Logstash:

input -> reading from multiple sources

filter -> grok

output/storing -> filter data storing

Elastic Search:

Store

retrival

->ElasticHQ plugin to monitoring and troubleshooting

bigdesk -> monitoring

elasticsearch-head plugin for chrome

start elastic:

go to C:\elastic\_stack\elasticsearch-7.5.2\bin

Run the service:

>elasticsearch

From browser:

http//: localhost:9200

From cygwin:

* Curl -XGET 'localhost:9200'
* Curl –XGET ‘localhost:9200/nodes?pretty’
* curl -XPUT 'localhost:9200/test'

Index has multiple documents, document will be divided and stored as in sigments in shards.

* Segments are lucene index.

**Health check**:

$ curl -XGET 'localhost:9200/\_cat/health'

1581313837 05:50:37 elasticsearch yellow 1 1 2 2 0 0 2 0 - 50.0%

Administrator@DESKTOP-9A5G7S5 ~

$ curl -XGET 'localhost:9200/\_cat/health?v'

* Health color :yellow -> some replicas are not yet allocated.

For node details of elastic :

$ curl -XGET 'localhost:9200/\_cat/nodes?v'

To get the indices:

$ curl -XGET 'localhost:9200/\_cat/indices?v'

Ex: bank/chennai/2

**bank- index, chennai – type, id – 1;**

Create index:

$ curl --header 'content-type:application/JSON' -XPUT 'localhost:9200/test/external/1?pretty' -d '{"name":"eshan"}'

{

"\_index" : "test",

"\_type" : "external",

"\_id" : "1",

"\_version" : 1,

"result" : "created",

"\_shards" : {

"total" : 2,

"successful" : 1,

"failed" : 0

},

"\_seq\_no" : 0,

"\_primary\_term" : 2

}

To access: localhost:9200/bank/external/1

To access data of index:

$ curl -XGET 'localhost:9200/test/external/1/\_source?pretty'

{

"name" : "eshan"

}

* To use the sql based help
* Go to bin directory and run the below command

$ elasticsearch-sql-cli

sql> desc test;

column | type | mapping

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

name |VARCHAR |text

name.keyword |VARCHAR |keyword

from cygwin run the below command:

$ curl -H 'content-type:application/JSON' -XPOST 'localhost:9200/\_xpack/sql?format=txt' -d '{"query":"select \* from test"}'

name

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

eshan

Converting sql data into elastic based data:

$ curl -H 'content-type:application/JSON' -XPOST 'localhost:9200/\_xpack/sql/translate?format=txt&pretty' -d '{"query":"select \* from test"}'

{

"size" : 1000,

"\_source" : {

"includes" : [

"name"

],

"excludes" : [ ]

},

"sort" : [

{

"\_doc" : {

"order" : "asc"

}

}

]

}

To update the document/record command:

$ curl --header 'content-type:application/JSON' -XPOST 'localhost:9200/test/external/1/\_update?pretty' -d '{"doc":{"name":"Tanish"}}'

{

"\_index" : "test",

"\_type" : "external",

"\_id" : "1",

"\_version" : 2,

"result" : "updated",

"\_shards" : {

"total" : 2,

"successful" : 1,

"failed" : 0

},

"\_seq\_no" : 1,

"\_primary\_term" : 2

}

* To delete the document/record:

$ curl --header 'content-type:application/JSON' -XPOST 'localhost:9200/test/external/\_delete\_by\_query?pretty' -d '{"query":{"match":{"name":"Tanish"}}}'

{

"took" : 133,

"timed\_out" : false,

"total" : 1,

"deleted" : 1,

"batches" : 1,

"version\_conflicts" : 0,

"noops" : 0,

"retries" : {

"bulk" : 0,

"search" : 0

},

"throttled\_millis" : 0,

"requests\_per\_second" : -1.0,

"throttled\_until\_millis" : 0,

"failures" : [ ]

}

* Updating the record:

$ curl -H 'content-type:application/JSON' -XPOST 'localhost:9200/test/external/3/\_update?pretty' -d '{"doc": {"name":"jhon", "age":23}}'

{

"\_index" : "test",

"\_type" : "external",

"\_id" : "3",

"\_version" : 1,

"result" : "noop",

"\_shards" : {

"total" : 0,

"successful" : 0,

"failed" : 0

},

"\_seq\_no" : 5,

"\_primary\_term" : 2

}

* To update the particual column records:

$ curl --header 'content-type:application/JSON' -XPOST 'localhost:9200/test/external/1/\_update?pretty' -d '{"script":"ctx.\_source.age+=5"}'

{

"\_index" : "test",

"\_type" : "external",

"\_id" : "1",

"\_version" : 3,

"result" : "updated",

"\_shards" : {

"total" : 2,

"successful" : 1,

"failed" : 0

},

"\_seq\_no" : 6,

"\_primary\_term" : 2

}

* Adding bulk record into Elastic search:

$ curl --header 'content-type:application/JSON' -XPUT 'localhost:9200/test/external/\_bulk' -d'

{"index":{"\_id":"2"}}

{"name":"Paresh", "age":34}

{"index":{"\_id":"3"}}

{"name":"Minal","age":35}

'

{"took":61,"errors":false,"items":[{"index":{"\_index":"test","\_type":"external","\_id":"2","\_version":1,"result":"created","\_shards":{"total":2,"successful":1,"failed":0},"\_seq\_no":7,"\_primary\_

term":2,"status":201}},{"index":{"\_index":"test","\_type":"external","\_id":"3","\_version":2,"result":"updated","\_shards":{"total":2,"successful":1,"failed":0},"\_seq\_no":8,"\_primary\_term":2,"sta

tus":200}}]}

* Multiple operation can be done :

$ curl --header 'content-type:application/JSON' -XPUT 'localhost:9200/bank/external/\_bulk' -d'

> {"index":{"\_id":"4"}}

> {"name":"Kunal","age":54}

> {"update":{"\_id":"1"}}

> {"doc": { "name": "John Doe becomes Jane Doe" } }

'> {"delete":{"\_id":"2"}}

> '

{"took":804,"errors":true,"items":[{"index":{"\_index":"bank","\_type":"external","\_id":"4","\_version":1,"result":"created","\_shards":{"total":2,"successful":1,"failed":0},"\_seq\_no":0,"\_primary\_

term":1,"status":201}},{"update":{"\_index":"bank","\_type":"external","\_id":"1","status":404,"error":{"type":"document\_missing\_exception","reason":"[external][1]: document missing","index\_uuid"

:"ttLcWsbZQgOM0ucoXdxGEg","shard":"0","index":"bank"}}},{"delete":{"\_index":"bank","\_type":"external","\_id":"2","\_version":1,"result":"not\_found","\_shards":{"total":2,"successful":1,"failed":0

},"\_seq\_no":1,"\_primary\_term":1,"status":404}}]}

* Bulk insertion using json file:

$ curl --header 'content-type:application/JSON' -XPUT 'localhost:9200/account/\_bulk?pretty' --data-binary @C:\\accounts.json

**Kibana:**

**Run:**

C:\elastic\_stack\kibana-7.5.2-windows-x86\_64\bin>kibana.bat

Access from brower: <http://localhost:5601/app/kibana>

Go to dev tools option : (run the below query to find all the records of account index)

GET /account/\_search

Creating new index from exiting one:

POST \_reindex

{

"source": {

"index": "bank"

},

"dest": {

"index": "new\_bank"

}

}

POST \_reindex

{

"source":{

"index": "account",

"query": {

"term": {

"balance": 120020

}

}

},

"dest": {

"index": "find\_rec"

}

}

Salary ordering:

GET /account/\_search

{

"sort":[

{

"balance": {"order": "desc"}

}]

}

Get only one record:

GET /account/\_search

{

"size": 1,

"sort":[

{

"balance": {"order": "desc"}

}]

}

Get the record by the below source:

GET /account/\_search

{

"size": 1,

"sort":[

{

"balance": {"order": "desc"}

}

],

"\_source": ["firstname", "balance" ]

}

* Create new index with top 10 records from exiting index:

POST \_reindex

{

"size": 10,

"source": {

"index": "account",

"sort": [

{

"balance": {

"order": "desc"

}

}

],

"\_source":["firstname", "balance"]

},

"dest": {

"index": "new\_account4"

}

}

Finding the records between the range:

POST /account/\_search

{

"query": {

"range": {

"balance": {

"gte": 49355,

"lte": 49567

}

}

}

}

* Update the index with particualr constraints:

POST /new\_account4/\_update\_by\_query

{

"script": {

"source": "ctx.\_source.balance=0"

},

"query": {

"range": {

"balance": {

"gte": 40000,

"lte": 50000

}

}

}

}

Curl command from cygwin tool:

curl -XGET 'localhost:9200/account/\_search?pretty'

From browser:

[http://localhost:9200/account/\_search?pretty&q=\*](http://localhost:9200/account/_search?pretty&q=*)

From devtools :

GET account/\_search

{

"query": {"match\_all": {}}

}

From the browser Find the record based on Name:

<http://localhost:9200/account/_search?pretty&q=firstname:Amber>

**Get 2 records:**

GET account/\_search

{

"query": {"match\_all": {}},

"size": 2

}

**Get the records on pagination:**

GET account/\_search

{

"query": {"match\_all": {}},

"from": 10,

"size": 10

}

**Sorting the records:**

GET account/\_search

{

"query": {"match\_all": {}},

"sort": [

{

"firstname.keyword": {

"order": "desc"

}

}

]

}

**Find the records where the bala?**

GET account/\_search

{

"query": {

"match": {

"balance": 22776

}

}

* Match: with single word and muliple words
* Match\_phrase : for muliple words want to combine
* Match\_all: for all columns

GET account/\_search

{

"query": {

"match\_phrase": {

"address.keyword": "984 Woodside Avenue"

}

}

}

GET account/\_search

{

"query": {

"prefix": {

"firstname.keyword": {

"value": "A"

}

}

}

}

GET account/\_search

{

"query": {"match\_all": {}},

"from": 10,

"size": 10

}

GET account/\_search

{

"query": {"match\_all": {}},

"sort": [

{

"firstname.keyword": {

"order": "desc"

}

}

]

}

* Display the detilas of account holder whoes names starts with “Y” and having max balance in “y” starting names

GET account/\_search

{

"query": {

"prefix": {

"firstname.keyword": "Y"

}

},

"sort":[

{

"balance": {"order": "desc"}

}

]

}

Elastic HQ Analyasis plugin for elastic

Installation process of HQ

Download or clone the repository.

Open terminal and point to root of repository. Type: pip install -r requirements.txt

Run server with: python3 application.py. Alternatively: ./manage.py runserver

Access HQ with: <http://localhost:5000>