Exam Study Guide

To be fully prepared for the Elastic Certified Engineer exam, candidates should be able to complete all of the following exam objectives with only the assistance of the [**Elastic documentation**](https://www.elastic.co/guide/index.html):

Example questions:

<https://medium.com/ableneo/elastic-certified-engineer-exam-92260e9dabca>

cluster setup leveraging hot-warm nodes. ssh to each node, set elasticsearch.yml with correct properties and run elasticsearch process. cluster properties, modify already indexed data using ingest pipeline and Reindex API. create some custom analyzer and also one or two tasks about queries, nothing horrible just some aggregations and bool queries.

Reindex

# **Preparing for the Elastic Certified Engineer Exam - Get Elasticsearch Certified**

<https://www.youtube.com/watch?v=hsaLZSKCkF0>

Linuxacademy.com/ Elastic Certified Engineer Exam – Part 1

<https://linuxacademy.com/hands-on-lab/7ee0306e-4d2a-45c0-9419-5e795a22ba1e/>

Installation and Configuration

* **Deploy and start an Elasticsearch cluster that satisfies a given set of requirements**

FROM DOCUMENTATION: Install

**mac:**

curl -L -O https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-7.6.2-darwin-x86\_64.tar.gz

tar -xzvf elasticsearch-7.6.2-darwin-x86\_64.tar.gz

cd elasticsearch-7.6.2

./bin/elasticsearch

Test it’s up and running:

curl http://127.0.0.1:9200

or

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

Set Cluster Name and node name

-E syntax as follows:

./bin/elasticsearch -d -Ecluster.name=my\_cluster -Enode.name=node\_1

FROM COURSE: run an instance of Elastic

Ssh server1

//extract Elastic Search

tar -xf elasticsearch-7.3.1-linux-x86\_64.tar.gz

// start an instance of ElasticSearch

./elasticsearch-7.3.1/bin/elasticsearch

// once ^ is running, go to new tab

// and check the configuration of the node with :

curl -X GET <http://localhost:9200/>

// node name is “server1” and cluster\_name defaults as “elasticsearch”

* **Configure the nodes of a cluster to satisfy a given set of requirements**

Config files location

Elasticsearch has three configuration files:

* elasticsearch.yml for configuring Elasticsearch
* jvm.options for configuring Elasticsearch JVM settings
* log4j2.properties for configuring Elasticsearch logging

// TO CHANGE Cluster name: enter edit mode for yml config file to

nano elasticsearch-7.3.1/config/elasticsearch.yml

// ^ inside you can change cluster\_name or name of node

// TO CHANGE default heap size, open jvm.options

elasticsearch-7.3.1/config/jvm.options

// change from 4G to 512MB: edit -Xms4 to become 🡪 -Xms512m

Make sure to restart the instance to incorporate the changes

TO CHANGE the node name, use CLI

./elasticsearch-7.3.1/bin/elasticsearch -E **node.name=node1** -E http.host="localhost","server1"

* Secure a cluster using Elasticsearch Security

Getting started with security, xpack enabled true in yml file, discovery single node

Active passwords

* Define role-based access control using Elasticsearch Security

Kibana management dashboard

Indexing Data

* Define an index that satisfies a given set of requirements
* **Perform index, create, read, update, and delete operations on the documents of an index**

1. READ: GET
   1. Search for all the documents of an index: GET index\_name
   2. Two ways to do a match\_all query, search for all the documents in ANY index: A close up of text on a white background

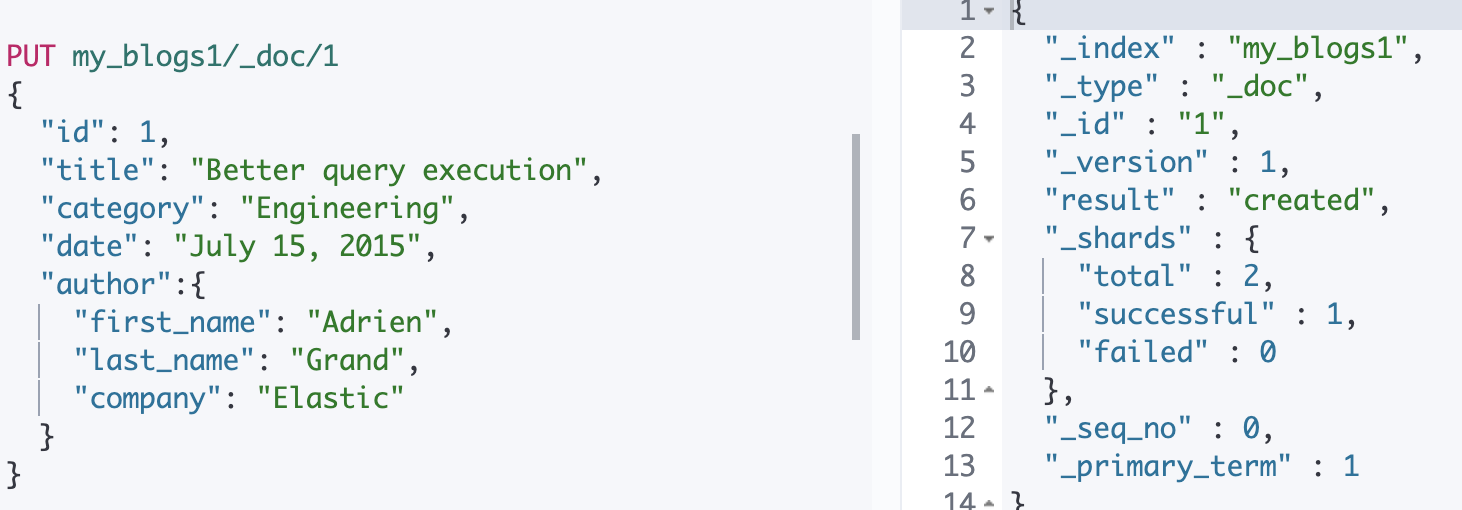
      Description automatically generated
   3. Match\_all query for documents in a specific index:



* 1. If you’d like to search for a document use “/\_doc” instead of \_Search  
      A picture containing drawing

     Description automatically generated

1. CREATE: POST or PUT
   1. Put requires an ID, POST doesn’t. Put is when you need to match the id of the document from its source for an example use case.
   2. PUT can take an id even if it’s a new doc, creates an entry if doesn’t exist, otherwise it’



1. DELETE
   1. A document in an index:
      1. \*be careful because it wont fail even if the document doesn’t exist
      2. DELETE my\_blogs/\_doc/1
   2. An index:
      1. DELETE /my\_blogs

* Define and use index aliases
* Define and use an index template for a given pattern that satisfies a given set of requirements
* Define and use a *dynamic template* that satisfies a given set of requirements
* Use the Reindex API and Update By Query API to reindex and/or update documents
* Define and use an ingest pipeline that satisfies a given set of requirements, including the use of Painless to modify documents
  + This example was in the youtube video

Queries

* Write and execute a search query for terms and/or phrases in one or more fields of an index

Multi\_match query



|  |
| --- |
| GET /courses/\_search  {  "query":{  "bool":{  "filter":{  "bool":{  "must":[  {"match":{"room": "e7"}},  {"range": {  "students\_enrolled":{  "gte": 12  }  }  },  {"match":{"professor.name": "baszo"}}  ],  "must\_not":[  {"match":{"course\_description": "economics"}},  {"match":{"professor.name": "bill"}}  ]  }  }  }  }  } |

* Write and execute a search query that is a Boolean combination of multiple queries and filters   
  A close up of a map

  Description automatically generated
* **Highlight** the search terms in the response of a query

there will be another element in each search hit, called highlight, which includes the highlighted fields and the highlighted fragments).

GET /courses/\_search

{

"size": 3,

"query":{

"match": {"name": "accounting"}

},

**"highlight": {**

**"fields": {**

**"name": {}**

**}**

}

}

A screenshot of a social media post

Description automatically generated

* Sort the results of a query by a given set of requirements
  + In a get request, include **“sort”: [ {“price”: {“order”: “desc”}} ]**
* Implement pagination of the results of a search query
  + Use the **from** and **size** properties to offset the results and set max # of docs returned in the results list
* Use the scroll API to retrieve large numbers of results

GET 'localhost:9200/twitter/tweet/\_search?scroll=1m'



<https://www.elastic.co/guide/en/elasticsearch/reference/7.5/search-request-body.html#request-body-search-scroll>

* Apply fuzzy matching to a query

<https://www.elastic.co/guide/en/elasticsearch/reference/7.6/common-options.html#fuzziness>

<https://www.elastic.co/guide/en/elasticsearch/reference/7.6/query-dsl-match-query.html#query-dsl-match-query-fuzziness>

* Define and use a *search template*

<https://www.elastic.co/guide/en/elasticsearch/reference/7.6/search-template.html>

* Write and execute a query that searches across multiple clusters

Aggregations

* Write and execute *metric* and bucket aggregations
* Write and execute aggregations that contain *sub-aggregations*
  + The aggregation below groups data by manufactuer (make) and then generates an average aggregation on the price field within the bucket.
  + { “aggs”: {

“popular\_manufacturers”: {“terms”: {“field”: “make.keyword”}},

“aggs”: { “average\_price”: {“avg”: {“field”: “price”}} } }

}}

Mappings and Text Analysis (hard)

* Define a mapping that satisfies a given set of requirements

|  |
| --- |
| PUT /customers  {  "mappings": {  "properties": {  "gender": {  "type": "text",  "analyzer": "standard"  },  "age": {  "type": "integer"  },  "total\_spend": {  "type": "float"  },  "is\_new": {  "type": "boolean"  },  "name": {  "type": "text",  "analyzer": "standard"  }  }  },  "settings": {  "number\_of\_replicas": 2,  "number\_of\_shards": 1  }  } |

A close up of text on a white background

Description automatically generated

* Define and use a custom analyzer that satisfies a given set of requirements
* Define and use multi-fields with different data types and/or analyzers

Min 18:14 <https://www.elastic.co/webinars/understanding-multi-fields-in-elasticsearch-mappings>

When mapping an index, and setting the type (text, float etc) of each field in the documents, you can set a secondary versions of a field that have different analyzers applied like the title of the blog uses the default standard analyzer, but you can also do query conditions on title.my\_english to remove stop words, lowercase and stem words so “Enterprise” will match with “enterprises”  
 A close up of text on a white background

Description automatically generated

In this example, the English analyzer makes “clouds” match with “Cloud”, while the standard analyzer on “title” wouldn’t match that.

A screenshot of a cell phone

Description automatically generated

* Configure an index so that it properly maintains the relationships of nested arrays of objects

Cluster Administration

* Allocate the shards of an index to specific nodes based on a given set of requirements
* Configure shard allocation awareness and forced awareness for an index
* Diagnose shard issues and repair a cluster’s health
* Backup and restore a cluster and/or specific indices
* Configure a cluster for use with a hot/warm architecture
* Configure a cluster for cross cluster search