# Elassandra Documentation

*Release 6.8.4.13*

**Strapdata**

**Dec 14, 2021**

# Contents

*Elassandra* closely integrates Elasticsearch in Cassandra.

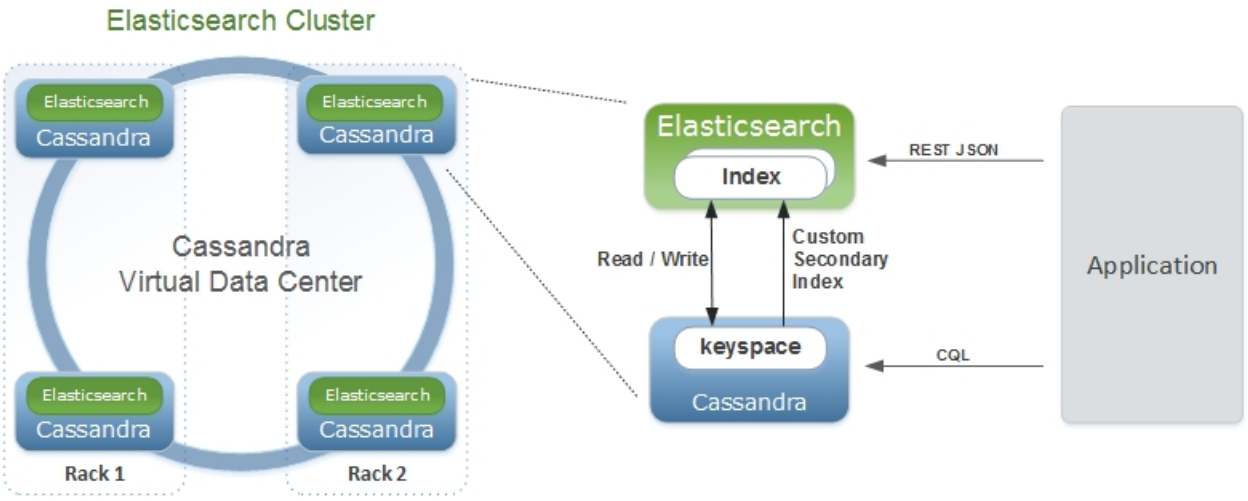
Contents:

CHAPTER 1

# Architecture

Elassandra closely integrates Elasticsearch within Apache Cassandra as a secondary index, allowing near-realtime search with all existing Elasticsearch APIs, plugins and tools like Kibana.

When you index a document, the JSON document is stored as a row in a Cassandra table and synchronously indexed in Elasticsearch.



## 1.1 Concepts Mapping

| Elastic-search | Cassan-dra | Description |
|---|---|---|
| Cluster | Virtual Datacenter | All nodes of a datacenter forms an Elasticsearch cluster |
| Shard | Node | Each Cassandra node is an Elasticsearch shard for each indexed keyspace |
| Index | Keyspace | An Elasticsearch index is backed by a keyspace |
| Type | Table | Each Elasticsearch document type is backed by a Cassandra table. Elasticsearch 6+ support only one document type, named "_doc" by default. |
| Document | Row | An Elasticsearch document is backed by a Cassandra row |
| Field | Cell | Each indexed field is backed by a Cassandra cell (row x column) |
| Object or nested field | User Defined Type | Automatically create a User Defined Type to store an Elasticsearch object |

From an Elasticsearch perspective :

- An Elasticsearch cluster is a Cassandra virtual datacenter.
- Every Elassandra node is a master primary data node.
- Each node only index local data and acts as a primary local shard.
- Elasticsearch data is no longer stored in Lucene indices, but in Cassandra tables.
  - An Elasticsearch index is mapped to a Cassandra keyspace,
  - Elasticsearch document type is mapped to a Cassandra table. Elasticsearch 6+ support only one document type, named "_doc" by default.
  - Elasticsearch document *_id* is a string representation of the Cassandra primary key.
- Elasticsearch discovery now relies on the cassandra gossip protocol. When a node joins or leaves the cluster, or when a schema change occurs, each node updates the nodes status and its local routing table.
- Elasticsearch gateway now store metadata in a Cassandra table and in the Cassandra schema. Metadata updates are played sequentially through a cassandra lightweight transaction. Metadata UUID is the cassandra hostId of the last modifier node.
- Elasticsearch REST and java API remain unchanged.
- Logging is now based on logback as in Cassandra.

From a Cassandra perspective :

- Columns with an ElasticSecondaryIndex are indexed in Elasticsearch.
- By default, Elasticsearch document fields are multivalued, so every field is backed by a list. Single valued document field can be mapped to a basic types by setting 'cql_collection: singleton' in our type mapping. See Elasticsearch document mapping for further details.
- Nested documents are stored using cassandra User Defined Type or map.
- Elasticsearch provides a JSON-REST API to cassandra, see Elasticsearch API.

## 1.2 Durability

All writes to a Cassandra node are recorded both in a memory table and in a commit log. When a memtable flush occurs, it flushes the elasticsearch secondary index on disk. When restarting after a failure, Cassandra replays commitlogs and re-indexes elasticsearch documents that were not flushed by Elasticsearch. This is the reason why elasticsearch translog is disabled in Elassandra.

## 1.3 Shards and Replicas

Unlike Elasticsearch, sharding depends on the number of nodes in the datacenter, and the number of replica is defined by your keyspace Replication Factor . Elasticsearch *numberOfShards* is just information about the number of nodes.

- When adding a new Elassandra node, the Cassandra boostrap process gets some token ranges from the existing ring and pull the corresponding data. Pulled data is automatically indexed and each node update its routing table to distribute search requests according to the ring topology.
- When updating the Replication Factor, you will need to run a nodetool repair <keyspace> on the new node to effectively copy and index the data.
- If a node becomes unavailable, the routing table is updated on all nodes to route search requests on available nodes. The current default strategy routes search requests on primary token ranges' owner first, then to replica nodes when available. If some token ranges become unreachable, the cluster status is in red, otherwise cluster status is in yellow.

After starting a new Elassandra node, data and Elasticsearch indices are distributed on 2 nodes (with no replication).

```
nodetool status twitter
Datacenter: DC1
===============
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address    Load       Tokens  Owns (effective)  Host ID                               ␣
↪    Rack
UN  127.0.0.1  156,9 KB   2       70,3%             74ae1629-0149-4e65-b790-
↪cd25c7406675  RAC1
UN  127.0.0.2  129,01 KB  2       29,7%             e5df0651-8608-4590-92e1-
↪4e523e4582b9  RAC2
```

The routing table now distributes search request on 2 Elassandra nodes covering 100% of the ring.

```
curl -XGET 'http://localhost:9200/_cluster/state/?pretty=true'
{
  "cluster_name" : "Test Cluster",
  "version" : 12,
  "master_node" : "74ae1629-0149-4e65-b790-cd25c7406675",
  "blocks" : { },
  "nodes" : {
    "74ae1629-0149-4e65-b790-cd25c7406675" : {
      "name" : "localhost",
      "status" : "ALIVE",
      "transport_address" : "inet[localhost/127.0.0.1:9300]",
      "attributes" : {
        "data" : "true",
        "rack" : "RAC1",
        "data_center" : "DC1",
        "master" : "true"
```

```
      }
    },
    "e5df0651-8608-4590-92e1-4e523e4582b9" : {
      "name" : "127.0.0.2",
      "status" : "ALIVE",
      "transport_address" : "inet[127.0.0.2/127.0.0.2:9300]",
      "attributes" : {
        "data" : "true",
        "rack" : "RAC2",
        "data_center" : "DC1",
        "master" : "true"
      }
    }
  },
  "metadata" : {
    "version" : 1,
    "uuid" : "e5df0651-8608-4590-92e1-4e523e4582b9",
    "templates" : { },
    "indices" : {
      "twitter" : {
        "state" : "open",
        "settings" : {
          "index" : {
            "creation_date" : "1440659762584",
            "uuid" : "fyqNMDfnRgeRE9KgTqxFWw",
            "number_of_replicas" : "1",
            "number_of_shards" : "1",
            "version" : {
              "created" : "1050299"
            }
          }
        },
        "mappings" : {
          "user" : {
            "properties" : {
              "name" : {
                "type" : "string"
              }
            }
          },
          "tweet" : {
            "properties" : {
              "message" : {
                "type" : "string"
              },
              "postDate" : {
                "format" : "dateOptionalTime",
                "type" : "date"
              },
              "user" : {
                "type" : "string"
              },
              "_token" : {
                "type" : "long"
              }
            }
          }
        },
```

```
        "aliases" : [ ]
      }
    }
  },
  "routing_table" : {
    "indices" : {
      "twitter" : {
        "shards" : {
          "0" : [ {
            "state" : "STARTED",
            "primary" : true,
            "node" : "74ae1629-0149-4e65-b790-cd25c7406675",
            "token_ranges" : [ "(-8879901672822909480,4094576844402756550]" ],
            "shard" : 0,
            "index" : "twitter"
          } ],
          "1" : [ {
            "state" : "STARTED",
            "primary" : true,
            "node" : "e5df0651-8608-4590-92e1-4e523e4582b9",
            "token_ranges" : [ "(-9223372036854775808,-8879901672822909480]",
↪"(4094576844402756550,9223372036854775807]" ],
            "shard" : 1,
            "index" : "twitter"
          } ]
        }
      }
    }
  },
  "routing_nodes" : {
    "unassigned" : [ ],
    "nodes" : {
      "e5df0651-8608-4590-92e1-4e523e4582b9" : [ {
        "state" : "STARTED",
        "primary" : true,
        "node" : "e5df0651-8608-4590-92e1-4e523e4582b9",
        "token_ranges" : [ "(-9223372036854775808,-8879901672822909480]",
↪"(4094576844402756550,9223372036854775807]" ],
        "shard" : 1,
        "index" : "twitter"
      } ],
      "74ae1629-0149-4e65-b790-cd25c7406675" : [ {
        "state" : "STARTED",
        "primary" : true,
        "node" : "74ae1629-0149-4e65-b790-cd25c7406675",
        "token_ranges" : [ "(-8879901672822909480,4094576844402756550]" ],
        "shard" : 0,
        "index" : "twitter"
      } ]
    }
  },
  "allocations" : [ ]
}
```

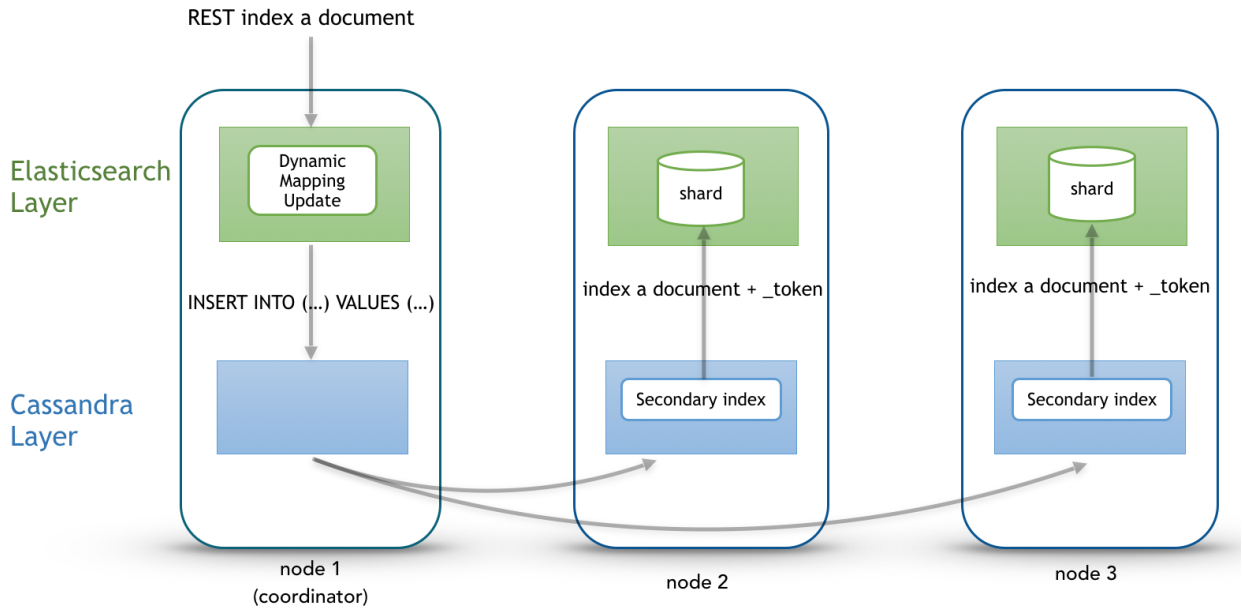

Internally, each node broadcasts its local shard status to the gossip application state X1 ( “twitter”:STARTED ) and its current metadata UUID/version to the application state X2.

**Note:** The payload of the gossip application state X1 maybe huge according to the number of indexes. If this field contains more than 64KB of data, the gossip will fail between nodes. That's why we introduce the *es.compress_x1* system property to compress the payload (default value is **false**). Before enabling this option, be sure that all your cluster nodes are in version 6.2.3.25 (or higher) or 6.8.4.2 (or higher)

```
nodetool gossipinfo
127.0.0.2/127.0.0.2
  generation:1440659838
  heartbeat:396197
  DC:DC1
  NET_VERSION:8
  SEVERITY:-1.3877787807814457E-17
  X1:{"twitter":3}
  X2:e5df0651-8608-4590-92e1-4e523e4582b9/1
  RELEASE_VERSION:2.1.8
  RACK:RAC2
  STATUS:NORMAL,-8879901672822909480
  SCHEMA:ce6febf4-571d-30d2-afeb-b8db9d578fd1
  INTERNAL_IP:127.0.0.2
  RPC_ADDRESS:127.0.0.2
  LOAD:131314.0
  HOST_ID:e5df0651-8608-4590-92e1-4e523e4582b9
localhost/127.0.0.1
  generation:1440659739
  heartbeat:396550
  DC:DC1
  NET_VERSION:8
  SEVERITY:2.220446049250313E-16
  X1:{"twitter":3}
  X2:e5df0651-8608-4590-92e1-4e523e4582b9/1
  RELEASE_VERSION:2.1.8
  RACK:RAC1
  STATUS:NORMAL,-4318747828927358946
  SCHEMA:ce6febf4-571d-30d2-afeb-b8db9d578fd1
  RPC_ADDRESS:127.0.0.1
  INTERNAL_IP:127.0.0.1
  LOAD:154824.0
  HOST_ID:74ae1629-0149-4e65-b790-cd25c7406675
```

## 1.4 Write path

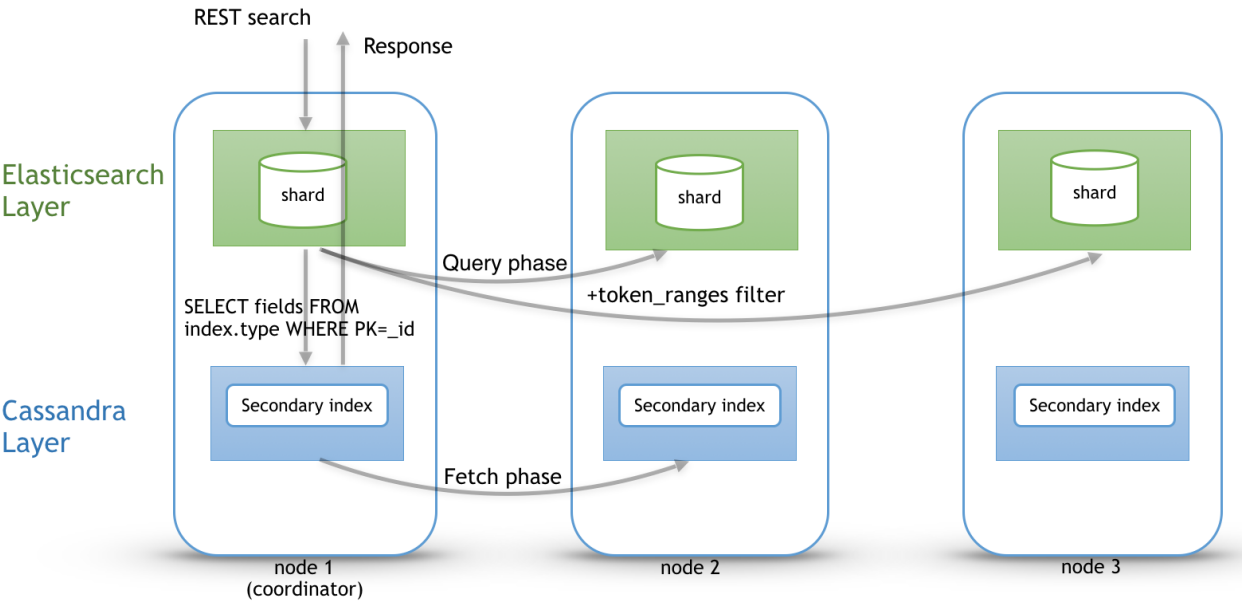
Write operations (Elasticsearch index, update, delete and bulk operations) are converted into CQL write requests managed by the coordinator node. The Elasticsearch document *_id* is converted into an underlying primary key, and the corresponding row is stored on many nodes according to the Cassandra replication factor. Then, on each node hosting this row, an Elasticsearch document is indexed through a Cassandra custom secondary index. Every document includes a _token fields used when for searching.




At index time, every node directly generates the Lucene fields without any JSON parsing overhead, and the Lucene files do not contain any version number, because the version-based concurrency management becomes meaningless in a multi-master database like Cassandra.

## 1.5 Search path

Search request is done in two phases. First, the query phase, the coordinator node adds a token_ranges filter to the query and broadcasts a search request to all nodes. This token_ranges filter covers the entire Cassandra ring and avoids duplicating results. Secondly, in the fetch phases, the coordinator fetches the required fields by issuing a CQL request in the underlying Cassandra table, and builds the final JSON response.




By default, an Elassandra search request is sub-queried to all nodes in the datacenter. With the

`RandomSearchStrategy`, the coordinator node requests the minimum of nodes to cover the whole Cassandra ring depending on the Cassandra Replication Factor, so this reduce the overall cost of a search and lower the CPU usage of nodes. For example, if you have a datacenter with four nodes and a replication factor of two, only two nodes will be requested with simplified token_ranges filters (adjacent token ranges are automatically merged).

Additionally, as these token_ranges filters only change when the datacenter topology changes (for example when a node is down or when adding a new node), Elassandra introduces a token_range bitset cache for each Lucene segment. With this cache, out of range documents are seen as deleted documents at the Lucene segment layer for subsequent queries using the same token_range filter. It drastically improves the search performances.

The CQL fetch overhead can also be mitigated by using keys and rows Cassandra caching, eventually using the off-heap caching features of Cassandra.

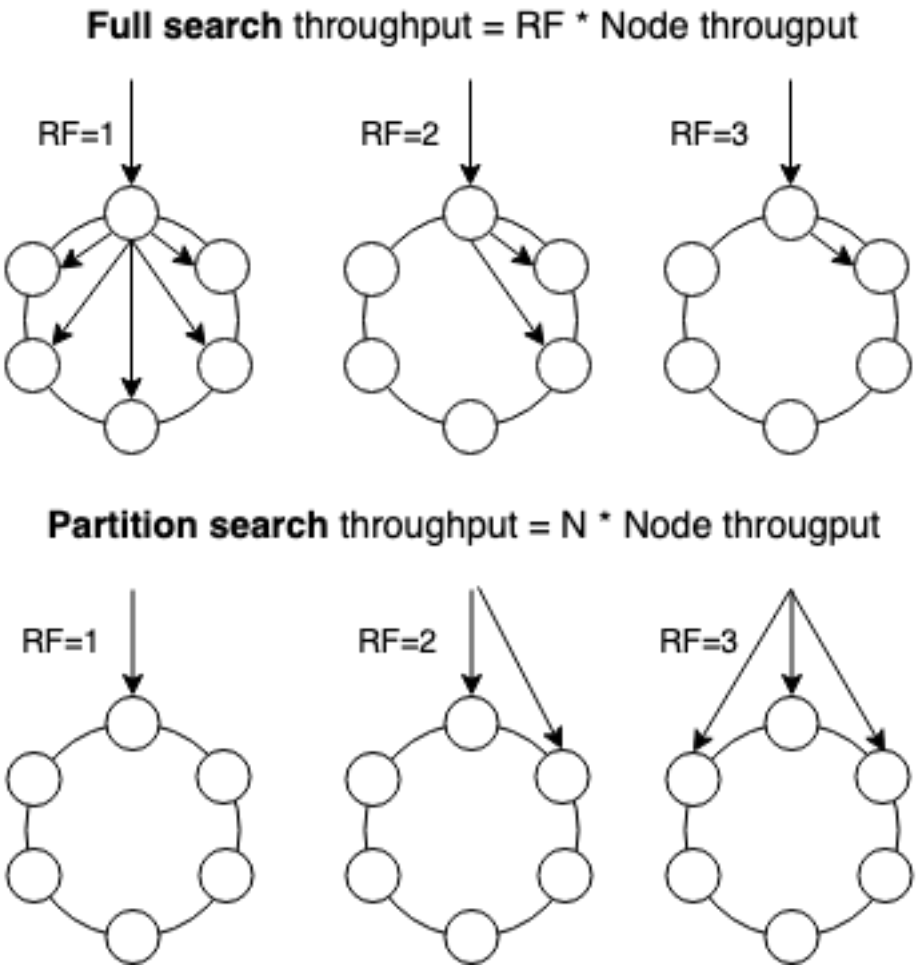
Finally, you can provide the Cassandra partition key as the routing parameter to route your search request to a Cassandra replica.

```
GET /books/_search?pretty&routing=xxx
{
        "query":{ ... }
}
```

Elasticsearch query over CQL automatically adds routing when partition key is present:

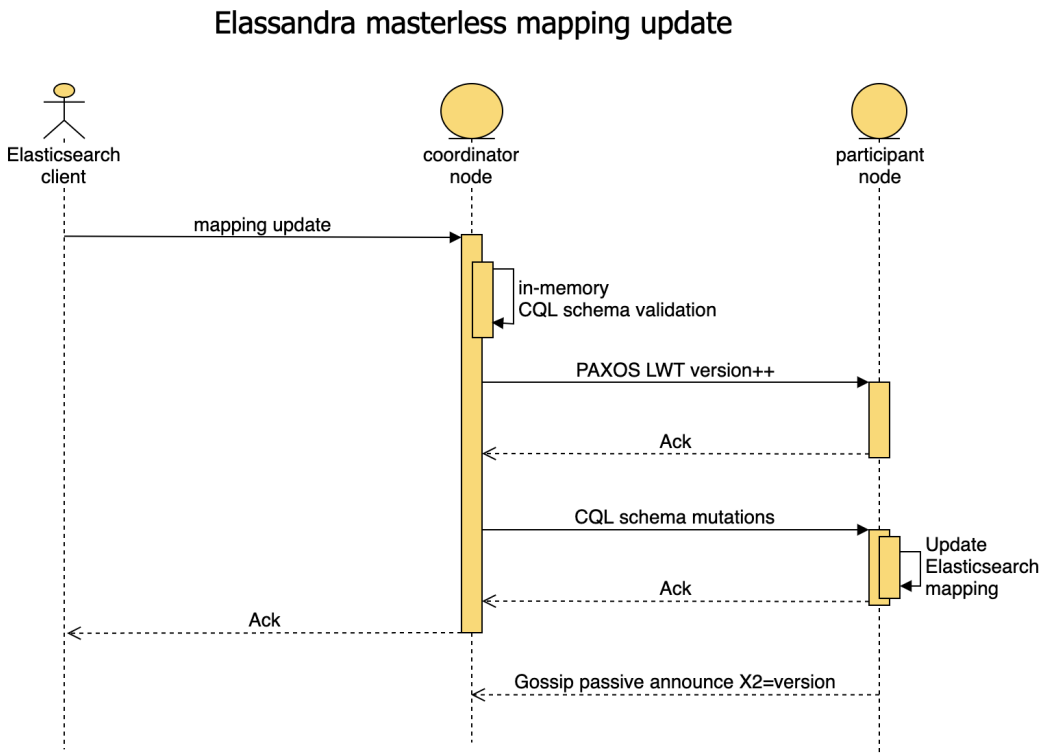
```
SELECT * FROM books WHERE id='xxx' AND es_query='{"query":{ ...}}'
```

Using partition search is definitely more scalable than full search on a datacenter:




## 1.6 Mapping and CQL schema management

Elassandra has no master node to manage the Elasticsearch mapping and all nodes can update the Elasticsearch mapping. In order to manage concurrent simultaneous mapping and CQL schema changes, Elassandra plays a PAXOS transaction to update the current Elasticsearch metadata version in the Cassandra table **elastic_admin.metadata_log** tracking all mapping updates. Here is the overall mapping update process including a PAXOS Light Weight Transaction and a CQL schema update:




Once the PAXOS transaction succeed, Elassandra coordinator node applies a batched-atomic (1) CQL schema update broadcasted to all nodes. Version number increase by one on each mapping update, and the **elastic_admin.metadata_log** tracks metadata update events, as shown in the following example.

```
SELECT * FROM elastic_admin.metadata_log;

 cluster_name  | v    | version | owner                                | source
↪                                  | ts
---------------+------+---------+--------------------------------------+--------------
↪----------------------------------+--------------------------------
 trial_cluster | 4545 |    4545 | fc11f3b2-8280-4a69-af45-aaf1e9d336ae | delete-index
↪[[index1574/q_xsELcBRFO2NITy62b6tg]] | 2019-09-16 15:06:31.054000+0000
 trial_cluster | 4544 |    4545 | a1fdf359-a0a0-4fd1-ad6c-1d2605248560 | delete-index
↪[[index1575/nsuu0CFiTkC2EH2gvLkXHw]] | 2019-09-16 15:02:44.511000+0000
 trial_cluster | 4543 |    4545 | a1fdf359-a0a0-4fd1-ad6c-1d2605248560 | delete-index
↪[[index2000/mEC5Bbx4T9m1ahi9LD1tIw]] | 2019-09-16 14:57:54.443000+0000
 trial_cluster | 4542 |    4545 | a1fdf359-a0a0-4fd1-ad6c-1d2605248560 | delete-index
↪[[index1576/sVaT7vjWS4e2ukuLoQNo_w]] | 2019-09-16 14:56:56.561000+0000
 trial_cluster | 4541 |    4545 | a1fdf359-a0a0-4fd1-ad6c-1d2605248560 | delete-index
↪[[index1570/DPmyeSB4Siyro9wbyEk9NA]] | 2019-09-16 14:55:59.507000+0000
 trial_cluster | 4540 |    4545 | a1fdf359-a0a0-4fd1-ad6c-1d2605248560 |
↪         cql-schema-mapping-update | 2019-09-16 14:54:06.280000+0000
 trial_cluster | 4539 |    4545 | a1fdf359-a0a0-4fd1-ad6c-1d2605248560 |
↪init table elastic_admin.metadata_log | 2019-09-16 14:44:57.243000+0000
```

**Tip:** The **elastic_admin.metadata_log** table contains one entry per metadata update event with a version number (column v), the host ID of the coordinator node (owner), the event origin (source) and timestamp (ts). If PAXOS update timeout occurs, Elassandra reads this table to transparently recover. If your cluster issues thousands of mapping updates, you should periodically delete old entries with a CQL range delete or add a default TTL to avoid an infinite

growth.

All nodes sharing the same Elasticsearch mapping should have the same X2 value and you can check this with **nodetool gossipinfo**, as show here with X2 = e5df0651-8608-4590-92e1-4e523e4582b9/1.

```
nodetool gossipinfo
127.0.0.2/127.0.0.2
  generation:1440659838
  heartbeat:396197
  DC:DC1
  NET_VERSION:8
  SEVERITY:-1.3877787807814457E-17
  X1:{"twitter":3}
  X2:e5df0651-8608-4590-92e1-4e523e4582b9/1
  RELEASE_VERSION:2.1.8
  RACK:RAC2
  STATUS:NORMAL,-8879901672822909480
  SCHEMA:ce6febf4-571d-30d2-afeb-b8db9d578fd1
  INTERNAL_IP:127.0.0.2
  RPC_ADDRESS:127.0.0.2
  LOAD:131314.0
  HOST_ID:e5df0651-8608-4590-92e1-4e523e4582b9
localhost/127.0.0.1
  generation:1440659739
  heartbeat:396550
  DC:DC1
  NET_VERSION:8
  SEVERITY:2.220446049250313E-16
  X1:{"twitter":3}
  X2:e5df0651-8608-4590-92e1-4e523e4582b9/1
  RELEASE_VERSION:2.1.8
  RACK:RAC1
  STATUS:NORMAL,-4318747828927358946
  SCHEMA:ce6febf4-571d-30d2-afeb-b8db9d578fd1
  RPC_ADDRESS:127.0.0.1
  INTERNAL_IP:127.0.0.1
  LOAD:154824.0
  HOST_ID:74ae1629-0149-4e65-b790-cd25c7406675
```

(1) All CQL changes involved by the Elasticsearch mapping update (CQL types and tables create/update) and the new Elasticsearch cluster state are applied in a *SINGLE* CQL schema update. The Elasticsearch metadata are stored in a binary format in the CQL schema as table extensions, stored in **system_schema.tables**, column **extensions** of type *frozen<map<text, blob>>*.

Elasticsearch metadata (indices, templates, aliases, ingest pipelines...) without document mapping is stored in **elastic_admin.metdata_log** table extensions:

```
admin@cqlsh> select keyspace_name, table_name, extensions from system_schema.tables␣
↪where keyspace_name='elastic_admin';

 keyspace_name | table_name   | extensions
---------------+--------------+------------------------------------------------------
↪----------------------------------------------------------------------------------
↪----------------------------------------------------------------------------------
↪----------------------------------------------------------------------------------
↪----------------------------------------------------------------------------------
↪----------------------------------------------------------------------------------
↪----------------------------------------------------------------------------------
↪----------------------------------------------------------------------------------
↪----------------------------------------------------------------------------------
↪--------------------------------
```

```
 elastic_admin | metadata_log | {'metadata':␣
↪0x3a290a05fa886d6574612d64617461fa8676657273696f6ec88b636c75737465725f7575696463663035613336343 62d6
↪ 'owner': 0xf05a3646ce6f4dfd9d7d592593b1eeee, 'version': 0x0000000000000004}

(1 rows)
```

For each document type backed by a Cassandra table, index metadata including the mapping is stored as an extension, where extension key is elastic_admin/<index_name> :

```
admin@cqlsh> select keyspace_name, table_name, extensions from system_schema.tables␣
↪where keyspace_name='myindex';

 keyspace_name | table_name | extensions
---------------+------------+--------------------------------------------------------------
↪------------------------------------------------------------------------------------------
↪----------------------------------------------------------------------
       myindex |     mytype | {'elastic_admin/myindex':␣
↪0x44464c00aa56caad2ca92c4855b2aa562a28ca2f482d2ac94c2d06f1d2f2f341144452a924b5a244494 7292d33352705 2
↪
```

When snapshoting a keyspace or a table (ex: nodetool snapshot <keyspace>), Cassandra also backups the CQL schema (in <snapshot_dir>/schema.cql) including the Elasticsearch index metadata and mapping, and thus, restoring the CQL schema for an indexed table also restore the associated Elasticsearch index definition in the current cluster state.

---

**Tip:** You can decode the SIMLE encoded mapping stored in table extensions by using the **elassandra-cli** utility, see *Tooling*.

---

CHAPTER 2

# Quick Start

## 2.1 Start your cluster

Start a docker-based Elassandra cluster using docker-compose file **ci/docker-compose.yml**:

```
version: '2.4'
services:
  seed_node:
    image: "docker.io/strapdata/elassandra:6.8.4.3"
    environment:
      - "JVM_OPTS=-Dcassandra.custom_query_handler_class=org.elassandra.index.
↪ElasticQueryHandler"
      - "MAX_HEAP_SIZE=1200m"
      - "HEAP_NEWSIZE=300m"
      - "CASSANDRA_CGROUP_MEMORY_LIMIT=true"
      - "DEBUG=true"
    cap_add:
      - IPC_LOCK
    ulimits:
      memlock: -1
    mem_limit: 2000m
    ports:
      - "9042:9042"
      - "9200:9200"
  node:
    image: "docker.io/strapdata/elassandra:6.8.4.3"
    environment:
      - "JVM_OPTS=-Dcassandra.custom_query_handler_class=org.elassandra.index.
↪ElasticQueryHandler"
      - "MAX_HEAP_SIZE=1200m"
      - "HEAP_NEWSIZE=300m"
      - "CASSANDRA_CGROUP_MEMORY_LIMIT=true"
      - "CASSANDRA_SEEDS=seed_node"
      - "DEBUG=true"
    links:
```

```
      - seed_node
    cap_add:
      - IPC_LOCK
    ulimits:
      memlock: -1
    mem_limit: 2000m

  kibana:
    image: docker.elastic.co/kibana/kibana-oss:6.8.4
    environment:
      - "ELASTICSEARCH_URL=http://seed_node:9200"
    ports:
      - "5601:5601"
    mem_limit: 500m
```

Start containers and scale up the elassandra cluster :

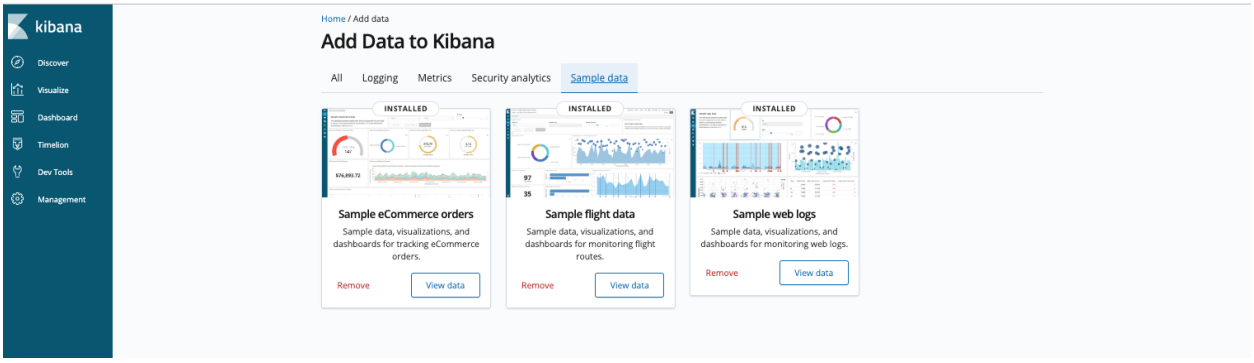
```
docker-compose --project-name test -f docker-compose.yml up -d --scale node=0
docker-compose --project-name test -f docker-compose.yml up -d --scale node=1
```

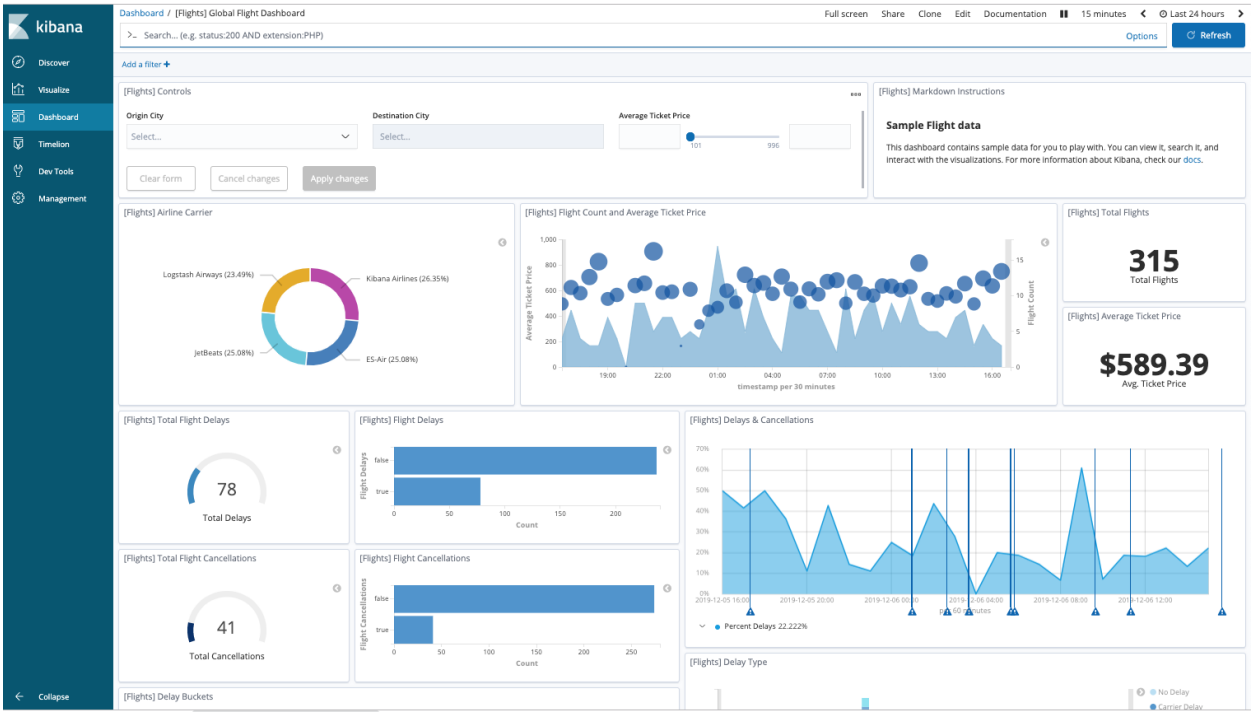
Check the cassandra nodes status:

```
docker exec -i test_seed_node_1 nodetool status
Datacenter: DC1
===============
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load       Tokens       Owns (effective)  Host ID
↪           Rack
UN  172.19.0.3  8.02 MiB   8            61.1%             14ac0af0-e51a-4f98-b57d-
↪7b012b584d84  r1
UN  172.19.0.4  3.21 MiB   8            38.9%             fec10e1f-4191-41d5-9a58-
↪7abcccc5972f  r1
```

## 2.2 Import sample data

After about 35 secondes to start Elassandra on node0, you should have access to kibana at http://localhost:5601, and you can insert sample data and browse sample dashboards.






View the kibana sample data in Cassandra:

```
docker exec -it test_seed_node_1 cqlsh

Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.5 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh> select * from kibana_sample_data_logs."_doc" limit 3;

 _id                  | agent
↪                                                  | bytes    | clientip            |
↪extension | geo
↪                              | host                                  | index
↪       | ip                   | machine                                    | memory
↪| message
↪
↪                                                                                    |
↪phpmemory | referer                                                                          |
↪request                                                                 | response | tags
↪               | timestamp                                 | url
↪                                                                            | utc_time
----------------------+-------------------------------------------------------------------------------
↪-----------------------------------------------+---------+----------------------+--------
↪----+-------------------------------------------------------------------------------------
↪--------------------+-----------------------------------------+-----------------------------
↪+--------------------+-----------------------------------------------+--------------+------
↪-------------------------------------------------------------------------------------------
↪-------------------------------------------------------------------------------------------
↪-------------------------------------------------------------------+------------+----
↪-----------------------------------------------------------------+------------------
↪-----------------------------------------------+----------+-------------------------+--
↪-------------------------------------+-----------------------------------------------------
↪-----------------------------------------------+-----------------------------------
↪---
```

```
 _ISA224B3U12qk8z3Q78 | ['Mozilla/5.0 (X11; Linux i686) AppleWebKit/534.24 (KHTML,␣
↪like Gecko) Chrome/11.0.696.50 Safari/534.24'] |  [6465] | ['236.132.209.242'] |  ␣
↪  [''] | [{srcdest: ['CA:MY'], src: ['CA'], coordinates: [{lat: 43.10318, lon: -78.
↪70335}], dest: ['MY']}] | ['elastic-elastic-elastic.org'] | ['kibana_sample_data_
↪logs'] | ['236.132.209.242'] |  [{os: ['win 7'], ram: [18253611008]}] | [2.586e+05]␣
↪| ['236.132.209.242 - - [2018-08-26T10:51:51.506Z] "GET /people/type:astronauts/
↪name:john-david-f-bartoe/profile HTTP/1.1" 200 6465 "-" "Mozilla/5.0 (X11; Linux␣
↪i686) AppleWebKit/534.24 (KHTML, like Gecko) Chrome/11.0.696.50 Safari/534.24"'] | ␣
↪[258600] | ['http://www.elastic-elastic-elastic.com/success/john-o-creighton'] | ['/
↪people/type:astronauts/name:john-david-f-bartoe/profile'] |  ['200'] | ['success',
↪'security'] | ['2019-12-29 10:51:51.506000+0000'] | ['https://elastic-elastic-
↪elastic.org/people/type:astronauts/name:john-david-f-bartoe/profile'] | ['2018-08-
↪26 10:51:51.506000+0000']
 L4OA224B3U12qk8zxvxM |                                  ['Mozilla/5.0 (X11; Linux x86_
↪64; rv:6.0a1) Gecko/20110421 Firefox/6.0a1'] |  [9842] |     ['1.8.196.147'] |     ␣
↪[''] | [{srcdest: ['DE:CN'], src: ['DE'], coordinates: [{lat: 35.10117, lon: -75.
↪96595}], dest: ['CN']}] |             ['www.elastic.co'] | ['kibana_sample_data_
↪logs'] |     ['1.8.196.147'] | [{os: ['win xp'], ram: [12884901888]}] |        null␣
↪|                                                                              [
↪'1.8.196.147 - - [2018-08-05T16:38:26.871Z] "GET /enterprise HTTP/1.1" 200 9842 "-"
↪"Mozilla/5.0 (X11; Linux x86_64; rv:6.0a1) Gecko/20110421 Firefox/6.0a1"'] |     ␣
↪null |                    ['http://facebook.com/warning/stephen-robinson'] |       ␣
↪                                  ['/enterprise'] |  ['200'] |     ['success',
↪'info'] | ['2019-12-08 16:38:26.871000+0000'] |                                    ␣
↪           ['https://www.elastic.co/downloads/enterprise'] | ['2018-08-05 16:38:26.
↪871000+0000']
 R4SA224B3U12qk8z4hPC |                                  ['Mozilla/5.0 (X11; Linux x86_
↪64; rv:6.0a1) Gecko/20110421 Firefox/6.0a1'] | [19561] |    ['190.43.53.42'] |   [
↪'rpm'] | [{srcdest: ['BD:CN'], src: ['BD'], coordinates: [{lat: 36.28002, lon: -80.
↪78607}], dest: ['CN']}] |        ['artifacts.elastic.co'] | ['kibana_sample_data_
↪logs'] |    ['190.43.53.42'] |   [{os: ['win 8'], ram: [9663676416]}] |        null␣
↪|                                              ['190.43.53.42 - - [2018-08-
↪30T12:40:40.089Z] "GET /beats/metricbeat/metricbeat-6.3.2-i686.rpm HTTP/1.1" 200␣
↪19561 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:6.0a1) Gecko/20110421 Firefox/6.0a1"
↪'] |      null |   ['http://www.elastic-elastic-elastic.com/success/pavel-belyayev
↪'] |             ['/beats/metricbeat/metricbeat-6.3.2-i686.rpm'] |  ['200'] |     [
↪'success', 'info'] | ['2020-01-02 12:40:40.089000+0000'] |           ['https://
↪artifacts.elastic.co/downloads/beats/metricbeat/metricbeat-6.3.2-i686.rpm'] | [
↪'2018-08-30 12:40:40.089000+0000']

(3 rows)
```

## 2.3 Create an Elasticsearch index from a Cassandra table

Use the cassandra CQLSH to create a cassandra Keyspace, a User Defined Type, a Table and add two rows:

```
docker exec -i test_seed_node_1 cqlsh <<EOF
CREATE KEYSPACE IF NOT EXISTS test WITH replication = {'class':
↪'NetworkTopologyStrategy', 'DC1': 1};
CREATE TYPE IF NOT EXISTS test.user_type (first text, last text);
CREATE TABLE IF NOT EXISTS test.docs (uid int, username frozen<user_type>, login text,
↪ PRIMARY KEY (uid));
INSERT INTO test.docs (uid, username, login) VALUES (1, {first:'vince',last:'royer'},
↪'vroyer');
INSERT INTO test.docs (uid, username, login) VALUES (2, {first:'barthelemy',last:
↪'delemotte'}, 'barth');
```

```
EOF
```

Create an Elasticsearch index from the Cassandra table schema by discovering the CQL schema:

```
curl -XPUT -H 'Content-Type: application/json' http://localhost:9200/test -d'{
↪"mappings":{"docs":{"discover":".*"}}}'
{"acknowledged":true,"shards_acknowledged":true,"index":"test"}
```

This command discovers all column matching the provided regular expression, and creates the Eslasticsearch index.

## 2.4 Create an Elasticsearch index from scratch

Elassandra automatically generates the underlying CQL schema when creating an index or updating the mapping with a new field.

```
curl -XPUT -H 'Content-Type: application/json' http://localhost:9200/test2 -d'{
   "mappings":{
      "docs":{
         "properties": {
            "first": {
               "type":"text"
            },
            "last": {
               "type":"text",
               "cql_collection":"singleton"
            }
         }
      }
   }
}'
```

Generated CQL schema:

```
cqlsh> desc KEYSPACE test2;

CREATE KEYSPACE test2 WITH replication = {'class': 'NetworkTopologyStrategy', 'DC1':
↪'1'}  AND durable_writes = true;

CREATE TABLE test2.docs (
    "_id" text PRIMARY KEY,
    first list<text>,
    last text
) WITH bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.
↪SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.
↪compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
```

```
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';
CREATE CUSTOM INDEX elastic_docs_idx ON test2.docs () USING 'org.elassandra.index.
↪ExtendedElasticSecondaryIndex';
```

## 2.5 Search for a document

Search for a document through the Elasticsearch API:

```
curl "http://localhost:9200/test/_search?pretty"
{
  "took" : 10,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : 2,
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "test",
        "_type" : "docs",
        "_id" : "1",
        "_score" : 1.0,
        "_source" : {
          "uid" : 1,
          "login" : "vroyer",
          "username" : {
            "last" : "royer",
            "first" : "vince"
          }
        }
      },
      {
        "_index" : "test",
        "_type" : "docs",
        "_id" : "2",
        "_score" : 1.0,
        "_source" : {
          "uid" : 2,
          "login" : "barth",
          "username" : {
            "last" : "delemotte",
            "first" : "barthelemy"
          }
        }
      }
    ]
  }
}
```

In order to search a document through the CQL driver, add the following two dummy columns in your table schema.

Then, execute an Elasticsearch nested query. The dummy columns allow you to specify the targeted index when index name does not match the keyspace name.

```
docker exec -i test_seed_node_1 cqlsh <<EOF
ALTER TABLE test.docs ADD es_query text;
ALTER TABLE test.docs ADD es_options text;
cqlsh> SELECT uid, login, username FROM test.docs WHERE es_query='{ "query":{"nested":
↪{"path":"username","query":{"term":{"username.first":"barthelemy"}}}}}' AND es_
↪options='indices=test' ALLOW FILTERING;
uid | login | username
----+-------+-----------------------------------------
  2 | barth | {first: 'barthelemy', last: 'delemotte'}

(1 rows)
```

## 2.6 Manage Elasticsearch indices

Get the Elasticsearch cluster state:

```
curl "http://localhost:9200/_cluster/state?pretty"
{
  "cluster_name" : "Test Cluster",
  "compressed_size_in_bytes" : 579,
  "version" : 8,
  "state_uuid" : "mrE5raXOQO2SVA8AROJqwQ",
  "master_node" : "25457162-c5ef-44fa-a46b-a96434aae319",
  "blocks" : { },
  "nodes" : {
    "25457162-c5ef-44fa-a46b-a96434aae319" : {
      "name" : "172.17.0.2",
      "status" : "ALIVE",
      "ephemeral_id" : "25457162-c5ef-44fa-a46b-a96434aae319",
      "transport_address" : "172.17.0.2:9300",
      "attributes" : {
        "rack" : "r1",
        "dc" : "DC1"
      }
    }
  },
  "metadata" : {
    "version" : 1,
    "cluster_uuid" : "25457162-c5ef-44fa-a46b-a96434aae319",
    "templates" : { },
    "indices" : {
      "test" : {
        "state" : "open",
        "settings" : {
          "index" : {
            "creation_date" : "1553512833429",
            "number_of_shards" : "1",
            "number_of_replicas" : "0",
            "uuid" : "BOolxI89SqmrcbK7KM4sIA",
            "version" : {
              "created" : "6020399"
            },
            "provided_name" : "test"
```

```
          }
        },
        "mappings" : {
          "docs" : {
            "properties" : {
              "uid" : {
                "cql_partition_key" : true,
                "cql_primary_key_order" : 0,
                "type" : "integer",
                "cql_collection" : "singleton"
              },
              "login" : {
                "type" : "keyword",
                "cql_collection" : "singleton"
              },
              "username" : {
                "cql_udt_name" : "user_type",
                "type" : "nested",
                "properties" : {
                  "last" : {
                    "type" : "keyword",
                    "cql_collection" : "singleton"
                  },
                  "first" : {
                    "type" : "keyword",
                    "cql_collection" : "singleton"
                  }
                },
                "cql_collection" : "singleton"
              }
            }
          }
        },
        "aliases" : [ ],
        "primary_terms" : {
          "0" : 0
        },
        "in_sync_allocations" : {
          "0" : [ ]
        }
      }
    },
    "index-graveyard" : {
      "tombstones" : [ ]
    }
  },
  "routing_table" : {
    "indices" : {
      "test" : {
        "shards" : {
          "0" : [
            {
              "state" : "STARTED",
              "primary" : true,
              "node" : "25457162-c5ef-44fa-a46b-a96434aae319",
              "relocating_node" : null,
              "shard" : 0,
              "index" : "test",
```

```
              "token_ranges" : [
                "(-9223372036854775808,9223372036854775807]"
              ],
              "allocation_id" : {
                "id" : "dummy_alloc_id"
              }
            }
          ]
        }
      }
    }
  },
  "routing_nodes" : {
    "unassigned" : [ ],
    "nodes" : {
      "25457162-c5ef-44fa-a46b-a96434aae319" : [
        {
          "state" : "STARTED",
          "primary" : true,
          "node" : "25457162-c5ef-44fa-a46b-a96434aae319",
          "relocating_node" : null,
          "shard" : 0,
          "index" : "test",
          "token_ranges" : [
            "(-9223372036854775808,9223372036854775807]"
          ],
          "allocation_id" : {
            "id" : "dummy_alloc_id"
          }
        }
      ]
    }
  },
  "snapshots" : {
    "snapshots" : [ ]
  },
  "restore" : {
    "snapshots" : [ ]
  },
  "snapshot_deletions" : {
    "snapshot_deletions" : [ ]
  }
}
```


Get Elasticsearch index information:

```
curl "http://localhost:9200/_cat/indices?v"
health status index uuid                   pri rep docs.count docs.deleted store.size␣
↪pri.store.size
green  open   test  BOolxI89SqmrcbK7KM4sIA   1   0          4            0      4.1kb␣
↪         4.1kb
```

Delete the Elasticserach index (does not delete the underlying Cassandra table by default) :

```
curl -XDELETE http://localhost:9200/test
{"acknowledged":true}
```

## 2.7 Cleanup the cluster

Stop all containers:

```
docker-compose --project-name test -f docker-compose.yml stop
```

## 2.8 Docker Troubleshooting

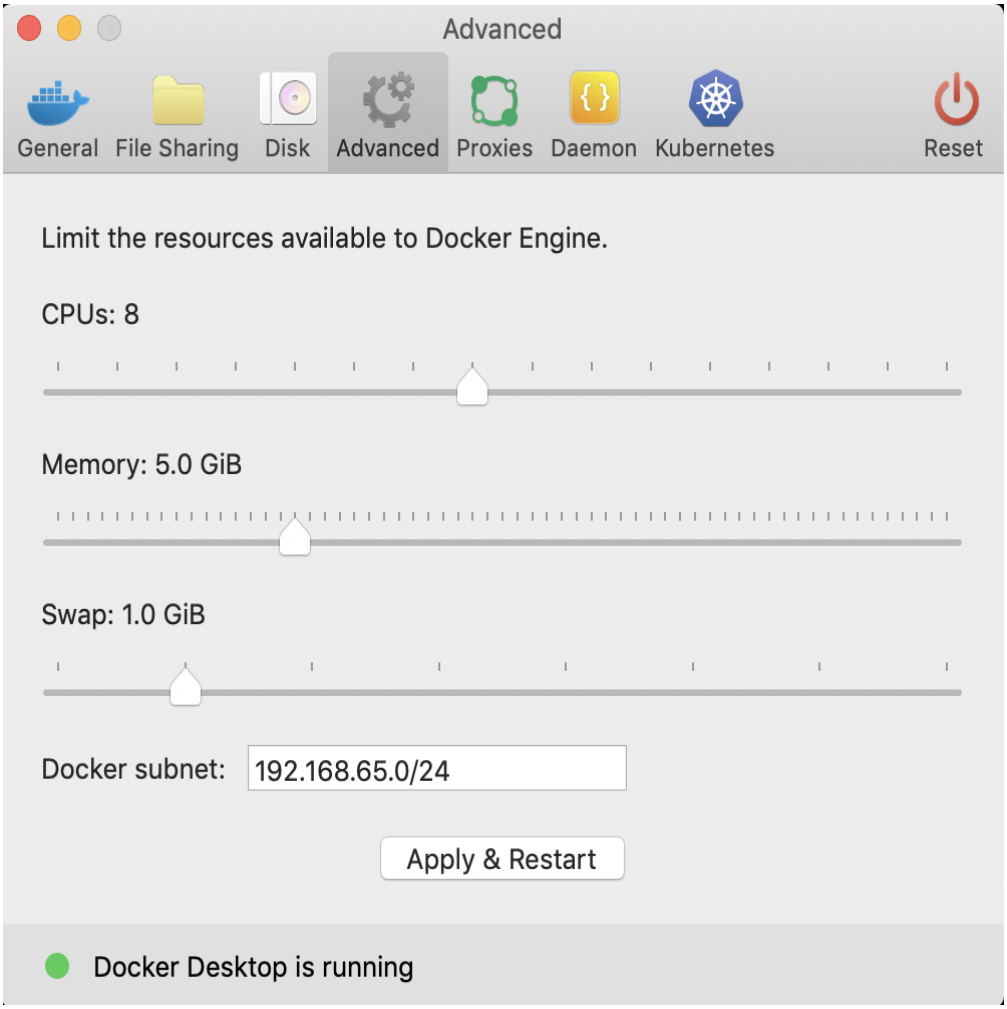
Because each Elassandra node require at least about 1.5Gb of RAM to work properly, small docker configuration can have memory issues. Here is 2 nodes configuration using 4.5Gb RAM.

```
docker stats
CONTAINER ID        NAME                CPU %               MEM USAGE / LIMIT     MEM
↪%                  NET I/O             BLOCK I/O           PIDS
ab91e8cf806b        test_node_1         1.53%               1.86GiB / 1.953GiB    95.
↪23%                10.5MB / 2.89MB     26MB / 89.8MB       113
8fe5f0cd6c38        test_seed_node_1    1.41%               1.856GiB / 1.953GiB   95.
↪01%                14.3MB / 16.3MB     230MB / 142MB       144
68cdabd681c6        test_kibana_1       1.25%               148.5MiB / 500MiB     29.
↪70%                5.97MB / 11.8MB     98.4MB / 4.1kB      11
```

If your containers exit, check the OOMKilled and the exit code in your docker container state, 137 is indicating the JVM ran out of memory.

```
docker inspect test_seed_node_1
...
"State": {
        "Status": "exited",
        "Running": false,
        "Paused": false,
        "Restarting": false,
        "OOMKilled": false,
        "Dead": false,
        "Pid": 0,
        "ExitCode": 137,
        "Error": "",
        "StartedAt": "2019-12-06T14:16:02.2636528Z",
        "FinishedAt": "2019-12-06T14:16:58.3260739Z"
    }
...
```

If needed, increase your docker memory quota from the docker advanced preferences and adjust memory setting in your docker-compose file:


Advanced
General
File Sharing
Disk
Advanced
Proxies
Daemon
Kubernetes
Reset
Limit the resources available to Docker Engine.
CPUs: 8
Memory: 5.0 GiB
Swap: 1.0 GiB
Docker subnet:
192.168.65.0/24
Apply & Restart
Docker Desktop is running

# CHAPTER 3

# Installation

There are a number of ways to install Elassandra:

- *tarball*
- *deb*
- *rpm*
- *docker image*.
- *helm chart* (kubernetes)
- *Google Kubernetes marketplace*

Elassandra is based on Cassandra and ElasticSearch, thus it will be easier if you're already familiar with one on these technologies.

**Important:** Be aware that Elassandra need more memory than Cassandra when Elasticsearch is used and should be installed on machine with at least 4Gb of RAM.

## 3.1 Tarball

Elassandra requires at least Java 8. Oracle JDK is the recommended version, but OpenJDK should also work as well. You need to check which version is installed on your computer:

```
$ java -version
java version "1.8.0_121"
Java(TM) SE Runtime Environment (build 1.8.0_121-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.121-b13, mixed mode)
```

Once java is correctly installed, download the Elassandra tarball:

```
wget https://github.com/strapdata/elassandra/releases/download/v6.8.4.13/
elassandra-6.8.4.13.tar.gz
```

Then extract its content:

```
tar -xzf elassandra-6.8.4.13.tar.gz
```

Go to the extracted directory:

```
cd elassandra-6.8.4.13
```

Configure `conf/cassandra.yaml` if necessary, and then run:

```
bin/cassandra -e
```

This has started cassandra with elasticsearch enabled (according to the `-e` option).

Get the node status:

```
bin/nodetool status
```

Now connect to the node with cqlsh:

```
bin/cqlsh
```

You're now able to type CQL commands. See the CQL reference.

Check the elasticsearch API:

```
curl -X GET http://localhost:9200/
```

You should get something like this:

```
{
  "name" : "127.0.0.1",
  "cluster_name" : "Test Cluster",
  "cluster_uuid" : "7cb65cea-09c1-4d6a-a17a-24efb9eb7d2b",
  "version" : {
    "number" : "6.8.4.13",
    "build_hash" : "b0b4cb025cb8aa74538124a30a00b137419983a3",
    "build_timestamp" : "2017-04-19T13:11:11Z",
    "build_snapshot" : true,
    "lucene_version" : "5.5.2"
  },
  "tagline" : "You Know, for Search"
}
```

You're done !

On a production environment, we recommand to to modify some system settings such as disabling swap. This guide shows you how to do it. On linux, you should install jemalloc.

## 3.2 Deb

**Important:** Cassandra and Elassandra packages conflict. You should remove Cassandra prior to install Elassandra.

The Java Runtime 1.8 is required to run Elassandra. On recent distributions it should be resolved automatically as a dependency. On Debian Jessie it can be installed from backports:

```
sudo apt-get install -t jessie-backports openjdk-8-jre-headless
```

You may need to install `apt-transport-https` and other utilities as well:

```
sudo apt-get install software-properties-common apt-transport-https gnupg2
```

Add our repository and gpg key:

```
sudo add-apt-repository 'deb [arch=all] https://nexus.repo.strapdata.com/repository/
↪apt-releases/ stretch main'
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys B335A4DD
```

And then install elassandra with:

```
sudo apt-get update && sudo apt-get install elassandra
```

Start Elassandra with Systemd:

```
sudo systemctl start cassandra
```

or SysV:

```
sudo service cassandra start
```

Files locations:

- `/usr/bin`: startup script, cqlsh, nodetool, elasticsearch-plugin
- `/etc/cassandra` and `/etc/default/cassandra`: configurations
- `/var/lib/cassandra`: data
- `/var/log/cassandra`: logs
- `/usr/share/cassandra`: plugins, modules, libs, ...
- `/usr/share/cassandra/tools`: cassandra-stress, sstabledump...
- `/usr/lib/python2.7/dist-packages/cqlshlib/`: python library for cqlsh

## 3.3 Rpm

**Important:** Cassandra and Elassandra packages conflict. You should remove Cassandra prior to install Elassandra.

The Java runtime 1.8 must be installed in order to run Elassandra. You can install it yourself or let the package manager pull it automatically as a dependency.

Create a file called `elassandra.repo` in the `/etc/yum.repos.d/` directory or similar according to your distribution (RedHat, OpenSuSe...):

```
[strapdata]
name=Strapdata
baseurl=https://nexus.repo.strapdata.com/repository/rpm-releases/
enabled=1
gpgcheck=0
priority=1
```

```
[strapdata-snapshots]
name=Strapdata Snapshots
baseurl=https://nexus.repo.strapdata.com/repository/rpm-snapshots/
enabled=1
gpgcheck=0
priority=1
```

And then install elassandra with:

```
sudo yum install elassandra
```

Start Elassandra with Systemd:

```
sudo systemctl start cassandra
```

or SysV:

```
sudo service cassandra start
```

Files locations:

- `/usr/bin`: startup script, cqlsh, nodetool, elasticsearch-plugin
- `/etc/cassandra` and `/etc/sysconfig/cassandra`: configurations
- `/var/lib/cassandra`: data
- `/var/log/cassandra`: logs
- `/usr/share/cassandra`: plugins, modules, libs...
- `/usr/share/cassandra/tools`: cassandra-stress, sstabledump...
- `/usr/lib/python2.7/site-packages/cqlshlib/`: python library for cqlsh

## 3.4 Docker image

We provide an image on docker hub:

```
docker pull strapdata/elassandra
```

This image is based on the official Cassandra image whose the documentation is valid as well for Elassandra.

The source code is on github at strapdata/docker-elassandra.

### 3.4.1 Start an Elassandra server instance

Starting an Elassandra instance is pretty simple:

```
docker run --name node0 -d strapdata/elassandra:6.8.4.13
```

Run nodetool, cqlsh and curl:

```
docker exec -it node0 nodetool status
docker exec -it node0 cqlsh
docker exec -it node0 curl localhost:9200
```

### 3.4.2 Environment Variables

When you start the Elassandra image, you can adjust the configuration of the Elassandra instance by passing one or more environment variables on the docker run command line.

| Variable Name | Description |
|---|---|
| CASSANDRA_LISTEN_ADDRESS | This variable is used for controlling which IP address to listen to for incoming connections on. The default value is auto, which will set the listen_address option in cassandra.yaml to the IP address of the container when it starts. This default should work in most use cases. |
| CASSANDRA_BROADCAST_ADDRESS | This variable is used for controlling which IP address to advertise on other nodes. The default value is the value of CASSANDRA_LISTEN_ADDRESS. It will set the broadcast_address and broadcast_rpc_address options in cassandra.yaml. |
| CASSANDRA_RPC_ADDRESS | This variable is used for controlling which address to bind the thrift rpc server to. If you do not specify an address, the wildcard address (0.0.0.0) will be used. It will set the rpc_address option in cassandra.yaml. |
| CASSANDRA_START_RPC | This variable is used for controlling if the thrift rpc server is started. It will set the start_rpc option in cassandra.yaml. As Elastic search used this port in Elassandra, it will be set ON by default. |
| CASSANDRA_SEEDS | This variable is the comma-separated list of IP addresses used by gossip for bootstrapping new nodes joining a cluster. It will set the seeds value of the seed_provider option in cassandra.yaml. The CASSANDRA_BROADCAST_ADDRESS will be added to the seeds passed on so that the sever can also talk to itself. |
| CASSANDRA_CLUSTER_NAME | This variable sets the name of the cluster. It must be the same for all nodes in the cluster. It will set the cluster_name option of cassandra.yaml. |
| CASSANDRA_NUM_TOKENS | This variable sets the number of tokens for this node. It will set the num_tokens option of cassandra.yaml. |
| CASSANDRA_DC | This variable sets the datacenter name of this node. It will set the dc option of cassandra-rackdc.properties. |
| CASSANDRA_RACK | This variable sets the rack name of this node. It will set the rack option of cassandra-rackdc.properties. |
| CASSANDRA_ENDPOINT_SNITCH | This variable sets the snitch implementation that will be used by the node. It will set the endpoint_snitch option of cassandra.yml. |
| CASSANDRA_DAEMON | The Cassandra entry-point class: `org.apache.cassandra.service.ElassandraDaemon` to start with ElasticSearch enabled (default), `org.apache.cassandra.service.ElassandraDaemon` otherwise. |

### 3.4.3 Files locations

Docker elassandra image is based on the debian package installation:

- `/etc/cassandra`: elassandra configuration
- `/usr/share/cassandra`: elassandra installation
- `/var/lib/cassandra`: data (sstables, lucene segment, commitlogs, ...)
- `/var/log/cassandra`: logs files.

`/var/lib/cassandra` is automatically managed as a docker volume. But it's a good target to bind mount from the host filesystem.

### 3.4.4 Exposed ports

- 7000: intra-node communication

- 7001: TLS intra-node communication
- 7199: JMX
- 9042: CQL
- 9160: thrift service
- 9200: ElasticSearch HTTP
- 9300: ElasticSearch transport

### 3.4.5 Create a cluster

In case there is only one elassandra instance per docker host, the easiest way is to start the container with `--net=host`.

When using the host network is not an option, you could just map the necessary ports with `-p 9042:9042`, `-p 9200:9200` and so on... but you should be aware that docker default network will considerably slow down performances.

---

**Note:** Creating a cluster from the standalone image is probably fine for testing environments. But if you plan to run long-lived Elassandra clusters on containers, Kubernetes is the way to go.

---

## 3.5 Helm chart

Helm Tiller must be initialised on the target kubernetes cluster.

Add our helm repository:

```
helm repo add strapdata https://charts.strapdata.com
```

Then create a cluster with the following command:

```
helm install -n elassandra --set image.tag="6.8.4.13" strapdata/elassandra
```

After installation succeeds, you can get a status of chart:

```
helm status elassandra
```

As show below, the Elassandra chart creates 2 clustered service for elasticsearch and cassandra:

```
kubectl get all -o wide -n elassandra
NAME                           READY     STATUS    RESTARTS   AGE
pod/elassandra-0               1/1       Running   0          51m
pod/elassandra-1               1/1       Running   0          50m
pod/elassandra-2               1/1       Running   0          49m

NAME                                 TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)
↪                                                          AGE
service/elassandra                   ClusterIP   None            <none>        7199/TCP,
↪7000/TCP,7001/TCP,9300/TCP,9042/TCP,9160/TCP,9200/TCP   51m
service/elassandra-cassandra         ClusterIP   10.0.174.13     <none>        9042/TCP,
↪9160/TCP                                                  51m
service/elassandra-elasticsearch     ClusterIP   10.0.131.15     <none>        9200/TCP
↪                                                          51m
```

```
NAME                            DESIRED   CURRENT   AGE
statefulset.apps/elassandra     3         3         51m
```

More information is available on github.

## 3.6 Google Kubernetes Marketplace

You can deploy an Elassandra cluster on GKE with a few clicks using our Elassandra Kubernetes App (require an existing GCP project and a running Google Kubernetes Cluster).

## 3.7 Running Cassandra only

In a cluster, you may need to run Cassandra datacenter without Elasticsearch indexing. In such case, change the CASSANDRA_DAEMON variable to **org.apache.cassandra.service.CassandraDaemon** in your `/etc/default/cassandra` on all nodes of your Cassandra only datacenter.

CHAPTER 4

# Configuration

## 4.1 Directory Layout

Elassandra merges the Cassandra and Elasticsearch directories as follows :

- `conf` : Cassandra configuration directory + elasticsearch.yml default configuration file.
- `bin` : Cassandra scripts + elasticsearch plugin script.
- `lib` : Cassandra and elasticsearch jar dependency
- `pylib` : Cqlsh python library.
- `tools` : Cassandra tools.
- `plugins` : Elasticsearch plugins installation directory.
- `modules` : Elasticsearch modules directory.
- `work` : Elasticsearch working directory.

Elasticsearch paths are set according to the following environment variables and system properties:

- `path.home` : **CASSANDRA_HOME** environment variable, cassandra.home system property, the current directory.
- `path.conf` : **CASSANDRA_CONF** environment variable, path.conf or path.home.
- `path.data` : **cassandra.storagedir**/data/elasticsearch.data, path.data system property or path.home/data/elasticsearch.data

## 4.2 Configuration files

Elasticsearch configuration relies on Cassandra configuration file **conf/cassandra.yaml** for the following parameters.

| Cassandra | Elasticsearch | Description |
|---|---|---|
| `cluster.name` | `cluster_name` | Elasticsearch cluster name is mapped to the cassandra cluster name. |
| `rpc_address` | `network.host` | Elasticsearch *network.host* is set to the cassandra *rpc_address*. |
| `broadcast_rpc_address` | `network.publish_host` | Elasticsearch *network.publish_host* is set to the cassandra *broadcast_rpc_address*. |
| `listen_address` | `transport.host` | Elasticsearch *transport_host* is set to the cassandra *listen_address*. |
| `broadcast_address` | `transport.publish_host` | Elasticsearch *transport.publish_host* is set to the cassandra *broadcast_address*. |

Node role (master, primary, and data) is automatically set by Elassandra, standard configuration should only set **cluster_name**, **rpc_address** in the `conf/cassandra.yaml`.

By default, Elasticsearch HTTP is bound to the Cassandra RPC address `rpc_address`, while Elasticsearch transport protocol is bound to the Cassandra internal address `listen_address`. You can overload these default settings by defining Elasticsearch network settings in conf/elasticsearch.yaml (in order to bind Elasticsearch transport on another interface).

By default, Elasticsearch transport publish address is the Cassandra broadcast address. However, in some network configurations (including multi-cloud deployment), the Cassandra broadcast address is a public address managed by a firewall, and it would involve network overhead for Elasticsearch inter-node communication. In such a case, you can set the system property `es.use_internal_address=true` to use the Cassandra `listen_address` as the Elasticsearch transport published address.

**Caution:** If you use the GossipingPropertyFile Snitch to configure your cassandra datacenter and rack properties in **conf/cassandra-rackdc.properties**, keep in mind that this snitch falls back to the PropertyFileSnitch when gossip is not enabled. So, when re-starting the first node, dead nodes can appear in the default DC and rack configured in **conf/cassandra-topology.properties**. It will also breaks the replica placement strategy and the computation of the Elasticsearch routing tables. So it is strongly recommended to set the same default rack and datacenter for both the **conf/cassandra-topology.properties** and the **conf/cassandra-rackdc.properties**.

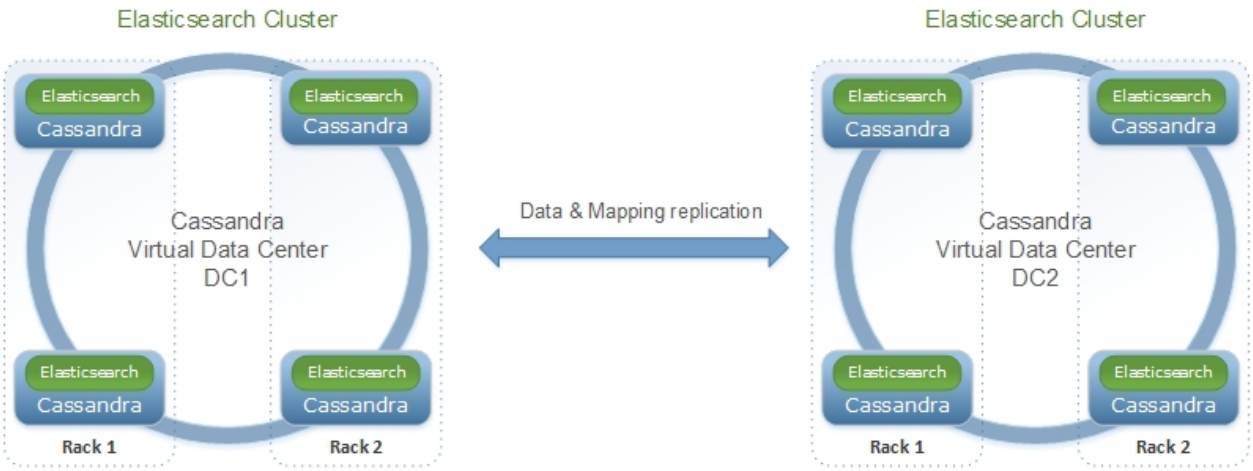
## 4.3 Logging configuration

The Cassandra logs in `logs/system.log` includes elasticsearch logs according to your `conf/logback.conf` settings. See cassandra logging configuration.

Per keyspace (or per table) logging level can be configured using the logger name `org.elassandra.index.ExtendedElasticSecondaryIndex.<keyspace>.<table>`.

## 4.4 Multi datacenter configuration

By default, all Elassandra datacenters share the same Elasticsearch cluster name and mapping. This mapping is stored in the `elastic_admin` keyspace.




If you want to manage various Elasticsearch clusters within a Cassandra cluster (when indexing different tables in different datacenters), you need to set a `datacenter.group` in **conf/elasticsearch.yml** and thus, all elassandra datacenters sharing the same datacenter group name will share the same mapping. These elasticsearch clusters will be named <cluster_name>@<datacenter.group> and mappings will be stored in a dedicated keyspace.table `elastic_admin_<datacenter.group>.metadata`.

All `elastic_admin[_<datacenter.group>]` keyspaces are configured with **NetworkReplicationStrategy** (see data replication). where the replication factor is **ONE** by default. When a mapping change occurs, Elassandra updates the Elasticsearch metadata in *elastic_admin[_<datacenter.group>].metadata* within a lightweight transaction to avoid conflict with concurrent updates. This transaction requires **QUORUM** available replicas and may involves cross-datacenter network latency for each Elasticsearch mapping update.

**Caution:** Elassandra cannot start Elasticsearch shards when the underlying keyspace is not replicated on the datacenter the node belongs to. In such case, the Elasticsearch shards remain UNASSIGNED and indices are red. You can fix that by manually altering the keyspace replication map, or use the Elassandra `index.replication` setting to properly configure it when creating the index.

If you want to deploy some indices to only a subset of the datacenters where your elastic_admin keyspace is replicated:

- Define a list of `datacenter.tags` in your **conf/elasticsearch.yml**.
- Add the index setting `index.datacenter_tag` to your local indices.

A tagged Elasticsearch index is visible from Cassandra datacenters having a matching tag in their `datacenter.tags`.

**Tip:** Cassandra cross-datacenter writes are not sent directly to each replica. Instead, they are sent to a single replica with a parameter telling to the replica to forward to the other replicas in that datacenter. These replicas will directly respond to the original coordinator. It reduces network traffic between datacenters when there are replicas.

## 4.5 Elassandra Settings

Most of the settings can be set at various levels :

- As a system property, default property is *es.<property_name>*
- At cluster level, default setting is *cluster.default_<property_name>*
- At index level, setting is *index.<property_name>*
- At table level, setting is configured as a *_meta:{ "<property_name> : <value> }* for a document type.

For example, `drop_on_delete_index` can be :

- set as a system property `es.drop_on_delete_index` for all created indices.
- set at cluster level with the `cluster.default_drop_on_delete_index` dynamic settings,
- set at index level with the `index.drop_on_delete_index` dynamic index settings,
- set as an Elasticsearch document type level with `_meta : { "drop_on_delete_index":true }` in the document type mapping.

Dynamic settings are only relevant for clusters, indexes and document type setting levels, system settings defined by a JVM property are immutable.

| Setting | Update | Levels | Default value | Description |
|---|---|---|---|---|
| `keyspace` | static | index | **index name** | Underlying cassandra keyspace name. |
| `replication` | static | index | *local_datacenter*:*number_of_replicas* | A comma separated list of *datacenter_name*:*replication_factor* used when creating the underlying cassandra keyspace (For exemple "DC1:1,DC2:2"). Remember that when a keyspace is not replicated to an elasticsearch-enabled datacenter, elassandra cannot open the keyspace and the associated elasticsearch index remains red. |
| `datacenter_tag` | dynamic | index | | Set a datacenter tag. A tagged index is only visible on the Cassandra datacenters having the tag in its `datacenter.tags` settings, see *Multi datacenter configuration*. |
| `table_options` | static | index | | Cassandra table options use when creating the underlying table (like "default_time_to_live = 300"). See the cassandra documentation for available options. |
| `secondary_index_class` | static | index, cluster | **ExtendedElasticSecondaryIndex** | Cassandra secondary index implementation class. This class needs to implements *org.apache.cassandra.index.Index* interface. |
| `search_strategy_class` | dynamic | index, cluster | **PrimaryFirstSearchStrategy** | Search strategy class. Available strategy are :<br>• *PrimaryFirstSearchStrategy* distributes search requests to all available nodes<br>• *RandomSearchStrategy* |

## 4.6 Sizing and tuning

Basically, Elassandra requires more CPU than the standalone Cassandra or Elasticsearch and Elassandra write throughput should be half the Cassandra write throughput if you index all columns. If you only index a subset of co lumns, write performance would be better.

Design recommendations :

- Increase number of Elassandra node or use partitioned index to keep shards size below 50Gb.
- Avoid huge wide rows, write-lock on a wide row can dramatically affect write performance.
- Choose the right Cassandra compaction strategy to fit your workload (See this blog post by Justin Cameron)

System recommendations :

- Turn swapping off.
- Configure less than half the total memory of your server and up to 30.5Gb. Minimum recommended DRAM for production deployments is 32Gb. If you are not aggregating on text fields, you can probably use less memory to improve file system cache used by Doc Values (See this excelent blog post by Chris Earle).
- Set -Xms to the same value as -Xmx.
- Ensure JNA and jemalloc are correctly installed and enabled.

### 4.6.1 Write performance

- By default, Elasticsearch analyzes the input data of all fields in a special **_all** field. If you don't need it, disable it.
- By default, Elasticsearch indexes all fields names in a special **_field_names** field. If you don't need it, disable it (elasticsearch-hadoop requires **_field_names** to be enabled).
- By default, Elasticsearch shards are refreshed every second, making new document visible for search within a second. If you don't need it, increase the refresh interval to more than a second, or even turn if off temporarily by setting the refresh interval to -1.
- Use the optimized version less Lucene engine (the default) to reduce index size.
- Disable `index_on_compaction` (Default is *false*) to avoid the Lucene segments merge overhead when compacting SSTables.
- Index partitioning may increase write throughput by writing to several Elasticsearch indexes in parallel, but choose an efficient partition function implementation. For example, *String.format()* is much more faster that *Message.format()*.

### 4.6.2 Search performance

- Use 16 to 64 vnodes per node to reduce the complexity of the token_ranges filter.
- Use the *RandomSearchStrategy* and increase the Cassandra Replication Factor to reduce the number of nodes requires for a search request.
- Enable the `token_ranges_bitset_cache`. This cache compute the token ranges filter once per Lucene segment. Check the token range bitset cache statistics to ensure this caching is efficient.
- Enable Cassandra row caching to reduce the overhead introduce by fetching the requested fields from the underlying Cassandra table.
- Enable Cassandra off-heap row caching in your Cassandra configuration.

- When possible, reduce the number of Lucene segments by forcing a merge.

CHAPTER 5

# Mapping

In essence, an Elasticsearch index is mapped to a Cassandra keyspace, and a document type to a Cassandra table.

## 5.1 Type mapping

Below is the mapping from Elasticsearch field basic types to CQL3 types :

| Elasticearch Types | CQL Types | Comment |
|---|---|---|
| keyword | text | Not analyzed text |
| text | text | Analyzed text |
| date | timestamp | |
| date | date | Existing Cassandra *date* columns mapped to an Elasticsearch date. (32-bit integer representing days since epoch, January 1, 1970) |
| byte | tinyint | |
| short | smallint | |
| integer | int | |
| long | bigint | |
| keyword | decimal | Existing Cassandra *decimal* columns are mapped to an Elasticsearch keyword. |
| long | time | Existing Cassandra *time* columns (64-bit signed integer representing the number of nanoseconds since midnight) stored as long in Elasticsearch. |
| double | double | |
| float | float | |
| boolean | boolean | |
| binary | blob | |
| ip | inet | Internet address |
| keyword | uuid | Existing Cassandra *uuid* columns are mapped to an Elasticsearch keyword. |
| keyword or date | timeuuid | Existing Cassandra *timeuuid* columns are mapped to an Elasticsearch keyword by default, or can explicitly be mapped to an Elasticsearch date. |
| geo_point | UDT geo_point or text | Built-In User Defined Type (1) |
| geo_shape | text | Requires *_source* enabled (2) |
| range | UDT xxxx_range | Elasticsearch range (integer_range, float_range, long_range, double_range, date_range, ip_range) |
| object, nested | Custom User Defined Type | User Defined Type should be frozen, as described in the Cassandra documentation. |

1. Geo shapes require _source to be enabled to store the original JSON document (default is disabled).
2. Existing Cassandra text columns containing a geohash string can be mapped to an Elasticsearch geo_point.

## 5.2 CQL mapper extensions

Elassandra adds some Elasticsearch mapper extensions in order to map Elasticsearch field to Cassandra:

| Parameter | Values | Description |
|---|---|---|
| cql_collection | **list**, set, singleton or none | Control how the field of type X is mapped to a column list<X>, set<X> or X. Default is **list** because Elasticsearch fields are multivalued. For copyTo fields, **none** means the field is not backed into Cassandra but just indexed by Elasticsearch. |
| cql_struct | **udt**, map or opaque_map | Control how an object or nested field is mapped to a User Defined Type or to a Cassandra. When using **map**, each new key is registred as a subfield in the elasticsearch mapping through a mapping update request. When using **opaque_map**, each new key is silently indexed as a new field, but the elasticsearch mapping is not updated. |
| cql_static_column | true or **false** | When *true*, the underlying CQL column is static. Default is **false**. |
| cql_primary_key_order | **integer** | Field position in the Cassandra the primary key of the underlying Cassandra table. Default is **-1** meaning that the field is not part of the Cassandra primary key. |
| cql_partition_key | true or **false** | When the cql_primary_key_order >= 0, specify if the field is part of the Cassandra partition key. Default is **false** meaning that the field is not part of the Cassandra partition key. |
| cql_clustering_key_desc | true or **false** | Indicates if the field is a clustering key in ascending or descending order, default is ascending (false). See Cassandra documentation about clustering key ordering. |
| cql_udt_name | **<table_name>** | Specify the Cassandra User Defined Type name to use to store an object. By default, this is **<field_name>** build (dots in *field_names* are replaced by underscores) |
| cql_type | **<CQL type>** | Specify the Cassandra type to use to store an elasticsearch field. By default, this is automatically set depending on the Elasticsearch field type, but in some situation, you can overwrite the default type by another one. |

For more information about Cassandra collection types and compound primary key, see CQL Collections and Compound keys.

---

**Tip:** For every update, Elassandra reads for missing fields in order to build a full Elasticsearch document. If some fields are backed by Cassandra collections (map, set or list), Elassandra force a read before index even if all fields are provided in the Cassandra upsert operation. For this reason, when you don't need multi-valued fields, use fields backed by native Cassandra types rather than the default list to avoid a read-before-index when inserting a row containing all its mandatory elasticsearch fields.

---

## 5.3 Elasticsearch multi-fields

Elassandra supports *Elasticsearch multi-fields <https://www.elastic.co/guide/en/elasticsearch/reference/6.2/multi-fields.html>* indexing, allowing to index a field in many ways for different purposes.

---

**Tip:** Indexing a wrong datatype into a field may throws an exception by default and reject the whole document. The ignore_malformed parameter, if set to true, allows the exception to be ignored. This parameter can also be set at the index level, to allow to ignore malformed content globally across all mapping types.

---

## 5.4 Bi-directional mapping

Elassandra supports the Elasticsearch Indice API and automatically creates the underlying Cassandra keyspaces and tables. For each Elasticsearch document type, a Cassandra table is created to reflect the Elasticsearch mapping. How-

ever, deleting an index does not remove the underlying keyspace, it only removes the Cassandra secondary indices associated to the mapped columns.

Additionally, with the new put mapping parameter `discover`, Elassandra creates or updates the Elasticsearch mapping for an existing Cassandra table. Columns matching the provided regular expression are mapped as Elasticsearch fields. The following command creates the Elasticsearch mapping for all columns starting with a 'a' in the Cassandra table *my_keyspace.my_table* and set a specific analyser for column *name*.

```
curl -XPUT -H 'Content-Type: application/json' "http://localhost:9200/my_keyspace/_
↪mapping/my_table" -d '{
    "my_table" : {
        "discover" : "a.*",
        "properties" : {
            "name" : {
                "type" : "text"
            }
        }
    }
}'
```

By default, all text columns are mapped with `"type":"keyword"`. Moreover, the discovery regular expression must exclude explicitly mapped fields to avoid inconsistent mapping. The following mapping update allows to discover all fields but the one named "name" and explicitly define its mapping.

```
curl -XPUT -H 'Content-Type: application/json' "http://localhost:9200/my_keyspace/_
↪mapping/my_table" -d '{
    "my_table" : {
        "discover" : "^((?!name).*)",
        "properties" : {
            "name" : {
                "type" : "text"
            }
        }
    }
}'
```

---

**Tip:** When creating the first Elasticsearch index for a given Cassandra table, Elassandra creates a custom CQL secondary index. Cassandra automatically builds indices on all nodes for all existing data. Subsequent CQL inserts or updates are automatically indexed in Elasticsearch.

If you then add a second or additional Elasticsearch indices to an existing indexed table, existing data are not automatically re-indexed because Cassandra has already indexed existing data. Instead of re-inserting your data into the Cassandra table, you may want to use the following command to force a Cassandra index rebuild. It will re-index your Cassandra table to all associated Elasticsearch indices :

```
nodetool rebuild_index --threads <N> <keyspace_name> <table_name> elastic_<table_name>
↪_idx
```

- *rebuild_index* reindexes SSTables from disk, but not from MEMtables. In order to index the very last inserted document, run a **nodetool flush <kespace_name>** before rebuilding your Elasticsearch indices.
- When deleting an elasticsearch index, elasticsearch index files are removed from the data/elasticsearch.data directory, but the Cassandra secondary index remains in the CQL schema until the last associated elasticsearch index is removed. Cassandra is acting as primary data storage, so keyspace and tables and data are never removed when deleting an elasticsearch index.

---

## 5.5 Meta-Fields

Elasticsearch meta-fields meaning is slightly different in Elassandra :

- `_index` is the index name mapped to the underlying Cassandra keyspace name (dash [-] and dot[.] are automatically replaced by underscore [_]).
- `_type` is the document type name mapped to the underlying Cassandra table name (dash [-] and dot[.] are automatically replaced by underscore [_]). Since Elasticsearch 6.x, there is only one type per index.
- `_id` is the document ID is a string representation of the primary key of the underlying Cassandra table. Single field primary key is converted to a string, compound primary key is converted into a JSON array converted to a string. For example, if your primary key is a string and a number, you will get `_id` = ["003011FAEF2E",1493502420000]. To get such a document by its `_id`, you need to properly escape brackets and double-quotes as shown below.

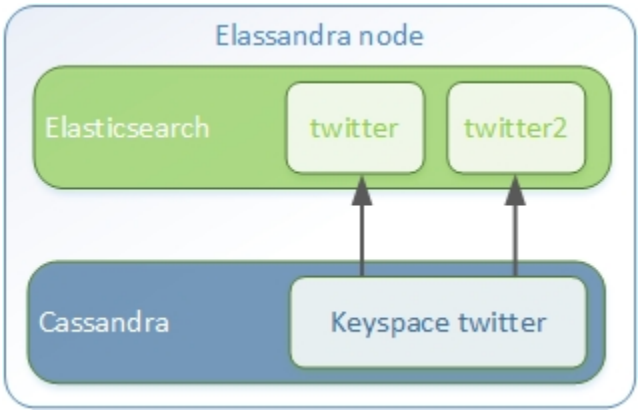
```
get 'twitter/tweet/\["003011FAEF2E",1493502420000\]?pretty'
{
  "_index" : "twitter",
  "_type" : "tweet",
  "_id" : "[\"003011FAEF2E\",1493502420000]",
  "_version" : 1,
  "found" : true,
  "_source" : {
    ...
  }
}
```

- `_source` is the indexed JSON document. By default, *_source* is disabled in Elassandra, meaning that *_source* is rebuild from the underlying Cassandra columns. If *_source* is enabled (see Mapping _source field) ELassandra stores documents indexed by with the Elasticsearch API in a dedicated Cassandra text column named *_source*. This allows to retreive the orginal JSON document for GeoShape Query.
- `_routing` is valued with a string representation of the partition key of the underlying Cassandra table. Single partition key is converted into a string, compound partition key is converted into a JSON array. Specifying `_routing` on get, index or delete operations is useless, since the partition key is included in `_id`. On search operations, Elassandra computes the Cassandra token associated with `_routing` for the search type, and reduces the search only to a Cassandra node hosting the token. (WARNING: Without any search types, Elassandra cannot compute the Cassandra token and returns with an error **all shards failed**).
- `_ttl` and `_timestamp` are mapped to the Cassandra TTL and WRITIME in Elassandra 5.x. The returned `_ttl` and `_timestamp` for a document will be the one of a regular Cassandra column if there is one in the underlying table. Moreover, when indexing a document through the Elasticsearch API, all Cassandra cells carry the same WRITETIME and TTL, but this could be different when upserting some cells using CQL.
- `_parent` is string representation of the parent document primary key. If the parent document primary key is composite, this is string representation of columns defined by `cql_parent_pk` in the mapping. See Parent-Child Relationship.
- `_token` is a meta-field introduced by Elassandra, valued with **token(<partition_key>)**.
- `_host` is an optional meta-field introduced by Elassandra, valued with the Cassandra host id, allowing to check the datacenter consistency.

## 5.6 Mapping change with zero downtime

You can map several Elasticsearch indices with different mappings to the same Cassandra keyspace. By default, an index is mapped to a keyspace with the same name, but you can specify a target `keyspace` in your index settings.

For example, you can create a new index **twitter2** mapped to the Cassandra keyspace **twitter** and set a mapping for the type **tweet** associated to the existing Cassandra table **twitter.tweet**.




```
curl -XPUT -H 'Content-Type: application/json' "http://localhost:9200/twitter2/" -d '{
   "settings" : { "keyspace" : "twitter" } },
   "mappings" : {
      "tweet" : {
            "properties" : {
              "message" : { "type" : "text" },
              "post_date" : { "type" : "date", "format": "yyyy-MM-dd" },
              "user" : { "type" : "keyword" },
              "size" : { "type" : "long" }
            }
      }
   }
}
```

You can set a specific mapping for **twitter2** and re-index existing data on each Cassandra node with the following command (indices are named **elastic_<tablename>_idx**).

```
nodetool rebuild_index [--threads <N>] twitter tweet elastic_tweet_idx
```

By default, **rebuild_index** uses only one thread, but Elassandra supports multi-threaded index rebuild with the new parameter **–threads**. Index name is <elastic>_<table_name>_idx where *column_name* is any indexed column name. Once your **twitter2** index is ready, set an alias **twitter** for **twitter2** to switch from the old mapping to the new one, and delete the old **twitter** index.

```
curl -XPOST -H 'Content-Type: application/json' "http://localhost:9200/_aliases" -d '
↪{ "actions" : [ { "add" : { "index" : "twitter2", "alias" : "twitter" } } ] }'
curl -XDELETE "http://localhost:9200/twitter"
```

## 5.7 Partitioned Index

Elasticsearch TTL support is deprecated since Elasticsearch 2.0 and the Elasticsearch TTLService is disabled in Elassandra. Rather than periodically looking for expired documents, Elassandra supports partitioned index allowing managing per time-frame indices. Thus, old data can be removed by simply deleting old indices.

Partitioned index also allows indexing more than 2^31 documents on a node (2^31 is the lucene max documents per index).

An index partition function acts as a selector when many indices are associated to a Cassandra table. A partition function is defined by 3 or more fields separated by a space character :

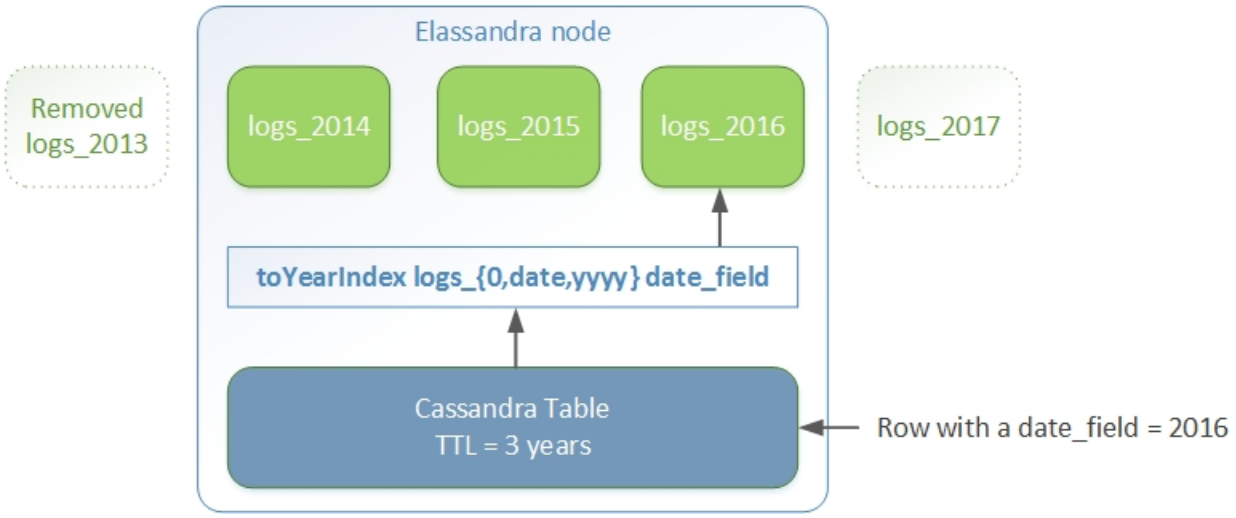
- Function name.
- Index name pattern.
- 1 to N document field names.

The target index name is the result your partition function,

A partition function must implements the java interface **org.elassandra.index.PartitionFunction**. Two implementation classes are provided :

- **StringFormatPartitionFunction** (the default) based on the JDK function String.format(Locale locale, <parttern>,<arg1>,...).
- **MessageFormatPartitionFunction** based on the JDK function MessageFormat.format(<parttern>,<arg1>,...).
- **TimeUUIDPartitionFunction** based on the JDK function String.format(Locale locale, <parttern>,<arg1>,...) (A TimeUUID argument will be converted as java.lang.Date).

Index partition function are stored in a map, so a given index function is executed exactly once for all mapped index. For example, the **toYearIndex** function generates the target index **logs_<year>** depending on the value of the **date_field** for each document (or row).



You can define each per-year index as follow, with the same `index.partition_function` for all **logs_<year>**. All these indices will be mapped to the keyspace **logs**, and all columns of the table **mylog** automatically mapped to the document type **mylog**.

```
curl -XPUT -H 'Content-Type: application/json' "http://localhost:9200/logs_2016" -d '{
  "settings": {
      "keyspace":"logs",
      "index.partition_function":"toYearIndex logs_{0,date,yyyy} date_field",
      "index.partition_function_class":"MessageFormatPartitionFunction"
  },
  "mappings": {
      "mylog" : { "discover" : ".*" }
  }
}'
```

**Tip:** Partition function is executed for each indexed document, so if write throughput is a concern, you should choose an efficient implementation class.

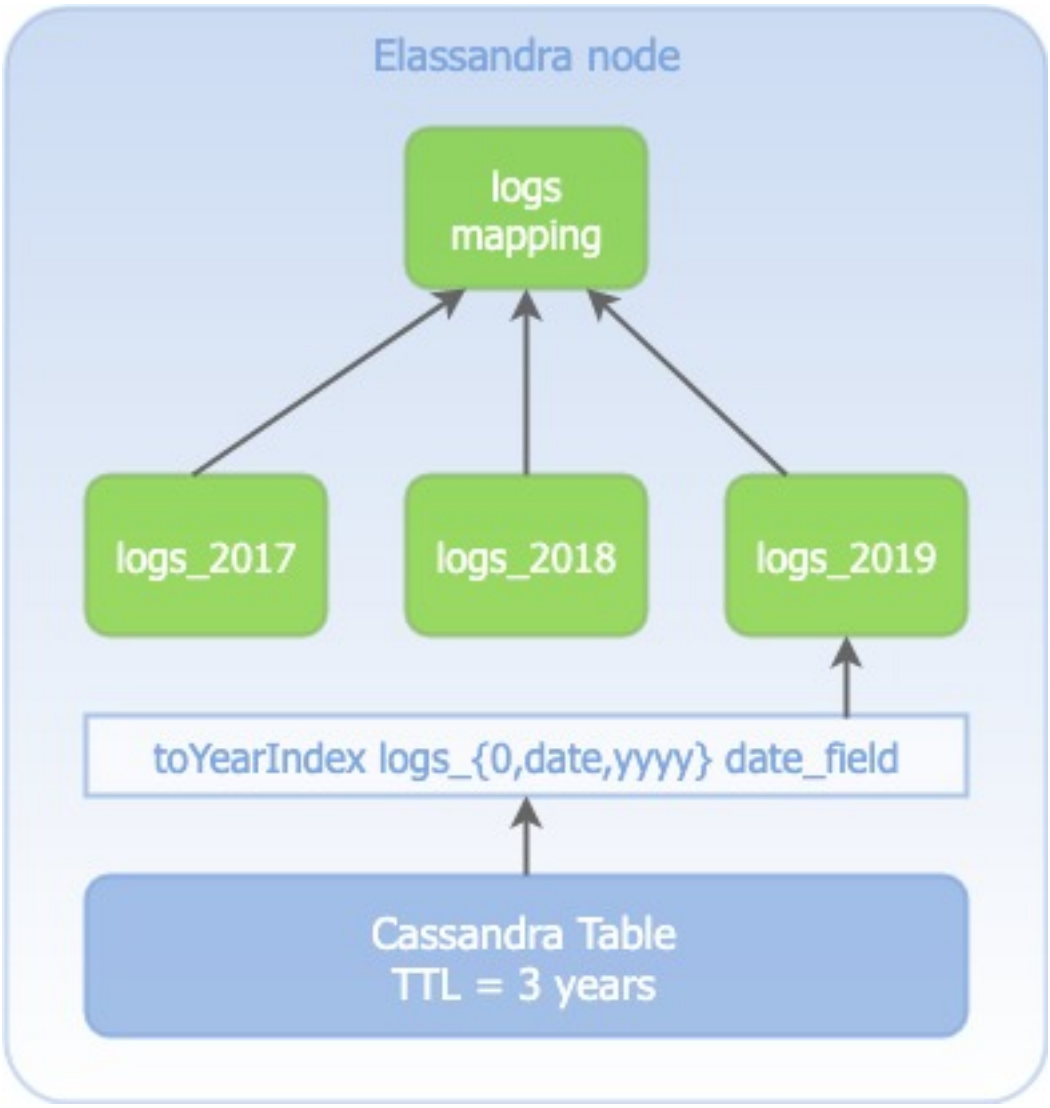
How To remove an old index.

```
curl -XDELETE "http://localhost:9200/logs_2013"
```

Cassandra TTL can be used in conjunction with partitioned index to automatically removed rows during the normal Cassandra compaction and repair processes when `index_on_compaction` is *true*, however it introduces a Lucene merge overhead because the document are re-indexed when compacting. You can also use the DateTieredCompactionStrategy to the TimeWindowTieredCompactionStrategy to improve performance of time series-like workloads.

### 5.7.1 Virtual index

In conjunction with partitioned indices, you can use a virtual index to share the same mapping for all partitioned indices.




A newly created index inherits the mapping created for other partitioned indices, and this drastically reduce the volume of Elasticsearch mappings stored in the CQL schema, and the number of mapping update across the cluster.

In order to create a partitioned index using the mapping of the virtual index, just add the name of the virtual index name as show bellow.

```
curl -XPUT -H 'Content-Type: application/json' "http://localhost:9200/logs_2016" -d '{
  "settings": {
      "keyspace":"logs",
      "index.partition_function":"toYearIndex logs_{0,date,yyyy} date_field",
      "index.partition_function_class":"MessageFormatPartitionFunction",
      "index.virtual_index":"logs"
  },
  "mappings": {
      "mylog" : { "discover" : ".*" }
```

```
  }
}'
```

The mappings section is only used to create the virtual index **logs** if it not exists when **logs_2016** is created. This virtual index **logs** have (or must have if you create it explicitly) the settings `index.virtual=true` and it will always be empty. Moreover, index templates can be used to specify common settings between partitioned index, including the virtual index name and its default mapping.

## 5.8 Object and Nested mapping

By default, Elasticsearch Object or nested types are mapped to dynamically created Cassandra User Defined Types.

```
curl -XPUT -H 'Content-Type: application/json' 'http://localhost:9200/twitter/tweet/1
↪' -d '{
     "user" : {
         "name" : {
             "first_name" : "Vincent",
             "last_name" : "Royer"
         },
         "uid" : "12345"
     },
     "message" : "This is a tweet!"
}'

curl -XGET 'http://localhost:9200/twitter/tweet/1/_source'
{"message":"This is a tweet!","user":{"uid":["12345"],"name":[{"first_name":["Vincent
↪"],"last_name":["Royer"]}]}}
```

The resulting Cassandra user defined types and table.

```
cqlsh>describe keyspace twitter;
CREATE TYPE twitter.tweet_user (
    name frozen<list<frozen<tweet_user_name>>>,
    uid frozen<list<text>>
);

CREATE TYPE twitter.tweet_user_name (
    last_name frozen<list<text>>,
    first_name frozen<list<text>>
);

CREATE TABLE twitter.tweet (
    "_id" text PRIMARY KEY,
    message list<text>,
    person list<frozen<tweet_person>>
)

cqlsh> select * from twitter.tweet;
_id  | message              | user
-----+----------------------+-------------------------------------------------------------
↪--------------------
1 | ['This is a tweet!'] | [{name: [{last_name: ['Royer'], first_name: ['Vincent']}],␣
↪uid: ['12345']}]
```

## 5.9 Dynamic mapping of Cassandra Map

By default, nested document are be mapped to User Defined Type. For top level fields only, you can also use a CQL map having a *text* key and a value of native or UDT type (using a collection in a map is not supported by Cassandra).

With `cql_struct=map`, each new key in the map involves an Elasticsearch mapping update (and a PAXOS transaction) to declare the key as a new field. Obviously, don't use such mapping when keys are versatile.

With `cql_struct=opaque_map`, Elassandra silently index each key as an Elasticsearch field, but does not update the mapping, which is far more efficient when using versatile keys. Every sub-fields (or every entry in the map) have the same type defined by the pseudo field name `_key` in the mapping. These fields are searchable, except with query string queries because Elasticsearch cannot lookup fields in the mapping.

Finally, when discovering the mapping from the CQL schema, Cassandra maps columns are mapped to an `opaque_map` by default. Adding explicit sub-fields to an `opaque_map` is still possible if you need to make these fields visible to Kibana for example.

In the following example, each new key entry in the map *attrs* is mapped as field.

```
CREATE KEYSPACE IF NOT EXISTS twitter WITH replication={ 'class':
↪'NetworkTopologyStrategy', 'DC1':'1' };
CREATE TABLE twitter.user (
   name text,
   attrs map<text,text>,
   PRIMARY KEY (name)
);
INSERT INTO twitter.user (name,attrs) VALUES ('bob',{'email':'bob@gmail.com',
↪'firstname':'bob'});
```

Create the type mapping from the Cassandra table and search for the *bob* entry.

```
curl -XPUT -H 'Content-Type: application/json' "http://localhost:9200/twitter" -d '{
    "mappings": {
       "user" : { "discover" : "^((?!attrs).*)" }
     }
}'

curl -XPUT -H 'Content-Type: application/json' 'http://localhost:9200/twitter/_
↪mapping/user?pretty=true' -d'{
 "properties" : {
     "attrs" : {
        "type" : "nested",
        "cql_struct" : "map",
        "cql_collection" : "singleton",
        "properties" : {
          "email" : {
            "type" : "keyword"
          },
          "firstname" : {
            "type" : "keyword"
          }
        }
      }
    }
}'

curl -XGET "http://localhost:9200/twitter/user/bob?pretty=true"
{
  "_index" : "twitter",
```

```
  "_type" : "user",
  "_id" : "bob",
  "_version" : 0,
  "found" : true,
  "_source":{"name":"bob","attrs":{"email":"bob@gmail.com","firstname":"bob"}}
}
```

Now insert a new entry in the attrs map column and search for a nested field *attrs.city:paris*.

```
UPDATE twitter.user SET attrs = attrs + { 'city':'paris' } WHERE name = 'bob';

curl -XGET  -H 'Content-Type: application/json' "http://localhost:9200/twitter/_
↪search?pretty=true" -d '{
"query":{
    "nested":{
            "path":"attrs",
            "query":{ "term": {"attrs.city":"paris" } }
             }
        }
}'
{
  "took" : 3,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "failed" : 0
  },
  "hits" : {
    "total" : 1,
    "max_score" : 2.3862944,
    "hits" : [ {
      "_index" : "twitter",
      "_type" : "user",
      "_id" : "bob",
      "_score" : 2.3862944,
      "_source":{"attrs":{"city":"paris","email":"bob@gmail.com","firstname":"bob"},
↪"name":"bob"}
    } ]
  }
}
```

With an `opaque_map`, search results are the same, and the Elasticsearch mapping is:

```
curl -XPUT -H 'Content-Type: application/json' "http://localhost:9200/twitter" -d '{
    "mappings": {
       "user" : { "discover" : ".*" }
     }
}'

curl -XGET "http://localhost:9200/twitter?pretty"
{
   "twitter" : {
     "aliases" : { },
     "mappings" : {
       "user" : {
         "properties" : {
           "attrs" : {
```

```
            "type" : "nested",
            "cql_struct" : "opaque_map",
            "cql_collection" : "singleton",
            "properties" : {
              "_key" : {
                "type" : "keyword",
                "cql_collection" : "singleton"
              }
            }
          },
          "name" : {
            "type" : "keyword",
            "cql_collection" : "singleton",
            "cql_partition_key" : true,
            "cql_primary_key_order" : 0
          }
        }
      }
    },
    "settings" : {
      "index" : {
        "creation_date" : "1568060813134",
        "number_of_shards" : "1",
        "number_of_replicas" : "0",
        "uuid" : "ZyolrbP9Qjm8rNezne7wUw",
        "version" : {
          "created" : "6020399"
        },
        "provided_name" : "twitter"
      }
    }
  }
}
```

### 5.9.1 Dynamic Template with Dynamic Mapping

Dynamic templates can be used when creating a dynamic field from a Cassandra map.

```
"mappings" : {
      "event_test" : {
         "dynamic_templates": [ {
                        "strings_template": {
                          "match": "strings.*",
                          "mapping": {
                              "type": "keyword"
                          }
                        }
                    } ],
        "properties" : {
          "id" : {
            "type" : "keyword",
            "cql_collection" : "singleton",
            "cql_partition_key" : true,
            "cql_primary_key_order" : 0
          },
          "strings" : {
```

```
            "type" : "object",
            "cql_struct" : "map",
            "cql_collection" : "singleton"
          }
        }
      }
}
```

A new entry *key1* in the underlying Cassandra map will have the following mapping:

```
"mappings" : {
       "event_test" : {
         "dynamic_templates" : [ {
           "strings_template" : {
             "mapping" : {
               "type" : "keyword",
               "doc_values" : true
             },
             "match" : "strings.*"
           }
         } ],
         "properties" : {
           "strings" : {
             "cql_struct" : "map",
             "cql_collection" : "singleton",
             "type" : "nested",
             "properties" : {
               "key1" : {
                 "type" : "keyword"
               }
           },
           "id" : {
             "type" : "keyword",
             "cql_partition_key" : true,
             "cql_primary_key_order" : 0,
             "cql_collection" : "singleton"
           }
         }
       }
     }
```

Note that because doc_values is true by default for a keyword field, it does not appear in the mapping.

## 5.10 Parent-Child Relationship

**Warning:** Parent child is supported in Elassandra 5.x.

Elassandra supports parent-child relationship when parent and child documents are located on the same Cassandra node. This condition is met :

- when running a single node cluster,
- when the keyspace replication factor equals the number of nodes or
- when the parent and child documents share the same Cassandra partition key, as shown in the following example.

Create an index company (a Cassandra keyspace), a Cassandra table, insert 2 rows and map this table as document type employee.

```
cqlsh <<EOF
CREATE KEYSPACE IF NOT EXISTS company WITH replication={ 'class':
↪'NetworkTopologyStrategy', 'dc1':'1' };
CREATE TABLE company.employee (
"_parent" text,
"_id" text,
name text,
dob timestamp,
hobby text,
primary key (("_parent"),"_id")
);
INSERT INTO company.employee ("_parent","_id",name,dob,hobby) VALUES ('london','1',
↪'Alice Smith','1970-10-24','hiking');
INSERT INTO company.employee ("_parent","_id",name,dob,hobby) VALUES ('london','2',
↪'Alice Smith','1990-10-24','hiking');
EOF

curl -XPUT -H 'Content-Type: application/json' "http://$NODE:9200/company2" -d '{
   "mappings" : {
    "employee" : {
            "discover" : ".*",
            "_parent"  : { "type": "branch", "cql_parent_pk":"branch" }
        }
    }
}'
curl -XPOST -H 'Content-Type: application/json' "http://127.0.0.1:9200/company/branch/
↪_bulk" -d '
{ "index": { "_id": "london" }}
{ "district": "London Westminster", "city": "London", "country": "UK" }
{ "index": { "_id": "liverpool" }}
{ "district": "Liverpool Central", "city": "Liverpool", "country": "UK" }
{ "index": { "_id": "paris" }}
{ "district": "Champs Élysées", "city": "Paris", "country": "France" }
'
```

Search for documents having children document of type *employee* with *dob* date greater than 1980.

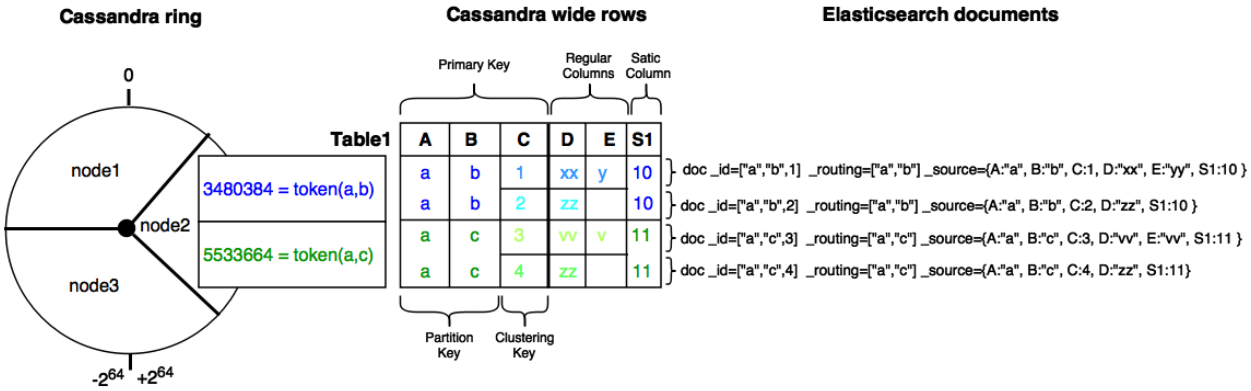
```
curl -XGET "http://$NODE:9200/company2/branch/_search?pretty=true" -d '{
  "query": {
    "has_child": {
      "type": "employee",
      "query": {
        "range": {
          "dob": {
            "gte": "1980-01-01"
          }
        }
      }
    }
  }
}'
```

Search for employee documents having a parent document where *country* match UK.

```
curl -XGET "http://$NODE:9200/company2/employee/_search?pretty=true" -d '{
  "query": {
    "has_parent": {
      "parent_type": "branch",
      "query": {
        "match": { "country": "UK"
        }
      }
    }
  }
}'
```

## 5.11 Indexing Cassandra static columns

When a Cassandra table has one or more clustering columns, a static columns is shared by all the rows with the same partition key.




Each time a static columns is modified, a document containing the partition key and only static columns is indexed in Elasticserach. By default, static columns are not indexed with every wide rows because any update on a static column would requires reindexation of all wide rows. However, you can request for fields backed by a static column on any get/search request.

The following example demonstrates how to use static columns to store meta information of a timeserie.

```
curl -XPUT -H 'Content-Type: application/json' "http://localhost:9200/test" -d '{
   "mappings" : {
       "timeseries" : {
         "properties" : {
           "t" : {
             "type" : "date",
             "format" : "strict_date_optional_time||epoch_millis",
             "cql_primary_key_order" : 1,
             "cql_collection" : "singleton"
           },
           "meta" : {
             "type" : "nested",
             "cql_struct" : "map",
             "cql_static_column" : true,
             "cql_collection" : "singleton",
             "include_in_parent" : true,
             "index_static_document": true,
```

```
              "index_static_columns": true,
              "properties" : {
                "region" : {
                  "type" : "keyword"
                }
              }
            },
            "v" : {
              "type" : "double",
              "cql_collection" : "singleton"
            },
            "m" : {
              "type" : "keyword",
              "cql_partition_key" : true,
              "cql_primary_key_order" : 0,
              "cql_collection" : "singleton"
            }
          }
        }
  }
}'

cqlsh <<EOF
INSERT INTO test.timeseries (m, t, v) VALUES ('server1-cpu', '2016-04-10 13:30', 10);
INSERT INTO test.timeseries (m, t, v) VALUES ('server1-cpu', '2016-04-10 13:31', 20);
INSERT INTO test.timeseries (m, t, v) VALUES ('server1-cpu', '2016-04-10 13:32', 15);
INSERT INTO test.timeseries (m, meta) VALUES ('server1-cpu', { 'region':'west' } );
SELECT * FROM test.timeseries;
EOF

 m           | t                           | meta                 | v
-------------+-----------------------------+----------------------+----
 server1-cpu | 2016-04-10 11:30:00.000000z | {'region': 'west'} | 10
 server1-cpu | 2016-04-10 11:31:00.000000z | {'region': 'west'} | 20
 server1-cpu | 2016-04-10 11:32:00.000000z | {'region': 'west'} | 15
```

Search for wide rows only where v=10 and fetch the meta.region field.

```
curl -XGET "http://localhost:9200/test/timeseries/_search?pretty=true&q=v:10&fields=m,
↪t,v,meta.region,_source"

"hits" : [ {
      "_index" : "test",
      "_type" : "timeseries",
      "_id" : "[\"server1-cpu\",1460287800000]",
      "_score" : 1.9162908,
      "_routing" : "server1-cpu",
      "_source" : {
            "t" : "2016-04-10T11:30:00.000Z",
            "v" : 10.0,
            "meta" : { "region" : "west" },
            "m" : "server1-cpu"
      },
      "fields" : {
        "meta.region" : [ "west" ],
        "t" : [ "2016-04-10T11:30:00.000Z" ],
        "m" : [ "server1-cpu" ],
        "v" : [ 10.0 ]
```

```
      }
    } ]
```

Search for rows where meta.region=west, returns only a static document (i.e. document containing the partition key and static columns) because `index_static_document` is true.

```
curl -XGET "http://localhost:9200/test/timeseries/_search?pretty=true&q=meta.
↪region:west&fields=m,t,v,meta.region"
"hits" : {
    "total" : 1,
    "max_score" : 1.5108256,
    "hits" : [ {
      "_index" : "test",
      "_type" : "timeseries",
      "_id" : "server1-cpu",
      "_score" : 1.5108256,
      "_routing" : "server1-cpu",
      "fields" : {
        "m" : [ "server1-cpu" ],
        "meta.region" : [ "west" ]
      }
    } ]
```

If needed, you can change the default behavior for a specific Cassandra table (or elasticsearch document type), by using the following custom metadata :

- `index_static_document` controls whether or not static document (i.e. document containing the partition key and static columns) are indexed (default is *false*).
- `index_static_only` if *true*, it only indexes static documents with partition key as `_id` and static columns as fields.
- `index_static_columns` controls whether or not static columns are included in the indexed documents (default is *false*).

Be careful, if `index_static_document` = *false* and `index_static_only` = *true*, it will not index any document. In our example with the following mapping, static columns are indexed in every document, allowing to search on.

```
curl -XPUT -H 'Content-Type: application/json' http://localhost:9200/test/_mapping/
↪timeseries -d '{
   "timeseries": {
      "discover" : ".*",
      "_meta": {
         "index_static_document":true,
         "index_static_columns":true
      }
   }
}'
```

## 5.12 Elassandra as a JSON-REST Gateway

When dynamic mapping is disabled and a mapping type has no indexed field, elassandra nodes can act as a JSON-REST gateway for Cassandra to get, set or delete a Cassandra row without any indexing overhead. In this case, the mapping may be use to cast types or format date fields, as shown below.

```
CREATE TABLE twitter.tweet (
    "_id" text PRIMARY KEY,
    message list<text>,
    post_date list<timestamp>,
    size list<bigint>,
    user list<text>
)

curl -XPUT -H 'Content-Type: application/json' "http://$NODE:9200/twitter/" -d'{
   "settings":{ "index.mapper.dynamic":false },
   "mappings":{
      "tweet":{
         "properties":{
            "size":     { "type":"long", "index":"no" },
            "post_date":{ "type":"date", "index":"no", "format" : "strict_date_
↪optional_time||epoch_millis" }
          }
      }
   }
}'
```

As a result, you can index, get or delete a Cassandra row, including any column from your Cassandra table.

```
curl -XPUT -H 'Content-Type: application/json' "http://localhost:9200/twitter/tweet/1?
↪consistency=one" -d '{
     "user" : "vince",
     "post_date" : "2009-11-15T14:12:12",
     "message" : "look at Elassandra !!",
     "size": 50
}'
{"_index":"twitter","_type":"tweet","_id":"1","_version":1,"_shards":{"total":1,
↪"successful":1,"failed":0},"created":true}

$ curl -XGET "http://localhost:9200/twitter/tweet/1?pretty=true&fields=message,user,
↪size,post_date'
{
  "_index" : "twitter",
  "_type" : "tweet",
  "_id" : "1",
  "_version" : 1,
  "found" : true,
  "fields" : {
    "size" : [ 50 ],
    "post_date" : [ "2009-11-15T14:12:12.000Z" ],
    "message" : [ "look at Elassandra !!" ],
    "user" : [ "vince" ]
  }
}

$ curl -XDELETE "http://localhost:9200/twitter/tweet/1?pretty=true'
{
  "found" : true,
  "_index" : "twitter",
  "_type" : "tweet",
  "_id" : "1",
  "_version" : 0,
  "_shards" : {
    "total" : 1,
```

```
    "successful" : 1,
    "failed" : 0
  }
}
```

## 5.13 Elasticsearch pipeline processors

Elassandra 6.x supports Elasticsearch pipeline processors when indexing through the Elasticsearch API. The following example illustrates how to generates a **timeuuid** clustering key when ingesting some logs into Elassandra (requires Elassandra 6.2.3.8+):

First, create a named pipeline as show below. This pipeline adds a new *timeuuid* field based on the existing date field *es_time* using the date format ISO8601 in europ timezone. The second processor set the document `_id` to a JSON compound key including the field *kubernetes.docker_id* (as the Cassandra partition key) and *ts* as a clustering key with CQL type `timeuuid`.

```
curl -H "Content-Type: application/json" -XPUT "http://localhost:9200/_ingest/
↪pipeline/fluentbit" -d'
{
  "description" : "fluentbit elassandra pipeline",
  "processors" : [
    {
      "timeuuid" : {
        "field": "es_time",
        "target_field": "ts",
        "formats" : ["ISO8601"],
        "timezone" : "Europe/Amsterdam"
      }
    },
    {
      "set" : {
        "field": "_id",
         "value": "[\"{{kubernetes.docker_id}}\",\"{{ts}}\"]"
      }
    }
  ]
}'
```

Because timeuuid is not an Elasticsearch type, this CQL type must be explicit in the Elasticsearch mapping using the `cql_type` field mapping attribute to replace the default `timestamp` by `timeuuid`. This can be acheived with an elasticsearch template. Your mapping must also defines a Cassandra partition key as text, and a clustering key of type `timeuuid`.

## 5.14 Check Cassandra consistency with Elasticsearch

When the `index.include_host = true` (default is false), the `_host` metafield containing the Cassandra host id is included in every indexed document. This allows distinguishing multiple copies of a document when the datacenter replication factor is greater than one. Then a token range aggregation allows counting the number of documents for each token range and for each Cassandra node.

In the following example, we have 1000 accounts documents in a keyspace with RF=2 in a two nodes datacenter, with each token ranges having the same number of document for the two nodes.

```
curl -XGET "http://$NODE:9200/accounts/_search?pretty=true&size=0" -d'{
        "aggs" : {
            "tokens" : {
                "token_range" : {
                   "field" : "_token"
                 },
                "aggs": {
                   "nodes" : {
                      "terms" : { "field" : "_host" }
                   }
                }
            }
        }
    }'
{
  "took" : 23,
  "timed_out" : false,
  "_shards" : {
    "total" : 2,
    "successful" : 2,
    "failed" : 0
  },
  "hits" : {
    "total" : 2000,
    "max_score" : 0.0,
    "hits" : [ ]
  },
  "aggregations" : {
    "tokens" : {
      "buckets" : [ {
        "key" : "(-9223372036854775807,-4215073831085397715]",
        "from" : -9223372036854775807,
        "from_as_string" : "-9223372036854775807",
        "to" : -4215073831085397715,
        "to_as_string" : "-4215073831085397715",
        "doc_count" : 562,
        "nodes" : {
          "doc_count_error_upper_bound" : 0,
          "sum_other_doc_count" : 0,
          "buckets" : [ {
            "key" : "528b78d3-fae9-49ae-969a-96668566f1c3",
            "doc_count" : 281
          }, {
            "key" : "7f0b782e-5b75-409b-85e9-f5f96a75a7dc",
            "doc_count" : 281
          } ]
        }
      }, {
        "key" : "(-4215073831085397714,7919694572960951318]",
        "from" : -4215073831085397714,
        "from_as_string" : "-4215073831085397714",
        "to" : 7919694572960951318,
        "to_as_string" : "7919694572960951318",
        "doc_count" : 1268,
        "nodes" : {
          "doc_count_error_upper_bound" : 0,
          "sum_other_doc_count" : 0,
          "buckets" : [ {
```

```
            "key" : "528b78d3-fae9-49ae-969a-96668566f1c3",
            "doc_count" : 634
          }, {
            "key" : "7f0b782e-5b75-409b-85e9-f5f96a75a7dc",
            "doc_count" : 634
          } ]
        }
      }, {
        "key" : "(7919694572960951319,9223372036854775807]",
        "from" : 7919694572960951319,
        "from_as_string" : "7919694572960951319",
        "to" : 9223372036854775807,
        "to_as_string" : "9223372036854775807",
        "doc_count" : 170,
        "nodes" : {
          "doc_count_error_upper_bound" : 0,
          "sum_other_doc_count" : 0,
          "buckets" : [ {
            "key" : "528b78d3-fae9-49ae-969a-96668566f1c3",
            "doc_count" : 85
          }, {
            "key" : "7f0b782e-5b75-409b-85e9-f5f96a75a7dc",
            "doc_count" : 85
          } ]
        }
      } ]
    }
  }
}
```


The same document count for all replica of a token range does not guarantee consistency between all replicas, but a different count allows to detect an inconsistency. Please note that according to your use case, you should add a filter to your query to ignore write operations occurring during the check.

# CHAPTER 6

# Operations

## 6.1 Indexing

Let's try and index some *twitter* like information (demo from Elasticsearch). First, let's create a twitter user, and add some tweets (the *twitter* index will be created automatically, see automatic index and mapping creation in Elasticsearch documentation):

```
curl -XPUT -H 'Content-Type: application/json' 'http://localhost:9200/twitter/user/
↪kimchy' -d '{ "name" : "Shay Banon" }'
curl -XPUT -H 'Content-Type: application/json' 'http://localhost:9200/twitter/tweet/1
↪' -d '
{
    "user": "kimchy",
    "postDate": "2009-11-15T13:12:00",
    "message": "Trying out Elassandra, so far so good?"
}'
curl -XPUT -H 'Content-Type: application/json' 'http://localhost:9200/twitter/tweet/2
↪' -d '
{
    "user": "kimchy",
    "postDate": "2009-11-15T14:12:12",
    "message": "Another tweet, will it be indexed?"
}'
```

You now have two rows in the Cassandra **twitter.tweet** table.

```
cqlsh
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 2.1.8 | CQL spec 3.2.0 | Native protocol v3]
Use HELP for help.
cqlsh> select * from twitter.tweet;
 _id | message                                       | postDate                      |␣
↪user
-----+-----------------------------------------------+-------------------------------+----
↪--------
```

```
   2 |      ['Another tweet, will it be indexed?'] | ['2009-11-15 15:12:12+0100'] | [
↪'kimchy']
   1 | ['Trying out Elassandra, so far so good?'] | ['2009-11-15 14:12:00+0100'] | [
↪'kimchy']
(2 rows)
```

Apache Cassandra is a column store that only support upsert operation. This means that deleting a cell or a row involves the creation of a tombstone (insert a null) kept until the compaction later removes both the obsolete data and the tombstone (See this blog about Cassandra tombstones).

By default, when using the Elasticsearch API to replace a document with a new one, Elassandra inserts a row corresponding to the new document including *null* for unset fields. Without these *null* (cell tombstones), old fields not present in the new document would be kept at the Cassandra level as zombie cells. If you store immutable data, you can set the index setting `index.index_insert_only` to true, to avoid the storage overhead generated by these tombstones.

Moreover, indexing with `op_type=create` (See Elasticsearch indexing ) requires a Cassandra PAXOS transaction to check if the document exists in the underlying datacenter. This comes with an unnecessary performance cost if you use an automatically generated document ID (See Automatic ID generation. ), as this ID will be the Cassandra primary key.

Depending on the *op_type* and document ID, CQL requests are issued as follows when indexing with the Elasticsearch API:

| *op_type* | Generated ID | Provided ID | Comment |
|---|---|---|---|
| `create` | INSERT INTO ... VALUES(...) | INSERT INTO ... VALUES(...) IF NOT EXISTS *(1)* | Index a new document if not exists. |
| `index` | INSERT INTO ... VALUES(...) | INSERT INTO ... VALUES(...) | |

(1) The *IF NOT EXISTS* comes with the cost of the PAXOS transaction. If you don't need to check the uniqueness of the provided ID, add parameter `check_unique_id=false`.

Insert operations are done with Cassandra Consistency Level `LOCAL_ONE` by default. If you want to insert or update with consistency level of `ONE`, `TWO`, `THREE`, `QUORUM`, `LOCAL_QUORUM`, `EACH_QUORUM` or `ALL`, you can set the `wait_for_active_shards` parameter (index settings or request parameter, see Wait For Active Shards ) to `1`, `2`, `3`, `quorum`, `local_quorum`, `each_quorum` or `all`.

## 6.2 GETing

Now, let's see if the information was added by GETting it:

```
curl -XGET 'http://localhost:9200/twitter/user/kimchy?pretty=true'
curl -XGET 'http://localhost:9200/twitter/tweet/1?pretty=true'
curl -XGET 'http://localhost:9200/twitter/tweet/2?pretty=true'
```

Elasticsearch state now reflects the new twitter index. Because we are currently running on one node, the **token_ranges** routing attribute matches 100% of the ring from Long.MIN_VALUE to Long.MAX_VALUE.

```
curl -XGET 'http://localhost:9200/_cluster/state/?pretty=true'
{
  "cluster_name" : "Test Cluster",
  "version" : 5,
  "master_node" : "74ae1629-0149-4e65-b790-cd25c7406675",
```

```
  "blocks" : { },
  "nodes" : {
    "74ae1629-0149-4e65-b790-cd25c7406675" : {
      "name" : "localhost",
      "status" : "ALIVE",
      "transport_address" : "inet[localhost/127.0.0.1:9300]",
      "attributes" : {
        "data" : "true",
        "rack" : "RAC1",
        "data_center" : "DC1",
        "master" : "true"
      }
    }
  },
  "metadata" : {
    "version" : 3,
    "uuid" : "74ae1629-0149-4e65-b790-cd25c7406675",
    "templates" : { },
    "indices" : {
      "twitter" : {
        "state" : "open",
        "settings" : {
          "index" : {
            "creation_date" : "1440659762584",
            "uuid" : "fyqNMDfnRgeRE9KgTqxFWw",
            "number_of_replicas" : "1",
            "number_of_shards" : "1",
            "version" : {
              "created" : "1050299"
            }
          }
        },
        "mappings" : {
          "user" : {
            "properties" : {
              "name" : {
                "type" : "string"
              }
            }
          },
          "tweet" : {
            "properties" : {
              "message" : {
                "type" : "string"
              },
              "postDate" : {
                "format" : "dateOptionalTime",
                "type" : "date"
              },
              "user" : {
                "type" : "string"
              }
            }
          }
        },
        "aliases" : [ ]
      }
    }
```

```
  },
  "routing_table" : {
    "indices" : {
      "twitter" : {
        "shards" : {
          "0" : [ {
            "state" : "STARTED",
            "primary" : true,
            "node" : "74ae1629-0149-4e65-b790-cd25c7406675",
            "token_ranges" : [ "(-9223372036854775808,9223372036854775807]" ],
            "shard" : 0,
            "index" : "twitter"
          } ]
        }
      }
    }
  },
  "routing_nodes" : {
    "unassigned" : [ ],
    "nodes" : {
      "74ae1629-0149-4e65-b790-cd25c7406675" : [ {
        "state" : "STARTED",
        "primary" : true,
        "node" : "74ae1629-0149-4e65-b790-cd25c7406675",
        "token_ranges" : [ "(-9223372036854775808,9223372036854775807]" ],
        "shard" : 0,
        "index" : "twitter"
      } ]
    }
  },
  "allocations" : [ ]
}
```


## 6.3 Updates

In Cassandra, an update is an upsert operation (if the row does not exists, it's an insert). As Elasticsearch, Elassandra issues a read-before-write operation before any update to build a full document.

Scripted updates, upsert (scripted_upsert and doc_as_upsert) are also supported.

## 6.4 Searching

Let's find all the tweets that *kimchy* posted:

```
curl -XGET 'http://localhost:9200/twitter/tweet/_search?q=user:kimchy&pretty=true'
```

We can also use the JSON query language Elasticsearch provides instead of a query string:

```
curl -XGET 'http://localhost:9200/twitter/tweet/_search?pretty=true' -d '
{
    "query" : {
        "match" : { "user": "kimchy" }
    }
}'
```

To avoid duplicate results when the Cassandra replication factor is greater than one, Elassandra adds a token_ranges filter to every query distributed to all nodes. Because every document contains a _token fields computed at index-time, this ensures that a node only retrieves documents for the requested token ranges. The `token_ranges` parameter is a conjunction of Lucene NumericRangeQuery built from the Elasticsearch routing tables to cover the entire Cassandra ring. .. code:

```
curl -XGET 'http://localhost:9200/twitter/tweet/_search?pretty=true&token_ranges=(0,
↪9223372036854775807)' -d '
{
    "query" : {
        "match" : { "user": "kimchy" }
    }
}'
```

Of course, if the token range filter covers all ranges (Long.MIN_VALUE to Long.MAX_VALUE), Elassandra automatically removes the useless filter.

Finally, you can restrict a query to the coordinator node with *preference=_only_local* parameter, for all token_ranges as shown below :

```
curl -XGET 'http://localhost:9200/twitter/tweet/_search?pretty=true&preference=_only_
↪local&token_ranges=' -d '
{
    "query" : {
        "match" : { "user": "kimchy" }
    }
}'
```

### 6.4.1 Optimizing search requests

#### The search strategy

Elassandra supports various search strategies to distribute a search request over the Elasticsearch cluster. A search strategy is configured at index-level with the `index.search_strategy_class` dynamic parameter.

| Strategy | Description |
|---|---|
| `org.elassandra.cluster.routing.PrimaryFirstSearchStrategy` (**Default**) | Search on all alive nodes in the datacenter. All alive nodes respond for their primary token ranges, and for replica token ranges when there are some unavailable nodes. This strategy is always used to build the routing table in the cluster state. |
| `org.elassandra.cluster.routing.RandomSearchStrategy` | For each query, randomly distribute a search request to a minimum of nodes to reduce the network traffic. For example, if your underlying keyspace replication factor is N, a search only invloves 1/N of the nodes. |

You can create an index with the `RandomSearchStrategy` as shown below (or change it dynamically).

```
curl -XPUT -H "Content-Type: application/json" "http://localhost:9200/twitter/" -d '{
   "settings" : {
      "index.search_strategy_class":"RandomSearchStrategy"
   }
}'
```

**Tip:** When changing a keyspace replication factor, you can force an Elasticsearch routing table update by closing and re-opening all associated Elasticsearch indices. To troubleshoot search request routing, set the logging level to **DEBUG** for **class org.elassandra.cluster.routing** in the **conf/logback.xml** file.

### 6.4.2 Caching features

Compared to Elasticsearch, Elassandra adds to each query a token ranges filter and by fetching fields through a CQL request at the Cassandra layer.

#### Token Ranges Query Cache

Token ranges filter depends on the node or vnodes configuration, are quite stable and shared for all keyspaces having the same replication factor. These filters only change when the datacenter topology changes, for example when a node is temporarily down or when a node is added to the datacenter. So, Elassandra uses a cache to keep these queries, a conjunction of Lucene NumericRangeQuery often reused for every search requests.

As a classic caching strategy, the `token_ranges_query_expire` controls the expiration time of useless token ranges filter queries into memory. The default is 5 minutes.

#### Token Ranges Bitset Cache

When enabled, the token ranges bitset cache keeps in memory the results of the token range filter for each Lucene segment. This in-memory bitset, acting as the liveDocs Lucene tombstones mechanism, is then reused for subsequent Lucene search queries. For each Lucene segment, this document bitset is updated when the Lucene tombstones count increases (it's a bitwise AND between the actual Lucene thumbstones and the token range filter result), or removed if the corresponding token ranges query is removed because unused from the token range query cache.

You can enable the token range bitset cache at index level by setting `index.token_ranges_bitset_cache` to *true* (Default is *false*), or configure the its default value for newly created indices at cluster or system levels.

You can also bypass this cache by adding *token_ranges_bitset_cache=false* in your search request :

```
curl -XGET "http://localhost:9200/twitter/_search?token_ranges_bitset_cache=false&
↪q=*:*"
```

Finally, you can check the in-memory size of the token ranges bitset cache with the Elasticsearch stats API, and clear it when clearing the Elasticsearch query_cache :

```
curl -XGET "http://localhost:9200/_stats?pretty=true"
...
"segments" : {
       "count" : 3,
       "memory_in_bytes" : 26711,
       "terms_memory_in_bytes" : 23563,
       "stored_fields_memory_in_bytes" : 1032,
       "term_vectors_memory_in_bytes" : 0,
       "norms_memory_in_bytes" : 384,
       "doc_values_memory_in_bytes" : 1732,
       "index_writer_memory_in_bytes" : 0,
       "index_writer_max_memory_in_bytes" : 421108121,
       "version_map_memory_in_bytes" : 0,
       "fixed_bit_set_memory_in_bytes" : 0,
       "token_ranges_bit_set_memory_in_bytes" : 240
```

```
    },
...
```

### Cassandra Key and Row Cache

To improve CQL fetch requests response time, Cassandra provides key and row caching features configured for each Cassandra table as follows :

```
ALTER TABLE ... WITH caching = {'keys': 'ALL', 'rows_per_partition': '1'};
```

To enable Cassandra row caching, set the `row_cache_size_in_mb` parameter in your **conf/cassandra.yaml**, and set `row_cache_class_name: org.apache.cassandra.cache.OHCProvider` to use off-heap memory.

**Tip:** Elasticsearch also provides a Lucene query cache, used for segments having more than 10k documents, and for some frequent queries (queries done more than 5 or 20 times depending of the nature of the query). The shard request cache, can also be enabled if the token range bitset cache is disabled.

## 6.5 Create, delete and rebuild index

In order to create an Elasticsearch index from an existing Cassandra table, you can specify the underlying keyspace. In the following example, all columns but *message* are automatically mapped with the default mapping, and the *message* is explicitly mapped with a custom mapping.

```
curl -XPUT -H 'Content-Type: application/json' 'http://localhost:9200/twitter_index' -
↪d '{
    "settings": { "keyspace":"twitter" }
    "mappings": {
        "tweet" : {
            "discover":"^(?!message).*",
            "properties" : {
               "message" : { "type":"keyword", "cql_collection":"singleton" }
            }

        }
    }
}'
```

**Caution:** Elassandra requires keyspaces configured with the *NetworkTopologyStrategy* in order to map the Elasticsearch *index.number_of_replicas* to the cassandra replication factor minus one. You can change your Cassandra replication factor as explained here.

**Tip:** By default, as the standard Elasticsearch, index creation only returns a response to the client when all primary shards have been started, or the request times out (default is 30 seconds). To emulate the Elasticsearch routing table, shards hosted by dead nodes are primary or not according to the underlying Cassandra replication factor. So, when there are some dead nodes, if the number of dead nodes is lower than the number of replicas in your create index request, index creation succeeds immediately with shards_acknowledged=true and index status is yellow, otherwise, index creation times out, shards_acknowledged=false and the index status is red, meaning that search requests will be

inconsistent. Finally, the Elasticsearch parameter wait_for_active_shards is useless in Elassandra, because Cassandra ensurea write consistency.

---

Deleting an Elasticsearch index does not remove any Cassandra data, it keeps the underlying Cassandra tables but removes Elasticsearch index files.

```
curl -XDELETE 'http://localhost:9200/twitter_index'
```

To re-index your existing data, for example after a mapping change to index a new column, run a **nodetool rebuild_index** as follows :

```
nodetool rebuild_index [--threads <N>] <keyspace> <table> elastic_<table>_idx
```

---

**Tip:** By default, rebuild index runs on a single thread. In order to improve re-indexing performance, Elassandra comes with a multi-threaded rebuild_index implementation. The **–threads** parameter allows to specify the number of threads dedicated to re-index a Cassandra table. Number of indexing threads should be tuned carefully to avoid CPU exhaustion. Moreover, indexing throughput is limited by locking at the lucene level, but this limit can be exceeded by using a partitioned index invloving many independent shards.

---

Re-index existing data relies on the Cassandra compaction manager. You can trigger a Cassandra compaction when :

- Creating the first Elasticsearch index on a Cassandra table with existing data automatically involves an index rebuild executed by the compaction manager,
- Running a nodetool rebuild_index command,
- Running a nodetool repair on a keyspace having indexed tables (a repair actually creates new SSTables triggering index build).

If the compaction manager is busy, secondary index rebuild is added as a pending task and executed later on. You can check current running compactions with a **nodetool compactionstats** and check pending compaction tasks with a **nodetool tpstats**.

```
nodetool -h 52.43.156.196 compactionstats
pending tasks: 1
                                     id          compaction type   keyspace      table  ␣
↪completed       total    unit   progress
052c70f0-8690-11e6-aa56-674c194215f6   Secondary index build     lastfm   playlist   ␣
↪66347424   330228366   bytes     20,09%
Active compaction remaining time :   0h00m00s
```

To stop a compaction task (including a rebuild index task), you can either use a **nodetool stop –compaction-id <uuid>** or use the JMX management operation **stopCompactionById** (on MBean org.apache.cassandra.db.CompactionManager).

## 6.6 Open, close index

Open and close operations allow an Elasticsearch index to be opened and closed. Even if the Cassandra secondary index remains in the CQL schema while the index is closed, it has no overhead. It's just a dummy function call. Obviously, when several Elasticsearch indices are associated with the same Cassandra table, data is indexed in opened indices, but not in closed ones.

```
curl -XPOST 'localhost:9200/my_index/_close'
curl -XPOST 'localhost:9200/my_index/_open'
```

**Warning:** Elasticsearch translog is disabled in Elassandra, so you might lose some indexed documents when closing an index if `index.flush_on_close` is *false*.

## 6.7 Flush, refresh index

A refresh makes all index updates performed since the last refresh available for search. By default, refresh is scheduled every second. By design, setting refresh=true on a index operation has no effect with Elassandra, because write operations are converted to CQL queries and documents are indexed later by a custom secondary index. So, the per-index refresh interval should be set carefully according to your needs.

```
curl -XPOST 'localhost:9200/my_index/_refresh'
```

A flush basically write a lucene index to disk. Because document **_source** is stored in the Cassandra table in Elassandra, it make sense to execute a `nodetool flush <keyspace> <table>` to flush both Cassandra Memtables to SSTables and lucene files for all associated Elasticsearch indices. Moreover, remember that a `nodetool snapshot` also involves a flush before creating a snapshot.

```
curl -XPOST 'localhost:9200/my_index/_flush'
```

**Tip:** Elasticsearch automatically triggers a flush when an index shard is inactive for more than `indices.memory.shard_inactive_time` (default is 5 minutes) or when Translog size is greater than `index.translog.flush_threshold_size` (Default is 512Mb). Elassandra implements a dummy Translog to track the size of indexed data and triggers a flush on the same size threashold. Elassandra also triggers an Elasticsearch flush when flushing Cassandra SSTables.

## 6.8 Managing Elassandra nodes

You can add, remove or replace an Elassandra node by using the same procedure as for Cassandra (see Adding nodes to an existing cluster). Even if it's technically possible, you should never boostrap more than one node at a time,

During the bootstrap process, pulled data from existing nodes are automatically indexed by Elasticsearch on the new node, involving a kind of an automatic Elasticsearch resharding. You can monitor and resume the Cassandra boostrap process with the nodetool bootstrap command.

After boostrap successfully ends, you should cleanup nodes to throw out any data that is no longer owned by that node, with a nodetool cleanup. Because cleanup involves by a Delete-by-query in Elasticsearch indices, it is recommended to smoothly schedule cleanups one at a time in you datacenter.

## 6.9 Backup and restore

By design, Elassandra synchronously updates Elasticsearch indices on the Cassandra write path. Flushing a Cassandra table involves a flush of all associated Elasticsearch indices. Therefore, Elassandra can backup data by taking a snapshot of Cassandra SSTables and Elasticsearch Lucene files on the same time on each node, as follows :

1. `nodetool snapshot --tag <snapshot_name> <keyspace_name>`
2. For all indices associated to <keyspace_name>

```
cp -al $CASSANDRA_DATA/elasticsearch.data/<cluster_name>/nodes/0/indices/
<index_name>/0/index/(_*|segment*) $CASSANDRA_DATA/elasticsearch.data/
snapshots/<index_name>/<snapshot_name>/
```

### 6.9.1 Restoring a snapshot

Restoring Cassandra SSTable and Elasticsearch Lucene files allows recovery of a keyspace and its associated Elasticsearch indices without stopping any node (but it is not intended to duplicate data to another virtual datacenter or cluster, this kind of operatio requires the sstableloader).

To perform a hot restore of Cassandra keyspace and its Elasticsearch indices :

1. Depending on your situation:
   - If you want to overwrite existing elasticsearch index, first truncate the underlying cassandra tables.
   - If you want to restore a deleted index or keyspace, first restore the CQL schema of the keyspace and lost tables by applying the **schema.cql** files from your snapshot. This re-creates empty elasticsearch indices.
2. Close the associated elasticsearch indices.
3. Restore the Cassandra table with your snapshot on each node.
4. Restore Elasticsearch snapshot data on each node (if ES index is open during nodetool refresh, this causes Elasticsearch index rebuild by the compaction manager, usually 2 threads).
5. Load restored SSTables with a `nodetool refresh`
6. Open all indices associated to the keyspace.

### 6.9.2 Point in time recovery

Point-in-time recovery is intended to recover the data at any time. This requires a restore of the last available Cassandra and Elasticsearch snapshot before your recovery point and then applies the commitlogs from this restore point to the recovery point. In this case, replaying commitlogs on startup also re-indexes data in Elasticsearch indices, ensuring consistency at the recovery point.

Of course, when stopping a production cluster is not possible, you should restore on a temporary cluster, make a full snapshot, and restore it on your production cluster as described by the hot restore procedure.

To perform a point-in-time-recovery of a Cassandra keyspace and its Elasticsearch indices, for all nodes at the same time :

1. Stop all the datacenter nodes.
2. Restore the last Cassandra snapshot before the restore point and commitlogs from that point to the restore point
3. Restore the last Elasticsearch snapshot before the restore point.
4. Restart your nodes

### 6.9.3 Restoring to a different cluster

When restoring data from another cluster, data distribution is not preserved, and the sstableloader send each restored rows to the appropriate nodes depending on token ranges distribution. If Elasticsearch indices are STARTED before restoring, data are automatically re-indexed in elasticsearch on each nodes while restoring with *sstableloader*.

To restore a Cassandra keyspace and its associated Elasticsearch indices from/to another cluster:

1. On the target cluster, create the same Cassandra schema without any custom secondary indices.

2. From the source cluster, extract the mapping of your associated indices and apply it to your destination cluster. Your keyspace and indices should be open and empty at this step.

If you are restoring into a new cluster having the same number of nodes, configure it with the same token ranges (see https://docs.datastax.com/en/Cassandra/2.1/cassandra/operations/ops_snapshot_restore_new_cluster.html). In this case, you can restore from Cassandra and Elasticsearch snapshots as described in steps 1, 3 and 4 of the snapshot restore procedure.

Otherwise, when the number of nodes and the token ranges from the source and destination cluster do not match, use the sstableloader to restore your Cassandra snapshots (see https://docs.datastax.com/en/cassandra/2.0/cassandra/tools/toolsBulkloader_t.html ). In this approach, all rows are read from the sstables and injected into the Cassandra cluster, causing a full Elasticsearch index rebuild.

## 6.10 Data migration

### 6.10.1 Migrating from Cassandra to Elassandra

Because Elassandra is Cassandra, you can upgrade an existing Cassandra cluster or just a datacenter to Elassandra, as soon as your Cassandra version is compatible with the Elassandra one :

- Stop your Cassandra nodes.
- Start Elassandra with your existing data directory (containing data, commitlog, saved_caches).

Before creating your first Elasticsearch index, deploy the following classes in a jar on all your Cassandra-only nodes to avoid a ClassNotFoundException. You can extract these classes from *lib/elasticsearch-<version>.jar* :

- org/elassandra/index/ExtendedElasticSecondaryIndex$DummySecondaryIndex.class
- org/elassandra/index/ExtendedElasticSecondaryIndex.class

You can move back to standard Cassandra by restarting Cassandra binaries or just starting Cassandra from your Elassandra installation:

- For tarball installation, run bin/cassandra (don't use the *-e* flag to enable Elasticsearch)
- For APT installation, set CASSANDRA_DAEMON in /etc/default/cassandra
- For RPM installation, set CASSANDRA_DAEMON in /etc/sysconfig/cassandra

Cassandra automatically builds new secondary indices with one thread. If you want to rebuild faster, stop the on-going rebuild on each node and restart it with the desired number of threads.

### 6.10.2 Migrating from Elasticsearch to Elassandra

Because of data distribution and because Elassandra stores the _source document in Cassandra SSTables, restoring an Elasticsearch snapshot won't work. In order to import data from an existing Elasticsearch cluster to Elassandra, you can use these solutions:

- Use logstash elasticsearch input plugin and the elasticsearch output plugin or the cassandra output plugin. The cassandra output plugin requires a more sophisticated configuration but distributes write load on all Elassandra nodes, while the Elasticsearch output plugin require some heap memory to process bulk inserts.
- Reindex from a remote Elasticsearch cluster as shown bellow:

```
curl -XPOST -H "Content-Type: application/json" "http://localhost:9200/_reindex" -d'{
  "source": {
    "remote": {
```

```
      "host": "http://localhost:9201",
      "socket_timeout": "1m",
      "connect_timeout": "10s"
    },
    "index": "source_index"
  },
  "dest": {
    "index": "dest_index"
  }
}'
```

As explain in the Elasticsearch documentation, remote hosts have to be explicitly whitelisted in **conf/elasticsearch.yml** using the `reindex.remote.whitelist` property.

**Warning:** Reindexing from a remote Elasticsearch cluster can cause a spike on the Elassandra node executing the reindex query.

**Note:** The reindex is only supported when reindexing from a remote Elasticsearch cluster. If you need to reindex data from Elassandra, you should use the nodetool rebuild_index utility.

## 6.11 Tooling

### 6.11.1 JMXMP support

Apache Cassandra rely on JMX over RMI to expose metrics and administration endpoints (used by nodetool). Unfortunately, the RMI approach makes management through a tunnel or a firewall hard or impossible. A simpler approach is to use the JMXMP protocol instead of RMI, java objects are just serialized over a TCP connection.

Elassandra provides JMXMP support when setting the system property `cassandra.jmxmp`:

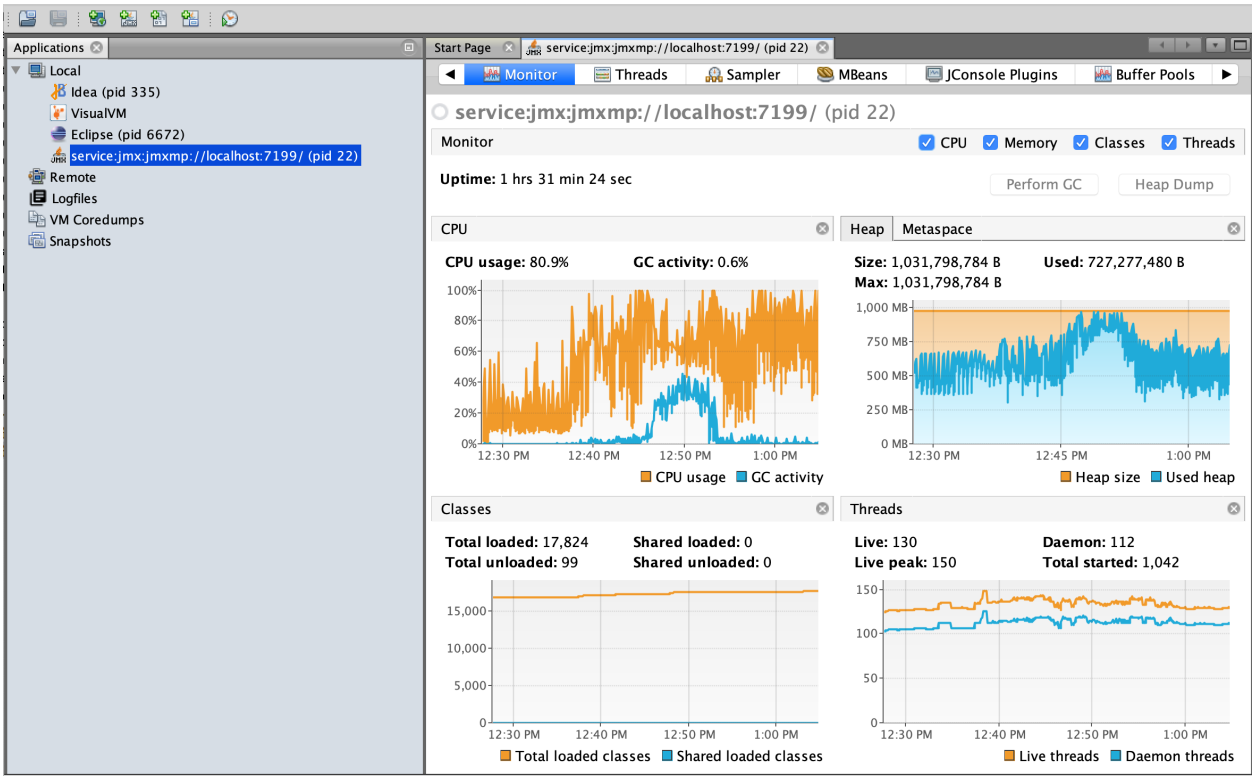
- Add -Dcassandra.jmxmp in **conf/jvm.options**
- Run the **nodetool** utility with the option `--jmxmp` (it adds the system property)

JMXMP TLS/SSL encryption and SASL plain text login/password authentication is also supported with the same settings as JMX over RMI.

In order to use VisualVM (or any other JMX client) :

- Add the jmxremote_optional-repackaged-5.0.jar in the classpath.
- Use the JMX URL `service:jmx:jmxmp://<myhost>:7199/`

Here is a screenshot of visualVM using JMXMP.




Unfortunately, the visualVM does not currently support SASL plain text client authentication.

### 6.11.2 Smile decoder

SIMLE is a binary data format that defines a binary equivalent of standard JSON data format. This is the format used by Elassandra to store the Elasticsearch metadata and index metadata into the 'extensions' of the cassandra CQL schema.

If you want to decode a smile encoded information, Elassandra CLI provides two decode commands: - **decodeMetaData** to decode the elasticsearch cluster state metadata stored in the CQL schema extension of the table **elastic_admin.metadata_log** - **decodeIndexMetadata** to decode the elasticsearch index metadata stored in the CQL schema extensions of the underlying table.

Here after you can see how to decode an index mapping from the extensions column of Cassandra schema_table.

```
$CASSANDRA_HOME/bin/cqlsh -e "select extensions from system_schema.tables where␣
↪keyspace_name = 'test' and table_name = 'docs';"

 extensions
------------------------------------------------------------------------------------------
↪------------------------------------------------------------------------------------------
↪------------------------------------------------------------------------------------------
↪------------------------------------------------------------------------------------------
↪------------------------------------------------------------------------------------------
↪------------------------------------------------------------------------------------------
↪------------------------------------------------------------------------------------------
↪------------------------------------------------------------------------------------------
↪------------------------------------------------------------------------------------------
↪------------------------------------------------------------
 {'elastic_admin/test': 0x3a290a05fa...986616c6961736573fafbfbfb}
```

```
(1 rows)


$CASSANDRA_HOME/bin/elassandra-cli decodeIndexMetadata -s "0x3a290a05fa...
↪986616c6961736573fafbfbfb"
Decoding : [0x3a290a05fa...986616c6961736573fafbfbfb]
{
  "test" : {
    "version" : 1,
    "state" : "open",
    "settings" : {
      "index.creation_date" : "1569831034865",
      "index.provided_name" : "test",
      "index.uuid" : "4tne58smR7e5nwOrdD1VvA",
      "index.version.created" : "6020399"
    },
    "mappings" : [ {
      "docs" : {
        "properties" : {
          "login" : {
            "type" : "keyword",
            "cql_collection" : "singleton"
          },
          "uid" : {
            "type" : "integer",
            "cql_collection" : "singleton",
            "cql_partition_key" : true,
            "cql_primary_key_order" : 0
          },
          "username" : {
            "type" : "nested",
            "cql_collection" : "singleton",
            "cql_udt_name" : "user_type",
            "properties" : {
              "first" : {
                "type" : "keyword",
                "cql_collection" : "singleton"
              },
              "last" : {
                "type" : "keyword",
                "cql_collection" : "singleton"
              }
            }
          }
        }
      }
    } ],
    "aliases" : { }
  }
}
```

CHAPTER 7

# Search through CQL

Executing Elasticsearch queries through through the Cassandra CQL driver provides several benefits:

- Eliminates the needs for an HTTP load balancer because the drivers are cluster aware and will load balance for you.
- Simplify the development of your application by using the same Data Access Objects for CQL and Elasticsearch requests.
- Get integrated security with Cassandra authentication and TLS encryption.
- Manage the Elasticsearch scrolling through the CQL paging.

## 7.1 Configuration

To enable Elasticsearch query over CQL:

- Set the Cassandra CQL query handler with the system property `cassandra.custom_query_handler_class` in your cassandra-env.sh and restart your nodes:

```
JVM_OPTS="$JVM_OPTS -Dcassandra.custom_query_handler_class=org.elassandra.index.
↪ElasticQueryHandler"
```

- Add a dummy column `es_query` to your cassandra table.
- Add a dummy column `es_options` to your cassandra table if you need to specify some specific options like target index names.

```
ALTER TABLE twitter.tweet ADD es_query text;
ALTER TABLE twitter.tweet ADD es_options text;
```

## 7.2 Search request through CQL

Then you can query the associated Elasticsearch index directly into a CQL SELECT request such as (document *_type* is the cassandra table name).

```
cassandra@cqlsh> SELECT "_id",foo FROM twitter.tweet WHERE es_query='{"query":{"query_
↪string":{"query":"bar2*"}}}';

 _id | foo
-----+-------
   2 |  bar2
  20 | bar20
  22 | bar22
  23 | bar23
  24 | bar24
  28 | bar28
  21 | bar21
  25 | bar25
  26 | bar26
  27 | bar27

(10 rows)
```

If your target index does not have the same name as the underlying keyspace one, you can specify targeted indices names in `es_options`, that must be followed by `ALLOW FILTERING`.

```
cassandra@cqlsh> SELECT "_id",foo FROM twitter.tweet WHERE es_query='{"query":{"query_
↪string":{"query":"bar2*"}}}' AND es_options='indices=twitter*' ALLOW FILTERING;
```

## 7.3 Paging

By default, when the Cassandra driver paging is enabled, the CQL query handler open a scroll cursor to retrieve large numbers of results (or even all results). The scroll context is automatically released when fetching the last page. The default scoll timeout is 60 seconds.

If you only need the first N results, use the CQL LIMIT clause as shown below. When the resquested LIMIT is lower than the CQL page size (default is 5000, see CQL Paging), the CQL query handler does not open a scroll cursor, but just set the elasticsearch query size.

```
cassandra@cqlsh> SELECT "_id",foo FROM twitter.tweet WHERE es_query='{"query":{"query_
↪string":{"query":"bar2*"}}}' LIMIT 3;

 _id | foo
-----+-------
   2 |  bar2
  20 | bar20
  22 | bar22

(3 rows)
```

If CQL paging is disabled and the LIMIT clause is absent, the CQL query handler execute a search request with an Elasticsearch query size limited to 10000, as the default *index.max_result_window*‘ (see dynamic index settings <https://www.elastic.co/guide/en/elasticsearch/reference/current/index-modules.html#dynamic-index-settings>‘_)

## 7.4 Routing

If all partition key columns are set in the where clause, the Elasticsearch query will be directly sent to a node hosting the data (no fan out).

```
cassandra@cqlsh> SELECT "_id", foo FROM twitter.tweet WHERE es_query='{"query":{
↪"query_string":{"query":"bar2*"}}}' AND "_id"='2';

 _id | foo
-----+------
   2 | bar2

(1 rows)
```

## 7.5 CQL Functions

Cassandra functions and User Defined Functions can be used in the CQL projection clause.

```
cassandra@cqlsh> SELECT "_id",foo,token("_id"),writetime(foo) FROM twitter.tweet␣
↪WHERE es_query='{"query":{"query_string":{"query":"bar2*"}}}';

 _id | foo    | system.token(_id)     | writetime(foo)
-----+--------+-----------------------+------------------
   2 |  bar2 |  5293579765126103566 | 1509275059354000
  20 | bar20 |  4866192165766252016 | 1509275059572000
  22 | bar22 |  5315788262387249245 | 1509275059591000
  23 | bar23 |  5502885531913083742 | 1509275059600000
  24 | bar24 |  5568379873904613205 | 1509275059614000
  28 | bar28 |  3168262793124788288 | 1509275059663000
  21 | bar21 | -3201810799627846645 | 1509275059580000
  25 | bar25 |  2509205981756244107 | 1509275059625000
  26 | bar26 | -6132418777949225301 | 1509275059633000
  27 | bar27 |  9060526884622895268 | 1509275059645000

(10 rows)
```

## 7.6 Elasticsearch aggregations through CQL

Elassandra supports the elasticsearch aggregation only in **regular CQL statement**. In this case :

- Returned columns are named with aggregations names.
- CQL functions are not supported.
- CQL projection clause, limit and pagination are ignored. It also implies that aggregation results must fit into the available memory.

```
cassandra@cqlsh> SELECT * FROM twitter2.doc WHERE es_query='{"aggs":{"sales_per_month
↪":{"date_histogram":{"field":"post_date","interval":"day"},"aggs":{"sales":{"sum":{
↪"field":"price"}}}}}}';

 sales_per_month.key             | sales_per_month.count | sales_per_month.sales.sum
---------------------------------+-----------------------+---------------------------
```

```
 2017-10-04 00:00:00.000000+0000 |                     3 |                          30
 2017-10-05 00:00:00.000000+0000 |                     1 |                          10
 2017-10-06 00:00:00.000000+0000 |                     1 |                          10
 2017-10-07 00:00:00.000000+0000 |                     3 |                          30

(4 rows)
```

When requesting multiple sibling aggregations, the tree result is flattened. In the following example, there are two top level aggregations named *sales_per_month* and *sum_monthly_sales*.

```
cassandra@cqlsh> SELECT * FROM twitter2.doc WHERE es_query='{"size":0,
      "aggs":{"sales_per_month":{"date_histogram":{"field":"post_date","interval":"day
↪"},"aggs":{"sales":{"sum":{"field":"price"}}}},
      "sum_monthly_sales":{"sum_bucket":{"buckets_path":"sales_per_month>sales"}}}}';

 sales_per_month.key             | sales_per_month.count | sales_per_month.sales.sum␣
↪| sum_monthly_sales.value

---------------------------------+-----------------------+---------------------------
↪+-------------------------
 2017-10-04 00:00:00.000000+0000 |                     3 |                          30␣
↪|                    null
 2017-10-05 00:00:00.000000+0000 |                     1 |                          10␣
↪|                    null
 2017-10-06 00:00:00.000000+0000 |                     1 |                          10␣
↪|                    null
 2017-10-07 00:00:00.000000+0000 |                     3 |                          30␣
↪|                    null
                            null |                  null |                        null␣
↪|                      80

(5 rows)
```

**Note:** If the aggregation type returns more than one value like *stats*, you have to request json output in *es_options*

ex : select * from twitter2.doc where es_query='{“size”:0, “aggs”:{{“sales”:{“stats”:{“field”:”price”}}}}}’ and es_options=’indices=twitter2;json=true’

**Note:** Elassandra only supports the following aggregation types over CQL:

- Term aggregation
- Histogram aggregation
- Date Histogram aggregation
- Percentiles aggregation
- Sum aggregation
- Avg aggregation
- Min aggregation
- Max aggregation
- Stats aggregation

## 7.7 Distributed Elasticsearch aggregation with Apache Spark

In order to use the Elasticsearch aggregation capabilities from Apache Spark, you must request Elassandra with a projection clause having the same CQL types as the returned aggregation results. Moreover, do not re-use the same column name more than once, otherwise you will get an **IndexOutOfBoundsException** while Apache Spark parses the result. In the following example, we used dummy columns count2, dc_power1, dc_power2 and dc_power3 to fit the aggregation results :

```
import org.apache.spark.{SparkConf, SparkContext}
import com.datastax.spark.connector._
import org.apache.spark.sql.cassandra._
val query = """{
  "query":{
    "bool":{
      "filter": [
        {"term": { "datalogger_name": "mysensor" }},
        {"range" : {
            "ts" : { "gte" : "2017-12-16", "lte" : "2018-01-20"  }
        }}
      ]
    }
  },
  "aggs":{
    "hour_agg":{
      "date_histogram":{"field":"ts","interval":"hour"},
      "aggs": {
        "agg_irradiance": {
          "avg": {
            "field": "irradiance"
          }
        },
        "agg_conso": {
          "avg": {
            "field": "altitude"
          }
        },
       "water1":{
            "terms":{"field":"azimuth"},
            "aggs":{
              "dc_power_agg":{ "sum":{"field":"dc_power"}}
            }
       }
      }
    }
  }
}"""
val t = sc.cassandraTable("iot", "sensors").select("ts","count","dc_power","dc_power1
↪","dc_power2","count2","dc_power3").where("es_query='"+query+"'");
t.collect.foreach(println)

CassandraRow{ts: 2017-12-31 00:00:00+0100, count: 204, dc_power: 0.0, dc_power1: null,
↪ dc_power2: 305.64675177506786, count2: 17, dc_power3: 0.0}
CassandraRow{ts: 2017-12-31 00:00:00+0100, count: 204, dc_power: 0.0, dc_power1: null,
↪ dc_power2: 308.4126297573829, count2: 17, dc_power3: 0.0}
CassandraRow{ts: 2017-12-31 00:00:00+0100, count: 204, dc_power: 0.0, dc_power1: null,
↪ dc_power2: 311.4319809865401, count2: 17, dc_power3: 0.0}
CassandraRow{ts: 2017-12-31 00:00:00+0100, count: 204, dc_power: 0.0, dc_power1: null,
↪ dc_power2: 314.7328283387269, count2: 17, dc_power3: 0.0}
```

```
CassandraRow{ts: 2017-12-31 00:00:00+0100, count: 204, dc_power: 0.0, dc_power1: null,
↪ dc_power2: 318.34321582364055, count2: 17, dc_power3: 0.0}
CassandraRow{ts: 2017-12-31 00:00:00+0100, count: 204, dc_power: 0.0, dc_power1: null,
↪ dc_power2: 322.28910238170704, count2: 17, dc_power3: 0.0}
CassandraRow{ts: 2017-12-31 00:00:00+0100, count: 204, dc_power: 0.0, dc_power1: null,
↪ dc_power2: 326.59122459682067, count2: 17, dc_power3: 0.0}
CassandraRow{ts: 2017-12-31 00:00:00+0100, count: 204, dc_power: 0.0, dc_power1: null,
↪ dc_power2: 331.2608198139219, count2: 17, dc_power3: 0.0}
CassandraRow{ts: 2017-12-31 00:00:00+0100, count: 204, dc_power: 0.0, dc_power1: null,
↪ dc_power2: 336.2944302705681, count2: 17, dc_power3: 0.0}
```

Alternatively, you can request an Apache Spark to get the aggregation results as JSON objects by adding the option **json=true** to the query `es_options` as follow :

```
val t = sc.cassandraTable("iot", "sensors").select("es_query").where("es_query='
↪"+query+"' AND es_options='json=true'");
t.collect.foreach(println)

CassandraRow{es_query: {"key_as_string":"2017-12-30T23:00:00.000Z","key
↪":1514674800000,"doc_count":204,"agg_irradiance":{"value":0.0},"water1":{"doc_count_
↪error_upper_bound":0,"sum_other_doc_count":34,"buckets":[{"key":305.64675177506786,
↪"doc_count":17,"dc_power_agg":{"value":0.0}},{"key":308.4126297573829,"doc_count
↪":17,"dc_power_agg":{"value":0.0}},{"key":311.4319809865401,"doc_count":17,"dc_
↪power_agg":{"value":0.0}},{"key":314.7328283387269,"doc_count":17,"dc_power_agg":{
↪"value":0.0}},{"key":318.34321582364055,"doc_count":17,"dc_power_agg":{"value":0.0}}
↪,{"key":322.28910238170704,"doc_count":17,"dc_power_agg":{"value":0.0}},{"key":326.
↪59122459682067,"doc_count":17,"dc_power_agg":{"value":0.0}},{"key":331.
↪2608198139219,"doc_count":17,"dc_power_agg":{"value":0.0}},{"key":336.2944302705681,
↪"doc_count":17,"dc_power_agg":{"value":0.0}},{"key":341.6684918842001,"doc_count
↪":17,"dc_power_agg":{"value":0.0}}]},"agg_conso":{"value":0.0}}}
CassandraRow{es_query: {"key_as_string":"2017-12-31T00:00:00.000Z","key
↪":1514678400000,"doc_count":204,"agg_irradiance":{"value":0.0},"water1":{"doc_count_
↪error_upper_bound":0,"sum_other_doc_count":34,"buckets":[{"key":5.253033308292965,
↪"doc_count":17,"dc_power_agg":{"value":0.0}},{"key":11.17937932261813,"doc_count
↪":17,"dc_power_agg":{"value":0.0}},{"key":16.9088341251606,"doc_count":17,"dc_power_
↪agg":{"value":0.0}},{"key":22.361824055627704,"doc_count":17,"dc_power_agg":{"value
↪":0.0}},{"key":27.483980631203153,"doc_count":17,"dc_power_agg":{"value":0.0}},{"key
↪":32.24594386978638,"doc_count":17,"dc_power_agg":{"value":0.0}},{"key":36.
↪63970141314307,"doc_count":17,"dc_power_agg":{"value":0.0}},{"key":40.
↪673315954868855,"doc_count":17,"dc_power_agg":{"value":0.0}},{"key":44.
↪36558478428467,"doc_count":17,"dc_power_agg":{"value":0.0}},{"key":47.
↪74149653565296,"doc_count":17,"dc_power_agg":{"value":0.0}}]},"agg_conso":{"value
↪":0.0}}}
CassandraRow{es_query: {"key_as_string":"2017-12-31T01:00:00.000Z","key
↪":1514682000000,"doc_count":204,"agg_irradiance":{"value":0.0},"water1":{"doc_count_
↪error_upper_bound":0,"sum_other_doc_count":34,"buckets":[{"key":53.65569068831377,
↪"doc_count":17,"dc_power_agg":{"value":0.0}},{"key":56.249279017946265,"doc_count
↪":17,"dc_power_agg":{"value":0.0}},{"key":58.63483107417463,"doc_count":17,"dc_
↪power_agg":{"value":0.0}},{"key":60.835352658997266,"doc_count":17,"dc_power_agg":{
↪"value":0.0}},{"key":62.87149505671871,"doc_count":17,"dc_power_agg":{"value":0.0}},
↪{"key":64.76161651252164,"doc_count":17,"dc_power_agg":{"value":0.0}},{"key":66.
↪52193854036197,"doc_count":17,"dc_power_agg":{"value":0.0}},{"key":68.
↪16674119813763,"doc_count":17,"dc_power_agg":{"value":0.0}},{"key":69.
↪70857084793244,"doc_count":17,"dc_power_agg":{"value":0.0}},{"key":71.
↪15844512445423,"doc_count":17,"dc_power_agg":{"value":0.0}}]},"agg_conso":{"value
↪":0.0}}}
```

## 7.8 CQL Driver integration

In order to build elasticsearch queries, query builders and helpers from elastic can be used as shown bellow and use it in a CQL prepared statement.

```
String esQuery = new SearchSourceBuilder().query(QueryBuilders.termQuery("user",
↪"vince")).toString(ToXContent.EMPTY_PARAMS);
ResultSet results = cassandraCQLUnit.session.execute("SELECT * FROM users WHERE es_
↪query = ? ALLOW FILTERING", esQuery);
```

You can also retrieve the Elasticsearch results summary **hits.total**, **hits.max_score**, **_shards.total**, **_shards.successful**, **_shards.skipped** and **_shards.failed** from the result custom payload.

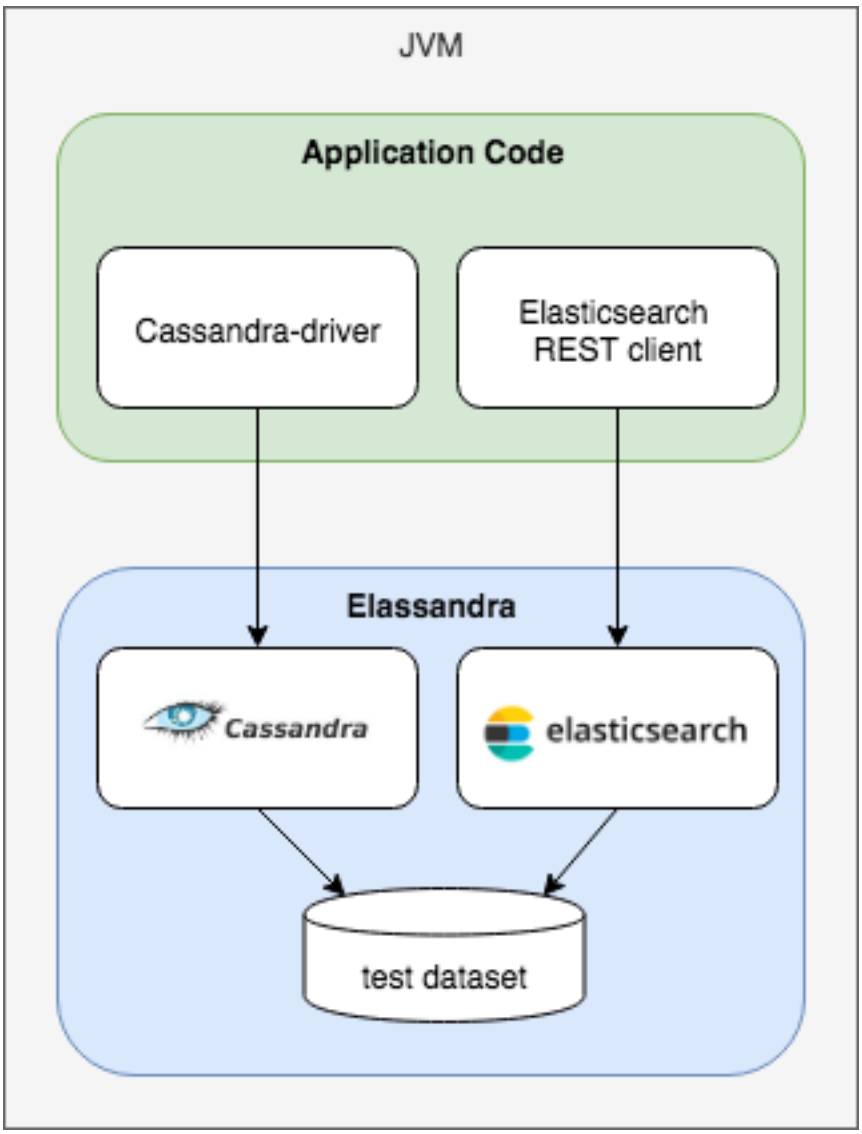
```
public static class IncomingPayload {
     public final long hitTotal;
     public final float hitMaxScore;
     public final int shardTotal;
     public final int shardSuccessful;
     public final int shardSkipped;
     public final int shardFailed;
     public IncomingPayload(Map<String,ByteBuffer> payload) {
          hitTotal = payload.get("hits.total").getLong();
          hitMaxScore = payload.get("hits.max_score").getFloat();
          shardTotal = payload.get("_shards.total").getInt();
          shardSuccessful = payload.get("_shards.successful").getInt();
          shardSkipped = payload.get("_shards.skipped").getInt();
          shardFailed = payload.get("_shards.failed").getInt();
     }
}

String esQuery = "{\"query\":{\"match_all\":{}}}";
ResultSet rs = session.execute("SELECT * FROM ks.table WHERE es_query=?", esQuery);
IncomingPayload payload = new IncomingPayload(rs.getExecutionInfo().
↪getIncomingPayload());
System.out.println("hits.total="+payload.hitTotal);
```

**Tip:** When sum of **_shards.successful**, **_shards.skipped** and **_shards.failed** is lower than **_shards.total**, it means the search is not consistent because of missing nodes. In such cases, index state is red.

## 7.9 Application UNIT Tests

Elassandra Unit helps you writing isolated JUnit tests in a Test Driven Development style with an embedded Elassandra instance.




In order to execute Elasticsearch queries through CQL with Elassandra unit, set the system property `cassandra.custom_query_handler_class` to `org.elassandra.index.ElasticQueryHandler`. Moreover set the system property `cassandra.ring_delay_ms` to 0 to quickly start elassandra.

Maven configuration:

```
<plugins>
   <plugin>
     <groupId>org.apache.maven.plugins</groupId>
     <artifactId>maven-surefire-plugin</artifactId>
     <version>3.0.0-M3</version>
     <configuration>
       <systemPropertyVariables>
         <cassandra.custom_query_handler_class>org.elassandra.index.
↪ElasticQueryHandler</cassandra.custom_query_handler_class>
         <cassandra.ring_delay_ms>0</cassandra.ring_delay_ms>
       </systemPropertyVariables>
     </configuration>
   </plugin>
</plugins>
```

## 7.10 CQL Tracing

Elasticsearch search request may invlove CQL requests to requested fields from the underlying Cassandra table. When searching through CQL, you can use Cassandra tracing capabilities to troubleshoot the Cassandra performance problems.

```
cassandra@cqlsh> tracing on;
Now Tracing is enabled
cassandra@cqlsh> SELECT * FROM twitter2.doc WHERE es_query='{"query":{"match_all":{}}}
↪';

 _id | es_options | es_query | message                                                 |␣
↪post_date                          | price | user
-----+------------+----------+---------------------------------------------------------+-----
↪--------------------------------+-------+------------
   2 |       null |     null | ['Elassandra adds dynamic mapping to Cassandra'] | [
↪'2017-10-04 14:12:00.000000+0000'] |  [10] | ['Poulpy']
   3 |       null |     null | ['Elassandra adds dynamic mapping to Cassandra'] | [
↪'2017-10-04 15:12:00.000000+0000'] |  [10] | ['Poulpy']
   5 |       null |     null | ['Elassandra adds dynamic mapping to Cassandra'] | [
↪'2017-10-06 13:12:00.000000+0000'] |  [10] | ['Poulpy']
   8 |       null |     null | ['Elassandra adds dynamic mapping to Cassandra'] | [
↪'2017-10-07 18:12:00.000000+0000'] |  [10] | ['Poulpy']
   1 |       null |     null | ['Elassandra adds dynamic mapping to Cassandra'] | [
↪'2017-10-04 13:12:00.000000+0000'] |  [10] | ['Poulpy']
   4 |       null |     null | ['Elassandra adds dynamic mapping to Cassandra'] | [
↪'2017-10-05 13:12:00.000000+0000'] |  [10] | ['Poulpy']
   6 |       null |     null | ['Elassandra adds dynamic mapping to Cassandra'] | [
↪'2017-10-07 13:12:00.000000+0000'] |  [10] | ['Poulpy']
   7 |       null |     null | ['Elassandra adds dynamic mapping to Cassandra'] | [
↪'2017-10-07 15:12:00.000000+0000'] |  [10] | ['Poulpy']

(8 rows)

Tracing session: 817762d0-c6d8-11e7-80c9-cf9ea31c7788

 activity
↪                                | timestamp                  | source    | source_␣
↪elapsed | client
-----------------------------------------------------------------------------------------
↪--------------------------------+----------------------------+-----------+-----------
↪----+-----------

↪                                                                                      ␣
↪          Elasticsearch query | 2017-11-11 13:04:44.544000 | 127.0.0.1 |           ␣
↪  0 | 127.0.0.1
     Parsing SELECT * FROM twitter2.doc WHERE es_query='{"query":{"match_all":{}}}';␣
↪[Native-Transport-Requests-1] | 2017-11-11 13:04:44.541000 | 127.0.0.1 |           ␣
↪192 | 127.0.0.1
                                                                 Preparing statement␣
↪[Native-Transport-Requests-1] | 2017-11-11 13:04:44.541000 | 127.0.0.1 |           ␣
↪382 | 127.0.0.1
                                                               Executing single-
↪partition query on roles [ReadStage-2] | 2017-11-11 13:04:44.542000 | 127.0.0.1 |  ␣
↪        1048 | 127.0.0.1
```

```
                                                                                Acquiring␣
↪sstable references [ReadStage-2] | 2017-11-11 13:04:44.542000 | 127.0.0.1 |          ␣
↪  1145 | 127.0.0.1
                                Skipped 0/1 non-slice-intersecting sstables, included 0␣
↪due to tombstones [ReadStage-2] | 2017-11-11 13:04:44.542000 | 127.0.0.1 |          ␣
↪ 1327 | 127.0.0.1
                                                                              Key cache␣
↪hit for sstable 1 [ReadStage-2] | 2017-11-11 13:04:44.542000 | 127.0.0.1 |          ␣
↪ 1475 | 127.0.0.1
                                                                  Merged data from␣
↪memtables and 1 sstables [ReadStage-2] | 2017-11-11 13:04:44.543000 | 127.0.0.1 |  ␣
↪         1724 | 127.0.0.1
                                                                Read 1 live and 0␣
↪tombstone cells [ReadStage-2] | 2017-11-11 13:04:44.543000 | 127.0.0.1 |          ␣
↪1830 | 127.0.0.1
                                                                  Executing single-
↪partition query on roles [ReadStage-4] | 2017-11-11 13:04:44.543000 | 127.0.0.1 |  ␣
↪         2279 | 127.0.0.1
                                                                                Acquiring␣
↪sstable references [ReadStage-4] | 2017-11-11 13:04:44.543000 | 127.0.0.1 |          ␣
↪  2360 | 127.0.0.1
                                Skipped 0/1 non-slice-intersecting sstables, included 0␣
↪due to tombstones [ReadStage-4] | 2017-11-11 13:04:44.543000 | 127.0.0.1 |          ␣
↪ 2432 | 127.0.0.1
                                                                              Key cache␣
↪hit for sstable 1 [ReadStage-4] | 2017-11-11 13:04:44.543000 | 127.0.0.1 |          ␣
↪ 2509 | 127.0.0.1
                                                                  Merged data from␣
↪memtables and 1 sstables [ReadStage-4] | 2017-11-11 13:04:44.544000 | 127.0.0.1 |  ␣
↪         2736 | 127.0.0.1
                                                                Read 1 live and 0␣
↪tombstone cells [ReadStage-4] | 2017-11-11 13:04:44.544000 | 127.0.0.1 |          ␣
↪2801 | 127.0.0.1
                                        Executing single-partition query on doc␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.552000 | 127.0.0.1 | ␣
↪          143 | 127.0.0.1
                                                    Acquiring sstable references␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.552000 | 127.0.0.1 | ␣
↪          311 | 127.0.0.1
                                                     Key cache hit for sstable 5␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.552000 | 127.0.0.1 | ␣
↪          438 | 127.0.0.1
                                                     Key cache hit for sstable 6␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.553000 | 127.0.0.1 | ␣
↪          553 | 127.0.0.1
  Skipped 0/2 non-slice-intersecting sstables, included 0 due to tombstones␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.553000 | 127.0.0.1 | ␣
↪          624 | 127.0.0.1
                                    Merged data from memtables and 2 sstables␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.553000 | 127.0.0.1 | ␣
↪          953 | 127.0.0.1
                                          Read 1 live and 0 tombstone cells␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.553000 | 127.0.0.1 | ␣
↪         1031 | 127.0.0.1
                                        Executing single-partition query on doc␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.553000 | 127.0.0.1 | ␣
↪         1280 | 127.0.0.1
                                                    Acquiring sstable references␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.553000 | 127.0.0.1 | ␣
↪         1335 | 127.0.0.1
```

```
                                                    Key cache hit for sstable 5␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.553001 | 127.0.0.1 | ␣
↪          1423 | 127.0.0.1
                                                    Key cache hit for sstable 6␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.554000 | 127.0.0.1 | ␣
↪          1515 | 127.0.0.1
  Skipped 0/2 non-slice-intersecting sstables, included 0 due to tombstones␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.554000 | 127.0.0.1 | ␣
↪          1593 | 127.0.0.1
                                        Merged data from memtables and 2 sstables␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.554000 | 127.0.0.1 | ␣
↪          1853 | 127.0.0.1
                                       Read 1 live and 0 tombstone cells␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.554000 | 127.0.0.1 | ␣
↪          1921 | 127.0.0.1
                                     Executing single-partition query on doc␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.554000 | 127.0.0.1 | ␣
↪          2091 | 127.0.0.1
                                         Acquiring sstable references␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.554000 | 127.0.0.1 | ␣
↪          2136 | 127.0.0.1
                                                    Key cache hit for sstable 5␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.554001 | 127.0.0.1 | ␣
↪          2253 | 127.0.0.1
                                                    Key cache hit for sstable 6␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.554001 | 127.0.0.1 | ␣
↪          2346 | 127.0.0.1
  Skipped 0/2 non-slice-intersecting sstables, included 0 due to tombstones␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.554001 | 127.0.0.1 | ␣
↪          2408 | 127.0.0.1
                                        Merged data from memtables and 2 sstables␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.555000 | 127.0.0.1 | ␣
↪          2654 | 127.0.0.1
                                     Executing single-partition query on doc␣
↪[elasticsearch[127.0.0.2][search][T#10]] | 2017-11-11 13:04:44.555000 | 127.0.0.2 |␣
↪            116 | 127.0.0.1
                                       Read 1 live and 0 tombstone cells␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.555000 | 127.0.0.1 | ␣
↪          2733 | 127.0.0.1
                                         Acquiring sstable references␣
↪[elasticsearch[127.0.0.2][search][T#10]] | 2017-11-11 13:04:44.555000 | 127.0.0.2 |␣
↪            303 | 127.0.0.1
                                     Executing single-partition query on doc␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.555000 | 127.0.0.1 | ␣
↪          2950 | 127.0.0.1
                                         Acquiring sstable references␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.555000 | 127.0.0.1 | ␣
↪          3002 | 127.0.0.1
                                                    Key cache hit for sstable 5␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.555000 | 127.0.0.1 | ␣
↪          3095 | 127.0.0.1
                                                    Key cache hit for sstable 6␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.555000 | 127.0.0.1 | ␣
↪          3191 | 127.0.0.1
  Skipped 0/2 non-slice-intersecting sstables, included 0 due to tombstones␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.555001 | 127.0.0.1 | ␣
↪          3253 | 127.0.0.1
                                        Merged data from memtables and 2 sstables␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.556000 | 127.0.0.1 | ␣
↪          3549 | 127.0.0.1
```

```
                                                    Key cache hit for sstable 5␣
↪[elasticsearch[127.0.0.2][search][T#10]] | 2017-11-11 13:04:44.556000 | 127.0.0.2 |␣
↪           480 | 127.0.0.1
                                                 Read 1 live and 0 tombstone cells␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.556000 | 127.0.0.1 | ␣
↪         3656 | 127.0.0.1
                                                    Key cache hit for sstable 6␣
↪[elasticsearch[127.0.0.2][search][T#10]] | 2017-11-11 13:04:44.556000 | 127.0.0.2 |␣
↪           650 | 127.0.0.1
 Skipped 0/2 non-slice-intersecting sstables, included 0 due to tombstones␣
↪[elasticsearch[127.0.0.2][search][T#10]] | 2017-11-11 13:04:44.556000 | 127.0.0.2 |␣
↪           747 | 127.0.0.1
                                       Merged data from memtables and 2 sstables␣
↪[elasticsearch[127.0.0.2][search][T#10]] | 2017-11-11 13:04:44.556000 | 127.0.0.2 |␣
↪          1245 | 127.0.0.1
                                                 Read 1 live and 0 tombstone cells␣
↪[elasticsearch[127.0.0.2][search][T#10]] | 2017-11-11 13:04:44.556000 | 127.0.0.2 |␣
↪          1362 | 127.0.0.1

↪                Request complete | 2017-11-11 13:04:44.563745 | 127.0.0.1 |          ␣
↪19745 | 127.0.0.1
```

You can then retreive tracing information stored into the system_traces keyspace for 24 hours as demonstrated below.

```
cassandra@cqlsh> select * from system_traces.sessions;

 session_id                           | client    | command | coordinator | duration␣
↪| parameters                                                                          ␣
↪                                                                                    ␣
↪       | request              | started_at
--------------------------------------+-----------+---------+-------------+----------
↪+-----------------------------------------------------------------------------------
↪-----------------------------------------------------------------------------------
↪-------+----------------------+---------------------------------
 817762d0-c6d8-11e7-80c9-cf9ea31c7788 | 127.0.0.1 |   QUERY |   127.0.0.1 |    19745␣
↪| {'consistency_level': 'ONE', 'page_size': '100', 'query': 'SELECT * FROM twitter2.
↪doc WHERE es_query=''{"query":{"match_all":{}}}'';', 'serial_consistency_level':
↪'SERIAL'} | Elasticsearch query | 2017-11-11 12:04:44.544000+0000
 7c49dae0-c6d8-11e7-80c9-cf9ea31c7788 | 127.0.0.1 |   QUERY |   127.0.0.1 |    20002␣
↪| {'consistency_level': 'ONE', 'page_size': '100', 'query': 'SELECT * FROM twitter2.
↪doc WHERE es_query=''{"query":{"match_all":{}}}'';', 'serial_consistency_level':
↪'SERIAL'} | Elasticsearch query | 2017-11-11 12:04:35.856000+0000
 6786c2d0-c6d8-11e7-80c9-cf9ea31c7788 | 127.0.0.1 |   QUERY |   127.0.0.1 |    16426␣
↪|                                                {'consistency_level': 'ONE', 'page_
↪size': '100', 'query': 'SELECT * FROM twitter2.doc ;', 'serial_consistency_level':
↪'SERIAL'} |  Execute CQL3 query | 2017-11-11 12:04:01.021000+0000
 6b49e550-c6d8-11e7-80c9-cf9ea31c7788 | 127.0.0.1 |   QUERY |   127.0.0.1 |    14129␣
↪|                                                 {'consistency_level': 'ONE', 'page_
↪size': '100', 'query': 'SELECT * FROM twitter2.doc;', 'serial_consistency_level':
↪'SERIAL'} |  Execute CQL3 query | 2017-11-11 12:04:07.333000+0000

(4 rows)
cassandra@cqlsh> SHOW SESSION 817762d0-c6d8-11e7-80c9-cf9ea31c7788;

Tracing session: 817762d0-c6d8-11e7-80c9-cf9ea31c7788

 activity
↪                                | timestamp                  | source    | source_
↪elapsed | client
```

```
-------------------------------------------------------------------------------------
↪-----------------------------+----------------------------+-----------+------------
↪----+-----------

↪                Elasticsearch query | 2017-11-11 13:04:44.544000 | 127.0.0.1 |
↪  0 | 127.0.0.1
     Parsing SELECT * FROM twitter2.doc WHERE es_query='{"query":{"match_all":{}}}';
↪[Native-Transport-Requests-1] | 2017-11-11 13:04:44.541000 | 127.0.0.1 |
↪192 | 127.0.0.1
                                                                Preparing statement
↪[Native-Transport-Requests-1] | 2017-11-11 13:04:44.541000 | 127.0.0.1 |
↪382 | 127.0.0.1
                                                          Executing single-
↪partition query on roles [ReadStage-2] | 2017-11-11 13:04:44.542000 | 127.0.0.1 |
↪        1048 | 127.0.0.1
                                                               Acquiring
↪sstable references [ReadStage-2] | 2017-11-11 13:04:44.542000 | 127.0.0.1 |
↪  1145 | 127.0.0.1
                              Skipped 0/1 non-slice-intersecting sstables, included 0
↪due to tombstones [ReadStage-2] | 2017-11-11 13:04:44.542000 | 127.0.0.1 |
↪ 1327 | 127.0.0.1
                                                                 Key cache
↪hit for sstable 1 [ReadStage-2] | 2017-11-11 13:04:44.542000 | 127.0.0.1 |
↪ 1475 | 127.0.0.1
                                                          Merged data from
↪memtables and 1 sstables [ReadStage-2] | 2017-11-11 13:04:44.543000 | 127.0.0.1 |
↪        1724 | 127.0.0.1
                                                             Read 1 live and 0
↪tombstone cells [ReadStage-2] | 2017-11-11 13:04:44.543000 | 127.0.0.1 |
↪1830 | 127.0.0.1
                                                          Executing single-
↪partition query on roles [ReadStage-4] | 2017-11-11 13:04:44.543000 | 127.0.0.1 |
↪        2279 | 127.0.0.1
                                                               Acquiring
↪sstable references [ReadStage-4] | 2017-11-11 13:04:44.543000 | 127.0.0.1 |
↪  2360 | 127.0.0.1
                              Skipped 0/1 non-slice-intersecting sstables, included 0
↪due to tombstones [ReadStage-4] | 2017-11-11 13:04:44.543000 | 127.0.0.1 |
↪ 2432 | 127.0.0.1
                                                                 Key cache
↪hit for sstable 1 [ReadStage-4] | 2017-11-11 13:04:44.543000 | 127.0.0.1 |
↪ 2509 | 127.0.0.1
                                                          Merged data from
↪memtables and 1 sstables [ReadStage-4] | 2017-11-11 13:04:44.544000 | 127.0.0.1 |
↪        2736 | 127.0.0.1
                                                             Read 1 live and 0
↪tombstone cells [ReadStage-4] | 2017-11-11 13:04:44.544000 | 127.0.0.1 |
↪2801 | 127.0.0.1
                                    Executing single-partition query on doc
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.552000 | 127.0.0.1 |
↪          143 | 127.0.0.1
                                              Acquiring sstable references
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.552000 | 127.0.0.1 |
↪          311 | 127.0.0.1
                                               Key cache hit for sstable 5
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.552000 | 127.0.0.1 |
↪          438 | 127.0.0.1
                                               Key cache hit for sstable 6
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.553000 | 127.0.0.1 |
↪          553 | 127.0.0.1
```

```
 Skipped 0/2 non-slice-intersecting sstables, included 0 due to tombstones␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.553000 | 127.0.0.1 | ␣
↪          624 | 127.0.0.1
                                    Merged data from memtables and 2 sstables␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.553000 | 127.0.0.1 | ␣
↪          953 | 127.0.0.1
                                  Read 1 live and 0 tombstone cells␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.553000 | 127.0.0.1 | ␣
↪         1031 | 127.0.0.1
                                Executing single-partition query on doc␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.553000 | 127.0.0.1 | ␣
↪         1280 | 127.0.0.1
                                    Acquiring sstable references␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.553000 | 127.0.0.1 | ␣
↪         1335 | 127.0.0.1
                                     Key cache hit for sstable 5␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.553001 | 127.0.0.1 | ␣
↪         1423 | 127.0.0.1
                                     Key cache hit for sstable 6␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.554000 | 127.0.0.1 | ␣
↪         1515 | 127.0.0.1
 Skipped 0/2 non-slice-intersecting sstables, included 0 due to tombstones␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.554000 | 127.0.0.1 | ␣
↪         1593 | 127.0.0.1
                                    Merged data from memtables and 2 sstables␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.554000 | 127.0.0.1 | ␣
↪         1853 | 127.0.0.1
                                  Read 1 live and 0 tombstone cells␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.554000 | 127.0.0.1 | ␣
↪         1921 | 127.0.0.1
                                Executing single-partition query on doc␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.554000 | 127.0.0.1 | ␣
↪         2091 | 127.0.0.1
                                    Acquiring sstable references␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.554000 | 127.0.0.1 | ␣
↪         2136 | 127.0.0.1
                                     Key cache hit for sstable 5␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.554001 | 127.0.0.1 | ␣
↪         2253 | 127.0.0.1
                                     Key cache hit for sstable 6␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.554001 | 127.0.0.1 | ␣
↪         2346 | 127.0.0.1
 Skipped 0/2 non-slice-intersecting sstables, included 0 due to tombstones␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.554001 | 127.0.0.1 | ␣
↪         2408 | 127.0.0.1
                                    Merged data from memtables and 2 sstables␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.555000 | 127.0.0.1 | ␣
↪         2654 | 127.0.0.1
                                Executing single-partition query on doc␣
↪[elasticsearch[127.0.0.2][search][T#10]] | 2017-11-11 13:04:44.555000 | 127.0.0.2 |␣
↪          116 | 127.0.0.1
                                  Read 1 live and 0 tombstone cells␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.555000 | 127.0.0.1 | ␣
↪         2733 | 127.0.0.1
                                    Acquiring sstable references␣
↪[elasticsearch[127.0.0.2][search][T#10]] | 2017-11-11 13:04:44.555000 | 127.0.0.2 |␣
↪          303 | 127.0.0.1
                                Executing single-partition query on doc␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.555000 | 127.0.0.1 | ␣
↪         2950 | 127.0.0.1
```

```
                                                    Acquiring sstable references␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.555000 | 127.0.0.1 | ␣
↪          3002 | 127.0.0.1
                                                     Key cache hit for sstable 5␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.555000 | 127.0.0.1 | ␣
↪          3095 | 127.0.0.1
                                                     Key cache hit for sstable 6␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.555000 | 127.0.0.1 | ␣
↪          3191 | 127.0.0.1
  Skipped 0/2 non-slice-intersecting sstables, included 0 due to tombstones␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.555001 | 127.0.0.1 | ␣
↪          3253 | 127.0.0.1
                                          Merged data from memtables and 2 sstables␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.556000 | 127.0.0.1 | ␣
↪          3549 | 127.0.0.1
                                                    Key cache hit for sstable 5␣
↪[elasticsearch[127.0.0.2][search][T#10]] | 2017-11-11 13:04:44.556000 | 127.0.0.2 |␣
↪           480 | 127.0.0.1
                                                Read 1 live and 0 tombstone cells␣
↪[elasticsearch[127.0.0.1][search][T#2]] | 2017-11-11 13:04:44.556000 | 127.0.0.1 | ␣
↪          3656 | 127.0.0.1
                                                    Key cache hit for sstable 6␣
↪[elasticsearch[127.0.0.2][search][T#10]] | 2017-11-11 13:04:44.556000 | 127.0.0.2 |␣
↪           650 | 127.0.0.1
 Skipped 0/2 non-slice-intersecting sstables, included 0 due to tombstones␣
↪[elasticsearch[127.0.0.2][search][T#10]] | 2017-11-11 13:04:44.556000 | 127.0.0.2 |␣
↪           747 | 127.0.0.1
                                          Merged data from memtables and 2 sstables␣
↪[elasticsearch[127.0.0.2][search][T#10]] | 2017-11-11 13:04:44.556000 | 127.0.0.2 |␣
↪          1245 | 127.0.0.1
                                                Read 1 live and 0 tombstone cells␣
↪[elasticsearch[127.0.0.2][search][T#10]] | 2017-11-11 13:04:44.556000 | 127.0.0.2 |␣
↪          1362 | 127.0.0.1

↪             Request complete | 2017-11-11 13:04:44.563745 | 127.0.0.1 |          ␣
↪19745 | 127.0.0.1
```

CHAPTER 8

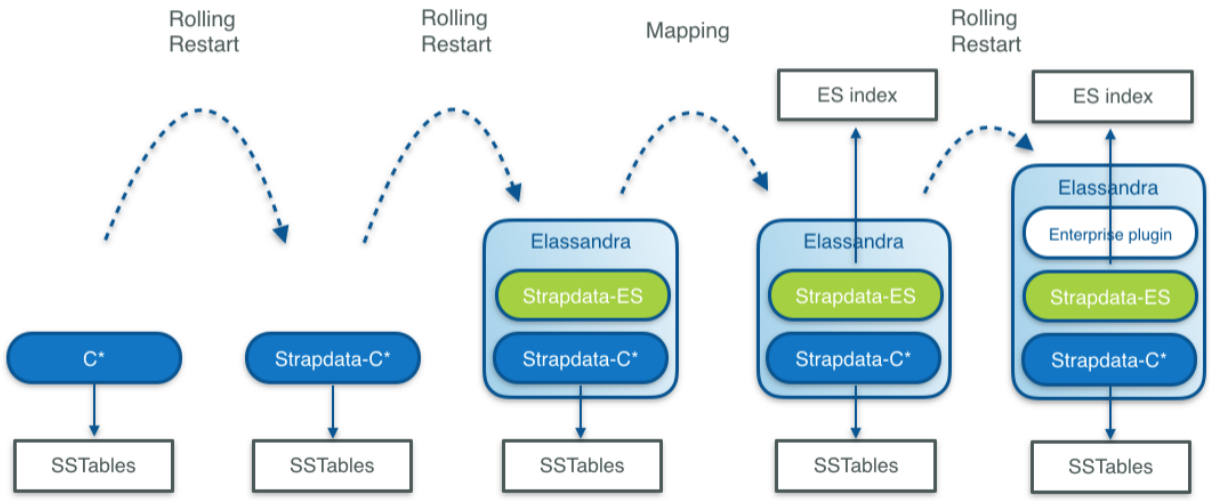
# Enterprise

Elassandra Enterprise is an Elasticsearch plugin installed on top of Elassandra community edition. Elassandra Enterprise plugin provides advanced features:

- Elasticsearch JMX management and monitoring.
- SSL encryption for Elasticsearch connections.
- Authentication, Authorization and Accounting for Elasticsearch.
- Elasticsearch Content-Based security (document-level security and field-level security).

If you are currently running a Cassandra cluster, you can progressively switch to Elassandra Enterprise by upgrading and activating Elasticsearch features:

- First switch to strapdata-cassandra, a fork of cassandra modified to support Elasticsearch.
- Second, restart nodes with Elasticsearch enabled (change the java main class).
- Third, create Elasticsearch indicies with a per table Elasticsearch mapping.
- And finally, deploy the Elassandra Enterprise plugin on your nodes to enable enterprise grade features.




## 8.1 Install

The Elassandra Enterprise plugin must be installed on all Elassandra nodes:

- Unzip the strapdata-enterprise-<version>.zip in a temporary directory.
- If you have installed the elassandra tarball, set CASSANDRA_HOME to your install directory.
- Run the install.sh script.
- Enable some enterprise features in your conf/elasticsearch.yaml (For example add *jmx.enabled: true*)
- Start (or restart) your node.

To check that your *Strapdata Enterprise plugin* is active:

```
GET _nodes/plugins?pretty
...
"plugins" : [
     {
       "name" : "org.apache.cassandra.service.ElassandraDaemon$ElassandraPlugin",
       "version" : "NA",
       "description" : "classpath plugin",
       "classname" : "org.apache.cassandra.service.ElassandraDaemon$ElassandraPlugin",
       "extended_plugins" : [ ],
       "has_native_controller" : false,
       "requires_keystore" : false
     },
     {
       "name" : "strapdata-plugin",
       "version" : "6.2.3.13",
       "description" : "Strapdata Enterprise plugin version 6.2.3.13",
       "classname" : "com.strapdata.elasticsearch.plugin.EnterprisePlugin",
       "extended_plugins" : [ ],
       "has_native_controller" : false,
       "requires_keystore" : false
     }
   ],
 ....
```

**Tip:** If is recommended to always run the same Elassandra Enterprise plugin as your Elassandra version. So after an Elassandra upgrade, reinstall the Elassandra Enterprise plugin in the same version.

If you run in a container, the strapdata/elassandra-enterprise docker image has the Enterprise plugin installed.

## 8.2 License management

Enterprise plugin requires a valid license. When you start your first node with the Enterprise plugin enabled, a 30 days license is generated with all features enabled. If you need more time to evaluate the product, you can request another 30 day license free of charge or purchase a souscription including technical support for Elassandra. If your license has expired, the enterprise plugin will operate in a restricted mode until a valid license is installed.

| Feature | Description | Restricted mode |
|---|---|---|
| JMX | JMX monotoring of Elasticsearch indices | Node restart required to see new index metrics, JMX attributes become read-only |
| SSL | SSL encryption of Elasticsearch connections | |
| AAA | User Authentication, Authorization and Audit | Node restart required to reload users'privileges, no more audit trails. |
| CBS | Content-Based Security rules | Node restart required to reload users'privileges. |

**Caution:** If the number of nodes of licensed datacenters becomes greater than your license maximum number of nodes, then the license becomes invalid on all your nodes.

### 8.2.1 License installation

Licenses are stored in a Cassandra table `elastic_admin.licenses`. You can also put a **conf/license.json** file, this file is automatically loaded at boot time if `elastic_admin.licenses` is empty.

```
cassandra@cqlsh> select * from elastic_admin.licenses;

 id                                   | clustername | company | datacenters | email
↪         | expire                              | features                          |␣
↪generated                           | issuer    | maxnodes | production | signature
↪                                                                                  |␣
↪start                               | type
-------------------------------------+-------------+---------+-------------+---------
↪--------+-------------------------------------+-----------------------------------+---------
↪-----------------------+-----------+----------+------------+-----------------------
↪-------------------------------------------------------------------------------+----------
↪-----------------------+-------
 bbbef903-bbea-401d-838d-faf696e53547 | TestCluster | thecorp |     ['DC1'] |␣
↪contact@thecorp | 2018-10-01 22:00:00.000000+0000 | ['JMX', 'SSL', 'AAA', 'CBS'] |␣
↪2017-10-02 13:23:09.227000+0000 | Strapdata |         3 |      False |␣
↪0x302c02141404c757c3d0e387a8f6194669d5b0a677fbb82102145b88c2785ffabc26b3aa9df72ba03b65f4a829fe␣
↪| 2017-10-01 22:00:00.000000+0000 | TRIAL
```

### 8.2.2 Checking your license

You can use the REST license API to check the currently active license. If your current configuration requires SSL encryption and user authentication, you must provide a valid login, password and root CA certificate.

```
$ curl --user <username>:<password> --cacert conf/cacert.pem -XGET "https://
↪localhost:9200/_license?pretty"
{
  "id" : "bbbef903-bbea-401d-838d-faf696e53547",
  "issuer" : "Strapdata",
  "company" : "thecorp",
  "email" : "contact@thecorp",
  "generated" : "2017-10-02T13:23:09.227Z",
  "start" : "2017-10-01T22:00:00.000Z",
  "expire" : "2018-10-01T22:00:00.000Z",
  "production" : false,
  "max_nodes" : 3,
  "cluster_name" : "TestCluster",
  "datacenters" : [
    "DC1"
  ],
  "type" : "TRIAL",
  "features" : [
    "JMX",
    "SSL",
    "AAA",
    "CBS"
  ],
  "status" : "valid",
  "signature" :
↪"0x302c02141404c757c3d0e387a8f6194669d5b0a677fbb82102145b88c2785ffabc26b3aa9df72ba03b65f4a829fe
↪"
}
```

### 8.2.3 Upgrading your license

You can update your licence by inserting additional license row in the Cassandra `elastic_admin.licenses` table.

```
cassandra@cqlsh> INSERT INTO elastic_admin.licenses JSON '{"id":"bb0a181c-dbc6-4255-
↪8d69-67b6e1d276ce","issuer":"Strapdata","company":"thecorp","email":"contact@thecorp
↪","type":"TRIAL","features":["JMX","SSL","AAA"],"production":false,"generated":
↪"2017-09-26 09:10:15.604Z","start":"2017-09-25 22:00:00.000Z","expire":"2018-09-25␣
↪22:00:00.000Z","clustername":"TestCluster","datacenters":["DC1"],"maxnodes":1,
↪"signature":
↪"0x302d02140b49e8c00b3606c66fe22378acb1ab781410460d02150092b666041dd97887b7d624fd6a12bbd434a955ed
↪"}';
```

Then reload the license with a POST REST request as shown below, each node returns to its active license If you have several licenses in `elastic_admin.licenses`, the **most recenlty generated valid** license will be used.

```
$ curl --user <username>:<password> --cacert <path/to/cacert.pem> -XPOST "https://
↪localhost:9200/_license?pretty"
{
  "_nodes" : {
    "total" : 2,
```

```
    "successful" : 2,
    "failed" : 0
  },
  "cluster_name" : "TestCluster",
  "nodes" : {
    "d607917d-8c68-4cc5-8dc2-2aa21f5ea986" : {
      "name" : "127.0.0.2",
      "license_id" : "bbbef903-bbea-401d-838d-faf696e53547"
    },
    "a1c5307c-5f5a-4676-a6f0-50f221dd655b" : {
      "name" : "127.0.0.1",
      "license_id" : "bbbef903-bbea-401d-838d-faf696e53547"
    }
  }
}
```

**Tip:** If you have several Elasticsearch clusters in your Cassandra cluster, reload the license for each datacenter where Elasticsearch has been enabled.

## 8.3 Index Join on Partition Key

Elassandra Enterprise supports query time join accross Elasticsearch indices under these two conditions:

- Elasticsearch indexes must have the same Cassandra keyspace and same partition key (same columns in the same order with same key validators).
- When partition key is composite, `doc_values` must be enabled on the `_routing` metafield.

### 8.3.1 Join query syntax

The join query requires an inner *FROM* index and a query. The following query returns the documents from *index1* having the partition key of documents in *index2* returned by the inner query. Of course, this is meaningfull when *index1* has no clustering key.

```
GET /index1/_search
{
   "query": {
      "join" : {
         "index" : "index2",
         "score_mode" : "avg",
         "query" : {
           ...
         }
      }
   }
}
```

The join query allows recusive join accross many indices sharing the same partition key by combining the *join* and *bool* queries:

```
GET /index1/_search
{
```

```
    "query":{
      "join":{
        "index":"index2",
        "query":{
          "bool":{
            "must": [
               { ... },
               { "join":{
                  "index":"index3",
                  "query":{ ... }
                }
              }
            ]
          }
        }
      }
    }
}
```

**Note:** A join query is not a relational join where the fields from the inner join queries are returned in the results. It's more like an SQL query SELECT ... FROM ... WHERE ... IN (SELECT ... FROM ...).

### 8.3.2 Join query example

In this example, we create three tables with the following CQL orders:

- The books table, where the UUID of a book is the partition and primary key.
- The citations and edition tables with compound primary key.

Those three tables share the same single partition key, the book UUID, in the same keyspace **example**. Because data distribution is driven by the same partition key, joining many Elasticsearch shards on each node is consistent.

```
CREATE KEYSPACE IF NOT EXISTS example WITH REPLICATION = { 'class' :
↪'NetworkTopologyStrategy', 'DC1' : 1 };

CREATE TABLE IF NOT EXISTS example.books (books uuid PRIMARY KEY, title text, author␣
↪text);
INSERT INTO example.books (books, title, author) VALUES (278078aa-095f-4aec-a048-
↪a138f5071431, 'A Brief History of Time', 'Stephen Hawking');
INSERT INTO example.books (books, title, author) VALUES (7c592b75-475c-420d-980b-
↪f035e1252787, 'The Universe in a Nutshell', 'Stephen Hawking');
INSERT INTO example.books (books, title, author) VALUES (f1662d47-afe7-4273-8544-
↪d7663dcb7498, 'Relativity', 'Albert Einstein');
INSERT INTO example.books (books, title, author) VALUES (72cc85db-4ec1-455a-b893-
↪e884607b3f9f, 'The World as I See It', 'Albert Einstein');

CREATE TABLE IF NOT EXISTS example.citations (books uuid, id uuid, words text,␣
↪PRIMARY KEY (books, id));
INSERT INTO example.citations (books, id, words) VALUES (278078aa-095f-4aec-a048-
↪a138f5071431, 0c46578e-4dcf-46d3-9136-8a2da187b8eb, 'Quiet people have the loudest␣
↪minds.');
INSERT INTO example.citations (books, id, words) VALUES (278078aa-095f-4aec-a048-
↪a138f5071431, f14942d8-281f-4835-813d-09254a0d70d8, 'Intelligence is the ability to␣
↪adapt to change.');
```

```
INSERT INTO example.citations (books, id, words) VALUES (278078aa-095f-4aec-a048-
↪a138f5071431, f9e3b0ba-2c52-484e-911e-d2e633baf41c, 'I don''t think the human race␣
↪will survive the next thousand years, unless we spread into space.');
INSERT INTO example.citations (books, id, words) VALUES (f1662d47-afe7-4273-8544-
↪d7663dcb7498, 72cc85db-4ec1-455a-b893-e884607b3f9f, 'Great spirits have always␣
↪encountered violent opposition from mediocre minds.');
INSERT INTO example.citations (books, id, words) VALUES (f1662d47-afe7-4273-8544-
↪d7663dcb7498, 70986be4-e586-4560-9405-12c290e9e0ab, 'If you can''t explain it to a␣
↪six year old, you don''t understand it yourself.');

CREATE TABLE IF NOT EXISTS example.editions (books uuid, editor text, isbn text,␣
↪available boolean, PRIMARY KEY ((books), editor));
INSERT INTO example.editions (books, editor, isbn, available) VALUES (278078aa-095f-
↪4aec-a048-a138f5071431, 'Bantam Press', '0857501003', true);
INSERT INTO example.editions (books, editor, isbn, available) VALUES (7c592b75-475c-
↪420d-980b-f035e1252787, 'Bantam Press', '9780593048153', false);
INSERT INTO example.editions (books, editor, isbn, available) VALUES (f1662d47-afe7-
↪4273-8544-d7663dcb7498, 'Penguin Classics', '0143039822', false);
INSERT INTO example.editions (books, editor, isbn, available) VALUES (f1662d47-afe7-
↪4273-8544-d7663dcb7498, 'GENERAL PRESS', 'B00R86QABW', false);
INSERT INTO example.editions (books, editor, isbn, available) VALUES (72cc85db-4ec1-
↪455a-b893-e884607b3f9f, 'Filiquarian Publishing', '1599869659', false);
```

We create three Elasticsearch indices backed by theses 3 tables in the **example** keyspace:

```
PUT /books
{
  "settings":{ "index.keyspace":"example" },
  "mappings":{
    "books":{
      "properties":{
        "title":{"type":"text", "cql_collection":"singleton"},
        "author":{"type":"text", "cql_collection":"singleton"},
        "books":{"type":"keyword", "cql_collection":"singleton","cql_partition_key":
↪"true", "cql_primary_key_order":"0"}
      }
    }
  }
}
PUT /citations
{
  "settings":{ "index.keyspace":"example" },
  "mappings":{
    "citations":{
      "properties":{
        "books":{"type":"keyword", "cql_collection":"singleton","cql_partition_key":
↪"true", "cql_primary_key_order":"0"},
        "id":{"type":"keyword", "cql_collection":"singleton","cql_partition_key":
↪"false", "cql_primary_key_order":"1"},
        "words":{"type":"text", "cql_collection":"singleton"}
      }
    }
  }
}
PUT /editions
{
  "settings":{ "index.keyspace":"example" },
  "mappings":{
```

```
    "editions":{
      "properties":{
        "books":{"type":"keyword", "cql_collection":"singleton","cql_partition_key":
↪"true", "cql_primary_key_order":"0"},
        "editor":{"type":"text", "cql_collection":"singleton","cql_partition_key":
↪"false", "cql_primary_key_order":"1"},
        "isbn":{"type":"text", "cql_collection":"singleton"},
        "available":{"type":"boolean", "cql_collection":"singleton"}
      }
    }
  }
}
```

We can search for books that have a citation containing the word *minds*:

```
GET /books/_search?pretty
{
  "query":{
    "join":{
      "index":"citations",
      "query":{
        "match":{ "words":"minds" }
      }
    }
  }
}
{
  "took" : 8,
  "timed_out" : false,
  "_shards" : {
    "total" : 3,
    "successful" : 3,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : 2,
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "books",
        "_type" : "books",
        "_id" : "f1662d47-afe7-4273-8544-d7663dcb7498",
        "_score" : 1.0,
        "_source" : {
          "books" : "f1662d47-afe7-4273-8544-d7663dcb7498",
          "author" : "Albert Einstein",
          "title" : "Relativity"
        }
      },
      {
        "_index" : "books",
        "_type" : "books",
        "_id" : "278078aa-095f-4aec-a048-a138f5071431",
        "_score" : 1.0,
        "_source" : {
          "books" : "278078aa-095f-4aec-a048-a138f5071431",
          "author" : "Stephen Hawking",
```

```
          "title" : "A Brief History of Time"
        }
      }
    ]
  }
}
```

Through a recusive join query, we can search for books that have a citation containing the word *mind* and *available* from table *editions*:

```
GET /books/_search?pretty
{
    "query":{
      "join":{
        "index":"citations",
        "query":{
          "bool":{
            "must": [
               { "match":{ "words":"minds"}},
               { "join":{
                  "index":"editions",
                  "query":{ "term": { "available":"true" }}
                 }
               }
            ]
          }
        }
      }
    }
}
{
  "took" : 7,
  "timed_out" : false,
  "_shards" : {
    "total" : 3,
    "successful" : 3,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : 1,
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "books",
        "_type" : "books",
        "_id" : "278078aa-095f-4aec-a048-a138f5071431",
        "_score" : 1.0,
        "_source" : {
          "books" : "278078aa-095f-4aec-a048-a138f5071431",
          "author" : "Stephen Hawking",
          "title" : "A Brief History of Time"
        }
      }
    ]
  }
}
```

## 8.4 JMX Managment & Monitoring

The JMX technology provides a standard solution for managing and monitoring java applications. With the JMX feature, you can manage and monitor both Cassandra and Elasticsearch.

### 8.4.1 JMX Monitoring

The JMX feature of Elassandra Enterprise exposes Elasticsearch stats over JMX, allowing monitoring the Elasticsearch cluster, index shards, threadpool and networks activities. You can browse these metrics with various JMX clients lsuch as VisualVM or jmxterm.

JMXTerm example :

```
java -jar jmxterm-1.0.0-uber.jar -l localhost:7199
$>domain org.elasticsearch.index
#domain is set to org.elasticsearch.index
$>bean org.elasticsearch.index:name=sales_2017,scope=search,type=IndexShard
#bean is set to org.elasticsearch.index:name=sales_2017,scope=search,type=IndexShard
$>get *
#mbean = org.elasticsearch.index:name=sales_2017,scope=search,type=IndexShard:
QueryTotal = 21;
FetchTotal = 0;
ScrollTotal = 0;
QueryTimeInMillis = 56038;
QueryCurrent = 0;
FetchTimeInMillis = 0;
FetchCurrent = 0;
ScrollTimeInMillis = 0;
ScrollCurrent = 0;
SuggestCount = 0;
SuggestTimeInMillis = 0;
SuggestCurrent = 0;
$>
```

These metrcis can be pulled, or pushed to various tools (graphite, ganglia or influxdb) using the popular Metrics Library embedded in Apache Cassandra.

### 8.4.2 Monitoring Elassandra with InfluxDB

Below is a sample configuration located in **conf/influxdb-reporting.yaml** sending JMX metrics to an InfluxDB database named *elassandra*.

```
influxdb:
-
  dbName: 'elassandra'
  protocol: 'http'
  tags:
    environment: 'test'
    cluster: 'test_cluster'
    host: 'vm1'
  hosts:
    - host: 'vm1'
      port: 8086
  timeunit: 'SECONDS'
  period: 60
```

```
prefix: ''
groupGauges: true
```

To enable this configuration, add **JVM_OPTS="$JVM_OPTS -Dcassandra.metricsReporterConfigFile=influxdb-reporting.yaml"** in your **conf/cassandra-env.sh**

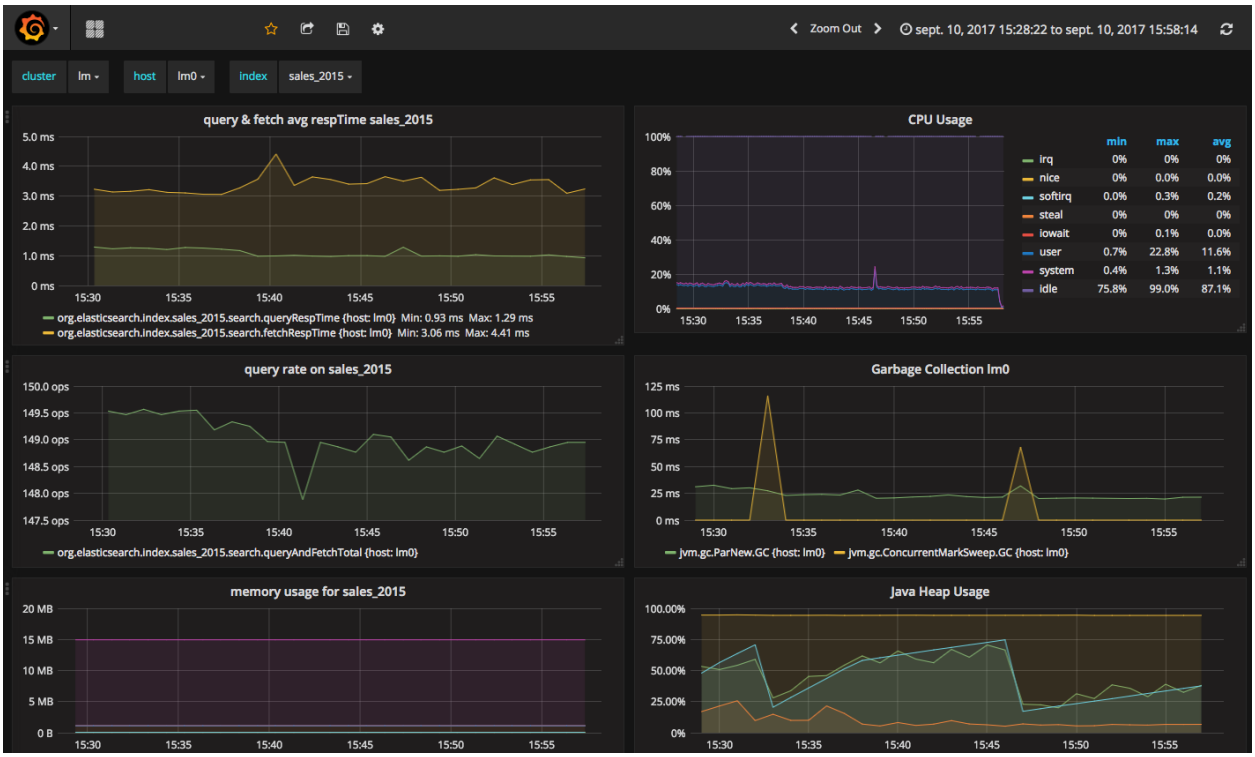
---

**Note:** When installing the Elassandra Enterprise plugin, the following jar files are added to the cassandra classpath :

- reporter-config-base-3.0.4.jar
- reporter-config3-3.0.4.jar
- metrics-influxdb-1.1.10-SNAPSHOT.jar
- dropwizard-metrics-influxdb-1.1.10-SNAPSHOT.jar

---

Then configure Grafana with an influxDB datasource and build your Elassandra dashboard.




## 8.4.3 Monitoring Elassandra with Prometheus

Prometheus can scrape both Elasticsearch and Cassandra JMX metrics through the standrard Prometheus JMX Exporter running as a java agent. To expose these metrics on TCP port 7500, add the following in your environnment or in the conf/cassandra-env.sh:

```
JVM_OPTS="$JVM_OPTS -javaagent:agents/jmx_prometheus_javaagent-0.3.1.jar=7500:conf/
↪jmx_prometheus_exporter.yml"
```

Here is the default JMX exporter configuration file **conf/jmx_prometheus_exporter.yml** for Elassandra.

```
lowercaseOutputName: true
lowercaseOutputLabelNames: true
whitelistObjectNames: [
"org.apache.cassandra.metrics:type=ColumnFamily,name=RangeLatency,*",
"org.apache.cassandra.metrics:type=ColumnFamily,name=LiveSSTableCount,*",
"org.apache.cassandra.metrics:type=ColumnFamily,name=SSTablesPerReadHistogram,*",
"org.apache.cassandra.metrics:type=ColumnFamily,name=SpeculativeRetries,*",
"org.apache.cassandra.metrics:type=ColumnFamily,name=MemtableOnHeapSize,*",
"org.apache.cassandra.metrics:type=ColumnFamily,name=MemtableSwitchCount,*",
"org.apache.cassandra.metrics:type=ColumnFamily,name=MemtableLiveDataSize,*",
"org.apache.cassandra.metrics:type=ColumnFamily,name=MemtableColumnsCount,*",
"org.apache.cassandra.metrics:type=ColumnFamily,name=MemtableOffHeapSize,*",
"org.apache.cassandra.metrics:type=ColumnFamily,name=BloomFilterFalsePositives,*",
"org.apache.cassandra.metrics:type=ColumnFamily,name=BloomFilterFalseRatio,*",
"org.apache.cassandra.metrics:type=ColumnFamily,name=BloomFilterDiskSpaceUsed,*",
"org.apache.cassandra.metrics:type=ColumnFamily,name=BloomFilterOffHeapMemoryUsed,*",
"org.apache.cassandra.metrics:type=ColumnFamily,name=SnapshotsSize,*",
"org.apache.cassandra.metrics:type=ColumnFamily,name=TotalDiskSpaceUsed,*",
"org.apache.cassandra.metrics:type=CQL,name=RegularStatementsExecuted,*",
"org.apache.cassandra.metrics:type=CQL,name=PreparedStatementsExecuted,*",
"org.apache.cassandra.metrics:type=Compaction,name=PendingTasks,*",
"org.apache.cassandra.metrics:type=Compaction,name=CompletedTasks,*",
"org.apache.cassandra.metrics:type=Compaction,name=BytesCompacted,*",
"org.apache.cassandra.metrics:type=Compaction,name=TotalCompactionsCompleted,*",
"org.apache.cassandra.metrics:type=ClientRequest,name=Latency,*",
"org.apache.cassandra.metrics:type=ClientRequest,name=Unavailables,*",
"org.apache.cassandra.metrics:type=ClientRequest,name=Timeouts,*",
"org.apache.cassandra.metrics:type=Storage,name=Exceptions,*",
"org.apache.cassandra.metrics:type=Storage,name=TotalHints,*",
"org.apache.cassandra.metrics:type=Storage,name=TotalHintsInProgress,*",
"org.apache.cassandra.metrics:type=Storage,name=Load,*",
"org.apache.cassandra.metrics:type=Connection,name=TotalTimeouts,*",
"org.apache.cassandra.metrics:type=ThreadPools,name=CompletedTasks,*",
"org.apache.cassandra.metrics:type=ThreadPools,name=PendingTasks,*",
"org.apache.cassandra.metrics:type=ThreadPools,name=ActiveTasks,*",
"org.apache.cassandra.metrics:type=ThreadPools,name=TotalBlockedTasks,*",
"org.apache.cassandra.metrics:type=ThreadPools,name=CurrentlyBlockedTasks,*",
"org.apache.cassandra.metrics:type=DroppedMessage,name=Dropped,*",
"org.apache.cassandra.metrics:type=Cache,scope=KeyCache,name=HitRate,*",
"org.apache.cassandra.metrics:type=Cache,scope=KeyCache,name=Hits,*",
"org.apache.cassandra.metrics:type=Cache,scope=KeyCache,name=Requests,*",
"org.apache.cassandra.metrics:type=Cache,scope=KeyCache,name=Entries,*",
"org.apache.cassandra.metrics:type=Cache,scope=KeyCache,name=Size,*",
"org.apache.cassandra.metrics:type=Streaming,name=TotalIncomingBytes,*",
"org.apache.cassandra.metrics:type=Streaming,name=TotalOutgoingBytes,*",
"org.apache.cassandra.metrics:type=Client,name=connectedNativeClients,*",
"org.apache.cassandra.metrics:type=Client,name=connectedThriftClients,*",
"org.apache.cassandra.metrics:type=Table,name=WriteLatency,*",
"org.apache.cassandra.metrics:type=Table,name=ReadLatency,*",
"org.apache.cassandra.net:type=FailureDetector,*",
"org.elasticsearch.cluster:*",
"org.elasticsearch.node:*",
"org.elasticsearch.index:*"
]
#blacklistObjectNames: ["org.apache.cassandra.metrics:type=ColumnFamily,*"]
rules:
  - pattern: org.apache.cassandra.metrics<type=(Connection|Streaming), scope=(\S*),␣
↪name=(\S*)><>(Count|Value)
```

```
    name: cassandra_$1_$3
    labels:
      address: "$2"
  - pattern: org.apache.cassandra.metrics<type=(ColumnFamily), name=(RangeLatency)><>
↪(Mean)
    name: cassandra_$1_$2_$3
  - pattern: org.apache.cassandra.net<type=(FailureDetector)><>(DownEndpointCount)
    name: cassandra_$1_$2
  - pattern: org.apache.cassandra.metrics<type=(Keyspace), keyspace=(\S*), name=(\S*)>
↪<>(Count|Mean|95thPercentile)
    name: cassandra_$1_$3_$4
    labels:
      "$1": "$2"
  - pattern: org.apache.cassandra.metrics<type=(Table), keyspace=(\S*), scope=(\S*),␣
↪name=(\S*)><>(Count|Mean|95thPercentile)
    name: cassandra_$1_$4_$5
    labels:
      "keyspace": "$2"
      "table": "$3"
  - pattern: org.apache.cassandra.metrics<type=(ClientRequest), scope=(\S*),␣
↪name=(\S*)><>(Count|Mean|95thPercentile)
    name: cassandra_$1_$3_$4
    labels:
      "type": "$2"
  - pattern: org.apache.cassandra.metrics<type=(\S*)(?:, ((?!scope)\S*)=(\S*))?(?:,␣
↪scope=(\S*))?, name=(\S*)><>(Count|Value)
    name: cassandra_$1_$5
    labels:
      "$1": "$4"
      "$2": "$3"
  - pattern: org.elasticsearch.cluster<name=([a-zA-Z_ 0-9]+)><>
↪(MetadataVersion|ClusterStateVersion|NumberOfPendingTasks|MaxTaskWaitTimeMillis|AliveNodeCount|Dea
    type: GAUGE
    name: elasticsearch_cluster_$2
  - pattern: org.elasticsearch.node<type=(transport)><>(\w*)
    name: elasticsearch_node_$1_$2
  - pattern: org.elasticsearch.node<type=(threadPool), name=(\S*)><>(\w*)
    name: elasticsearch_node_$1_$3
    type: GAUGE
    labels:
      "name": $2
  - pattern: org.elasticsearch.node<type=(httpServer)><>(\w*)
    type: COUNTER
    name: elasticsearch_node_$1_$2
    type: GAUGE
  - pattern: org.elasticsearch.index<type=(Index), name=(\S*)><>(IndexStatusCode)
    type: GAUGE
    name: elasticsearch_$1_$3
    labels:
      "name": $2
  - pattern: org.elasticsearch.index<type=(IndexShard), scope=(\S*)><>(\w*InBytes)
    type: GAUGE
    name: elasticsearch_$1_$3
    labels:
      "scope": $2
  - pattern: org.elasticsearch.index<type=(IndexShard), scope=(\S*)><>(\w*)
    type: COUNTER
    name: elasticsearch_$1_$3
```

```
    labels:
      "scope": $2
  - pattern: org.elasticsearch.index<type=(IndexShard), name=(\S*), scope=(\S*)><>
↪(\w*InBytes)
    type: GAUGE
    name: elasticsearch_$1_$4
    labels:
      "index": $2
      "scope": $3
  - pattern: org.elasticsearch.index<type=(IndexShard), name=(\S*), scope=(\S*)><>
↪(\w*)
    type: COUNTER
    name: elasticsearch_$1_$4
    labels:
      "index": $2
      "scope": $3
```

Then configure Grafana with a Prometheus datasource and build your Elassandra dashboard.


### 8.4.4 Monitoring Elassandra through the Prometheus Operator

When running Elassandra Enterprise under Kubernetes, you can use the Prometheus-Operator to monitor your Elassandra PODs.

Add the following annotations to automatically scrap Elassandra pods:

```
annotations.prometheus.io/scrape=true
annotations.prometheus.io/port=7500
```

Add the following kubernetes labels to your Elassandra PODs:

```
release: "my-release"
cluster: "my-cluster"
datacenter: "DC1"
```

If you deploy Elassandra through the Elassandra HELM chart, the **release** label is automatically added to your Elassandra PODs.

In your Prometheus Operator (in HELM values.yaml, prometheusSpec.additionalScrapeConfigs), add the following scrap config to properly map Kubernetes pod's labels to Grafana dashboard variables:

```
prometheusSpec:
  additionalScrapeConfigs:
    - job_name: 'kubernetes-pods'
      kubernetes_sd_configs:
      - role: pod
      relabel_configs:
      - source_labels: [__meta_kubernetes_pod_annotation_prometheus_io_scrape]
        action: keep
        regex: true
      - source_labels: [__meta_kubernetes_pod_annotation_prometheus_io_path]
        action: replace
        target_label: __metrics_path__
        regex: (.+)
      - source_labels: [__address__, __meta_kubernetes_pod_annotation_prometheus_io_
↪port]
        action: replace
        regex: ([^:]+)(?::\d+)?;(\d+)
        replacement: $1:$2
        target_label: __address__
      - action: labelmap
        regex: __meta_kubernetes_pod_label_(.+)
      - source_labels: [__meta_kubernetes_namespace]
        action: replace
        target_label: kubernetes_namespace
      - source_labels: [__meta_kubernetes_pod_name]
        action: replace
        target_label: kubernetes_pod_name
      - source_labels: [__meta_kubernetes_pod_name]
        action: replace
        target_label: instance
```

As the result, check that your Elassandra PODs have the expected tags in your Prometheus targets (release, cluster, datacenter and instance).

Finally, upload the elassandra-kubernetes-dashborad.json through the Grafana import wizard.

### 8.4.5 Enable/Disable search on a node

The JMX feature allows excluding/including a node from distributed search while still receiving CQL write, repairing or rebuilding its elasticsearch indices, by setting the following attributes on the JMX Bean `org.elasticsearch.node:type=node`

| JMX Attribute | Default value | Description |
|---|---|---|
| `SearchEnabled` | **true** | Set wether or not the node is invloved in distributed search queries from other nodes. When `SearchEnabled` is false on a node, all its shards are seen UNASSIGNED from other nodes. |
| `AutoEnableSearch` | **true** | If true, the node automatically set `SearchEnabled` to true when it becomes available, participating to distributed search queries. In order to restart a node in a maintenance mode for search requests, you can set `AutoEnableSearch` to **false** with the system property `es.auto_enable_search`. |

To set `SearchEnabled` on command line, just use **jmxterm** as in the following exemple.

```
echo "set -b org.elasticsearch.node:type=node SearchEnabled false" | java -jar␣
↪jmxterm-1.0.0-uber.jar -l localhost:7199
```

## 8.5 SSL Network Encryption

The SSL Feature provides trafic encryption for both HTTP and Elasticsearch transport connections.

---

**Note:** Elasticsearch transport protocol is the native binary protocol used for Elasticsearch node-to-node communication. You can also use the transport protocol from a client application written in java, as described in the elasticsearch documentation.

---

SSL configuration is defined in your **conf/cassandra.yaml** for both Cassandra and Elasticsearch :

- Server options define node-to-node encryption for both Cassandra and Elasticsearch. Obviously, Elasticsearch transport connections are encrypted when *internode_encryption* is set to **all** or **rack** (there is no elasticsearch cross-datacenter traffic).
- Client options define client-to-node encryption to request both Cassandra and Elasticsearch. If *optional* is **true**, Elasticsearch will accept the clear connections for HTTP and transport request.

To ensure support for all encryption algorithms, it is highly recommended to install the JCE Unlimited Strength Jurisdiction policy files on all nodes.

Here an illustrattion of a SSL configuration in your **conf/cassandra.yaml** file :

```
# Enable or disable inter-node encryption
# Default settings are TLS v1, RSA 1024-bit keys (it is imperative that
# users generate their own keys) TLS_RSA_WITH_AES_128_CBC_SHA as the cipher
# suite for authentication, key exchange and encryption of the actual data transfers.
# Use the DHE/ECDHE ciphers if running in FIPS 140 compliant mode.
# NOTE: No custom encryption options are enabled at the moment
# The available internode options are : all, none, dc, rack
#
# If set to dc cassandra will encrypt the traffic between the DCs
# If set to rack cassandra will encrypt the traffic between the racks
#
# The passwords used in these options must match the passwords used when generating
# the keystore and truststore.  For instructions on generating these files, see:
# http://download.oracle.com/javase/6/docs/technotes/guides/security/jsse/
↪JSSERefGuide.html#CreateKeystore
#
server_encryption_options:
    internode_encryption: all
    keystore: /etc/cassandra/.keystore.jks
    keystore_password: changeit
    truststore: /etc/cassandra/.truststore.jks
    truststore_password: changeit
    # More advanced defaults below:
    protocol: TLSv1.2
    # algorithm: SunX509
    # store_type: JKS
    # cipher_suites: [TLS_RSA_WITH_AES_128_CBC_SHA,TLS_RSA_WITH_AES_256_CBC_SHA,TLS_
↪DHE_RSA_WITH_AES_128_CBC_SHA,TLS_DHE_RSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_RSA_WITH_
↪AES_128_CBC_SHA,TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA]
```

```
    # require_client_auth: true

# enable or disable client/server encryption.
client_encryption_options:
    enabled: true
    # If enabled and optional is set to true encrypted and unencrypted connections␣
↪are handled.
    optional: true
    keystore: /etc/cassandra/.keystore.jks
    keystore_password: changeit
    require_client_auth: true
    # Set trustore and truststore_password if require_client_auth is true
    truststore: /etc/cassandra/.truststore.jks
    truststore_password: changeit
    # More advanced defaults below:
    protocol: TLSv1.2
    # algorithm: SunX509
    # store_type: JKS
    # cipher_suites: [TLS_RSA_WITH_AES_128_CBC_SHA,TLS_RSA_WITH_AES_256_CBC_SHA,TLS_
↪DHE_RSA_WITH_AES_128_CBC_SHA,TLS_DHE_RSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_RSA_WITH_
↪AES_128_CBC_SHA,TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA]
```

**Caution:** If paths to keystores are relative, you could faced an issue when starting Elassandra from another directory than the installed directory. You should use the absolute keystore paths to avoid such an issue.

Elassandra nodes certificates should have the following X509 v3 extensions to work properly with HTTPS clients:

```
ObjectId: 2.5.29.37 Criticality=false
ExtendedKeyUsages [
  clientAuth
  serverAuth
]

ObjectId: 2.5.29.15 Criticality=true
KeyUsage [
  DigitalSignature
  Key_Encipherment
]
```

Moreover, SSL/TLS hostname verification requires that the requested hostname matches the certificate subject common name, or at least one of the Subject Alternative Names (SANs). In order to use the same certificate for all elassandra nodes, add a wildcard SAN to your certificate and use a matching DNS name to connect to the elassandra nodes. Usually, **localhost** and **127.0.0.1** are also included in the SANs to allow local connections.

### 8.5.1 Elasticsearch SSL configuration

SSL for Elasticsearch is actived according to the following settings in your **conf/elasticsearch.yml** :

| Setting | Default | Description |
|---|---|---|
| `https.enabled` | false | Enable HTTPS on client-to-node Elasticsearch connections |
| `ssl.transport.enabled` | false | Enable SSL on Elastisearch transport connections (node-to-node connections) |

Once HTTPS is enabled, accessing your Elasticsearch cluster requires the HTTPS protocol and a trusted certificate to validate the server side certificate :

```
curl -XGET --cacert conf/cacert.pem "https://localhost:9200/my_index/_search"
```

You can also check your SSL configuration with a `GET /_sslinfo` request.

```
curl -XGET --cacert conf/cacert.pem "https://localhost:9200/_sslinfo"
{
   "https_protocol" : "TLSv1.2",
   "https_cipher" : "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384"
}
```

If client encryption is enabled in your **conf/cassandra.yaml**, and `require_client_auth=true`, a client certificate is required to connect to elasticsearch nodes.

```
curl -XGET --cacert conf/cacert.pem --key client.key.pem --pass <private_key_pass> --
↪cert client.crt.pem "https://localhost:9200/_sslinfo"
{
   "https_protocol" : "TLSv1.2",
   "https_cipher" : "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384"
}
```

When using curl, if NSS database is installed on your system, make sure your private key file is PKCS1 (or RSA). If it's not the case, convert your PKCS8 private key file to a PKCS1 with the following command:

```
openssl rsa -in client.key.pem -out client.rsakey.pem
```

**Tip:** In order to enable SSL/TLS encryption without any downtime, you'll need first to deploy keystores and enable TLS/SSL for cassandra only on all nodes in a rolling restart, then enable SSL for elasticsearch in your **conf/elasticsearch.yml** in a second rolling restart. Finally, enable internode SSL/TLS client authentication in a third rolling restart by configuring `server_encryption_options.require_client_auth: true` (If this is configured before SSL/TLS for Elasticsearch is enabled, Elasticsearch sub queries won't be able to reach some nodes).

### 8.5.2 JMX traffic Encryption

Enable SSL for JMX by setting the following parameters.

```
JVM_OPTS="$JVM_OPTS -Dcom.sun.management.jmxremote.ssl=true"
JVM_OPTS="$JVM_OPTS -Dcom.sun.management.jmxremote.ssl.need.client.auth=true"
JVM_OPTS="$JVM_OPTS -Dcom.sun.management.jmxremote.registry.ssl=true"
#JVM_OPTS="$JVM_OPTS -Dcom.sun.management.jmxremote.ssl.enabled.protocols=<enabled-
↪protocols>"
#JVM_OPTS="$JVM_OPTS -Dcom.sun.management.jmxremote.ssl.enabled.cipher.suites=
↪<enabled-cipher-suites>"

JVM_OPTS="$JVM_OPTS -Djavax.net.ssl.keyStore=<install_dir>/conf/server-keystore.jks"
JVM_OPTS="$JVM_OPTS -Djavax.net.ssl.keyStorePassword=changeit"
JVM_OPTS="$JVM_OPTS -Djavax.net.ssl.trustStore=<install_dir>/cassandra/conf/server-
↪truststore.jks"
JVM_OPTS="$JVM_OPTS -Djavax.net.ssl.trustStorePassword=changeit"
```

Once SSL is enabled on JMX, `nodetool` utility will require the *–ssl* option.

## 8.6 Authentication and Authorization

Elasticsearch authentifcation and autorization are based on the Cassandra internal Authentication and Role-Based Access Control, allowing getting an homogeneous security policy.

### 8.6.1 Authenticated search request through CQL

In order to propagate Cassandra authentication to Elasticsearch when searching through the CQL driver, use the **EnterpriseElasticQueryHandler** by adding the following system property to your cassandra-env.sh and restart your nodes :

```
JVM_OPTS="$JVM_OPTS -Dcassandra.custom_query_handler_class=org.elassandra.index.
↪EnterpriseElasticQueryHandler"
```

### 8.6.2 Cassandra internal authentication

To enable Cassandra authentication, set the following settings in your **conf/cassandra.yaml**, and restart your node :

```
authenticator: PasswordAuthenticator
authorizer: CassandraAuthorizer
```

Once the authentication is enabled, create a new Cassandra superuser to avoid issue with the default "Cassandra" superuser (Authentication with the Cassandra superuser requires QUORUM nodes to be available in your cluster), and change the default Cassandra password.

```
CREATE ROLE admin WITH PASSWORD='******' AND LOGIN=true AND SUPERUSER=true;
ALTER ROLE cassandra WITH PASSWORD='******';
```

Then configure the replication factor for the *system_auth* keyspace according to your cluster configuration (see Configure Native Authentication). Finally, adjust roles and credential cache settings and disable JMX configuration of authentifcation and authorization cache.

### 8.6.3 Cassandra LDAP authentication

The Cassandra LDAPAuthenticator provides external LDAP authentication for both Cassandra and Elasticsearch access.

For performance reasons, the LDAPAuthenticator tries first to authenticate users through the Cassandra PasswordAuthenticator. If local authentication failed, the Cassandra LDAPAuthenticator search for the username in the LDAP directory and tries to bind with the provided password.

To enable Cassandra LDAP user authentication, set the following settings in your **conf/cassandra.yaml** :

```
authorizer: CassandraAuthorizer
authenticator: com.strapdata.cassandra.ldap.LDAPAuthenticator
role_manager: com.strapdata.cassandra.ldap.LDAPRoleManager
```

Update the **$CASSANDRA_CONF/ldap.properties** file according to your LDAP configuration:

```
# For extra settings, see https://docs.oracle.com/javase/7/docs/technotes/guides/jndi/
↪jndi-ldap.html
# Ldap server URI including the base search DN.
# Specify ldaps when using a secure LDAP port (strongly recommended)
```

```
# see https://docs.oracle.com/javase/jndi/tutorial/ldap/misc/url.html
ldap_uri: ldaps://localhost:636/

# Service user distinguished name. This user will be a SUPERUSER and be used for␣
↪looking up
# user details on authentication.
service_dn: cn=admin,dc=example,dc=org
service_password: password

# User search base distinguished name and filter pattern
user_base_dn: dc=example,dc=org
user_filter: (cn={0})

# When storing password in cache, store a hashed copy. Note this will have a␣
↪performance impact as the password will need to be hashed on each authentication.
# If false, password will be stored in memory on the Cassandra server as plain text␣
↪and you should ensure appropriate security controls to mitigate risk of compromise␣
↪of LDAP passwords.
cache_hashed_password: true
```

Add the following system property in your JVM options:

```
JVM_OPTS="$JVM_OPTS -Dldap.properties.file=$CASSANDRA_CONF/ldap.properties"
```

Restart Elassandra nodes.

When LDAP user authentication succeed, the associated Cassandra role is automatically created with the user distinguished name:

```
$ cqlsh -u alice -p *****
[cqlsh 5.0.1 | Cassandra 3.11.3.5 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cassandra@cqlsh> list roles;

 role                          | super | login | options
-------------------------------+-------+-------+---------
                     cassandra |  True |  True |        {}
 cn=admin,dc=example,dc=org    |  True |  True |        {}
 cn=alice,dc=example,dc=org    | False |  True |        {}
```

Cassandra permissions or elasticsearch privileges (in table `elastic_admin.privileges`) can be granted to these LDAP roles, but usually, it's preferable to assign permissions and privileges to a base role, and grant LDAP users to this role. In the following example, the role *logstash* is autorized to manage elasticsearch indicies matching the regex 'logstash-.*' and LDAP user *alice* inherits this role:

```
CREATE ROLE logstash WITH LOGIN = false;
INSERT INTO elastic_admin.privileges (role, actions, indices) VALUES ( 'logstash',
↪'indices:.*','logstash-.*');
GRANT logstash TO 'cn=alice,dc=example,dc=org';
```

**Tip:** By default, the LDAPAuthenticator relies on the JSSE (Java Socket Secure Extension) SSL implementation supporting some customization. You can specify the LDAP trusted root certificated by setting the system property `javax.net.ssl.trustStore`.

You can also specify your own *SSLSocketFactory* through the JNDI property `java.naming.ldap.factory.socketjava.naming.ldap.factory.socket`. Strapdata provides a

**com.strapdata.cassandra.ldap.TrustAllSSLSocketFactory** for tests purposes allowing to accept any root certificates.

For tests, hostname verification can also be disabled by setting the system property `com.sun.jndi.ldap.object.disableEndpointIdentification` to *true*.

### 8.6.4 Elasticsearch Authentication, Authorization and Content-Based Security

Elasticsearch authentication settings are defined in **conf/elasticsearch.yml**. To be effective, these settings must be the same on all the nodes of a Cassandra datacenter.

| Setting | Default | Description |
|---|---|---|
| `aaa.enabled` | **false** | Enable Elasticsearch authentication and authorization. |
| `aaa.rest.prompt` | **true** | By default, a rejected HTTP request returns with a 403 code, meaning access is forbidden. When prompt is configured, rejected anonymous HTTP requests return a code 401 this prompt in the authorization header. |
| `aaa.rest.realm` | **${cluster_name} authentication required** | Prompted realm when HTTP authentifcation is required. |
| `aaa.user_header` | | When user is already authenticated by an HTTP proxy, you can define the HTTP header name used to carry the cassandra user's name used to execute an elasticsearch request. To avoid security breach, you should properly restrict unauthenticated access to Elassandra when using such mechanism. |
| `aaa.anonymous_user` | | Defines the cassandra user's name used to execute unauthenticated request. If undefined, unauthenticated requests are rejected. |
| `aaa.shared_secret` | **Base64 encoded cluster name** | Shared secret used to tag authorized requests on the coordinator node. This should be a confidential per datacenter secret. |
| `cbs.enabled` | **false** | Enable or disable Content-Based Security. |

**Tip:** Elasticsearch **user authentication requires HTTPS**. (User authentication without HTTPS is not supported).

In order to grant an Elasticsearch request, Elassandra will check two levels of access rights :

1. First, Elassandra will look for a **privilege** matching your Elasticsearch request in the Cassandra table `elastic_admin.privileges`.
2. If no privilege matches and request is related to indices, Elassandra will look for a Cassandra **permission** associated with the user's roles.

### 8.6.5 Privileges

Privileges are defined in the Cassandra table `elastic_admin.privileges`.

```
CREATE TABLE elastic_admin.privileges (
   role text,
   actions text,
```

```
   indices text,
   fields set<text>,
   query text,
   PRIMARY KEY (role, actions, indices)
);
```

- `role`: The user's role.
- `actions`: Regular expression defining the authorized actions.
- `indices`: Regular expression defining the authorized target indices. If null, all indices backed by keyspaces associated to the role.
- `fields`: List of visible fields of documents when the Content-Base Security is enabled. The support wilcards, for example foo* will match all fields starting by *foo*. If your request matches multiple privileges, returned document will contain all associated fields.
- `query`: Filter query when Content-Base Security is enabled. If your request matches multiple privileges, returned document are filtered with all queries.

**Important:**

- Cassandra roles with *superuser* = **true** have full access to Elasticsearch.
- All cluster-level access should be granted the user privileges.
- Content-Based Security should be used with read-only accounts.

**Tip:** To authorize Elasticsearch template and pipeline management and allow creation of indices with for example name **kubernetes_cluster.*** for user fluentbit, add the following privileges:

- INSERT INTO elastic_admin.privileges (role,actions,indices) VALUES ('fluentbit','cluster:monitor/nodes/info','.*');
- INSERT INTO elastic_admin.privileges (role,actions,indices) VALUES ('fluentbit','cluster:admin/ingest/pipeline/put','.*');
- INSERT INTO elastic_admin.privileges (role,actions,indices) VALUES ('fluentbit','indices:.*','kubernetes_cluster.*');

### 8.6.6 Permissions

Cassandra permission associated to a role are granted or revoked as shown below :

```
GRANT SELECT ON KEYSPACE sales TO sales;
LIST ALL PERMISSIONS;

 role      | username  | resource          | permission
-----------+-----------+-------------------+------------
 cassandra | cassandra |     <role sales> |      ALTER
 cassandra | cassandra |     <role sales> |       DROP
 cassandra | cassandra |     <role sales> |  AUTHORIZE
    sales |      sales | <keyspace sales> |     MODIFY

(4 rows)
```

```
cassandra@cqlsh> REVOKE SELECT ON KEYSPACE sales FROM sales;
```

Cassandra permissions associated to a role are mapped into Elasticserach Document and Indices APIs as follows.

| Cassandra privilege | Cassandra Permissions | Elasticsearch Action | Elasticsearch API |
|---|---|---|---|
| CREATE | CREATE KEYSPACE and CREATE TABLE in any keyspace. | indices:admin/create | Create Index |
| ALTER | ALTER KEYSPACE and ALTER TABLE in any keyspace. | indices:admin/mapping indices:admin/alias indices:admin/template indices:admin/settings/update | Put Mapping Index Alias Index Templates Update Indices Settings |
| DROP | DROP KEYSPACE and DROP TABLE in any keyspace. | indices:admin/delete | Delete Index |
| EXECUTE | Execute operations on any Elasticsearch indices associated to the granted keyspaces. | indices:admin/refresh indices:admin/flush indices:admin/optimize indices:admin/open indices:admin/close indices:admin/cache/clear indices:admin/analyze | Refresh Flush Force Merge Open Index Close Index Clear Cache Analyze |
| DESCRIBE | Retrieve stats about Elasticsearch indices associated with the granted mbeans. | indices:monitor/stats indices:monitor/segments | Indices Stats Indices Segments |
| SELECT | SELECT on any table. | indices:data/read/.* indices:admin/get indices:admin/exists indices:admin/types/exists indices:admin/mapping indices:admin/mappings/fields/get | All document reading API Get Index Indices Exists Type Exists Get Mapping Get Field Mapping |
| MODIFY | INSERT, UPDATE, DELETE on any table. | indices:data/write/.* | All document writing API |

### 8.6.7 Privilege caching

For performance reasons, the Elasticsearch privilege table is cached into the memory, according the following settings in **conf/elasticsearch.yml** :

| Setting | Default | Description |
|---|---|---|
| `aaa.privilege_cache_expire` | **1h** | Privlege cache entry TTL |
| `aaa.privilege_cache_size` | **1024** | Privilege cache max entries. |

When changing a privilege in `elastic_admin.privileges`, you should clear the cache with the follwing REST request to put the change into effect on available nodes :

```
curl -XPOST --user admin:admin --cacert conf/cacert.pem "https://localhost:9200/_aaa_
↪clear_privilege_cache?pretty"
{
  "_nodes" : {
    "total" : 2,
    "successful" : 2,
```

```
    "failed" : 0
  },
  "cluster_name" : "TestCluster",
  "nodes" : {
    "d607917d-8c68-4cc5-8dc2-2aa21f5ea986" : {
      "name" : "127.0.0.2"
    },
    "a1c5307c-5f5a-4676-a6f0-50f221dd655b" : {
      "name" : "127.0.0.1"
    }
  }
}
```

If you just want to invalidate the privilege cache for some roles, you can just specify the roles :

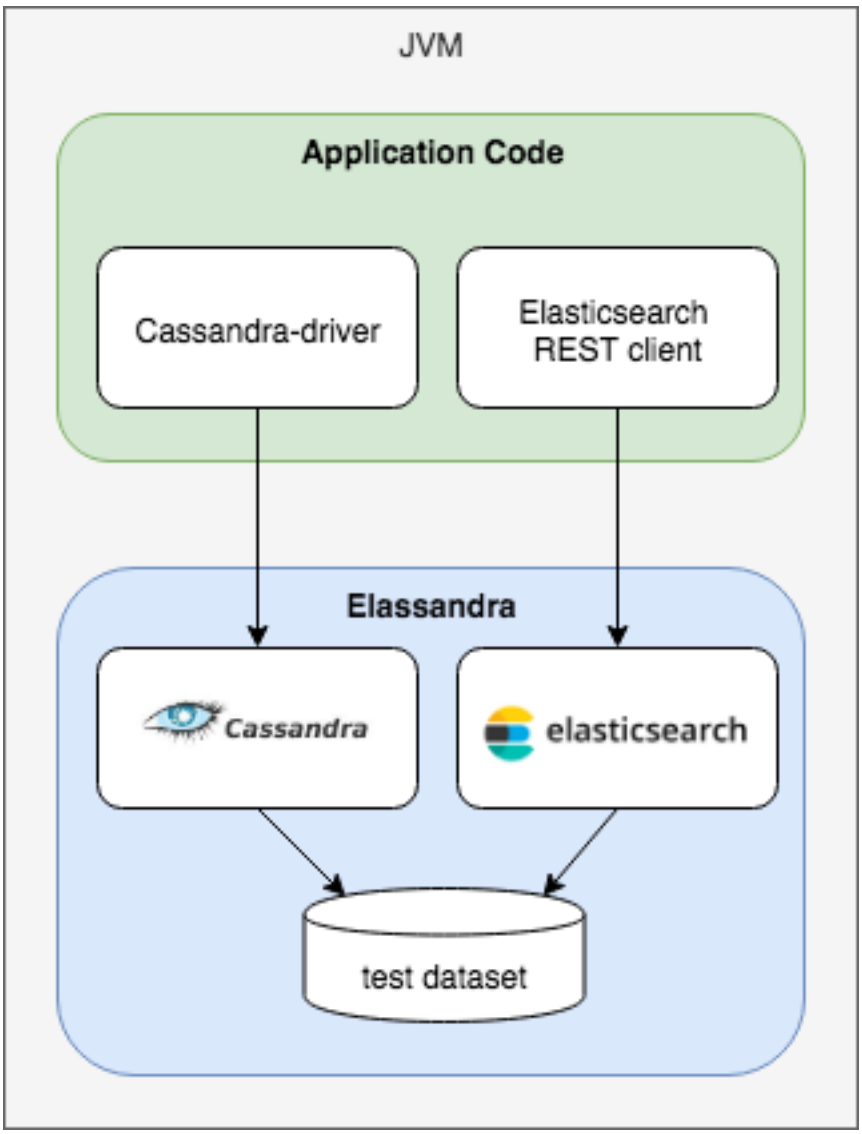
```
POST _aaa_clear_privilege_cache?pretty&roles=sales,kibana"
```

**Tip:** If you are running multiple Elasticsearch clusters in your Cassandra cluster, you should clear the privilege cache on each datacenter where Elasticsearch has been enabled.

## 8.7 Integration

### 8.7.1 Application UNIT Tests

Elassandra Unit helps you writing isolated JUnit tests in a Test Driven Development style with an embedded Elassandra instance.




In order to execute authenticated Elasticsearch queries through CQL with Elassandra unit:

- Set the system property `cassandra.custom_query_handler_class` to `org.elassandra.index.EnterpriseElasticQueryHandler`.
- Add the following test dependencies to your project.

Maven configuration:

```
<dependency>
   <groupId>com.strapdata.elasticsearch.plugin.enterprise</groupId>
   <artifactId>strapdata-plugin</artifactId>
   <version>${elassandra.version}</version>
   <scope>test</scope>
 </dependency>
 <dependency>
   <groupId>com.strapdata.elasticsearch.plugin.enterprise</groupId>
   <artifactId>strapdata-core</artifactId>
   <version>${elassandra.version}</version>
   <scope>test</scope>
 </dependency>
```

```
...
<plugins>
   <plugin>
     <groupId>org.apache.maven.plugins</groupId>
     <artifactId>maven-surefire-plugin</artifactId>
     <version>3.0.0-M3</version>
     <configuration>
       <systemPropertyVariables>
         <cassandra.custom_query_handler_class>org.elassandra.index.
↪EnterpriseElasticQueryHandler</cassandra.custom_query_handler_class>
       </systemPropertyVariables>
     </configuration>
   </plugin>
</plugins>
```

Gradle configuration:

```
dependencies {
   test 'com.strapdata.elasticsearch.plugin:strapdata-plugin:${elassandra.version}'
   test 'com.strapdata.elasticsearch.plugin:strapdata-core:${elassandra.version}'
}
```

### 8.7.2 Secured Transport Client

The elasticsearch transport protocol used for the inter-node communication can be used directly from your java application (deprecated). It is very efficient as it does not have to deal with the JSON serialzation. Strapdata provides a SSL transport client to work with a secured Elassandra cluster :

1. If your Elassandra cluster requires user authentification, check that your user have access to the cluster topology with the *Nodes Info API* (action **cluster:monitor/nodes/info**).
2. Add the **ssl-transport-client.jar** and its dependencies in your CLASSPATH.
3. Add the desired configuration to your client settings, including SSL settings as shown in the following example.
4. Add an `ssl.transport_client_credential` containing *username*:*password* to monitor the cluster state. This account must be authorized to do `cluster:monitor/state` and `cluster:monitor/nodes/liveness` in the `elastic_admin.privileges` table.

```
CREATE ROLE monitor WITH PASSWORD = 'monitor' AND LOGIN = true;
INSERT INTO elastic_admin.privileges (role, actions,indices) VALUES('monitor',
↪'cluster:monitor/state','.*');
INSERT INTO elastic_admin.privileges (role, actions,indices) VALUES('monitor',
↪'cluster:monitor/nodes/liveness','.*');
```

#. Add an **Authorization** header to your client containing your based-64 encoded login and password. This account must have the appropriate Cassandra permissions or privileges in the `elastic_admin.privileges` table.

```
...
import com.strapdata.elasticsearch.plugins.ssl.PreBuiltSslTransportClient;

TransportClient client = new PreBuiltSslTransportClient(Settings.builder()
     .put("cluster.name", "myClusterName")
     .put("client.transport.sniff",true)
     .put("ssl.transport.enabled", true)
     .put("ssl.truststore.path", "/path/to/truststore.jks")
```

```
    .put("ssl.truststore.password", "******")
    .put("ssl.transport_client_credential", "monitor:password")   // Add credential␣
↪to monitor Elasticsearch
    ...
    .build())
 .addTransportAddress(new InetSocketTransportAddress(InetAddress.getByName("localhost
↪"), 9300))

 // Add user credential to request Elasticsearch
 client.filterWithHeader(Collections.singletonMap("Authorization",␣
↪PreBuiltSslTransportClient.encodeBasicHeader("bob","password")));
```

Available security settings for the secured transport client for Elassandra :

| Setting | Default | Description |
|---|---|---|
| ssl.transport.enabled | **false** | Enable SSL on transport connections. |
| ssl.algorithm | **SunX509** | Algorithm used to manage keys and certificates. |
| ssl.storetype | **JKS** | Crytptographic stores file format. |
| ssl.trust_all_cert | **false** | Trust all certificates |
| ssl.truststore.path | **conf/.truststore** | Path to your truststore. |
| ssl.truststore.password | **cassandra** | Truststore password. |
| ssl.protocol | **TLSv1.2** | Secure protocol. |
| ssl.ciphers | **JCE default** | SSL Cipher suite |
| ssl.require_client_auth | **false** | Enable SSL client authentication. |
| ssl.keystore.path | **conf/.truststore** | Path to your keystore when using SSL client authentication. |
| ssl.keystore.password | **cassandra** | Truststore password when using SSL client authentication. |
| ssl.require_endpoint_verification | **false** | Enable server hostname verification. |
| ssl.transport_client_credential | | *login*:*password* used to monitor the Elasticsearch cluster state. |

### 8.7.3 Multi-user Kibana configuration

Kibana needs a dedicated kibana account to manage the kibana configuration, with the CREATE, ALTER, MODIFY, SELECT cassandra permissions.

```
CREATE ROLE kibana WITH PASSWORD = '*****' AND LOGIN = true;
CREATE KEYSPACE "_kibana" WITH replication = {'class': 'NetworkTopologyStrategy', 'DC1
↪':'1'};
GRANT CREATE ON KEYSPACE "_kibana" TO kibana;
GRANT ALTER ON KEYSPACE "_kibana" TO kibana;
GRANT SELECT ON KEYSPACE "_kibana" TO kibana;
GRANT MODIFY ON KEYSPACE "_kibana" TO kibana;
LIST ALL PERMISSIONS OF kibana;

 role   | username | resource            | permission
--------+----------+---------------------+------------
 kibana |   kibana | <keyspace _kibana> |     CREATE
 kibana |   kibana | <keyspace _kibana> |      ALTER
 kibana |   kibana | <keyspace _kibana> |     SELECT
 kibana |   kibana | <keyspace _kibana> |     MODIFY
```

Add the cluster monitoring and kibana indices access rights to the *kibana* user, and refresh the privileges cache.

```
INSERT INTO elastic_admin.privileges (role,actions,indices) VALUES ('kibana',
↪'cluster:monitor/.*','.*');
INSERT INTO elastic_admin.privileges (role,actions,indices) VALUES ('kibana',
↪'indices:.*','\.kibana');
INSERT INTO elastic_admin.privileges (role,actions,indices) VALUES ('kibana',
↪'indices:.*','\.kibana_.*');
SELECT * FROM elastic_admin.privileges;

 role   | actions             | indices      | fields | query
--------+---------------------+--------------+--------+-------
 kibana | cluster:monitor/.* |       .*     |   null |  null
 kibana |          indices:.* |    \.kibana |   null |  null
 kibana |          indices:.* | \.kibana_.* |   null |  null
```

Finally, Kibana user accounts must have :

- the SELECT permission on vizualized indices, especially on your default kibana index.
- the SELECT permission on the Kibana keyspace to read kibana configuration.
- the MODIFY permission on the Kibana keyspace to store kibana configuration if authorized to create/update Kibana objects.

**Tip:** Once a user has been authenticated by Kibana, Kibana will keep this information. In order to logout from your browser, clear the cookies and data associated with your Kibana server.

### 8.7.4 Kibana and Content-Based Security

As explained in the cassandra documentation, you can grant a role to another role and create a hierarchy of roles. Next, you can give some elasticsearch privileges to a base role inherited by some user roles allowed to login, and specify a query filter or field-level filter to this base role.

In the following example, the base role *group_a* has a read access to index *my_index* with a document-level filter defined by a term query. Thereafter, the user role *bob* (allowed to log in) will inherit of the privileges from the base role *group_a* to read the kibana configuration and the index *my_index* only for documents where *category* is *A*.

```
REVOKE SELECT ON KEYSPACE my_index FROM kibana;
CREATE ROLE group_a WITH LOGIN = false;
GRANT SELECT ON KEYSPACE "_kibana" to group_a;
INSERT INTO elastic_admin.privileges (role, actions, indices, query) VALUES('group_a',
↪'indices:data/read/.*','my_index', '{ "term" : { "category" : "A" }}');
CREATE ROLE bob WITH PASSWORD = 'bob' AND LOGIN = true;
GRANT group_a TO bob;
```

Don't forget to refresh the privileges cache by issuing the following command :

```
POST /_aaa_clear_privilege_cache
```

### 8.7.5 Elasticsearch Spark connector

The elasticsearch-hadoop connector can access a secured Elassandra cluster by providing the same SSL/TLS and Username/Pasword authentication parameters as the orginal connector. Below is an example of a spark-shell.

```
ES_OPTS="$ES_OPTS --conf spark.es.nodes=127.0.0.1"
ES_OPTS="$ES_OPTS --conf spark.es.net.ssl=true"
ES_OPTS="$ES_OPTS --conf spark.es.net.ssl.truststore.location=file:///path/to/
↪truststore.jks"
ES_OPTS="$ES_OPTS --conf spark.es.net.ssl.truststore.pass=*******"
ES_OPTS="$ES_OPTS --conf spark.es.net.http.auth.user=john"
ES_OPTS="$ES_OPTS --conf spark.es.net.http.auth.pass=*******"

bin/spark-shell --driver-class-path path/to/elasticsearch-hadoop-5.5.0.jar $ES_OPTS
```

In order to work, the Elasticsearch spark connector will require some privileges to monitor your cluster and request for availables shards for search. You can associate these privileges to a dedicated Cassandra role *spark*, and grant this role to the account used in your spark application. The *spark* role has no Cassandra permission, but user *john* inherits its privileges from the `elastic_admin.privileges` table.

```
CREATE ROLE spark;
INSERT INTO elastic_admin.privileges (role,actions,indices) VALUES ('spark',
↪'cluster:monitor/.*','.*');
INSERT INTO elastic_admin.privileges (role,actions,indices) VALUES ('spark',
↪'indices:admin/shards/search_shards','.*');
SELECT * FROM elastic_admin.privileges WHERE role='spark';

 role   | actions                              | indices | fields | query
--------+--------------------------------------+---------+--------+-------
  spark |                 cluster:monitor/.* |      .* |   null |  null
  spark | indices:admin/shards/search_shards |      .* |   null |  null

(2 rows)
GRANT spark TO john;
LIST ROLES of john;

 role  | super | login | options
-------+-------+-------+---------
 spark | False | False |        {}
  john | False |  True |        {}

(2 rows)
```

### 8.7.6 Cassandra Spark Connector

The cassandra-spark-connector can request both Cassandra and Elasticsearch through the CQL driver.

## 8.8 Elasticsearch Auditing

Elasticsearch auditing tracks security events using the following fields :

| Field | Description |
|---|---|
| status | GRANTED(200), UNAUTHORIZED(401), FORBIDDEN(403), BLOCKED(409) |
| type | PRIVILEGE, PERMISSION, UNAUTHORIZED, UNSUPPORTED, TAMPERED |
| login | User login |
| role | Cassandra role |
| source | Source IP of the elasticsearch request, or the value of the **X-Forwarded-For** HTTP header if present. |
| action | Elasticsearch action |
| indices | Requested indices |

Audits events are recorded in a Cassandra table or in a log file configured as an appender in your **conf/logback.xml** file.

| Setting | Default | Description |
|---|---|---|
| `aaa.audit.enabled` | **false** | Enable or disable Elasticsearch auditing. |
| `aaa.audit.appender` | **none** | Audit events are recorded in a Cassandra table (**cql**) or in a logback appender (**log**). |
| `aaa.audit.include_login` | | Comma separated list of logins to audit |
| `aaa.audit.exclude_login` | | Comma separated list of logins not audited |

### 8.8.1 Logback Audit

When using the **log** appender for audit, you should configure a dedicated logback appender in your **conf/logback.xml** file :

```
<appender name="AUDIT" class="ch.qos.logback.core.rolling.RollingFileAppender">
   <file>${cassandra.logdir}/audit.log</file>
   <rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
      <fileNamePattern>${cassandra.logdir}/audit.log.%i.zip</fileNamePattern>
      <minIndex>1</minIndex>
      <maxIndex>20</maxIndex>
   </rollingPolicy>
   <triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
      <maxFileSize>500MB</maxFileSize>
   </triggeringPolicy>
   <encoder>
      <pattern>%date{ISO8601} %msg%n</pattern>
   </encoder>
</appender>
```

And add a logger named **LogbackAuditor** with additiviy set to **false** :

```
<logger name="LogbackAuditor" level="DEBUG" additivity="false" >
     <appender-ref ref="AUDIT" />
</logger>
```

Below is an exemple of audit logs in the **logs/audit.log** file :

```
2017-10-20 14:11:49,854 200,PERMISSION,sales,roles/sales,/10.0.1.5,indices:data/read/
↪search,[sales_*]
2017-10-20 14:11:51,607 200,PERMISSION,sales,roles/sales,/10.0.1.5,indices:data/read/
↪search,[.kibana]
```

```
2017-10-20 14:11:52,377 200,PRIVILEGE,kibana,roles/kibana,/10.0.1.5,cluster:monitor/
↪main,null
2017-10-20 14:11:52,501 200,PRIVILEGE,kibana,roles/kibana,/10.0.1.5,cluster:monitor/
↪nodes/info,null
2017-10-20 14:11:52,627 200,PRIVILEGE,kibana,roles/kibana,/10.0.1.5,cluster:monitor/
↪nodes/info,null
2017-10-20 14:11:52,679 200,PERMISSION,sales,roles/sales,/10.0.1.5,indices:data/read/
↪mget[shard],[.kibana]
2017-10-20 14:11:52,751 200,PERMISSION,kibana,roles/kibana,/10.0.1.5,indices:data/
↪read/mget[shard],[.kibana]
2017-10-20 14:11:52,868 200,PRIVILEGE,kibana,roles/kibana,/10.0.1.5,cluster:monitor/
↪health,[.kibana]
2017-10-20 14:11:52,990 200,PERMISSION,kibana,roles/kibana,/10.0.1.5,indices:data/
↪read/search,[.kibana]
```

### 8.8.2 CQL Audit

When using the **cql** appender for audit, audit events are recorded in the cassandra table `elastic_audit.events`.

```
cassandra@cqlsh> select * from elastic_audit.events ;

 node      | event                                | action                          |␣
↪indices     | level      | login   | role          | source    | status
-----------+--------------------------------------+---------------------------------+----
↪---------+------------+---------+--------------+-----------+--------
 10.0.0.4 | cf74fed0-b5a2-11e7-9508-157b11ac2561 |          cluster:monitor/main |  ␣
↪    null |  PRIVILEGE | kibana | roles/kibana | 10.0.1.5 |    200
 10.0.0.4 | d2026070-b5a2-11e7-9508-157b11ac2561 |         cluster:monitor/state |  ␣
↪    null |  PRIVILEGE | kibana | roles/kibana | 10.0.1.5 |    200
 10.0.0.4 | da709470-b5a2-11e7-9508-157b11ac2561 |      indices:data/read/search | [
↪'sales_*'] | PERMISSION |  sales |  roles/sales | 10.0.1.5 |    200
 10.0.0.4 | d8025390-b5a2-11e7-9508-157b11ac2561 |        cluster:monitor/health | ['.
↪kibana'] |  PRIVILEGE | kibana | roles/kibana | 10.0.1.5 |    200
 10.0.0.4 | cf9de390-b5a2-11e7-9508-157b11ac2561 |    cluster:monitor/nodes/info |  ␣
↪    null |  PRIVILEGE | kibana | roles/kibana | 10.0.1.5 |    200
```

If you want to have multiple copies of audit events in your cluster, you can alter the following default settings :

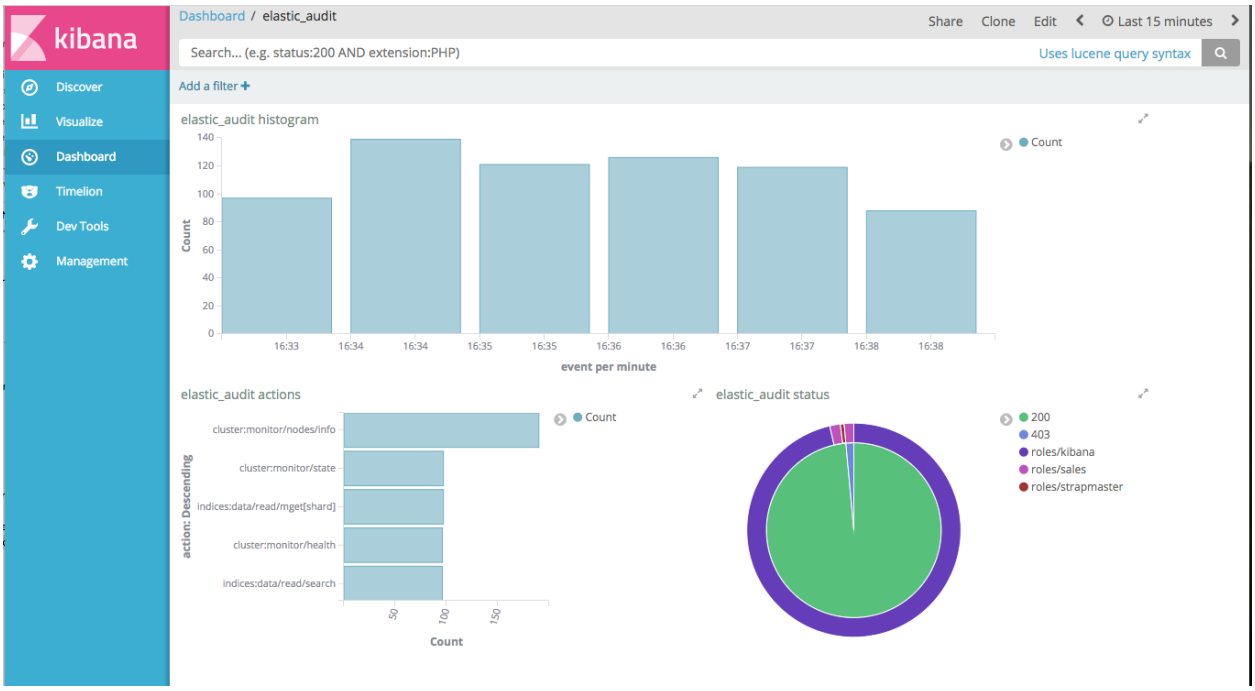
| Setting | Default | Description |
|---|---|---|
| `aaa.audit.cql.rf` | **1** | Cassandra *Replication Factor* used when creating the `elastic_audit` keyspace. |
| `aaa.audit.cql.cl` | **LOCAL_ONE** | Write *Consistency Level* for audit events. |

You can index with Elasticsearch the `elastic_audit.events` table using the following mapping, where the *event* timeuuid column is explicitly mapped to a date :

```
curl -XPUT --user admin:admin --cacert conf/cacert.pem "https://localhost:9200/
↪elastic_audit/" -d'
{
   "mappings":{
      "events":{
         "discover":"^((?!event).*)",
         "properties":{
            "event":{
```

```
                "type":"date",
                "cql_collection":"singleton"
            }
        }
    }
}'
```

Next, you can build your audit trail Kibana report.




**Tip:** Keep in mind that CQL audit trail involves a network overhead because each node sends some events to all other nodes. For better performance, you should use the Logback audit and collect the events with Beat+Logstash into a dedicated Elassandra cluster.

## 8.9 Limitations

### 8.9.1 Content-Based Security Limitations

- The request cache is disabled for search requests.
- The following queries are not supported for document-level filtering :
  - **Has Parent**, **Has Child** queries.
  - **Terms** queries with lookups.
  - **Geo Shape** queries without inline shape definition.
  - **Percolate** queries.

If you try to insert an unsupported query in `elastic_admin.privileges.query`, you will get a syntax error as shown below :

```
cassandra@cqlsh> insert into elastic_admin."privileges" (role,actions,indices,query)␣
↪VALUES ('blogger','indices:data/read/.*','blog','{"query":{ "has_parent":{"parent_
↪type":"blog","query":{"term":{"tag":"something"}}}}}');
SyntaxException: Unsupported query for content-based filtering
```

CHAPTER 9

# Integration

## 9.1 Integration with an existing Cassandra cluster

Elassandra includes a modified version of Cassandra, available at strapdata-cassandra repro, so **all nodes of a cluster should run Elassandra binaries**. However, you can start a node with or without the Elasticsearch support. Obviously, all nodes of a datacenter should run Cassandra only or Cassandra with Elasticsearch.

### 9.1.1 Rolling upgrade from Cassandra to Elassandra

Before starting any Elassandra node with Elasticsearch enabled, do a rolling replace of the Cassandra binaries with the Elassandra ones. For each node :

- Install Elassandra.
- Replace the Elassandra configuration files (cassandra.yaml and snitch configuration file) with the ones from your existing cluster.
- Bind the Elassandra data folder to the existing Cassandra data folder
- Stop your Cassandra node.
- Restart Cassandra `elassandra bin/cassandra` or Cassandra with Elasticsearch enabled `elassandra bin/cassandra -e`

### 9.1.2 Create a new Elassandra datacenter

The overall procedure is similar to the Cassandra one described in Adding a datacenter to a cluster.

For each node in your new datacenter :

- Install Elassandra.
- Set `auto_bootstrap: false` in your **conf/cassandra.yaml**.
- Start Cassandra-only nodes in your new datacenter and check that all nodes join the cluster.

```
bin/cassandra
```

- Restart all nodes in your new datacenter with Elasticsearch enabled. You should see started shards but empty indices.

```
bin/cassandra -e
```

- Set the replication factor of indexed keyspaces to one or more in your new datacenter.
- Pull data from your existing datacenter.

```
nodetool rebuild <source-datacenter-name>
```

After rebuilding all of your new nodes, you should see the same number of documents for each index in your new and existing datacenters.

- Set `auto_bootstrap: true` (default value) in your **conf/cassandra.yaml**
- Create new Elasticsearch index or map some existing Cassandra tables.

---

**Tip:** If you need to replay this procedure for a node :

- stop your node
- nodetool removenode <id-of-node-to-remove>
- clear data, commitlogs and saved_cache directories.

---

## 9.2 Installing Elasticsearch plugins

Elasticsearch plugin installation remains unchanged, see Elasticsearch plugin installation.

- bin/plugin install <url>

## 9.3 Running Kibana with Elassandra

Kibana can run with Elassandra, providing a visualization tool for Cassandra and Elasticsearch data.

- If you want to load sample data from the Kibana Getting started, apply the following changes to logstash.jsonl with a sed command.

```
s/logstash-2015.05.18/logstash_20150518/g
s/logstash-2015.05.19/logstash_20150519/g
s/logstash-2015.05.20/logstash_20150520/g

s/article:modified_time/articleModified_time/g
s/article:published_time/articlePublished_time/g
s/article:section/articleSection/g
s/article:tag/articleTag/g

s/og:type/ogType/g
s/og:title/ogTitle/g
s/og:description/ogDescription/g
s/og:site_name/ogSite_name/g
```

```
s/og:url/ogUrl/g
s/og:image:width/ogImageWidth/g
s/og:image:height/ogImageHeight/g
s/og:image/ogImage/g

s/twitter:title/twitterTitle/g
s/twitter:description/twitterDescription/g
s/twitter:card/twitterCard/g
s/twitter:image/twitterImage/g
s/twitter:site/twitterSite/g
```

## 9.4 JDBC Driver sql4es + Elassandra

The Elasticsearch JDBC driver. can be used with Elassandra. Here is a code example :

```
Class.forName("nl.anchormen.sql4es.jdbc.ESDriver");
Connection con = DriverManager.getConnection("jdbc:sql4es://localhost:9300/twitter?
↪cluster.name=Test%20Cluster");
Statement st = con.createStatement();
ResultSet rs = st.executeQuery("SELECT user,avg(size),count(*) FROM tweet GROUP BY␣
↪user");
ResultSetMetaData rsmd = rs.getMetaData();
int nrCols = rsmd.getColumnCount();
while(rs.next()){
    for(int i=1; i<=nrCols; i++){
         System.out.println(rs.getObject(i));
     }
}
rs.close();
con.close();
```

## 9.5 Running Spark with Elassandra

For Elassandra 5.5, a modified version of the elasticsearch-hadoop connector is available for Elassandra on the strapdata repository. This connector works with spark as described in the Elasticsearch documentation available at elasticsearch/hadoop.

For example, in order to submit a spark job in client mode:

```
bin/spark-submit --driver-class-path <yourpath>/elasticsearch-spark_2.10-2.2.0.jar  --
↪master spark://<sparkmaster>:7077 --deploy-mode client <application.jar>
```

# CHAPTER 10

# Testing

Elasticsearch comes with a testing framework based on JUNIT and RandomizedRunner provided by the randomized-testing project. Most of these tests work with Elassandra to ensure compatibility between Elasticsearch and Elassandra.

## 10.1 Testing environnement

By default, JUnit creates one instance for each test class and executes each *@Test* method in parallel with many threads. Because Cassandra uses many static variables, concurrent testing is not possible, so each test is executed sequentially (using a semaphore to serialize tests) on a single node Elassandra cluster listening on localhost, see ESSingleNodeTestCase]). Test configuration is located in **core/src/test/resources/conf**, data and logs are generated in **core/build/testrun/test/J0**.

Between each test, all indices (and underlying keyspaces and tables) are removed to have idempotent testings and avoid conflicts with index names. System settings `es.synchronous_refresh` and `es.drop_on_delete_index` are set to *true* in the parent *pom.xml*.

Finally, the testing framework randomizes the local settings representing a specific geographical, political, or cultural region, but Apache Cassandra does not support such setting because string manipulation are implemented with the default locale settings (see CASSANDRA-12334). For exemple, *String.format("SELECT %s FROM ...",...)* is computed as *String.format(Local.getDefault(),"SELECT %s FROM ...",...)*, involving errors for some Locale setting. As a workaround, a javassit byte-code manipulation in the Ant build step adds a *Locale.ROOT* argument to weak the method calls in all Cassandra classes.

## 10.2 Elassandra build tests

Elassandra build unit tests allows using both the Elasticsearch API and CQL requests as shown in the following example.

```
public class BasicTests extends ESSingleNodeTestCase {

    @Test
```

```
    public void testTest() throws Exception {
     createIndex("cmdb");
     ensureGreen("cmdb");

     process(ConsistencyLevel.ONE,"CREATE TABLE cmdb.server ( name text, ip inet,␣
↪netmask int, prod boolean, primary key (name))");
     assertAcked(client().admin().indices().preparePutMapping("cmdb")
             .setType("server")
             .setSource("{ \"server\" : { \"discover\" : \".*\", \"properties\": { \
↪"name\":{ \"type\":\"keyword\" }}}}")
             .get());

     process(ConsistencyLevel.ONE,"insert into cmdb.server (name,ip,netmask,prod)␣
↪VALUES ('localhost','127.0.0.1',8,true)");
     process(ConsistencyLevel.ONE,"insert into cmdb.server (name,ip,netmask,prod)␣
↪VALUES ('my-server','123.45.67.78',24,true)");

     assertThat(client().prepareGet().setIndex("cmdb").setType("server").setId("my-
↪server").get().isExists(), equalTo(true));
     assertThat(client().prepareGet().setIndex("cmdb").setType("server").setId(
↪"localhost").get().isExists(), equalTo(true));

     assertEquals(client().prepareIndex("cmdb", "server", "bigserver234")
         .setSource("{\"ip\": \"22.22.22.22\", \"netmask\":32, \"prod\" : true, \
↪"description\": \"my big server\" }")
         .get().getResult(), DocWriteResponse.Result.CREATED);

     assertThat(client().prepareSearch().setIndices("cmdb").setTypes("server").
↪setQuery(QueryBuilders.queryStringQuery("*:*")).get().getHits().getTotalHits(),␣
↪equalTo(3L));
    }
}
```

To run this specific test :

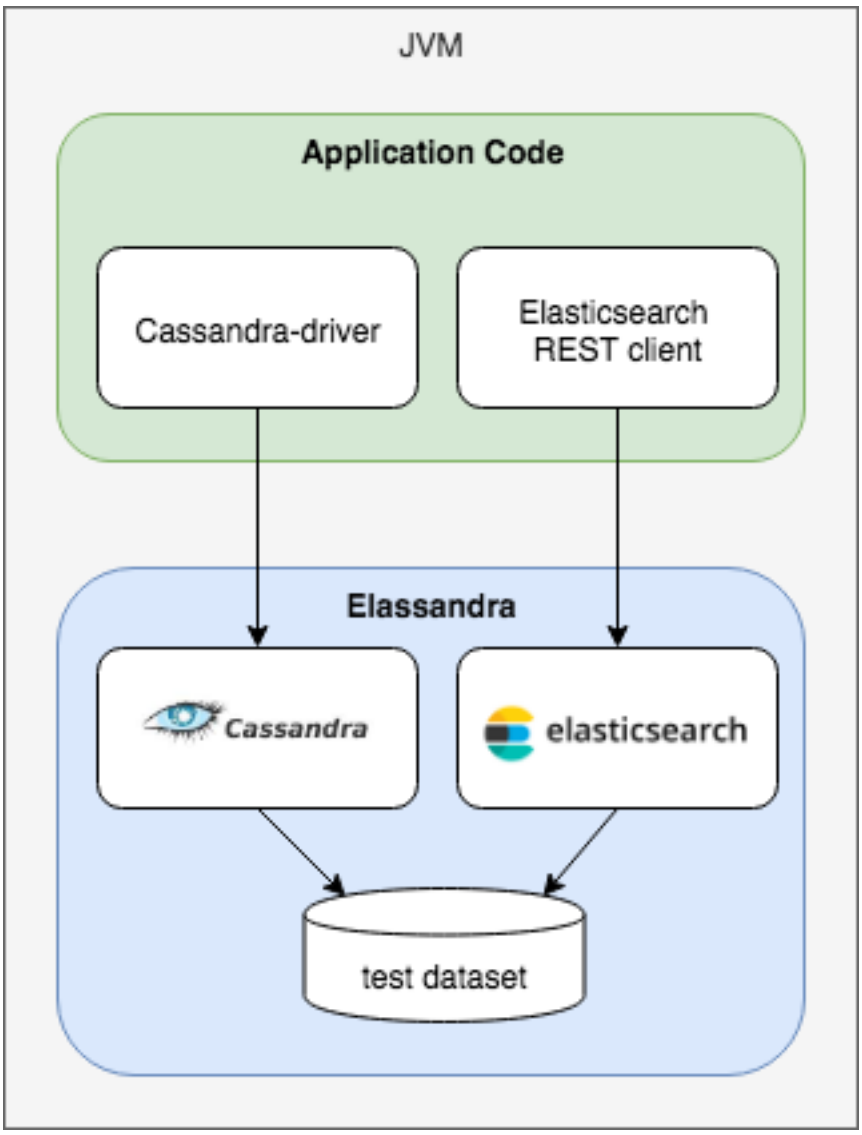
```
$gradle :server:test -Dtests.seed=96A0B026F3E89763 -Dtests.class=org.elassandra.
↪BasicTests  -Dtests.security.manager=false -Dtests.locale=it-IT -Dtests.
↪timezone=Asia/Tomsk
```

To run all core unit tests :

```
$gradle server:test
```

## 10.3 Application tests with Elassandra-Unit

Elassandra-Unit helps you writing isolated JUnit tests in a Test Driven Development style with an embedded Elassandra instance.




- Start an embedded Elassandra (including both Cassandra and Elasticsearch).
- Create structure (keyspace and Column Families) and load data from an XML, JSON or YAML DataSet.
- Execute a CQL script.
- Query Cassandra through the Cassandra driver.
- Query Elasticsearch through the Elasticsearch REST API.

See the Elassandra-Unit README for more information.

CHAPTER 11

# Breaking changes and limitations

## 11.1 Deleting an index does not delete cassandra data

By default, Cassandra is considered as a primary data storage for Elasticsearch, so deleting an Elasticsearch index does not delete Cassandra content, keyspace and tables remain unchanged. If you want to use Elassandra as Elasticsearch, you can configure your cluster or only some indices with the `drop_on delete_index` like this.

```
$curl -XPUT -H "Content-Type: application/json" "$NODE:9200/twitter/" -d'{
   "settings":{ "index":{ "drop_on_delete_index":true } }
}'
```

Or to set `drop_on delete_index` at cluster level :

```
$curl -XPUT -H "Content-Type: application/json" "$NODE:9200/_cluster/settings" -d'{
   "persistent":{ "cluster.drop_on_delete_index":true }
}'
```

## 11.2 Nested or Object types cannot be empty

Because Elasticsearch nested and object types are backed by a Cassandra User Defined Type, it requires at least one sub-field in the mapping.

## 11.3 Document _version, _seq_no and _primary_term are meaningless

Elasticsearch's versioning system helps to cope with conflicts, but in a multi-master database like Apache Cassandra, versioning cannot ensure global consistency of compare-and-set operations.

In Elassandra, Elasticsearch version management is disabled by default, document version is not more indexed in lucene files and **document version is always 1**. Elasticsearch version 6.x introduced the _primary_term and _seq_no to uniquely identify updates on a document, but again, in a multi-master system, these counters are not more relevant, and your applications should not rely on it.

Finally, if you need to avoid conflicts on write operations, you should use Cassandra lightweight transactions (or PAXOS transaction). Such lightweight transactions is also used when updating the Elassandra mapping or when indexing a document with *op_type=create*, but of course, it comes with a network cost.

## 11.4 Primary term and Sequence Number

As explained here, Elasticsearch introduced **_primary_term** and **_seq_no** in order to manage shard replication consistently and store these fields in lucene documents. But in Elassandra, replication is fully managed by cassandra and all shard are considered as primary. Thus, these two fields are not more stored in lucene by the default elassandra lucene engine named **VersionLessInternalEngine**. Consequently, all search results comes with *_primary_term = 1* and *_seq_no = 1*.

## 11.5 Index and type names

Because Cassandra does not support special characters in keyspace and table names, Elassandra automatically replaces dots (.) and hyphens (-) characters by underscore (_) in index names, and hyphen (-) characters by underscore (_) in type names to create underlying Cassandra keyspaces and tables.

When such a modification occurs for document type names, Elassandra keeps type names translation in memory to correctly translate back table names to documents types. Obviously, if you have types names like *xxx-xxx* and *xxx_xxx* in the same underlying keyspace, bijective translation is not possible and you can get some trouble.

Moreover, Cassandra table names are limited to 48 characters, so Elasticsearch type names are also limited to 48 characters. If you need longer Elasticsearch index names, you can increase this limit with the Cassandra system property `cassandra.max_name_length`, but be careful with the maximum filename length on your platform in you data directory.

## 11.6 Column names

For Elasticsearch, field mapping is unique in an index. So, two columns having the same name, indexed in an index, should have the same CQL type and share the same Elasticsearch mapping.

## 11.7 Null values

To be able to search for null values, Elasticsearch can replace null by a default value (see https://www.elastic.co/guide/en/elasticsearch/reference/2.4/null-value.html ). In Elasticsearch, an empty array is not a null value, wheras in Cassandra, an empty array is stored as null and replaced by the default null value at index time.

## 11.8 Refresh on write

Elasticsearch write operations support a `refresh` parameter to control when changes made by this request are made visible to search. Possible values are *true*, *false*, or *wait_for* and in this last case, the coordinator node waits until

a refresh happens. But in elassandra, replication is managed by Cassandra and can be asynchronous. As the result managing a refresh on involved shards or waiting for a refresh to happen in not possible.

If we need to search right after a write operation, you can force a refresh before search or, if you have a reasonably low level of updates, set the index settings `index.synchronous_refresh` to true. This provides *Real Time Search* by refreshing shards after each update, but of course, its comes with a cost.

If you have legacy applications using `refresh=true` or `refresh=wait_for`, you can set the system property `es.synchronous_refresh` to a regexp of index name to automatically set `synchronous_refresh` to **true**. By default, because Kibana sometimes updates elasticsearch with `refresh=wait_for`, this system property `es.synchronous_refresh` is set by default to (.kibana.*).

## 11.9 Elasticsearch unsupported features

- Tribe node allows to query multiple Elasticsearch clusters. This feature is not currently supported by Elassandra.
- Elasticsearch snapshot and restore operations are disabled (See Elassandra backup and restore in operations).
- Elasticsearch percolator, reindex and shrink API are not supported.
- Elasticsearch range fiels are supported in version 6.2
- Parent-Child join is currently supported only in Elassandra version 5.5
- Running Elassandra with a java security manager is not supported.

## 11.10 Cassandra limitations

- Elassandra only supports the murmur3 partitioner.
- The thrift protocol is supported only for read operations.
- Elassandra synchronously indexes rows into Elasticsearch. This may increases the write duration, particulary when indexing complex document like GeoShape, so Cassandra `write_request_timeout_in_ms` is set to 5 seconds (Cassandra default is 2000ms, see Cassandra config)
- In order to avoid concurrent mapping or persistent cluster settings updates, Elassandra plays a PAXOS transaction that require QUORUM available nodes for the keyspace *elastic_admin* to succeed. So it is recommended to have at least 3 nodes in 3 distinct racks (A 2 nodes datacenter won't accept any mapping update when a node is unavailable).
- CQL3 **TRUNCATE** on a Cassandra table deletes all associated Elasticsearch documents by playing a delete_by_query where *_type* = <*table_name*>. Of course, such a delete_by_query comes with a performance cost and won't notify IndexingOperationListeners for preDelete and postDelete events if used in an Elasticsearch plugin.

CHAPTER 12

# Indices and tables

- genindex
- modindex
- search