# Introduction

Elasticsearch is a search and analytics engine in what is called the Elastic Stack.

# CRUD Operations

## Create with Index API

The PUT keyword is used to create an empty index, which can be filled later:

|  |
| --- |
| PUT index\_name |

An index can be created with settings and/or mappings. Mappings contain the index’s schema, meaning its fields and their configuration (type, format, properties/sub-fields, etc.). For example:

|  |
| --- |
| PUT index\_name {  "settings": {  "number\_of\_shards": 3,  "number\_of\_replicas": 2  },  "mappings": {  "properties": {  "field1": {  "type": "date",  "format": "yyyy-MM-dd"  },  "field2": {  "type": "text",  "fields": {  "keyword": {  "type": "keyword",  "ignore\_above": 256  }  }  }  }  }  } |

Mappings, for example, can directly be set from index\_name/\_mapping. The following two examples are thus equivalent:

|  |  |
| --- | --- |
| PUT index\_name {  "mappings": {  "properties": {  ...  }  }  } | PUT index\_name/\_mapping {  "properties": {  ...  }  } |

## Insert and CRUD with Bulk API

The POST keyword to the \_bulk index is used to perform multiple operations, including inserting (creating) new data in an index:

|  |
| --- |
| POST \_bulk  { "create": { "\_index": "index\_name", "\_id": "1" } }  { "field1": "value1", "field2": "value2", "field3": "value3" }  { "create": { "\_index": "index\_name", "\_id": "2" } }  { "field1": "value4", "field2": "value5", "field3": "value6" }  { "create": { "\_index": "index\_name", "\_id": "3" } }  { "field1": "value7", "field2": "value8", "field3": "value9" } |

Bulk can also be used to perform all CRUD operations, not only insertion, and can be mixed up inside a single request:

|  |
| --- |
| POST \_bulk  { "index": { "\_index": "index\_name", "\_id": "1" } }  { "field1": "value1" }  { "delete": { "\_index": "index\_name", "\_id": "2" } }  { "create": { "\_index": "index\_name", "\_id": "3" } }  { "field1": "value3" }  { "update": { "\_index": "index\_name", "\_id": "1" } }  { "doc": { "field2": "value2" } } |

## Read with Index API

The GET keyword is used to read an existing index, including its data and metadata:

|  |
| --- |
| GET index\_name |

It is also possible to read multiple indices:

|  |
| --- |
| GET index1, index2, index3 |

It is possible to query the index for a specific set of data with the query parameter and its own parameters, including all, open, closed, hidden, and include\_defaults. For example, the following will match all results:

|  |
| --- |
| GET recordings/\_search {  "query": {  "match\_all": {}  },  } |

## Delete with Index API

The DELETE keyword is used to delete an existing index, including its data and metadata:

|  |
| --- |
| DELETE index\_name |

It is also possible to delete multiple indices:

|  |
| --- |
| DELETE index1, index2, index3 |

The same options that apply for GET can also be applied for DELETE (like, for example, deleting entries that match a specific pattern only).

## Update with Index API

The PUT keyword is used to update an existing index mapping, but also create one as seen in 2.1 above:

|  |
| --- |
| PUT index\_name/\_mapping {  "properties": {  ...  }  } |

The same options that apply for GET and DELETE can also be applied for PUT (like, for example, updating entries that match a specific pattern only).

Changing the mapping of an existing field is sometimes dangerous, since it could invalidate existing data. So it is recommended to create a new index with the correct mapping, then re-index to copy from the original index to the new one:

|  |
| --- |
| POST \_reindex {  "source": {  "index": "original\_index"  },  "dest": {  "index": "new\_index"  }  } |