## How to get and verify data in the database

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
| Use the nodetool cfstats –H keysapcace |

**Post Test**

|  |
| --- |
|  |

## Log description and analyzing

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
| *nodetool getlogginglevels*  Get the runtime logging levels  Setting the log levels  nodetool set logginglevel |

**Post Test**

|  |
| --- |
|  |

## Stat description and analyzing

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
| ***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  ---------------- |

**Post Test**

|  |
| --- |
|  |

## Alarm description and analyzing

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
|  |

**Post Test**

|  |
| --- |
|  |

## Binary version check

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
| **nodetool version** |

**Post Test**

|  |
| --- |
|  |

## Configuration version check

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
|  |

**Post Test**

|  |
| --- |
|  |

## How to monitor software component workload

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
| **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. |

**Post Test**

|  |
| --- |
|  |

## How to recovery node in case IP change

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
| 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. |

**Post Test**

|  |
| --- |
|  |

## How to change node member name in cluster

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
| --Change the cluster name for cassandra node  UPDATE system.local SET cluster\_name = '<new\_cluster\_name>' where key='local';  --Then flush the sstable  ./nodetool flush system  --stop the cassandra node  kill -9 <cassandra-pid>  --Change the cluster name in the cassandra.yaml  cluster\_name = 'new\_cluster\_name'  --start the cassandra node when you change the datacenter of the node  ./cassandra -Dcassandra.ignore\_dc=true |

**Post Test**

|  |
| --- |
|  |

## Data management on D11

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
|  |

**Post Test**

|  |
| --- |
|  |

## Log file rotation and log management.

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
| Cassandra do the log rotation and zip the old log file also  You change the configuration by editing  Vim $CASSANDRA\_HOME/logback.xml |

**Post Test**

|  |
| --- |
|  |

# Software upgrade

## How to check package version

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
|  |

**Post Test**

|  |
| --- |
|  |

## Package version update

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
|  |

**Post Test**

|  |
| --- |
|  |

## How to check compatible between binary and configuration file

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
|  |

**Post Test**

|  |
| --- |
|  |

# Schema

## Online Update Schema

**Objective:**

|  |
| --- |
| **1.** |

**Impact:**

|  |
| --- |
|  |

**Deployment Instruction:**

|  |
| --- |
|  |

**Post Test:**

|  |
| --- |
|  |

## Offline Update Schema

**Objective:**

|  |
| --- |
| **1.** |

**Impact:**

|  |
| --- |
|  |

**Deployment Instruction:**

|  |
| --- |
|  |

**Post Test:**

|  |
| --- |
|  |

## Schema compare and verification

**Objective:**

|  |
| --- |
| **1.** |

**Impact:**

|  |
| --- |
|  |

**Deployment Instruction:**

|  |
| --- |
|  |

**Post Test:**

|  |
| --- |
|  |

# Database

## Header provisioning and verification

**Objective:**

|  |
| --- |
| **1.** |

**Impact:**

|  |
| --- |
|  |

**Deployment Instruction:**

|  |
| --- |
|  |

**Post Test:**

|  |
| --- |
|  |

## How to restore database from other node in the same cluster

**Objective:**

|  |
| --- |
| **1.** |

**Impact:**

|  |
| --- |
|  |

**Deployment Instruction:**

**Post Test**

|  |
| --- |
| **Use** nodetool rebuild |

## Database backup and restore

**Objective:**

|  |
| --- |
| **1.** |

**Impact:**

|  |
| --- |
|  |

**Deployment Instruction:**

|  |  |
| --- | --- |
| 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** |  |

**Post Test:**

|  |
| --- |
|  |

## Database Daily backup

**Objective:**

|  |
| --- |
| **1.** |

**Impact:**

|  |
| --- |
|  |

**Deployment Instruction:**

|  |
| --- |
| **use cqlsh for direct to the database** **cqlsh -p <password> -u <username> <host\_ip>  <port> -k <key\_space> --request-timeout=9000** |

**Post Test:**

|  |
| --- |
|  |

## Direct access to database

**Objective:**

|  |
| --- |
| **1.** |

**Impact:**

|  |
| --- |
|  |

**Deployment Instruction:**

|  |
| --- |
|  |

**Post Test**

|  |
| --- |
|  |

## Test query data via D01/D02

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
|  |

**Post Test**

|  |
| --- |
|  |

# Expansion

## Add new cluster

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
| 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. |

**Post Test**

|  |
| --- |
|  |

## Add new component

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
| * 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. |

**Post Test**

|  |
| --- |
|  |

## Delete cluster

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
| 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  **Do this procedure to all the nodes in the cluster** |

**Post Test**

|  |
| --- |
|  |

## Delete component

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
| 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 |

**Post Test**

|  |
| --- |
|  |