elastic

wp2ss2xAxTY5UnRUdgo2tUgW

Why Kibana ?

Easiest way of running queries is using Kibana tool. It also sends requests to Elastic search API.

It formats the response for us and makes it easier to read.

It also sets the correct-content type header.

Auto typing.

Postman / Curl can be used.

1. GET /\_cluster/health

\_cluster is the api

health is the command

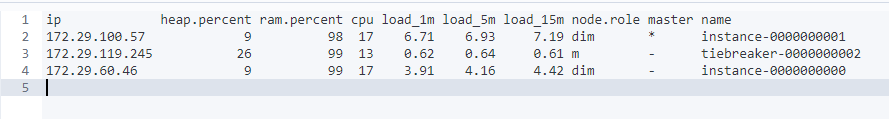
2. \_cat api which outputs data in a human readable format.

\_cat/nodes : all nodes in the cluster

command :

GET /\_cat/nodes?v

v : query parameter : instructs elastic search to include a descriptive header in the output- to identify each peice of info

o/p:  


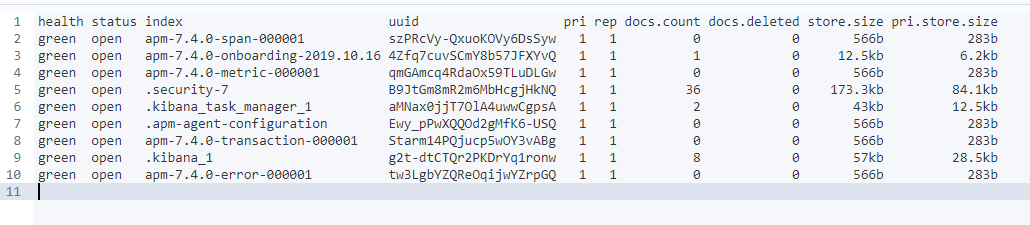
3. GET /\_nodes/stats

API : nodes

To inspect nodes in a lot of detail.  
  


4. GET /\_cat/indices?v

curl -XGET "http://24e374bd44bb4bf884cfd220f7f55cf0.containerhost:9244/\_cat/indices?v"

List all indices  


If its elastic cloud : we can send elastic search requests with the CURL http client

curl –XGET –u elastic: wp2ss2xAxTY5UnRUdgo2tUgW ‘<https://24e374bd44bb4bf884cfd220f7f55cf0.ap-southeast-1.aws.found.io:9243/.kibana/_serch> –H “Content-Type: application/json” –d { “query” : { “match\_all”:{} }}’

5. Sharding and scalability.

Elastic search has a cluster with nodes.

Each node has a capacity and you can store 200gb of data in 2 nodes each having 100gb of data.

Sharding : Makes it possible to scale the amount of documents we can store.

It’s a way to divide indices into small pieces called shard[any number of shards] – done at index level.  
 Indices could contain a few hundred records to a million records.

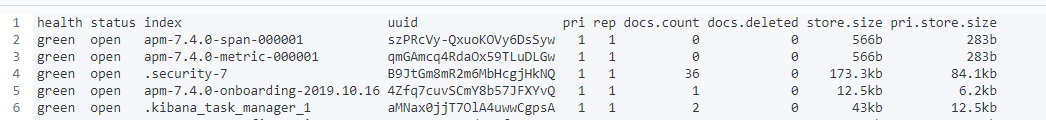
To horizontally scale the data volume indices are divided into shards.

If an index has 5 shards : There is no need that all shards be in different nodes – they could be in the same node as well.

Search query on an index can be run independently on different nodes – increasing the speed of output [as shards could be stored in different nodes.].

Sharding : it is a process to sub divide an index in smaller pieces.

: it increases the number of documents an index can store.

6. Get /cat/indices?v  
  


Pri – primary shard : the number of shards a given index has.

spilt api : to increase the number of shards in an index.

shrik api : to reduce the number of shards for an index.

7. Replication.

- it is configured at the index level

- Copies of shards are created – replica shards.

- When an index is created we can choose how many replicas of each shard we want. 1 being the default.

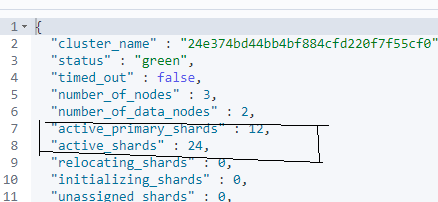
- Replica shard is never stored in the same node.

- Throughput can be increased as the query is distributed.

8. snapshot: to store an index completely.

Used for daily backup, manual backups are taken just before applying changes – just to be sure the changes are good.

9. Create and Index: PUT /indexname

10. Check the cluster Health: GET /\_cluster/health  


The cluster health is **green** as there is replication of shards.

11. check the shards : GET /\_cat/shards?v  


p- primary shard.

r – replica shard.

state = STARTED: both primary and replica shards are available for requests.

12. different roles of a node.

Master node :

Data node :

Ingest Node : This enables a node to run Ingest pipelines.

Node.ml : identifies a node as machine learning

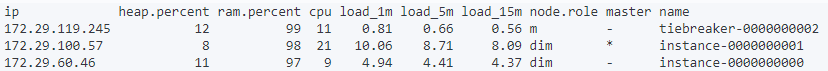
Xpack.ml.enabled : Determines if the node should respond to machine learning requests.

Co-ordination node : how ES process a request and delegates it internally to data nodes[by removing other roles – no role exists., kind of a load balances].

13. Roles of our nodes ?

Command : GET /\_cat/nodes?v

o/p :



dim – data ingest and master

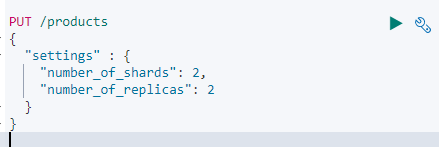
m - master

14. ES exposes a rest API.

- HTTP verb is important to perform a particular action.

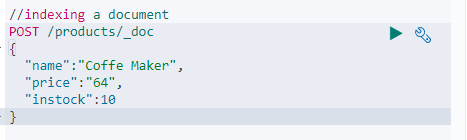
Delete an index :   
  
Create an index by specify the number of shards and replicas. [ For indices that we use to create for production purposes we should stick to the default values]  
first line : http verb + end point

Other lines : json request body AND index settings should be passed as a json object



15. INDEXING A DOCUMENT BY sending a request to an end point.

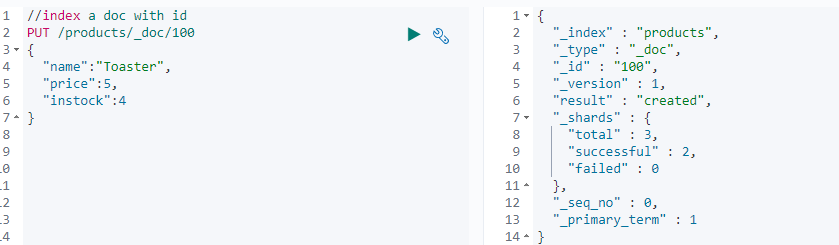
*- we need to define the document within the request body as a json object*

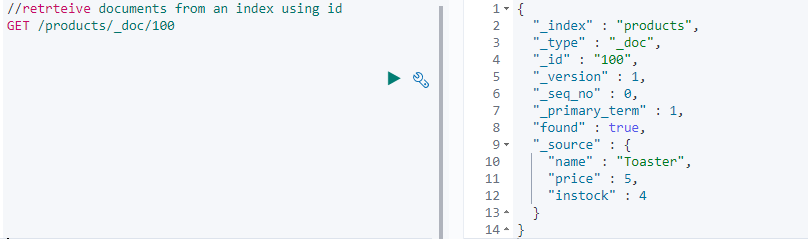


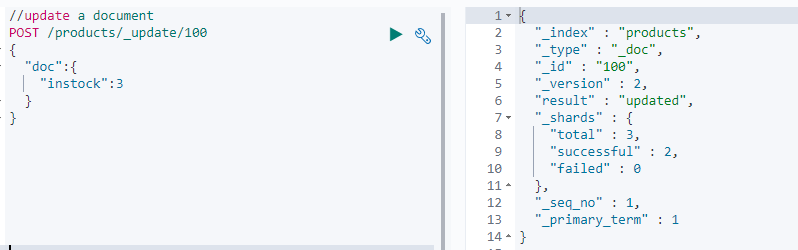
o/p :   
  
  
\_shards : The number of shards in which you successfully stored a document.  
In step 14 : replics are 2, so document was added to primary and also to the 2 replicas so total is 3.

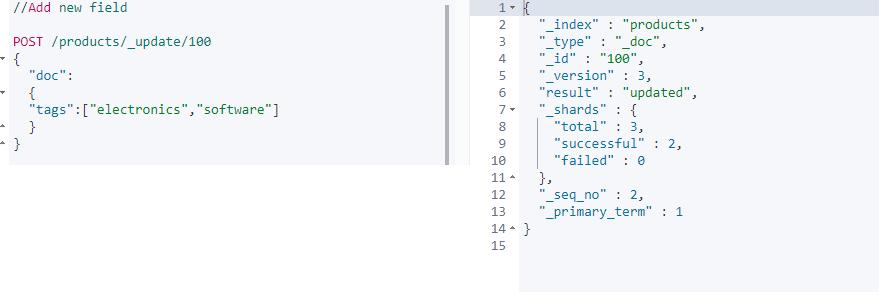
\_id : was created automatically but we can add that too.

16. Add \_id while indexing a document.

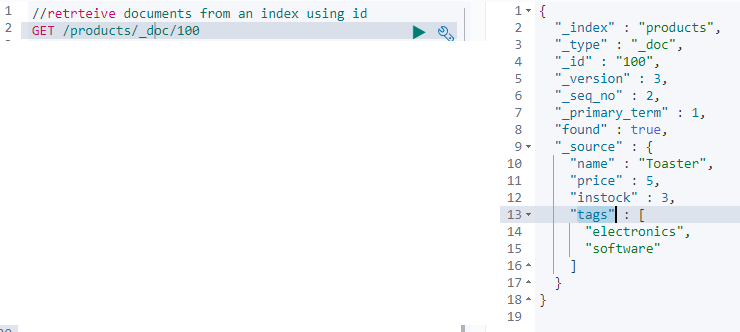
To specify an id : we need to change the http verb to put as this is a convention for REST api.  
  


17. Retrieving documents from an index.  


18. Updating documents: by sending a update request   


19. Adding new fields to existing documents.  


Now we see the new field with data.

****

How this works internally.

* ES documents are immutable (!).
* The \_update api : retrieved the document/changed its fields/re-indexed the document with the same id.

20. Scripted updates

- There is a use case where you retrieve a document first + update a field value + update the document.

- all of this can be done in one go.

- with scripting : we can use if statements

- use the \_update api + script update with the request body