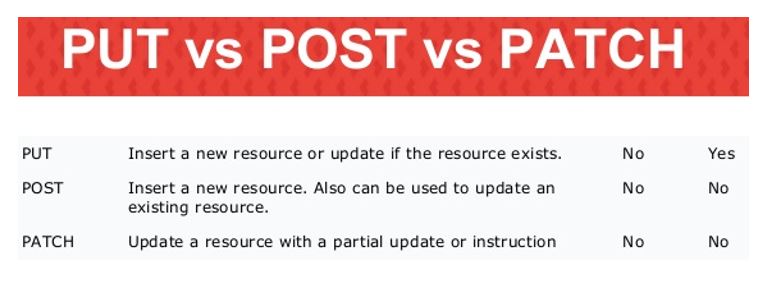
**ELASTIC SEARCH**



* **bool** indicates we are using boolean
* **must** is for **AND**
* **should** is for **OR**
  1. **Some types of FILTERS**

**term**: filter by exact values

{“term”: {“field”: value}}

**terms**: match if any exact values in a list match

{“terms”: {“field”: [“value”, “value1”] } }

**range**: Find numbers or dates in a given range (gt, gte, lt, lte)

{“range”: {“field”: {“gte”: value}}}

**exists**: Find documents where a field exists

{“exists”: {“field”: “doc\_name”}}

**missing**: Find documents where a field is missing

{“missing”: {“field”: “doc\_name”}}

**bool**: Combine filters with Boolean logic (must, must\_not, should)

**2. Some Types of Queries**

**match\_all**: returns all documents and is the default. Normally used with a filter.

{“match\_all”: {}}

**match**: searches analyzed results, such as full text search.

{“match”: {“filed”: “value”}}

**multi\_match**: run the same query on multiple fields.

{“multi\_match”: {“query”: “value”, “fields”: [“title”, “synopsis” ] } }

**bool**: Works like a bool filter, but results are scored by relevance.

**SYNTAX:**

**queries** are wrapped in a “**query**”: { } block,

**filters** are wrapped in a “**filter**”: { } block.

**Phrase Matching:**

It will must find all terms, in the right order

{

"query": {

"match\_phrase": {

"title": "star wars"

}

}

}

**Sloap:**

the slop represents how far you’re willing to let a term move to satisfy a phrase (in either direction!)

another example: “HARSH P ATEL” would match “HARSH PATEL” with a slop of 1

Eg. {

"query": {

"match\_phrase": {

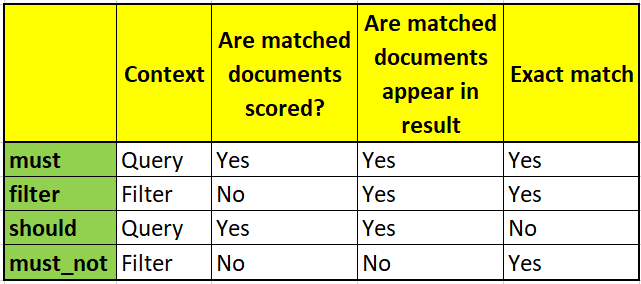
"title": {"query": "HARSH PATEL", "slop": 1}

}

}

}

**Difference:**



**Elasticsearch Boolean Clauses**

The four boolean clauses used for bool queries are filter, must, must\_not, and should.

**filter** – Filter is used to par down the dataset; a document will either fit into a filter or be excluded by it. Filter queries can be used to reduce datasets to a particular date or date range, specific location, or other exact matches. It is important to understand that filtering increases search performance.

Filter queries are automatically stored in the Elasticsearch cache. The next time the exact same filter query is run, the results will be pulled instantly from the cache. We will go into more depth about filtering below.

**must** – Must is similar to the “and” operator used when making a Google search. Using must tells Elasticsearch that document matches need to include all of the queries that fall under the must clause. If you have more than one query, then all of those queries need to match.

**must-not** – Must\_not is similar to the “not” operator used when making a Google search. It is the opposite of the must clause. Using must\_not tells Elasticsearch that document matches cannot include any of the queries that fall under the must\_not clause.

**should** – It would be ideal for the matching documents to include all of the queries in the should clause, but they do not have to be included. Scoring is used to rank the matches. Further down in this article we have a section on how scoring is used in Elasticsearch.

Eg.

{

"query":{

"bool": {

"must": {"match": {"genre": "Sci-Fi"}},

"must\_not": {"match": {"title": "trek"}},

"filter": {"range": {"year": {"gte": 2010, "lt": 2015}}}

}

}

}

RANGE==>

Stone below 100$

{

"query": {

"range": {

"price\_srk": {

"lte": 100

}

}

},

"size": 10,

"from": 0,

"sort": [

{

"price\_srk": {

"unmapped\_type": "keyword",

"order": "desc"

}

}

]

}

**Prefix/Wildcard Queries:**

{ "query": { "prefix": { "year": "201" } } }

{ "query": { "wildcard": { "year": "1**\***" } } }

Match only 5 star ratings using aggs:

{

"query": {

"match": {

"rating": 5.0

}

},

"aggs" : {

"ratings": {

"terms": {

"field" : "rating"

}

}

}

}

AVG rating for perticular movie:

{

"query": {

"match\_phrase": {

"title": "Star Wars Episode IV"

}

},

"aggs": {

"avg\_rating": {

"avg": {

"field": "rating"

}

}

}

}

**Snapshot and restore**

A snapshot is a backup of a running Elasticsearch cluster. You can use snapshots to:

* Regularly back up a cluster with no downtime
* Recover data after deletion or a hardware failure
* Transfer data between clusters
* Reduce your storage costs by using [searchable snapshots](https://www.elastic.co/guide/en/elasticsearch/reference/current/searchable-snapshots.html) in the cold and frozen data tiers

**Histogram**

A values array of [double](https://www.elastic.co/guide/en/elasticsearch/reference/current/number.html" \o "Numeric field types) numbers, representing the buckets for the histogram. These values must be provided in ascending order.

Eg.

{

"aggs": {

"whole\_stone": {

"histogram": {

"field": "total\_depth\_percent",

"interval": 2

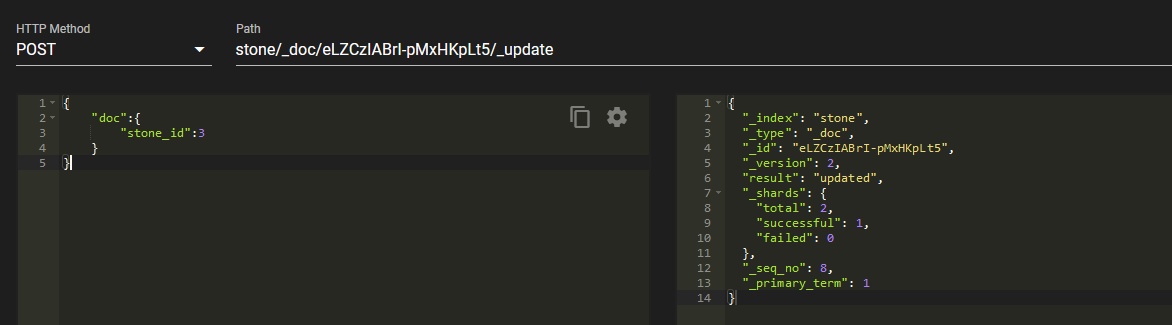
}

}

}

}

**UPDATE DATA :**



Body contain new data which you want to add/update

In url starts with

Index/type/\_id(on which you want to update)/\_update

**View Timestamp of hour:**

{

"aggs": {

"timestamp": {

"date\_histogram": {

"field": "@timestamp",

"interval": "hour"

}

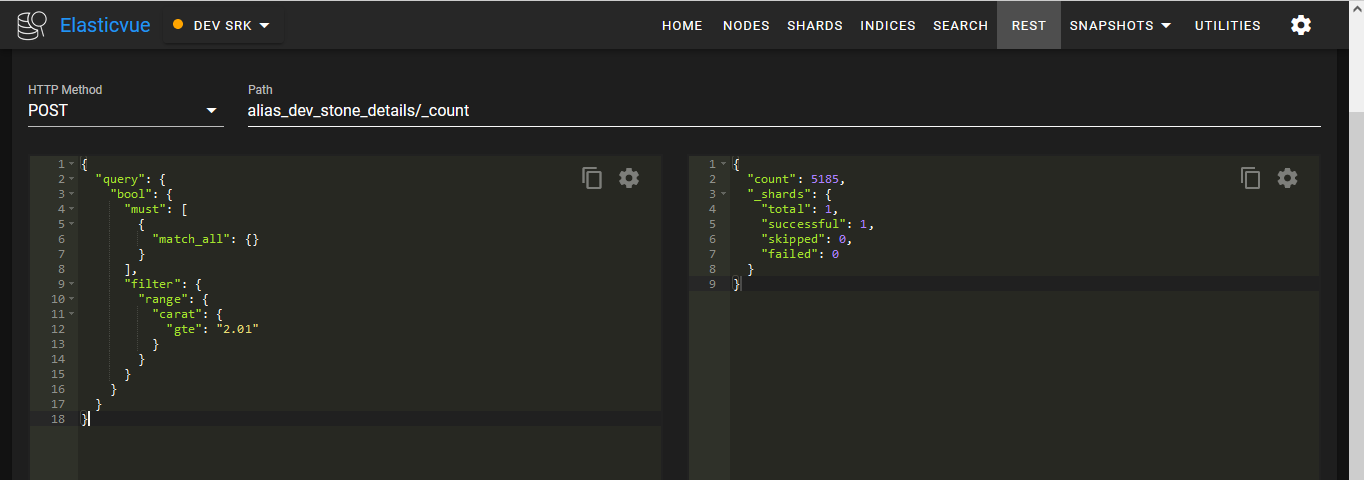
}

}

}

**View count of api:**

Use \_count for get the count instead of \_search



Eg. Range critiria

{

"query": {

"bool": {

"must": [

{

"match\_all": {}

}

],

"filter": {

"range": {

"carat": {

"gte": "2.01"

}

}

}

}

},

"\_source": {

"includes": [

"carat"

]

},

"track\_total\_hits": true

}

**Insert document at particular id**



Syntax: same for update in existing document

POST index\_name/index\_type/ :id (which you want to add/update)

\_version => give count of document Is updated/created/deleted

Update in kibana:

POST favourite\_candy/\_update/2

{"doc":{

"first\_name":"Ravi",

"Last\_name":"Patel",

"Team":"UI/UX!"

}

}

FIND NOT EMPTY URL QUERY

{"query":{"bool":{"filter":{"bool":{"must\_not":[{"match":{"digiplot\_url.keyword":""}},{"match":{"std\_movie\_url.keyword":""}},{"match":{"hna\_url.keyword":""}},{"match":{"plot\_url.keyword":""}},{"match":{"cert\_url.keyword":""}},{"match":{"image\_url.keyword":""}}]}}}}}

**NOT QUERY:**

{

    "query": {

        "bool": {

            "must\_not": {

                "term": {

                    "history\_type": "combine\_search"

                }

            }

        }

    },

    "\_source": {

        "includes": [

            "history\_type"

        ]

    },

    "size": 20,

    "from": 0,

    "sort": []

}

\* ) must field exists or not query

GET /\_search{

"query": {

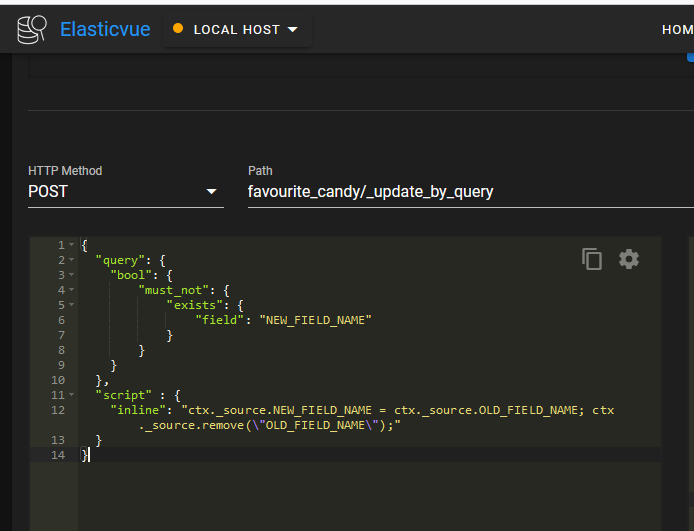
"exists": {

"field": "user"

}

}}

RENAME FIELD in INDEX:



Query:

{

"query": {

"bool": {

"must\_not": {

"exists": {

"field": "NEW\_FIELD\_NAME"

}

}

}

},

"script" : {

"inline": "ctx.\_source.NEW\_FIELD\_NAME = ctx.\_source.OLD\_FIELD\_NAME; ctx.\_source.remove(\"OLD\_FIELD\_NAME\");"

}

}

**Add mapping in existing index:**

{

"properties": {

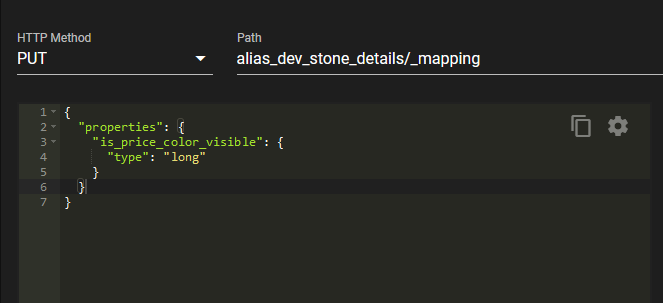
"is\_price\_color\_visible": {

"type": "long"

}

}

}



Set max-bucket-size:

PUT \_cluster/settings

{

"transient": {

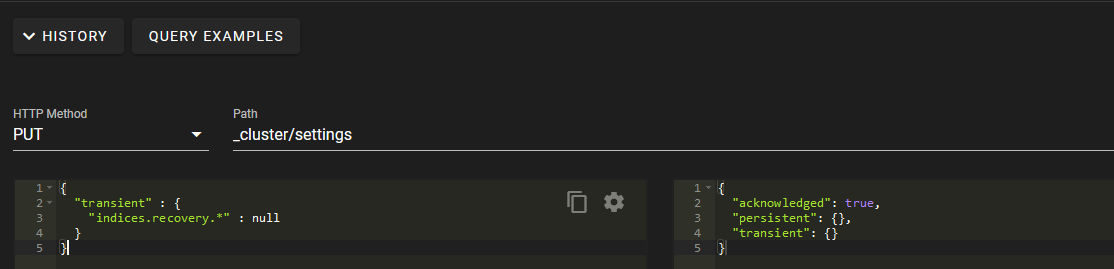
"search.max\_buckets": 20000

}

}

OR You can write those property and assign null value to it.  
  
**Property.Property = null**

Reset Set max-bucket-size:



curl --location --request PUT 'http://40.81.240.12:9200/\_cluster/settings' \

--header 'User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:100.0) Gecko/20100101 Firefox/100.0' \

--header 'Accept: application/json' \

--header 'Accept-Language: en-US,en;q=0.5' \

--header 'Accept-Encoding: gzip, deflate' \

--header 'Authorization: Basic ZWxhc3RpYzpTcmtheWNnMDAwMDk=' \

--header 'Content-Type: application/json' \

--header 'Connection: keep-alive' \

--data-raw '{

"transient" : {

"indices.recovery.\*" : null

}

}'

**MUST and MUST\_not example:**

{

"query": {

"bool": {

"must": {

"match": {

"b2b\_state": 1

}

},

"must\_not": {

"match": {

"std\_grp\_no": 0

}

}

}

}

}

\* FANCY STONE LAER QUERY SHOULD :

{"query":{"bool":{"filter":{"bool":{"should":[{"bool":{"must":[{"wildcard":{"color.id":"2,\*,\*"}},[{"bool":{"must":{"match":{"is\_ecommerce":1}}}}]]}},{"bool":{"must":[{"terms":{"btbv\_code":[0,979,982]}},{"terms":{"color.id":["5"]}},[{"bool":{"must":{"match":{"is\_ecommerce":1}}}}]]}}]}}}},"sort":[{"shape\_order":{"order":"asc"}},{"size\_order":{"order":"desc"}},{"clarity\_order":{"order":"asc"}},{"color\_order":{"order":"asc"}},{"carat":{"order":"asc"}},{"stone\_id.keyword":{"order":"asc"}}],"\_source":{"excludes":[],"includes":["stone\_id","b2b\_state","std\_grp\_no","price\_srk","carat"]}}

**BODY-BUILDER**

LINK: [Bodybuilder | An elasticsearch query body builder](https://bodybuilder.js.org/)



bodybuilder() .filter('term',('terms', 'product\_id'),1060617188).build()

Find document at perticuclar id:

bodybuilder()

.query("terms",("","\_id"),["Harsh.Patel"])

.build()

Fetch max\_id:

bodybuilder()

.agg("max","id")

.build()

**ACCESS NESTED [{}] in es :**

**QUery:**

{"query":{"nested":{"path":"company\_assigned\_roles","query":{"terms":{"company\_assigned\_roles.role\_name.keyword":["Buyer","Certification Vendor"]}}}}}

\

Add Mapping into existing index.

**CURL :**

curl --location --request PUT 'http://localhost:9200/alias\_dev\_party\_details/\_mapping' \ --header 'User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:108.0) Gecko/20100101 Firefox/108.0' \ --header 'Accept: application/json' \ --header 'Accept-Language: en-US,en;q=0.5' \ --header 'Accept-Encoding: gzip, deflate, br' \ --header 'Content-Type: application/json' \ --header 'Authorization: Basic ZWxhc3RpYzpTcmtheWNnMDAwMDk=' \ --header 'Origin: moz-extension://2b41486f-b6f0-4873-aad4-55edd5e08ade' \ --header 'Connection: keep-alive' \ --header 'Sec-Fetch-Dest: empty' \ --header 'Sec-Fetch-Mode: cors' \ --header 'Sec-Fetch-Site: same-origin' \ --header 'Cookie: sails.sid=s%3A\_ZNa-KlXTsPmkdPh7BK9JtfnvDqBbNzU.4GBZkhdMGK1dHIa5iyT6SzGVXuXRkF4xIu4210sDOtU' \ --data-raw '{ "properties": { "kam\_details": { "type": "nested", "properties": { "user\_code": { "type": "text", "fields": { "keyword": { "type": "keyword", "ignore\_above": 256 } } }, "kam\_name": { "type": "text", "fields": { "keyword": { "type": "keyword", "ignore\_above": 256 } } }, "short\_name": { "type": "text", "fields": { "keyword": { "type": "keyword", "ignore\_above": 256 } } }, "PARTY\_CODE": { "type": "text", "fields": { "keyword": { "type": "keyword", "ignore\_above": 256 } } }, "DEPARTMENT\_CODE": { "type": "long" } } } } }'

**MUST with nested [{}] :**

{"query":{"bool":{"must":[{"nested":{"path":"company\_assigned\_roles","query":{"terms":{"company\_assigned\_roles.role\_name.keyword":["Shipping","Certification Vendor"]}}}},{"nested":{"path":"kam\_details","query":{"terms":{"kam\_details.DEPARTMENT\_CODE":[10]}}}}]}}}

**BULK operation in elasticsearch**

Ref : https://www.elastic.co/guide/en/elasticsearch/reference/7.17/docs-bulk.html