LAB – ELASTICSEARCH

Pre-Requisite:

1. Install JDK 1.6 or higher in your machine.
2. Install curl in your machine. If you are using windows machine download the curl from the below link:

   https://curl.haxx.se/download.html

   Copy the curl.exe into "C:\Windows" path. Now you can use curl in your command prompt itself.

3. Download Elasticsearch from below link: (I have downloaded "**elasticsearch-5.5.0**" version throughout my demo)

   https://www.elastic.co/downloads/elasticsearch

   Extract the zip. Locate the bin directory where you can see bat file list.

   a. You can open the command prompt which is locating the Elasticsearch installation directory

```
C:\Windows\System32\cmd.exe

D:\vidavid\ElasticSearch\elasticsearch-5.5.0\elasticsearch-5.5.0>dir
 Volume in drive D is Data
 Volume Serial Number is 3608-2657

 Directory of D:\vidavid\ElasticSearch\elasticsearch-5.5.0\elasticsearch-5.5.0

07/20/2017  07:24 PM    <DIR>          .
07/20/2017  07:24 PM    <DIR>          ..
07/20/2017  07:23 PM    <DIR>          bin
07/20/2017  07:24 PM    <DIR>          config
07/20/2017  07:24 PM    <DIR>          data
07/20/2017  07:23 PM    <DIR>          lib
06/30/2017  11:14 PM            11,358 LICENSE.txt
07/20/2017  07:25 PM    <DIR>          logs
07/20/2017  07:23 PM    <DIR>          modules
06/30/2017  11:20 PM           194,187 NOTICE.txt
06/30/2017  11:20 PM    <DIR>          plugins
06/30/2017  11:14 PM             9,548 README.textile
               3 File(s)        215,093 bytes
               9 Dir(s)  145,931,939,840 bytes free

D:\vidavid\ElasticSearch\elasticsearch-5.5.0\elasticsearch-5.5.0>
```

   b. Use below command to start ElasticSearch cluster.

```
.\bin\elasticsearch
```

This will start the elastic search with default cluster name, as you see in the below window: the default node name here is pBtoz7C.

You can kill the below terminal window by press **Ctrl + C**

```
Elasticsearch 5.5.0
[2017-07-27T15:55:58,850][INFO ][o.e.n.Node               ] JVM arguments [-Xms2g, -Xmx2g, -XX:+UseConcMarkSweepGC, -XX:CMSInitiati
ngOccupancyFraction=75, -XX:+UseCMSInitiatingOccupancyOnly, -XX:+DisableExplicitGC, -XX:+AlwaysPreTouch, -Xss1m, -Djava.awt.headles
s=true, -Dfile.encoding=UTF-8, -Djna.nosys=true, -Djdk.io.permissionsUseCanonicalPath=true, -Dio.netty.noUnsafe=true, -Dio.netty.no
KeySetOptimization=true, -Dio.netty.recycler.maxCapacityPerThread=0, -Dlog4j.shutdownHookEnabled=false, -Dlog4j2.disable.jmx=true,
-Dlog4j.skipJansi=true, -XX:+HeapDumpOnOutOfMemoryError, -Delasticsearch, -Des.path.home=D:\vidavid\ElasticSearch\elasticsearch-5.5
.0\elasticsearch-5.5.0]
[2017-07-27T15:56:00,613][INFO ][o.e.p.PluginsService     ] [pBtoz7C] loaded module [aggs-matrix-stats]
[2017-07-27T15:56:00,615][INFO ][o.e.p.PluginsService     ] [pBtoz7C] loaded module [ingest-common]
[2017-07-27T15:56:00,618][INFO ][o.e.p.PluginsService     ] [pBtoz7C] loaded module [lang-expression]
[2017-07-27T15:56:00,621][INFO ][o.e.p.PluginsService     ] [pBtoz7C] loaded module [lang-groovy]
[2017-07-27T15:56:00,627][INFO ][o.e.p.PluginsService     ] [pBtoz7C] loaded module [lang-mustache]
[2017-07-27T15:56:00,630][INFO ][o.e.p.PluginsService     ] [pBtoz7C] loaded module [lang-painless]
[2017-07-27T15:56:00,635][INFO ][o.e.p.PluginsService     ] [pBtoz7C] loaded module [parent-join]
[2017-07-27T15:56:00,640][INFO ][o.e.p.PluginsService     ] [pBtoz7C] loaded module [percolator]
[2017-07-27T15:56:00,644][INFO ][o.e.p.PluginsService     ] [pBtoz7C] loaded module [reindex]
[2017-07-27T15:56:00,648][INFO ][o.e.p.PluginsService     ] [pBtoz7C] loaded module [transport-netty3]
[2017-07-27T15:56:00,652][INFO ][o.e.p.PluginsService     ] [pBtoz7C] loaded module [transport-netty4]
[2017-07-27T15:56:00,661][INFO ][o.e.p.PluginsService     ] [pBtoz7C] no plugins loaded
[2017-07-27T15:56:03,262][INFO ][o.e.d.DiscoveryModule    ] [pBtoz7C] using discovery type [zen]
[2017-07-27T15:56:04,049][INFO ][o.e.n.Node               ] initialized
[2017-07-27T15:56:04,051][INFO ][o.e.n.Node               ] [pBtoz7C] starting ...
[2017-07-27T15:56:04,701][INFO ][o.e.t.TransportService   ] [pBtoz7C] publish_address {127.0.0.1:9300}, bound_addresses {127.0.0.1:
9300}, {[::1]:9300}
[2017-07-27T15:56:07,794][INFO ][o.e.c.s.ClusterService   ] [pBtoz7C] new_master {pBtoz7C}{pBtoz7CbRqCejPyceefM9A}{hHNgSlHLThSEKXNX
AOvXNQ}{127.0.0.1}{127.0.0.1:9300}, reason: zen-disco-elected-as-master ([0] nodes joined)
[2017-07-27T15:56:08,187][INFO ][o.e.g.GatewayService     ] [pBtoz7C] recovered [2] indices into cluster_state
[2017-07-27T15:56:08,291][INFO ][o.e.h.n.Netty4HttpServerTransport] [pBtoz7C] publish_address {127.0.0.1:9200}, bound_addresses {12
7.0.0.1:9200}, {[::1]:9200}
[2017-07-27T15:56:08,299][INFO ][o.e.n.Node               ] [pBtoz7C] started
[2017-07-27T15:56:10,121][INFO ][o.e.c.r.a.AllocationService] [pBtoz7C] Cluster health status changed from [RED] to [YELLOW] (reaso
n: [shards started [[products][3]] ...]).
```

4. If you want to create cluster with cluster name and node name you can use the below command:

```
.\bin\elasticsearch -Ecluster.name=capgemini_es -Enode.name=cap_node
```

5. Now you can see the below terminal with node details:

Elasticsearch 5.5.0

```
[2017-07-27T16:03:10,862][INFO ][o.e.n.Node               ] [cap_node] node name [cap_node], node ID [pBtoz7CbRqCejPyceefM9A]
[2017-07-27T16:03:10,878][INFO ][o.e.n.Node               ] [cap_node] version[5.5.0], pid[3572], build[260387d/2017-06-30T23:16:05
.735Z], OS[Windows 8.1/6.3/amd64], JVM[Oracle Corporation/Java HotSpot(TM) 64-Bit Server VM/1.8.0_45/25.45-b02]
[2017-07-27T16:03:10,878][INFO ][o.e.n.Node               ] [cap_node] JVM arguments [-Xms2g, -Xmx2g, -XX:+UseConcMarkSweepGC, -XX:
CMSInitiatingOccupancyFraction=75, -XX:+UseCMSInitiatingOccupancyOnly, -XX:+DisableExplicitGC, -XX:+AlwaysPreTouch, -Xss1m, -Djava.
awt.headless=true, -Dfile.encoding=UTF-8, -Djna.nosys=true, -Djdk.io.permissionsUseCanonicalPath=true, -Dio.netty.noUnsafe=true, -D
io.netty.noKeySetOptimization=true, -Dio.netty.recycler.maxCapacityPerThread=0, -Dlog4j.shutdownHookEnabled=false, -Dlog4j2.disable
.jmx=true, -Dlog4j.skipJansi=true, -XX:+HeapDumpOnOutOfMemoryError, -Delasticsearch, -Des.path.home=D:\vidavid\ElasticSearch\elasti
csearch-5.5.0\elasticsearch-5.5.0]
[2017-07-27T16:03:11,972][INFO ][o.e.p.PluginsService     ] [cap_node] loaded module [aggs-matrix-stats]
[2017-07-27T16:03:11,974][INFO ][o.e.p.PluginsService     ] [cap_node] loaded module [ingest-common]
[2017-07-27T16:03:11,977][INFO ][o.e.p.PluginsService     ] [cap_node] loaded module [lang-expression]
[2017-07-27T16:03:11,979][INFO ][o.e.p.PluginsService     ] [cap_node] loaded module [lang-groovy]
[2017-07-27T16:03:11,981][INFO ][o.e.p.PluginsService     ] [cap_node] loaded module [lang-mustache]
[2017-07-27T16:03:11,984][INFO ][o.e.p.PluginsService     ] [cap_node] loaded module [lang-painless]
[2017-07-27T16:03:11,986][INFO ][o.e.p.PluginsService     ] [cap_node] loaded module [parent-join]
[2017-07-27T16:03:11,988][INFO ][o.e.p.PluginsService     ] [cap_node] loaded module [percolator]
[2017-07-27T16:03:11,991][INFO ][o.e.p.PluginsService     ] [cap_node] loaded module [reindex]
[2017-07-27T16:03:11,993][INFO ][o.e.p.PluginsService     ] [cap_node] loaded module [transport-netty3]
[2017-07-27T16:03:11,997][INFO ][o.e.p.PluginsService     ] [cap_node] loaded module [transport-netty4]
[2017-07-27T16:03:12,001][INFO ][o.e.p.PluginsService     ] [cap_node] no plugins loaded
[2017-07-27T16:03:13,914][INFO ][o.e.d.DiscoveryModule    ] [cap_node] using discovery type [zen]
[2017-07-27T16:03:14,720][INFO ][o.e.n.Node               ] [cap_node] initialized
[2017-07-27T16:03:14,722][INFO ][o.e.n.Node               ] [cap_node] starting ...
[2017-07-27T16:03:15,235][INFO ][o.e.t.TransportService   ] [cap_node] publish_address {127.0.0.1:9300}, bound_addresses {127.0.0.1
:9300}, {[::1]:9300}
[2017-07-27T16:03:18,289][INFO ][o.e.c.s.ClusterService   ] [cap_node] new_master {cap_node}{pBtoz7CbRqCejPyceefM9A}{YvfpC0b1RQC5LH
r8R1cKJg}{127.0.0.1}{127.0.0.1:9300}, reason: zen-disco-elected-as-master ([0] nodes joined)
[2017-07-27T16:03:18,644][INFO ][o.e.h.n.Netty4HttpServerTransport] [cap_node] publish_address {127.0.0.1:9200}, bound_addresses {1
27.0.0.1:9200}, {[::1]:9200}
[2017-07-27T16:03:18,644][INFO ][o.e.n.Node               ] [cap_node] started
[2017-07-27T16:03:18,673][INFO ][o.e.g.GatewayService     ] [cap_node] recovered [2] indices into cluster_state
```

Demo: Health Check Up

1. Once the cluster is up and running by default it will take 9200 port, you can use below command to check the cluster details:

```
curl http://localhost:9200
```

```
Command Prompt

C:\Users\vidavid>curl http://localhost:9200
{
  "name" : "cap_node",
  "cluster_name" : "capgemini_es",
  "cluster_uuid" : "RksOwRIGQmKZPxLQeC6kKA",
  "version" : {
    "number" : "5.5.0",
    "build_hash" : "260387d",
    "build_date" : "2017-06-30T23:16:05.735Z",
    "build_snapshot" : false,
    "lucene_version" : "6.6.0"
  },
  "tagline" : "You Know, for Search"
}

C:\Users\vidavid>
```

2. To know the **health** details use the below command:

localhost:9200/_cat/health?v&pretty

```
localhost:9200/_cat/health?v&pretty

epoch      timestamp cluster      status node.total node.data shards pri relo init unassign pending_tasks max_task_wait_time active_shards_percent
1501152522 16:18:42  capgemini_es yellow          1         1     10  10    0    0       10             0                  -                 50.0%
```

3. You can also try with curl in your command prompt:

```
curl "localhost:9200/_cat/health?v&pretty"
```

```
Command Prompt
C:\Users\vidavid>curl "localhost:9200/_cat/health?v&pretty"
epoch      timestamp cluster      status node.total node.data shards pri relo init unassign pending_tasks max_task_wait_time activ
e_shards_percent
1501152619 16:20:19  capgemini_es yellow          1         1     10  10    0    0       10             0                  -
           50.0%
```

Note : I will use curl in my command prompt for REST API request

4. You can use below command to check node details:

```
curl "localhost:9200/_cat/nodes?v&pretty"
```

Demo: List All Indices:

1. To list all the available indices in your cluster use the below command:

```
curl -XGET "localhost:9200/_cat/indices?v&pretty"
```

Demo: Create an index:

2. **To create new index use the below command:**

```
curl -XPUT "localhost:9200/products?&pretty"
curl -XGET "localhost:9200/_cat/indices?v&pretty"
```

```
curl -XPUT "localhost:9200/customers?&pretty"
curl -XPUT "localhost:9200/orders?&pretty"

curl -XGET "localhost:9200/_cat/indices?v&pretty"
```

By using the above command we have created products, customers and orders indices.

Demo: Index and Query a document

1. You can create/update document by using below command:

```
curl -XPUT 'localhost:9200/products/mobiles/1?pretty' -d'
{
  "name": "iPhone 7",
  "camera": "12MP",
  "storage": "256GB",
  "display": "4.7inch",
  "battery": "1,960mAh",
  "reviews": ["Incredibly happy after having used it for one week", "Best iPhone so far", "Very expensive, stick to Android"]
}
```

```
'
```

Note: Single quote will not be parsed by your windows. Hence we have to use double quotes(''). For convenience I have used .json file to provide document.

```
curl -XPUT "localhost:9200/products/mobiles/1?pretty" -d @mobile.json
curl -XPUT "localhost:9200/products/mobiles/2?pretty" -d @mobile1.json
curl -XPUT "localhost:9200/products/mobiles/3?pretty" -d @mobile2.json
curl -XPUT "localhost:9200/products/laptops/1?pretty" -d @laptop.json
curl -XPUT "localhost:9200/products/laptops/2?pretty" -d @laptop1.json
curl -XPOST "localhost:9200/products/mobiles?pretty" -d @mobiles.json
curl -XGET "localhost:9200/_cat/indices?v&pretty"
```

Demo: Fetching whole and partial documents

```
curl -XGET "localhost:9200/products/mobiles/1?pretty"
curl -XGET "localhost:9200/products/laptops/1?pretty"
curl -XGET "localhost:9200/products/laptops/10?pretty"
```

Partial documents

```
curl -XGET "localhost:9200/products/mobiles/1?pretty&_source=false"
curl -XGET "localhost:9200/products/mobiles/1?pretty&_source=name,reviews"
curl -XGET "localhost:9200/products/mobiles/1?pretty&_source=name,reviews,storage"
curl -XGET "localhost:9200/products/laptops/1?pretty&_source=name,RAM,storage"
```

Demo: Updating whole and partial documents

```
curl -XGET "localhost:9200/products/mobiles/3?pretty"
curl -XPUT "localhost:9200/products/mobiles/3?pretty" -d @mobiles3.json
```

Updates using the _update API with "doc"

```
curl -XGET "localhost:9200/products/mobiles/2?pretty"
curl -XPOST "localhost:9200/products/mobiles/2/_update?pretty" -d @doc.json
curl -XGET "localhost:9200/products/mobiles/2?pretty"
curl -XPOST "localhost:9200/products/mobiles/2/_update?pretty" -d @mobile_update.json
```

Updates using the _update API with "script"

```
curl -XPUT "localhost:9200/products/shoes/1?pretty" -d @shoes1.json
curl -XPUT "localhost:9200/products/shoes/2?pretty" -d @shoes2.json
curl -XGET "localhost:9200/products/shoes/1?pretty"
curl -XPOST "localhost:9200/products/shoes/1/_update?pretty" -d @shoes_update.json
curl -XGET "localhost:9200/products/shoes/1?pretty"
curl -XGET "localhost:9200/products/shoes/2?pretty"
curl -XPOST "localhost:9200/products/shoes/2/_update?pretty" -d @shoes_update1.json
curl -XGET "localhost:9200/products/shoes/2?pretty"
```

Demo: Deleting an index

```
curl -XDELETE "localhost:9200/products/mobiles/2?pretty"
curl -XGET "localhost:9200/products/mobiles/2?pretty"
curl -i -XHEAD "localhost:9200/products/mobiles/2?pretty"
curl -i -XHEAD "localhost:9200/products/mobiles/1?pretty"
curl -XGET "localhost:9200/_cat/indices?v&pretty"
curl -XDELETE "localhost:9200/customerss?pretty"
curl -XGET "localhost:9200/_cat/indices?v&pretty"
```

Demo: Bulk indexing documents

Multi-get

```
curl -XGET "localhost:9200/_mget?pretty" -d @get.json
curl -XGET "localhost:9200/products/_mget?pretty" -d @get1.json
curl -XGET "localhost:9200/products/laptops/_mget?pretty" -d @get2.json
```

Index multiple documents

```
curl -XPOST "localhost:9200/_bulk?pretty" --data-binary @bulk.json

curl -XPOST "localhost:9200/products/_bulk?pretty" --data-binary @bulk.json

curl -XPOST "localhost:9200/products/shoes/_bulk?pretty" --data-binary @shoes_bulk.json
```

Auto-generate ids

```
curl -XPOST "localhost:9200/products/shoes/_bulk?pretty" --data-binary @autoid.json
```

(or)

```
curl -XPOST "localhost:9200/products/shoes/_bulk?pretty" -d"

{ "index" : {} }

{ "name": "Puma","size": 9,"color": "black" }

{ "index" : {} }

{ "name": "New Balance","size": 8,"color": "black" }

"
```

Note: this may not work in windows, you can go with above command.

Bulk operations in one go (paste these one operation at a time)

```
curl -XPOST "localhost:9200/products/shoes/_bulk?pretty" -H "Content-Type: application/json" -d"

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

{ "name": "Puma","size": 9,"color": "black" }

{ "index" : {"_id" : "4" } }

{ "name": "New Balance","size": 8,"color": "black" }

{ "delete" : {"_id" : "2" } }

{ "create" : {"_id" : "5" } }

{ "name": "Nike Power","size": 12,"color": "black" }

{ "update" : {"_id" : "1"} }

{ "doc" : {"color" : "orange"} }
```

```
"
```

Demo: Bulk indexing documents from a JSON file

Create customers.json

```
curl -H "Content-Type: application/x-ndjson" -XPOST
"localhost:9200/customers/personal/_bulk?pretty&refresh" --data-binary @"customers.json"

curl -XGET "localhost:9200/_cat/indices?v&pretty"
```

Searching and Analysis

1. Open the below URL to generate bulk amount of JSON document.
   www.json-generator.com

   Go to: http://www.json-generator.com/

```
[
  '{{repeat(1000, 1000)}}',
  {
    name: '{{firstName()}} {{surname()}}',
    age: '{{integer(18, 75)}}',
    gender: '{{gender()}}',
    email: '{{email()}}',
    phone: '+1 {{phone()}}',
    street: '{{integer(100, 999)}} {{street()}}',
    city: '{{city()}}',
    state: '{{state()}}, {{integer(100, 10000)}}'
  }
]
```

   Download and save as **customers_full.json**

   Open customers_full.json in sublimetext

   - Remove the array brackets
   - Find-Replace },{ with }\n{ in the regex mode on sublime text
   - Find-Replace {"name" with {"index" : {}}\n{"name" in the regex model on sublime text

   Now the file is in a format that can be parsed by elastic search

```
curl -XGET "localhost:9200/_cat/indices?v&pretty"
curl -XDELETE "localhost:9200/customers?pretty"
curl -XGET "localhost:9200/_cat/indices?v&pretty"
```

   Re-create the customers index

```
curl -H "Content-Type: application/x-ndjson" -XPOST
"localhost:9200/customers/personal/_bulk?pretty&refresh" --data-binary
@"customers_full.json"

curl -XGET "localhost:9200/_cat/indices?v&pretty"
```

Searching using the query parameter

To query the document we can try this by using curl or directly hit your browser. I prefer to go with browser for better formatting:

```
curl -XGET "localhost:9200/customers/_search?q=wyoming&pretty"

localhost:9200/customers/_search?q=wyoming&pretty

localhost:9200/customers/_search?q=wyoming&sort=age:desc&pretty

localhost:9200/customers/_search?q=state:kentucky&sort=age:asc&pretty

localhost:9200/customers/_search?q=state:kentucky&from=10&size=2&pretty

localhost:9200/customers/_search?q=state:kentucky&explain&pretty


localhost:9200/products,customers/_search?pretty

localhost:9200/products/shoes,mobiles/_search?pretty
```

Searching using the request body

```
curl -XGET "localhost:9200/products/_search?pretty" -d @reqBody_query1.json

curl -XGET 'localhost:9200/products/_search?pretty' -d'

{

  "query": { "match_all": {} }

}

'

curl -XGET "localhost:9200/products/_search?pretty" -d @reqBody_query2.json

curl -XGET 'localhost:9200/products/_search?pretty' -d'

{
```

```
  "query": {}
}
'
curl -XGET "localhost:9200/products/_search?pretty" -d @reqBody_query3.json
curl -XGET 'localhost:9200/products/_search?pretty' -d'
{
  "query": { "match_all": {} },
  "size": 3
}
'
curl -XGET "localhost:9200/products/_search?pretty" -d @reqBody_query4.json
curl -XGET 'localhost:9200/products/_search?pretty' -d'
{
  "query": { "match_all": {} },
  "from": 5,
  "size": 3
}
'
curl -XGET "localhost:9200/customers/_search?pretty" -d @reqBody_query5.json
curl -XGET 'localhost:9200/customers/_search?pretty' -d'
{
  "query": { "match_all": {} },
  "sort": { "age": { "order": "desc" } },
  "size": 20
}
'
```

Query terms and source filtering

Term Filtering:

```
curl -XGET 'localhost:9200/customers/_search?pretty' -d'
{
  "query" : {
    "term" : { "name" : "gates" }
  }
}
'
curl -XGET "localhost:9200/customers/_search?pretty" -d@term.json

curl -XGET 'localhost:9200/customers/_search?pretty' -d'
{
  "query" : {
    "term" : { "street" : "chestnut" }
  }
}
'
curl -XGET "localhost:9200/customers/_search?pretty" -d@term_query.json
```

Source Filtering:

```
curl -XGET 'localhost:9200/customers/_search?pretty' -d'
{
  "_source" : false,
  "query" : {
    "term" : { "street" : "chestnut" }
  }
}
```

```
'
curl -XGET "localhost:9200/customers/_search?pretty" -d@source.json

curl -XGET 'localhost:9200/customers/_search?pretty' -d'
{
   "_source" : "st*",
   "query" : {
      "term" : { "state" : "washington" }
   }
}
'
curl -XGET "localhost:9200/customers/_search?pretty" -d@source_field_pat.json

curl -XGET 'localhost:9200/customers/_search?pretty' -d'
{
   "_source" : ["st*", "*n*"],
   "query" : {
      "term" : { "state" : "washington" }
   }
}
'
curl -XGET "localhost:9200/customers/_search?pretty" -d@source_field_mul_pat.json

curl -XGET 'localhost:9200/customers/_search?pretty' -d'
{
    "_source": {
      "includes": ["st*", "*n*"],
```

```
      "excludes": [ "*der" ]
    },
    "query" : {
      "term" : { "state" : "washington" }
    }
}
'
curl -XGET "localhost:9200/customers/_search?pretty" -d@source_field_inc_exc.json
```

Full Text Queries:

The match keyword

```
curl -XGET 'localhost:9200/customers/_search?pretty' -d'
{
    "query": {
      "match" : {
        "name" : "webster"
      }
    }
}
'
curl -XGET "localhost:9200/customers/_search?pretty" -d@match.json


curl -XGET 'localhost:9200/customers/_search?pretty' -d'
{
    "query": {
      "match" : {
          "name" : {
```

```
                "query" : "frank norris",
                "operator" : "or"
            }
        }
    }
}
'
curl -XGET "localhost:9200/customers/_search?pretty" -d@match_phrase.json



curl -XGET 'localhost:9200/customers/_search?pretty' -d'
{
    "query": {
        "match" : {
            "street" : "tompkins place"
        }
    }
}
'
curl -XGET "localhost:9200/customers/_search?pretty" -d@match_phrase_no_operator.json
```

The match_phrase keyword

```
curl -XGET 'localhost:9200/customers/_search?pretty' -d'
{
    "query": {
        "match_phrase" : {
            "street" : "tompkins place"
```

```
      }
    }
}
'
curl -XGET "localhost:9200/customers/_search?pretty" -d@match_phrase1.json

curl -XGET 'localhost:9200/customers/_search?pretty' -d'
{
   "query": {
      "match_phrase" : {
         "state" : "puerto rico"
      }
   }
}
'
curl -XGET "localhost:9200/customers/_search?pretty" -d@match_phrase2.json
```

The match_phrase_prefix

```
curl -XGET 'localhost:9200/customers/_search?pretty' -d'
{
   "query": {
      "match_phrase_prefix" : {
         "name" : "ma"
      }
   }
}
```

```
'
curl -XGET "localhost:9200/customers/_search?pretty" -d@match_phrase_prefix.json

curl -XGET 'localhost:9200/customers/_search?pretty' -d'
{
  "query": {
    "match_phrase_prefix" : {
      "street" : "clymer st"
    }
  }
}
'
curl -XGET "localhost:9200/customers/_search?pretty" -d@match_phrase_prefix1.json
```

Common terms queries

```
curl -XGET 'localhost:9200/products/_search?pretty' -d'
{
  "query": {
    "common": {
      "reviews": {
        "query": "this is great",
         "cutoff_frequency": 0.001
      }
    }
  }
}
```

```
'

curl -XGET "localhost:9200/products/_search?pretty" -d@query_term.json
```

Boolean compound queries

Must

```
curl -XGET 'localhost:9200/customers/_search?pretty'  -d'
{
  "query": {
    "bool": {
      "must": [
        { "match": { "street": "ditmas" } },
        { "match": { "street": "avenue" } }
      ]
    }
  }
}
'

curl -XGET "localhost:9200/customers/_search?pretty"  -d@boolean_must.json
```

should

```
curl -XGET 'localhost:9200/customers/_search?pretty'  -d'
{
  "query": {
    "bool": {
      "should": [
        { "match": { "street": "ditmas" } },
        { "match": { "street": "street" } }
      ]
```

```
  }
 }
}
'

curl -XGET "localhost:9200/customers/_search?pretty"  -d@boolean_should.json
```

must_not

```
curl -XGET 'localhost:9200/customers/_search?pretty'  -d'
{
  "query": {
    "bool": {
      "must_not": [
        { "match": { "state": "california texas" } },
        { "match": { "street": "lane street" } }
      ]
    }
  }
}
'

curl -XGET "localhost:9200/customers/_search?pretty"  -d@boolean_must_not.json
```

Term queries

```
curl -XGET 'localhost:9200/customers/_search?pretty' -d'
{
  "query": {
    "bool": {
      "should": [
```

```
      {
        "term": {
          "state": {
            "value": "idaho"
          }
        }
      },
      {
        "term": {
          "state": {
            "value": "california"
          }
        }
      }
    ]
  }
 }
}
'
curl -XGET "localhost:9200/customers/_search?pretty" -d@term_query_1.json



curl -XGET 'localhost:9200/customers/_search?pretty' -d'
{
  "query": {
    "bool": {
      "should": [
```

```
      {
        "term": {
          "state": {
            "value": "idaho",
            "boost": 2.0
          }
        }
      },
      {
        "term": {
          "state": {
            "value": "california"
          }
        }
      }
    ]
  }
 }
}
'
curl -XGET "localhost:9200/customers/_search?pretty" -d@term_query_boost.json
```

Filters

```
curl -XGET 'localhost:9200/customers/_search?pretty' -d'
{
  "query": {
```

```
   "bool": {
    "must": { "match_all": {} },
    "filter": {
     "range": {
      "age": {
       "gte": 20,
       "lte": 30
      }
     }
    }
   }
  }
}
'

curl -XGET "localhost:9200/customers/_search?pretty" -d@filters.json
```

Using filters along with search terms

```
curl -XGET 'localhost:9200/customers/_search?pretty' -H 'Content-Type: application/json' -d'
{
  "query": {
   "bool": {
    "must":
     { "match": {
        "state":   "alabama"
     }
    },
```

```
    "filter": [
      { "term":  { "gender": "female" }},
      { "range": { "age": { "gte": "50" }}}
    ]
  }
 }
}
'

curl -XGET "localhost:9200/customers/_search?pretty" -H "Content-Type: application/json" -
d@ilters_with_search.json
```

Metrics aggregations

Average

```
curl -XPOST 'localhost:9200/customers/_search?&pretty' -d'
{
  "size" : 0,
  "aggs" : {
     "avg_age" : {
        "avg" : {
           "field" : "age"
        }
     }
  }
}
'

curl -XPOST "localhost:9200/customers/_search?&pretty" -d@agg_avg.json
```

```
curl -XPOST 'localhost:9200/customers/_search?&pretty' -d'
{
  "size" : 0,
  "aggregations" : {
    "avg_age" : {
      "avg" : {
        "field" : "age"
      }
    }
  }
}
'
curl -XPOST "localhost:9200/customers/_search?&pretty" -d@aggs_avg.json

curl -XPOST 'localhost:9200/customers/_search?&pretty' -d'
{
  "size" : 1,
  "aggs" : {
    "avg_age" : {
      "avg" : {
        "field" : "age"
      }
    }
  }
}
'
curl -XPOST "localhost:9200/customers/_search?&pretty" -d@aggs_avg_size.json
```

Average with some search terms

```
curl -XPOST 'localhost:9200/customers/_search?&pretty' -d'
{
   "size" : 0,
    "query" : {
       "bool" : {
           "filter" : {
              "match" : { "state" : "minnesota" }
           }
        }
    },
   "aggs" : {
      "avg_age" : {
         "avg" : {
            "field" : "age"
         }
      }
   }
}
'
curl -XPOST "localhost:9200/customers/_search?&pretty" -d@aggs_avg_search_terms.json
```

Stats

```
curl -XPOST 'localhost:9200/customers/_search?&pretty' -d'
{
  "size" : 0,
  "aggs" : {
    "age_stats" : {
      "stats" : {
        "field" : "age"
      }
    }
  }
}
'

curl -XPOST "localhost:9200/customers/_search?&pretty" -d@stats.json
```

Aggregations might need fielddata, e.g. cardinality aggregation Cardinality

```
curl -XPOST 'localhost:9200/customers/_search?&pretty' -d'
{
  "size" : 0,
  "aggs" : {
    "age_count" : {
      "cardinality" : {
        "field" : "age"
      }
    }
```

```
  }
}
'
curl -XPOST "localhost:9200/customers/_search?&pretty" -d@card_age.json


curl -XPOST 'localhost:9200/customers/_search?&pretty' -d'
{
  "size" : 0,
   "aggs" : {
      "gender_count" : {
          "cardinality" : {
             "field" : "gender"
          }
      }
   }
}
'
curl -XPOST "localhost:9200/customers/_search?&pretty" -d@card_gender.json
```

This will throw the exception, use fielddata and mapping to enable text cardinality.

```
curl -XPUT 'localhost:9200/customers/_mapping/personal?pretty' -d'
{
  "properties": {
   "gender": {
     "type":    "text",
```

```
    "fielddata": true
   }
  }
}
'
curl -XPUT "localhost:9200/customers/_mapping/personal?pretty" -d@card_gender_true.json
```

Now it will return below:

```
{
  "acknowledged" : true
}
```

Now re-run the original request

```
curl -XPOST 'localhost:9200/customers/_search?&pretty' -d'
{
  "size" : 0,
   "aggs" : {
      "gender_count" : {
          "cardinality" : {
             "field" : "gender"
          }
      }
   }
}
'
curl -XPOST "localhost:9200/customers/_search?&pretty" -d@card_gender_accepted.json
```

Bucketing aggregation by field values

Term aggregations

```
curl -XPOST 'localhost:9200/customers/_search?&pretty' -d'
{
  "size" : 0,
  "aggs" : {
     "gender_bucket" : {
        "terms" : {
           "field" : "gender"
        }
     }
  }
}
'
curl -XPOST "localhost:9200/customers/_search?&pretty" -d@agg_bucket.json


curl -XPOST 'localhost:9200/customers/_search?&pretty' -d'
{
  "size" : 0,
  "aggs" : {
     "age_bucket" : {
        "terms" : {
           "field" : "age"
        }
     }
  }
}
```

```
'

curl -XPOST "localhost:9200/customers/_search?&pretty" -d@agg_bucket_age.json
```

Range aggregation

```
curl -XPOST 'localhost:9200/customers/_search?&pretty' -d'
{
  "size" : 0,
  "aggs" : {
    "age_ranges" : {
      "range" : {
        "field" : "age",
        "ranges" : [
          { "to" : 30 },
          { "from" : 30, "to" : 40 },
          { "from" : 40, "to" : 55 },
          { "from" : 55 }
        ]
      }
    }
  }
}
'

curl -XPOST "localhost:9200/customers/_search?&pretty" -d@agg_bucket_age_range.json


curl -XPOST 'localhost:9200/customers/_search?&pretty' -d'
{
  "size" : 0,
```

```
  "aggs" : {
    "age_ranges" : {
      "range" : {
        "field" : "age",
        "keyed" : true,
        "ranges" : [
          { "to" : 30 },
          { "from" : 30, "to" : 40 },
          { "from" : 40, "to" : 55 },
          { "from" : 55 }
        ]
      }
    }
  }
}
'
curl -XPOST "localhost:9200/customers/_search?&pretty" -d@agg_bucket_age_range_key.json


curl -XPOST 'localhost:9200/customers/_search?&pretty' -d'
{
  "size" : 0,
  "aggs" : {
    "age_ranges" : {
      "range" : {
        "field" : "age",
        "keyed" : true,
```

```
                "ranges" : [
                  { "key": "young", "to" : 30 },
                  { "key": "quarter-aged", "from" : 30, "to" : 40 },
                  { "key": "middle-aged", "from" : 40, "to" : 55 },
                  { "key": "senior", "from" : 55 }
                ]
              }
            }
          }
        }
        '

        curl -XPOST "localhost:9200/customers/_search?&pretty" -
d@agg_bucket_age_range_key_name.json
```

Nesting aggregations

2 level nesting

```
curl -XPOST 'localhost:9200/customers/_search?&pretty' -d'
{
  "size" : 0,
  "aggs" : {
     "gender_bucket" : {
        "terms" : {
           "field" : "gender"
        },
        "aggs": {
           "average_age": {
              "avg": {
                 "field": "age"
              }
           }
        }
     }
  }
}
'
curl -XPOST "localhost:9200/customers/_search?&pretty" -d@nested_aggs.json
```

3 level nesting

```
curl -XPOST 'localhost:9200/customers/_search?&pretty' -d'
{
  "size" : 0,
  "aggs" : {
     "gender_bucket" : {
        "terms" : {
           "field" : "gender"
        },
        "aggs" : {
           "age_ranges" : {
              "range" : {
                 "field" : "age",
                 "keyed" : true,
                 "ranges" : [
                    { "key": "young", "to" : 30 },
                    { "key": "middle-aged", "from" : 30, "to" : 55 },
                    { "key": "senior", "from" : 55 }
                 ]
              },
              "aggs": {
                 "average_age": {
                    "avg": {
                       "field": "age"
                    }
                 }
              }
```

```
            }
          }
        }
      }
    }
'

curl -XPOST "localhost:9200/customers/_search?&pretty" -d@3_level_nested_aggs.json
```

Filter and filters aggregations

```
curl -XPOST 'localhost:9200/customers/_search?size=0&pretty' -d'
{
  "aggs" : {
    "state" : {
      "filter" : { "term": { "state": "texas" } },
      "aggs" : {
        "avg_age" : { "avg" : { "field" : "age" } }
      }
    }
  }
}
'
curl -XPOST "localhost:9200/customers/_search?size=0&pretty" -d@aggs_bucket_filter.json
```

Multiple Filters

```
curl -XGET 'localhost:9200/customers/_search?pretty'  -d'
{
 "size": 0,
 "aggs" : {
  "states" : {
   "filters" : {
    "filters" : {
     "washington" :   { "match" : { "state" : "washington"   }},
     "north carolina" :   { "match" : { "state" : "north carolina"   }},
     "south dakota" : { "match" : { "state" : "south dakota" }}
    }
```

```
    }
   }
  }
 }
'

curl -XGET "localhost:9200/customers/_search?pretty"  -d@aggs_bucket_filters.json
```

(Anonymous filters, returned in the same order as the original filter specification)

```
curl -XGET 'localhost:9200/customers/_search?pretty'  -d'

{
  "size": 0,
  "aggs" : {
    "states" : {
      "filters" : {
        "filters" : [
          { "match" : { "state" : "washington"   }},
          { "match" : { "state" : "north carolina"   }},
          { "match" : { "state" : "south dakota" }}
        ]
      }
    }
  }
}
'

curl -XGET "localhost:9200/customers/_search?pretty"  -d@anonymous_filters.json
```

Other bucket

```
curl -XGET 'localhost:9200/customers/_search?pretty'  -d'
{
  "size": 0,
  "aggs" : {
    "states" : {
      "filters" : {
        "other_bucket_key": "other_states",
        "filters" : {
          "washington" :   { "match" : { "state" : "washington"   }},
          "north carolina" :   { "match" : { "state" : "north carolina"   }},
          "south dakota" : { "match" : { "state" : "south dakota" }}
        }
      }
    }
  }
}
'
curl -XGET "localhost:9200/customers/_search?pretty"  -d@aggs_bucket_filters_keys.json
```