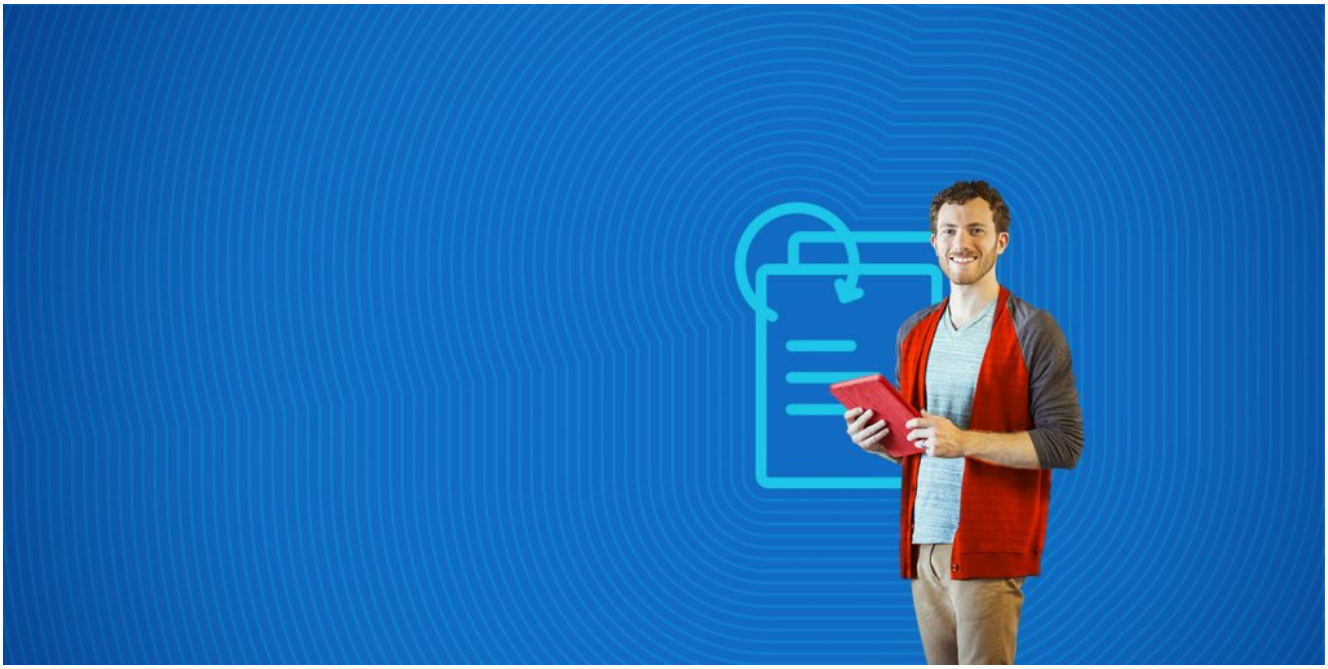

ElasticSearch Commands Cheat Sheet

Here we show some of the most common ElasticSearch commands using curl. ElasticSearch is sometimes complicated. So here we make it simple.

delete index

Below the index is named **samples**.

```
curl -X DELETE 'http://localhost:9200/samples'
```

list all indexes

```
curl -X GET 'http://localhost:9200/_cat/indices?v'
```

list all docs in index

```
curl -X GET 'http://localhost:9200/sample/_search'
```

query using URL parameters

Here we use Lucene query format to write q=school:Harvard.

```
curl -X GET http://localhost:9200/samples/_search?q=school:Harvard
```

Query with JSON aka Elasticsearch Query DSL

You can query using parameters on the URL. But you can also use JSON, as shown in the next example. JSON would be easier to read and debug when you have a complex query than one giant string of URL parameters.

```
curl -XGET --header 'Content-Type: application/json' http://localhost:9200/samples/_search -d '{
      "query" : {
        "match" : { "school": "Harvard" }
    }
}'
```

list index mapping

All Elasticsearch fields are indexes. So this lists all fields and their types in an index.

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

Add Data

```
curl -XPUT --header 'Content-Type: application/json' http://localhost:9200/samples/_doc/1 -d '{
   "school" : "Harvard"
}'
```

Update Doc

Here is how to add fields to an existing document. First we create a new one. Then we update it.

```
curl -XPUT --header 'Content-Type: application/json' http://localhost:9200/samples/_doc/2 -d '
{
    "school": "Clemson"
}'

curl -XPOST --header 'Content-Type: application/json' http://localhost:9200/samples/_doc/2/_update -d '{
"doc" : {
               "students": 50000}
}'
```

backup index

```
curl -XPOST --header 'Content-Type: application/json' http://localhost:9200/_reindex -d '{
  "source": {
    "index": "samples"
  },
  "dest": {
    "index": "samples_backup"
  }
}'
```

Bulk load data in JSON format

```
export pwd="elastic:"

curl --user $pwd  -H 'Content-Type: application/x-ndjson' -
```

```
XPOST
'https://58571402f5464923883e7be42a037917.eu-central-1.aws.cloud.es.io:9243/0/_bulk?pretty' --data-binary @<file>
```

Show cluster health

```
curl --user $pwd  -H 'Content-Type: application/json' -XGET https://58571402f5464923883e7be42a037917.eu-central-1.aws.cloud.es.io:9243/_cluster/health?pretty
```

Aggregation and Bucket Aggregation

For an nginx web server this produces web hit counts by user city:

```
curl -XGET --user $pwd --header 'Content-Type: application/json' https://58571402f5464923883e7be42a037917.eu-central-1.aws.cloud.es.io:9243/logstash/_search?pretty -d '{
        "aggs": {
            "cityName": {
                  "terms": {
                   "field": "geoip.city_name.keyword",
                             "size": 50

        }
   }
  }
}
'
```

This expands that to product response code count by city in an nginx web server log

```
curl -XGET --user $pwd --header 'Content-Type: application/json' https://58571402f5464923883e7be42a037917.eu-central-1.aws.cloud.es.io:9243/logstash/_search?pretty -d '{
        "aggs": {
          "city": {
                "terms": {
```

```
                        "field": "geoip.city_name.keyword"
                },
        "aggs": {
          "responses": {
                "terms": {
                    "field": "response"
                 }
           }
         }
      },
      "responses": {
                "terms": {
                    "field": "response"
                 }
        }
   }
}'
```

Using ElasticSearch with Basic Authentication

If you have turned on security with ElasticSearch then you need to supply the user and password like shown below to every curl command:

```
curl -X GET 'http://localhost:9200/_cat/indices?v' -u elastic:(password)
```

Pretty Print

Add ?pretty=true to any search to pretty print the JSON. Like this:

```
curl -X GET 'http://localhost:9200/(index)/_search'?pretty=true
```