BIG DATA

TOO BIG TO IGNORE

SÜMEYYE KAYNAK

OUTLINE

ElasticSearch

ELASTICSEARCH

- It is the Nosql technology used in text search operations in big data.
- Elasticsearch infrastructure includes Apache Lucene and Solr libraries.
- Apache Lucene and Solr libraries are open source.
- In the Elasticsearch module, data is indexed while it is being saved.

ELASTICSEARCH

- In the word search, not all data is searched. Results can be found quickly through the created index list.

Pfizer

Implementing Elastic as a fundamental core to a drug-manufacturing data cloud

ST MARY'S UNIVERSITY

Customizing search results across multiple domains to make their students' lives easier

Adobe

Making search smarter with machine learning at scale

COMPANIES USING ELASTICSEARCH

IEEE

Monitoring application infrastructure across a major financial institution

ebay

Creating a custom Elasticsearch as a Service platform

Delivering a better help experience for over a billion users

CISCO

Reducing system downtime with Elasticsearch at the basis of Cisco's Cloud Native platform

bankwest

Creating a real-time search solution across over 4 million customer records

THOMSON REUTERS

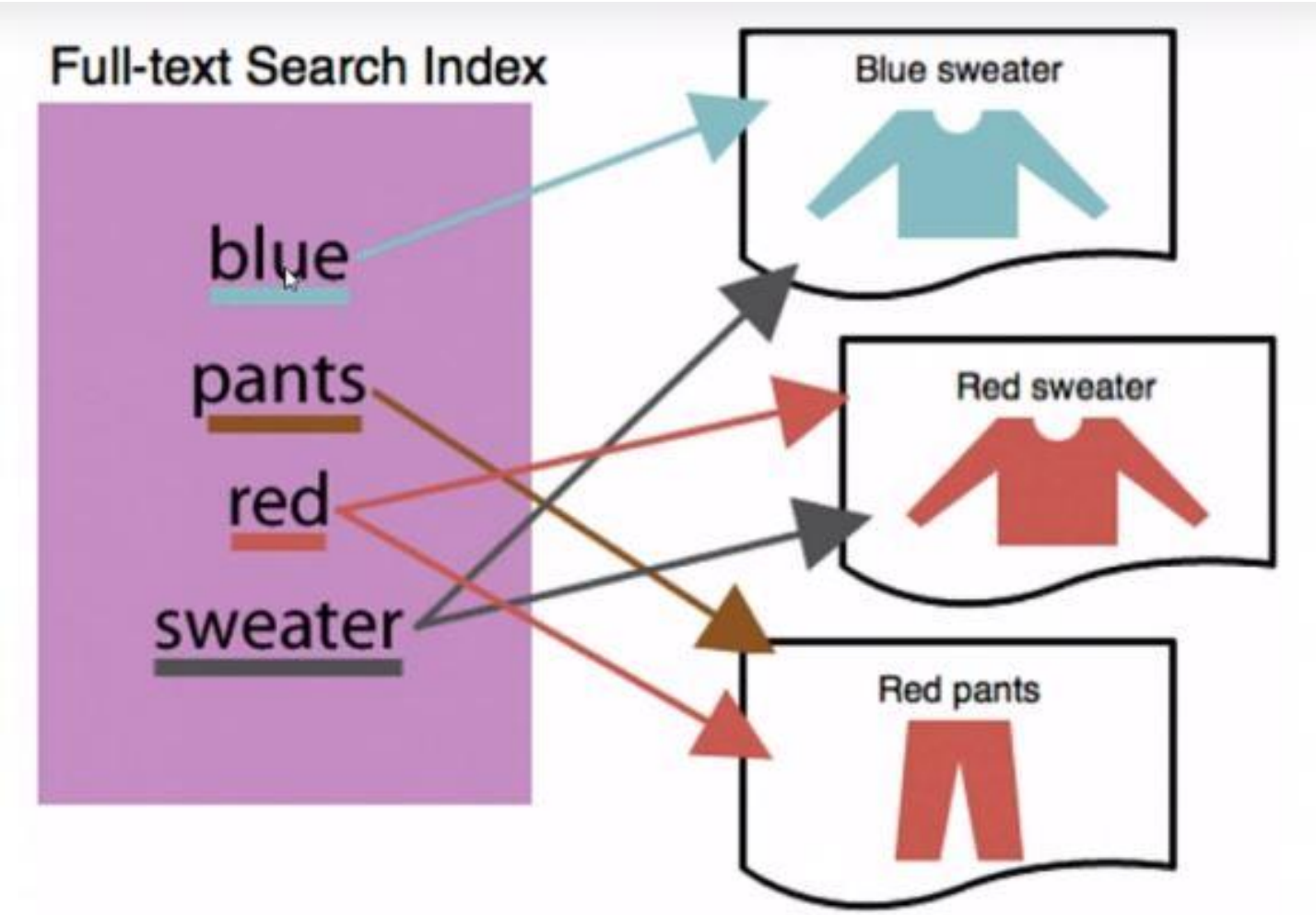
Driving better research, analysis, and journalism

1	The old night keeper the keep in the town
2	In the big old house in the big old gown.
3	The house in the town had the big old keep.
4	Where the old night keeper never did sleep.
5	The night keeper keeps the keep in the night.
6	And keeps in the dark and sleeps in the light.

6 document to index

Term	Documents
Sleeps	<6>
The	<1> <2> <3> <4> <5> <6>
Town	<1> <3>
Where	<4>

Term	Documents
And	<6>
Big	<2> <3>
Dark	<6>
Did	<4>
Gown	<2>
Had	<3>
House	<2> <3>
In	<1> <2> <3> <5> <6>
Keep	<1> <3> <5>
Keeper	<1> <4> <5>
Keeps	<1> <5> <6>
Light	<6>
Never	<4>
Night	<1> <4> <5>
Old	<1> <2> <3> <4>
Sleep	<4>


Full-text Search Index
blue
pants
red
sweater
Blue sweater
Red sweater
Red pants

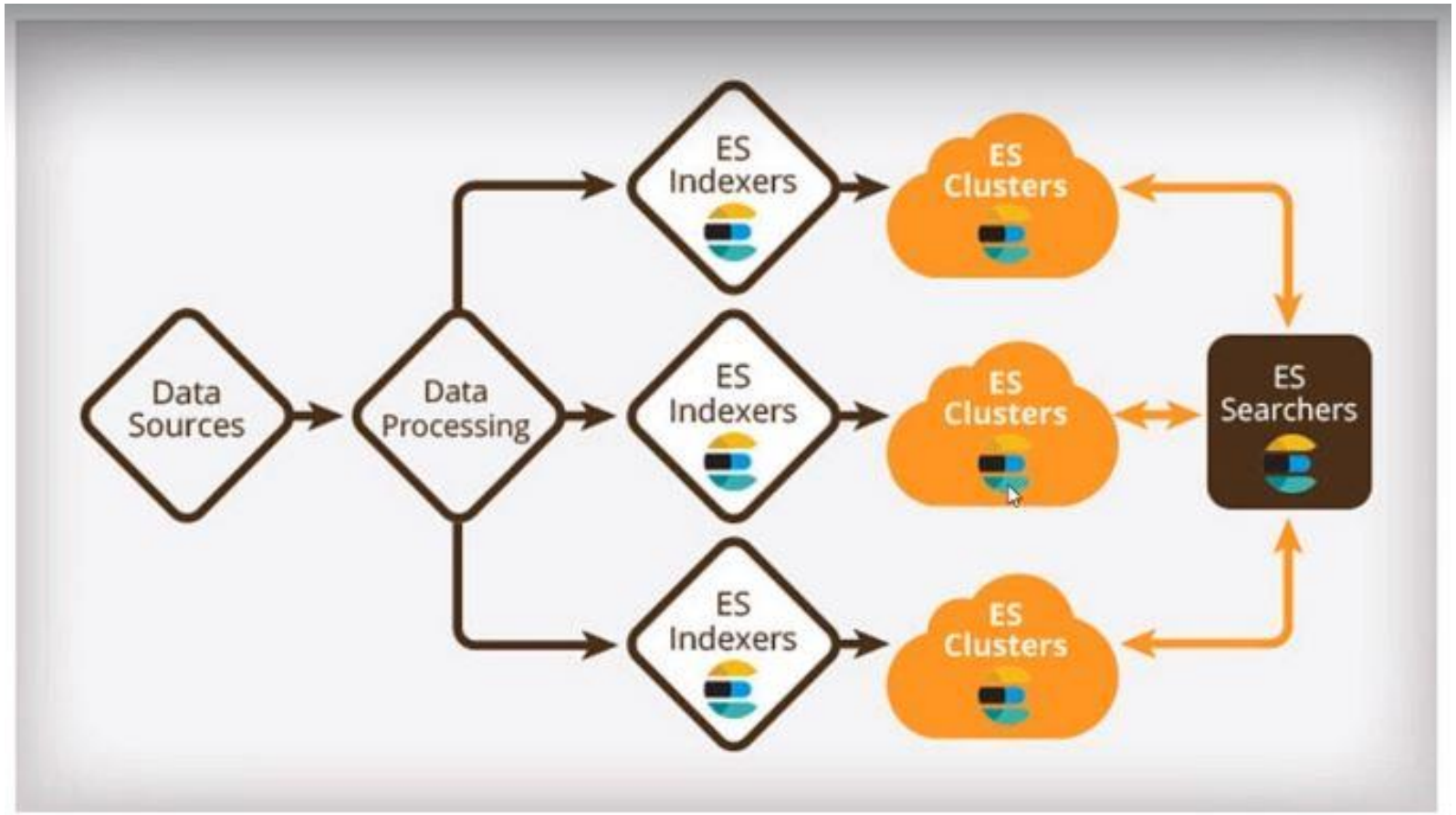
TERMINOLOGY

Relation Databases

- Database
- Table
- Row
- Column
- Schema

Elasticsearch

- Index
- Type
- Document
- Fields
- Mapping


Data Sources
Data Processing
ES Indexers
ES Indexers
ES Indexers
ES Clusters
ES Clusters
ES Clusters
ES Searchers

DOCUMENT AS JSON

```
{
  "id"           : "abc123",
  "title"        : "A JSON Document",
  "body"         : "A JSON document is a ...",
  "published_on" : "2013/06/27 10:00:00",
  "featured"     : true,
  "tags"         : ["search", "json"],
  "author"       : {
       "first_name"  : "Clara",
       "last_name"   : "Rice",
       "email"       : "clara@rice.org"
  }
}
```

- Since Elasticsearch is a document-type Nosql database, documents are stored in json format.

GENERAL FEATURES

- Elasticsearch is scalable up to petabytes of structured and unstructured data.
- Elasticsearch uses denormalization to improve the search performance.
- Elasticsearch is one of the popular enterprise search engines, and is currently being used by many big organizations like Wikipedia, The Guardian, StackOverflow, GitHub etc.
- Elasticsearch is an open source and available under the Apache license version 2.0

KEY CONCEPTS

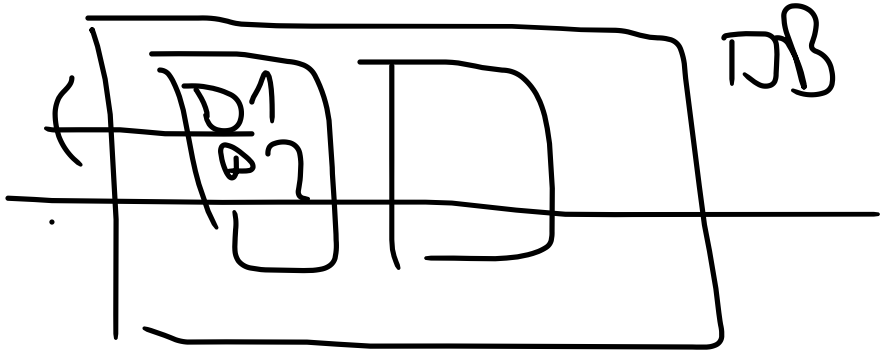
- **Node:** It refers to a single running instance of Elasticsearch.
- **Cluster**: It is a collection of one or more nodes. Cluster provides collective indexing and search capabilities across all the nodes for entire data.
- **Index**: It is a collection of different type of documents and their properties. Index also uses the concept of shards to improve the performance.

KEY CONCEPTS

- **Document**: It is a collection of fields in a specific manner defined in JSON format. Every document belongs to a type and resides inside an index. Every document is associated with a unique identifier called the UID.
- **Shard**: Indexes are horizontally subdivided into shards. This means each shard contains all the properties of document but contains less number of JSON objects than index. The horizontal separation makes shard an independent node, which can be store in any node.

DB

Table



DB

KEY CONCEPTS

- **Replicas**: Elasticsearch allows a user to create replicas of their indexes and shards. Replication not only helps in increasing the availability of data in case of failure, but also improves the performance of searching by carrying out a parallel search operation in these replicas.

ADVANTAGES

- Elasticsearch is developed on Java, which makes it compatible on almost every platform.
- Elasticsearch is real time, in other words after one second the added document is searchable in this engine.
- Elasticsearch is distributed, which makes it easy to scale and integrate in any big organization.
- Creating full backups are easy by using the concept of gateway, which is present in Elasticsearch.

ADVANTAGES

- Handling multi-tenancy is very easy in Elasticsearch when compared to Apache Solr.
- Elasticsearch uses JSON objects as responses, which makes it possible to invoke the Elasticsearch server with a large number of different programming languages.

DISADVANTAGES

- Elasticsearch does not have multi-language support in terms of handling request and response data (only possible in JSON) unlike in Apache Solr, where it is possible in CSV, XML and JSON formats.

ELASTIC SEARCH-INSTALLATION

For Windows OS;

- Download Elasticsearch from https://www.elastic.co/downloads/elasticsearch
- The downloaded file is copied to the C:

ELASTIC SEARCH SERVER

To run the Elastic Search server;

```
Microsoft Windows [Version 10.0.18363.1556]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Sumeyye>cd ..

C:\Users>cd ..

C:\>cd elasticsearch-7.15.0

C:\elasticsearch-7.15.0>cd bin

C:\elasticsearch-7.15.0\bin>elasticsearch.bat
```

CRUD OPERATION

- In client/server architecture, a request-response communication is performed between client-server.
- Http methods determine the purpose of the request made to the server side.

HTTP METHODS

Common request types are:

- GET
- POST
- HEAD
- OPTIONS
- TRACE
- PUT
- DELETE
- CONNECT

Method	Description
GET	Request to read a Web page
HEAD	Request to read a Web page's header
PUT	Request to store a Web page
POST	Append to a named resource (e.g. a Web page)
DELETE	Remove the web page
TRACE	Echo the incoming request
CONNECT	Reserved for future use
OPTIONS	Query certain options

GET AND POST METHODS

- **Get method:** GET is used to request data from a specified resource.
- **Post method:** POST is used to send data to a server to create/update a resource.

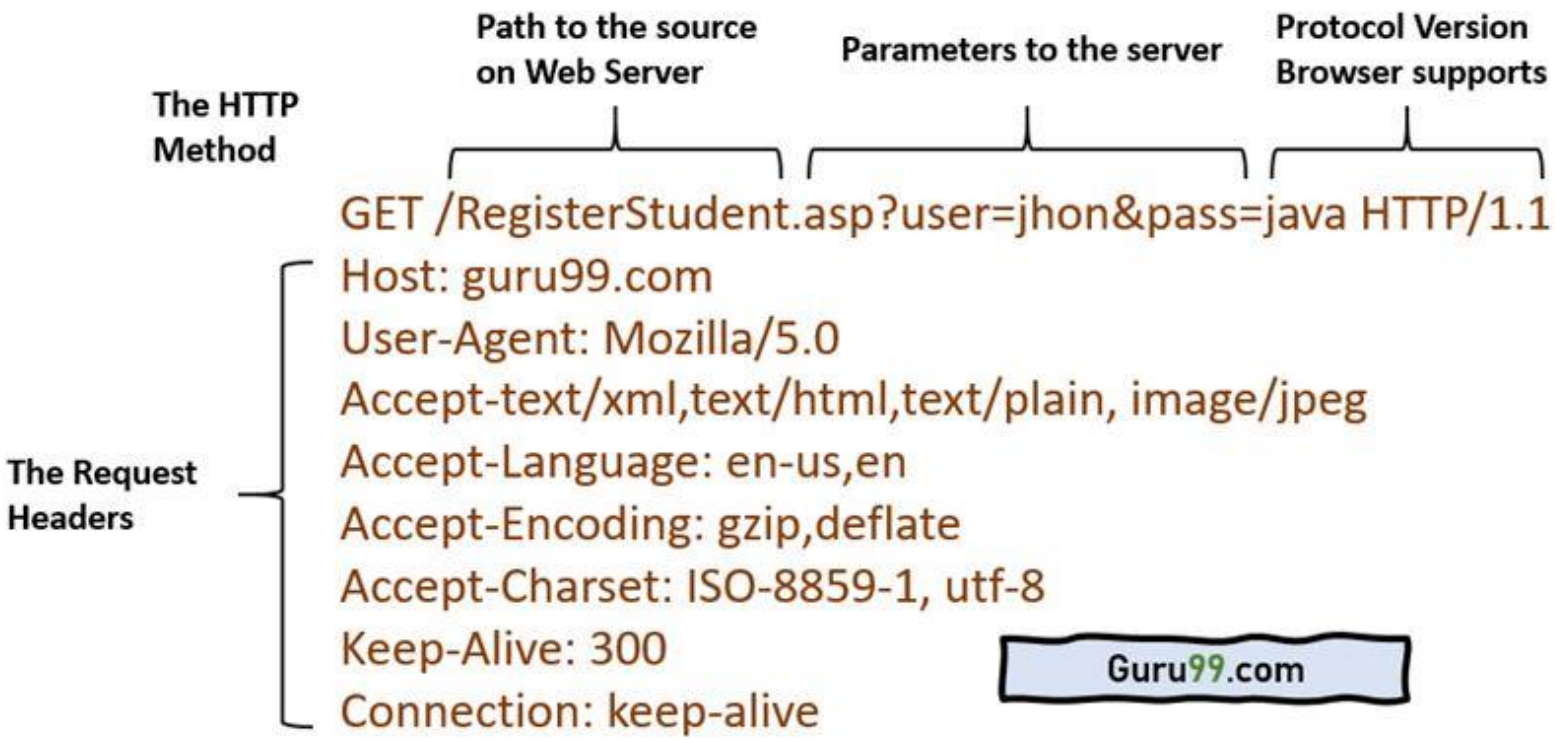
GET METHOD

- Get requests can be cached.
- In GET method, values are visible in the URL.
- Get requests remain in the browser history.
- Get requests can be bookmarked.
- Get requests should never be used when dealing with sensitive data
- GET requests have length restrictions
- GET requests are only used to request data (not modify)

/test/demo_form.php?name1=value1&name2=value2

GET METHOD

```
GET/RegisterStudent.asp?user=value1&pass=value2
```



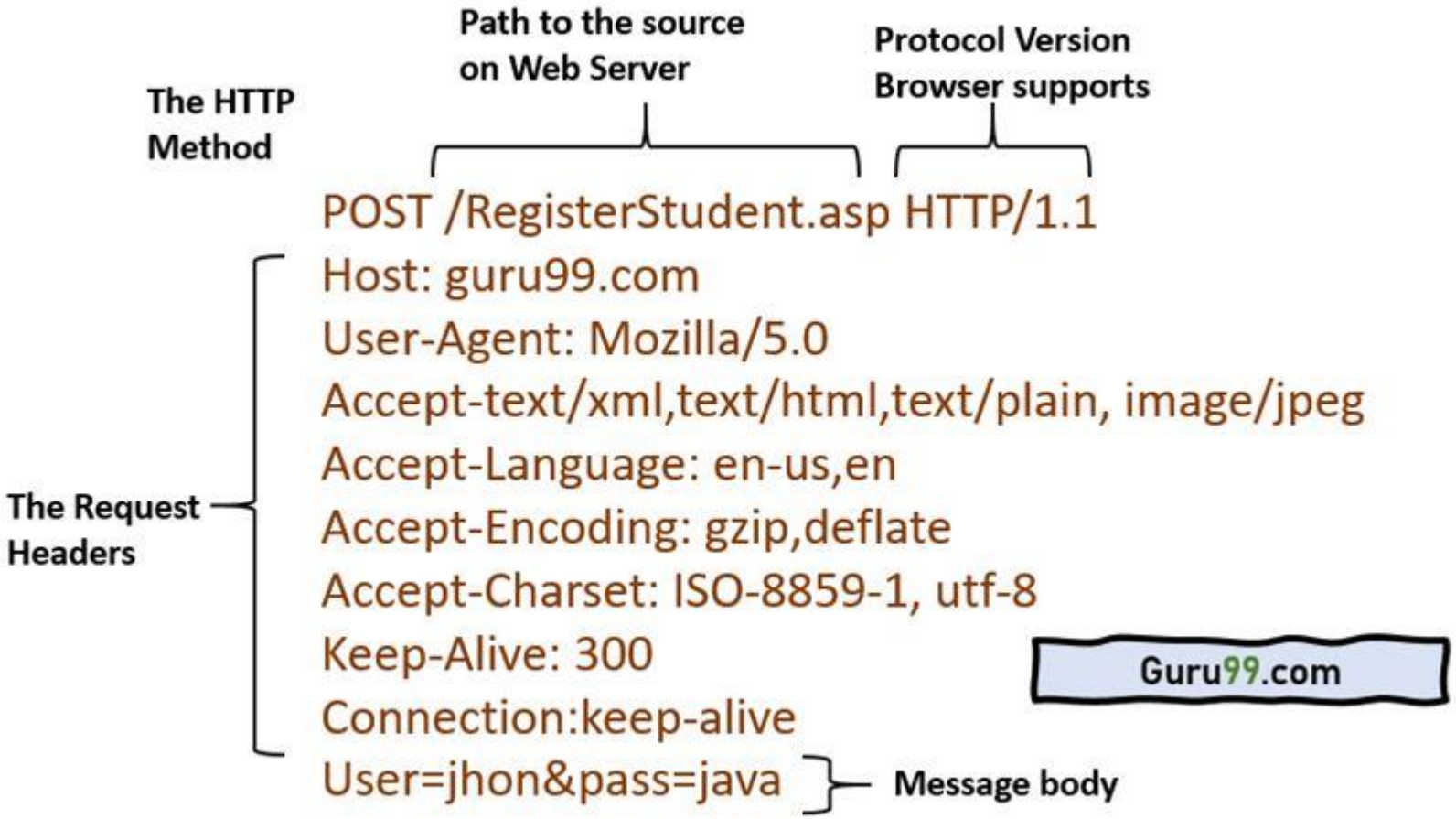
POST METHOD

- POST requests are never cached
- In POST method, values are not visible in the URL.
- POST requests do not remain in the browser history
- POST requests cannot be bookmarked
- POST requests have no restrictions on data length

/test/demo_form.php

POST METHOD

```
POST/RegisterStudent.asp HTTP/1.1
Host: www.guru99.com
user=value1&pass=value2
```



Product Pricing Enterprise Resources and support Explore

Sign In Sign

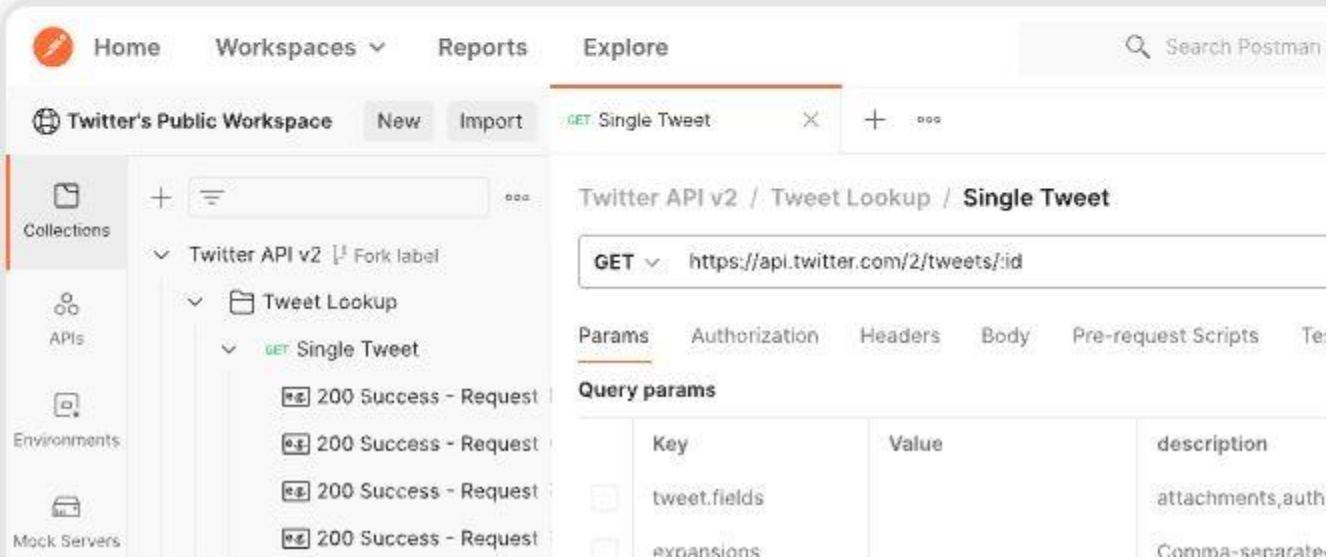
Download Postman

Download the app to quickly get started using the Postman API Platform. Or, if you prefer a browser experience, you can try the new web version of Postman.

The Postman app

The ever-improving Postman app (a new release every two weeks) gives you a full-featured Postman experience.

Download the App



Home Workspaces Reports Explore Search Postman Invite Upgrade

My Workspace New Import

GET https://localhost:4... GET https://localhost:4...

No Environment

Collections
APIs
Environments
Mock Servers
Monitors
History

January 4
GET https://localhost:44341/api/Todolt...
DEL https://localhost:44341/api/Todolt...
GET https://localhost:44341/api/Todolt...
GET https://localhost:44341/api/Todolt...
POST https://localhost:44341/api/Todolt...
GET https://localhost:44341/api/Todolt...
POST https://localhost:44341/api/Todolt...

January 3
GET https://localhost:44390/api/Todolt...
POST https://localhost:44390/api/Todolt...
GET https://localhost:44390/api/Todolt...
GET https://reqres.in/api/users?page=2

January 2
GET https://localhost:44390/api/Todolt...
GET https://localhost:44390/api/Todolt...
DEL https://localhost:44390/api/Todolt...

https://localhost:44390/api/TodoItems Save

GET https://localhost:44341/api/TodoItems/2 Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

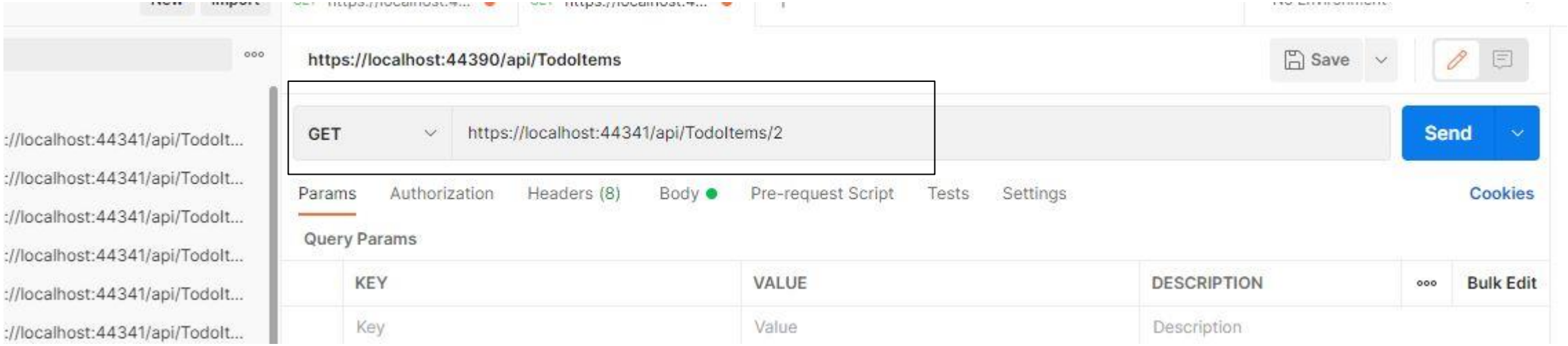
Query Params

KEY	VALUE	DESCRIPTION	Bulk Edit
Key	Value	Description	

Body Cookies Headers Test Results Status: 200 OK Time: 829 ms Size: 184 B Save Response

Pretty Raw Preview Visualize Text

1


https://localhost:44390/api/TodoItems
Save
GET
https://localhost:44341/api/TodoItems/2
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Key
Value
Description
://localhost:44341/api/TodoIt...

Command Prompt - elasticsearch.bat

```
Microsoft Windows [Version 10.0.18363.1556]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Sumeyye>cd ..

C:\Users>cd ..

C:\>cd elasticsearch-7.15.0

C:\elasticsearch-7.15.0>cd bin

C:\elasticsearch-7.15.0\bin>elasticsearch.bat
[2021-10-14T11:29:36,238][INFO ][o.e.n.Node               ] [DESKTOP-GK3VE45] version[7.15.0], pid[18184], build[defau
/zip/79d65f6e357953a5b3cbcc5e2c7c21073d89aa29/2021-09-16T03:05:29.143308416Z], OS[Windows 10/10.0/amd64], JVM[Eclipse
undation/OpenJDK 64-Bit Server VM/16.0.2/16.0.2+7]
[2021-10-14T11:29:36,335][INFO ][o.e.n.Node               ] [DESKTOP-GK3VE45] JVM home [C:\elasticsearch-7.15.0\jdk],
ing bundled JDK [true]
[2021-10-14T11:29:36,350][INFO ][o.e.n.Node               ] [DESKTOP-GK3VE45] JVM arguments [-Des.networkaddress.cache
tl=60, -Des.networkaddress.cache.negative.ttl=10, -XX:+AlwaysPreTouch, -Xss1m, -Djava.awt.headless=true, -Dfile.encodi
=UTF-8, -Djna.nosys=true, -XX:-OmitStackTraceInFastThrow, -XX:+ShowCodeDetailsInExceptionMessages, -Dio.netty.noUnsafe
rue, -Dio.netty.noKeySetOptimization=true, -Dio.netty.recycler.maxCapacityPerThread=0, -Dio.netty.allocator.numDirectA
nas=0, -Dlog4j.shutdownHookEnabled=false, -Dlog4j2.disable.jmx=true, -Djava.locale.providers=SPI,COMPAT, --add-opens=j
a.base/java.io=ALL-UNNAMED, -XX:+UseG1GC, -Djava.io.tmpdir=C:\Users\Sumeyye\AppData\Local\Temp\elasticsearch, -XX:+Hea
umpOnOutOfMemoryError, -XX:HeapDumpPath=data, -XX:ErrorFile=logs/hs_err_pid%p.log, -Xlog:gc*,gc+age=trace,safepoint:fi
```

```
, version: 50, reason: Publication{term=2, version=50}
2021-10-14T11:30:19,503][INFO ][o.e.h.AbstractHttpServerT
ound_addresses {127.0.0.1:9200}, {[::1]:9200}
2021-10-14T11:30:19,515][INFO ][o.e.n.Node
2021-10-14T11:30:19,927][INFO ][o.e.l.LicenseService
```

GET https://localhost:4... | GET https://localhost:4... | No Environment

https://localhost:44390/api/TodoItems

Save

GET localhost:9200 Send

Params | Authorization | Headers (6) | Body | Pre-request Script | Tests | Settings | Cookies

none | form-data | x-www-form-urlencoded | raw | binary | GraphQL

This request does not have a body

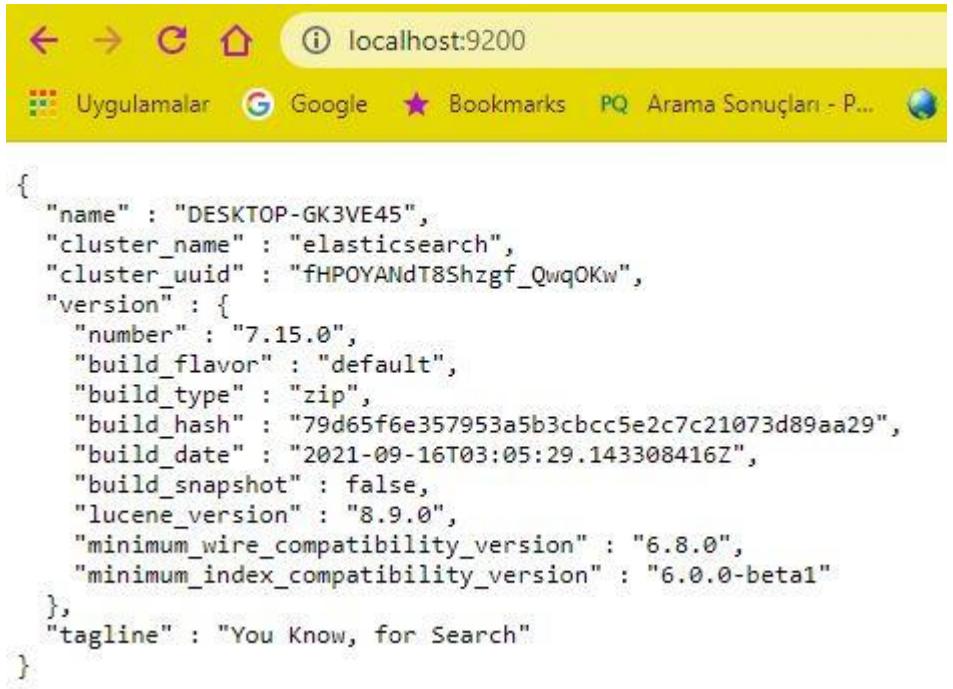
Body | Cookies | Headers (5) | Test Results | Status: 200 OK | Time: 12 ms | Size: 801 B | Save Response

Pretty | Raw | Preview | Visualize | JSON


```
{
    "name": "DESKTOP-GK3VE45",
    "cluster_name": "elasticsearch",
    "cluster_uuid": "fHPOYANdT8Shzgf_QwqOKw",
    "version": {
        "number": "7.15.0",
        "build_flavor": "default",
        "build_type": "zip",
        "build_hash": "79d65f6e357953a5b3cbcc5e2c7c21073d89aa29",
        "build_date": "2021-09-16T03:05:29.143308416Z",
        "build_snapshot": false,
        "lucene_version": "8.9.0",
        "minimum_wire_compatibility_version": "6.8.0",
```

MAPPING CONCEPTS ACROSS SQL AND ELASTIC SEARCH

SQL	ELASTIC SEARCH
column	field
row	document
table	index
schema	implicit



PUT METHOD

https://localhost:44390/api/TodoItems

Save

PUT | localhost:9200/exam/product/123 | Send

Params | Authorization | Headers (8) | Body | Pre-request Script | Tests | Settings | Cookies

none | form-data | x-www-form-urlencoded | raw | binary | GraphQL | JSON | Beautify

```
1 {
2     "brand":"apple",
3     "name": "iphone x",
4     "color": "black",
5     "date":"2018"
6 }
```

This information is sent to the table named product of the database named exam with the id number of 123.

PUT localhost:9200/exam/product/123 Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1  {
2      "brand":"apple",
3      "name": "iphone x",
```

Body Cookies Headers (7) Test Results Status: 201 Created Time: 1651 ms Size: 889 B Save Response

Pretty Raw Preview Visualize JSON

```
1  {
2      "_index": "exam",
3      "_type": "product",
4      "_id": "123",
5      "_version": 1,
6      "result": "created",
7      "_shards": {
8          "total": 2,
9          "successful": 1,
10         "failed": 0
11     },
12     "_seq_no": 0,
```

UPDATE WITH POSTMAN

PUT localhost:9200/exam/product/124

Params Authorization Headers (8) Body Pre-rec

none form-data x-www-form-urlencoded raw

```
1 {
2     "brand":"apple",
3     "name": "iphone 8",
4     "color": "blue",
5     "date":"2017"
6 }
```

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2     "_index": "exam",
3     "_type": "product",
4     "_id": "124",
5     "_version": 1,
6     "result": "created",
7     "_shards": {
```

https://localhost:44390/api/TodoItems

PUT localhost:9200/exam/product/124

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2     "brand":"apple",
3     "name": "iphone 8",
4     "color": "blue",
5     "date":"2017",
6     "camera_resolution":"12mp"
7 }
```

Body Cookies Headers (6) Test Results Status: 200

Pretty Raw Preview Visualize JSON

```
1 {
2     "_index": "exam",
3     "_type": "product",
4     "_id": "124",
5     "_version": 2,
6     "result": "updated",
7     "_shards": {
```

POST METHOD

https://localhost:44390/api/TodoItems

Save

POST | localhost:9200/exam/product | Send

Params | Authorization | Headers (8) | Body | Pre-request Script | Tests | Settings | Cookies

none | form-data | x-www-form-urlencoded | raw | binary | GraphQL | JSON | Beautify

```
1 {
2     "brand":"apple",
3     "name": "galaxy s6",
4     "color": "black",
5     "date":"2014"
6 }
```

Body | Cookies | Headers (7) | Test Results | Status: 201 Created | Time: 33 ms | Size: 927 B | Save Response

Pretty | Raw | Preview | Visualize | JSON

```
1 {
2     "_index": "exam",
3     "_type": "product",
4     "_id": "1V3wfnwBgkfNxnN50LUf",
5     "_version": 1,
6     "result": "created",
7     "_shards": {
```

QUERY WITH POSTMAN

GET localhost:9200/exam/product/123 Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

Body Cookies Headers (6) Test Results Status: 200 OK Time: 22 ms Size: 829 B Save Response

Pretty Raw Preview Visualize JSON


```
{
    "_index": "exam",
    "_type": "product",
    "_id": "123",
    "_version": 1,
    "_seq_no": 0,
    "_primary_term": 1,
```

QUERY WITH POSTMAN

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize JSON


```
{
    "_index": "exam",
    "_type": "product",
    "_id": "123",
    "_version": 1,
    "_seq_no": 0,
    "_primary_term": 1,
    "found": true,
    "_source": {
        "brand": "apple",
        "name": "iphone x",
        "color": "black",
        "date": "2018"
    }
```


https://localhost:44390/api/TodoItems

Save

GET localhost:9200/exam/product/123?_source=false Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

Body Cookies Headers (6) Test Results Status: 200 OK Time: 32 ms Size: 769 B Save Response

Pretty Raw Preview Visualize JSON


```
{
    "_index": "exam",
    "_type": "product",
    "_id": "123",
    "_version": 1,
    "_seq_no": 0,
    "_primary_term": 1,
    "found": true
}
```

QUERY WITH POSTMAN

https://localhost:44390/api/TodoItems

Save

GET localhost:9200/exam/product/123?_source_includes=color,name Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

KEY	VALUE	DESCRIPTION
_source_includes	color,name	
Key	Value	Description

Bulk Edit

Body Cookies Headers (6) Test Results Status: 200 OK Time: 10 ms Size: 799 B Save Response

Pretty Raw Preview Visualize JSON


```
{
    "_index": "exam",
    "_type": "product",
    "_id": "123",
    "_version": 1,
    "_seq_no": 0,
    "_primary_term": 1,
    "found": true,
    "_source": {
        "color": "black",
        "name": "iphone x"
    }
}
```

DELETE COMMAND WITH POSTMAN

https://localhost:44390/api/TodoItems

GET localhost:9200/exam/product/123

Params Authorization Headers (6) Body Pre

Query Params

KEY	V
Key	V

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize JSON

```
1  {
2      "_index": "exam",
3      "_type": "product",
4      "_id": "123",
5      "_version": 1,
6      "_seq_no": 0,
7      "_primary_term": 1,
8      "found": true,
9      "_source": {
10         "brand": "apple",
11         "name": "iphone x",
12         "color": "black",
13         "date": "2018"
```

https://localhost:44390/api/TodoItems

DELETE localhost:9200/exam/product/123

Params Authorization Headers (6) Body Pre-re

Query Params

KEY	VALU
Key	Valu

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize JSON

```
1  {
2      "_index": "exam",
3      "_type": "product",
4      "_id": "123",
5      "_version": 2,
6      "result": "deleted",
7      "_shards": {
8          "total": 2,
9          "successful": 1,
10         "failed": 0
11     },
12     "_seq_no": 4,
13     "_primary_term": 1
```

https://localhost:44390/api/TodoItems

GET localhost:9200/exam/product/123

Params Authorization Headers (6) Body Pre-re

Query Params

KEY	VAL
Key	Val

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize JSON

```
1  {
2      "_index": "exam",
3      "_type": "product",
4      "_id": "123",
5      "found": false
6  }
```

UPDATE COMMAND WITH POSTMAN

https://localhost:44390/api/TodoItems

GET localhost:9200/exam/product/124

Params Authorization Headers (6) Body Pre-request Scri

Query Params

KEY	VALUE
Key	Value

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize JSON


```
{
    "_index": "exam",
    "_type": "product",
    "_id": "124",
    "_version": 2,
    "_seq_no": 2,
    "_primary_term": 1,
    "found": true,
    "_source": {
        "brand": "apple",
        "name": "iphone 8",
        "color": "blue",
        "date": "2017",
```


https://localhost:44390/api/TodoItems

POST localhost:9200/exam/product/124/_update

Params Authorization Headers (8) Body Pre-reque

none form-data x-www-form-urlencoded raw


```
{
    "doc":{
        "color":"black"
    }
}
```


Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize JSON


```
    "_index": "exam",
    "_type": "product",
    "_id": "124",
    "_version": 3,
    "result": "updated",
    "_shards": {
        "total": 2,
        "successful": 1,
        "failed": 0
```


https://localhost:44390/api/TodoItems

GET localhost:9200/exam/product/124

Params Authorization Headers (6) Body

none form-data x-www-form-urlencoded

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize JSO


```
    "_seq_no": 6,
    "_primary_term": 1,
    "found": true,
    "_source": {
        "brand": "apple",
        "name": "iphone 8",
        "color": "black",
        "date": "2017",
        "camera_resolution": "12mp"
```

DELETE COMMAND WITH POSTMAN

https://localhost:44390/api/TodoItems

PUT localhost:9200/exam/product/125

Params Authorization Headers (8) Body Pre-requ

none form-data x-www-form-urlencoded raw

```
1 {
2   "brand":"samsung",
3   "name":"s5",
4   "color":"pink",
5   "date":"2015"
6 }
```

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2     "_index": "exam",
3     "_type": "product",
4     "_id": "125",
5     "_version": 1,
6     "_seq_no": 7,
7     "_primary_term": 1,
8     "found": true,
9     "_source": {
```

https://localhost:44390/api/TodoItems

POST localhost:9200/exam/product/_delete_by_query

Params Authorization Headers (8) Body Pre-request

none form-data x-www-form-urlencoded raw

```
1 {
2     "query": {
3         "match": {
4             "brand":"samsung"
5         }
6     }
```

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2     "_index": "exam",
3     "_type": "product",
4     "_id": "125",
5     "_version": 1,
6     "_seq_no": 7,
7     "_primary_term": 1,
8     "found": true,
9     "_source": {
```

POST localhost:9200/exam/product/_delete_by_query

Params Authorization Headers (8) Body Pre-request S

none form-data x-www-form-urlencoded raw

```
1 {
2     "query": {
3         "match": {
4             "brand":"samsung"
5         }
6     }
```

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2     "took": 261,
3     "timed_out": false,
4     "total": 1,
5     "deleted": 1,
6     "batches": 1,
7     "version_conflicts": 0,
8     "noops": 0,
9     "retries": {
```

SEARCH COMMAND WITH POSTMAN

GET localhost:9200/exam/product/_search?q=brand:apple

Params Authorization Headers (8) Body Pre-request Script

none form-data x-www-form-urlencoded raw binary

```
{
    "query": {
        "match": {
            "brand":"samsung"
        }
    }
```

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "took": 6,
    "timed_out": false,
    "_shards": {
        "total": 1,
        "successful": 1,
        "skipped": 0,
        "failed": 0
    },
```

```
    },
    "hits": {
        "total": {
            "value": 2,
            "relation": "eq"
        },
        "max_score": 0.13353139,
        "hits": [
            {
                "_index": "exam",
```

Pretty Raw Preview Visualize JSON

```
        "max_score": 0.13353139,
        "hits": [
            {
                "_index": "exam",
                "_type": "product",
                "_id": "1V3wfnwBgkfNxnN50LUf",
                "_score": 0.13353139,
                "_source": {
                    "brand": "apple",
```

Pretty Raw Preview Visualize JSON

```
                "_source": {
                    "brand": "apple",
                    "name": "galaxy s6",
                    "color": "black",
                    "date": "2014"
                }
            },
            {
                "_index": "exam",
```

Pretty Raw Preview Visualize JSON

```
                "_type": "product",
                "_id": "124",
                "_score": 0.13353139,
                "_source": {
                    "brand": "apple",
                    "name": "iphone 8",
                    "color": "black",
                    "date": "2017",
                    "camera_resolution": "12mp"
```

GET ALL INFORMATION IN THE DATABASE

GET localhost:9200/_all

Params Authorization Headers (6) Body Pre-request S

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON


```
{
    "exam": {
        "aliases": {},
        "mappings": {
            "properties": {
                "brand": {
                    "type": "text",
                    "fields": {
                        "keyword": {
                            "type": "keyword",
                            "ignore_above": 256
                        }
                    }
                },
                "camera_resolution": {
                    "type": "text",
                    "fields": {
                        "keyword": {
```

ELASTIC SEARCH ASP.NET INTEGRATION

- You can install NEST from the package manager console.

```
PM> Install-Package NEST
```

- Alternatively, simply search for NEST in the package manager UI.

CONNECTION

- **Connecting to a single node**

```
var node = new Uri("http://localhost:9200");
var settings = new ConnectionSettings(node);
var client = new ElasticClient(settings);
```

- **Connecting to multiple nodes using a connection pool**

```
var nodes = new Uri[]
{
        new Uri("http://myserver1:9200"),
        new Uri("http://myserver2:9200"),
        new Uri("http://myserver3:9200")
};

var pool = new StaticConnectionPool(nodes);
var settings = new ConnectionSettings(pool);
var client = new ElasticClient(settings);
```

INDEXING

```
var tweet = new Tweet
{
    Id = 2,
    User = "kimchy",
    PostDate = new DateTime(2009, 11, 15),
    Message = "Trying out NEST, so far so good?"
};

var response1 = client.Index(tweet, idx => idx.Index("mytweetindex"));
```

All the calls have async variants:

var response = await client.IndexAsync(tweet, idx => idx.Index("mytweetindex"));
// awaits a Task<IndexResponse>

GETTING A DOCUMENT

```
var response2 = await client.GetAsync<Tweet>(2, idx => idx.Index("mytweetindex"));
var tweetData = response2.Source; // the original document
```

```
0 references
public async Task<IActionResult> Index()
{
    var node = new Uri("http://localhost:9200");
    var settings = new ConnectionSettings(node);
    var client = new ElasticClient(settings);

    var tweet = new Tweet
    {
        Id = 2,
        User = "kimchy",
        PostDate = new DateTime(2009, 11, 15),
        Message = "Trying out NEST, so far so good?"
    };

    var response1 = client.Index(tweet, idx => idx.Index("mytweetindex"));

    var response2 = await client.GetAsync<Tweet>(2, idx => idx.Index("mytweetindex"));
    var tweetData = response2.Source; // the original document


    return View();
}
```

SEARCH

```
public IActionResult Search()
{
    var node = new Uri("http://localhost:9200");
    var settings = new ConnectionSettings(node);
    var client = new ElasticClient(settings);

    var response = client.Search<Tweet>(s => s
.Index("mytweetindex") //or specify index via settings.DefaultIndex("mytweetindex");
.From(0)
.Size(10)
.Query(q => q
.Term(t => t.User, "kimchy") || q
.Match(mq => mq.Field(f => f.User).Query("nest"))
)
);
    return View();
}
```

UPDATE

```
0 references
public IActionResult Update()
{
    var node = new Uri("http://localhost:9200");
    var settings = new ConnectionSettings(node);
    var client = new ElasticClient(settings);

    var response = client.Update<Tweet>(2, x => x.Index("mytweetindex").Doc(new Tweet { Message = "güncellendi" }));
    return View();
}
```

DELETE

```
0 references
public IActionResult Delete()
{
    var node = new Uri("http://localhost:9200");
    var settings = new ConnectionSettings(node);
    var client = new ElasticClient(settings);

    var response = client.Delete<Tweet>(2, x => x.Index("mytweetindex"));
    return View();

}
```