

ExercAu02-Elastic - Bulk API e Importação

1. Importar os dados na Guia Arquivos para os índices

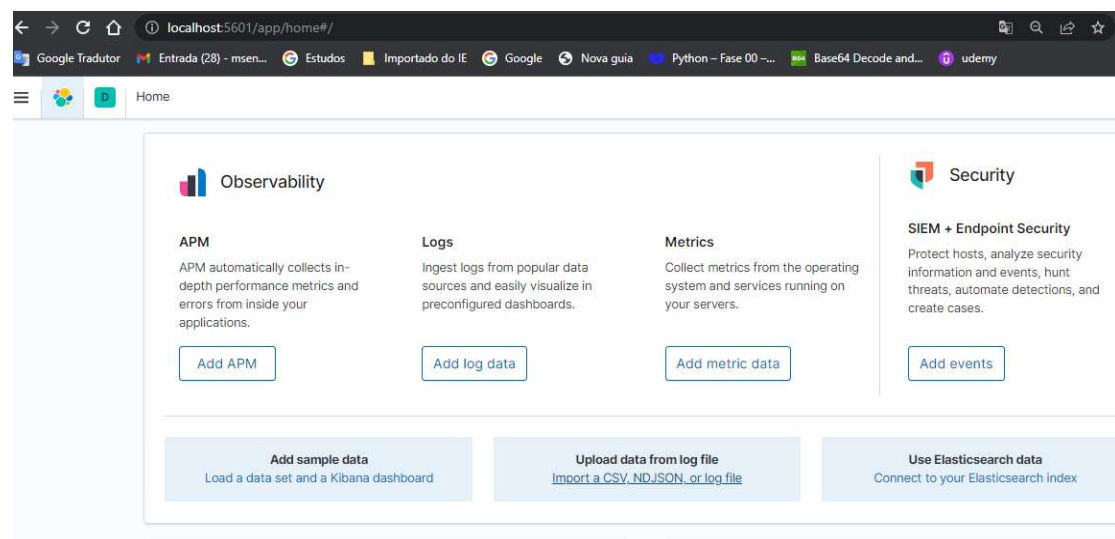
Índice: concessionaria2

dataset/cars.bulk

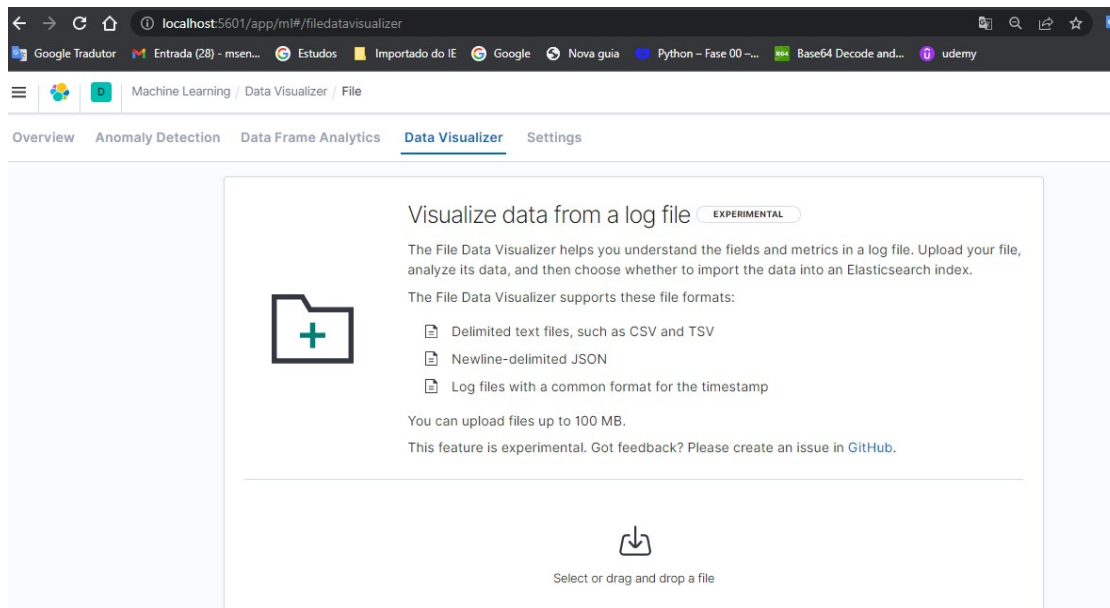
Índice: populacao

dataset/populacaoLA.csv

Resposta:

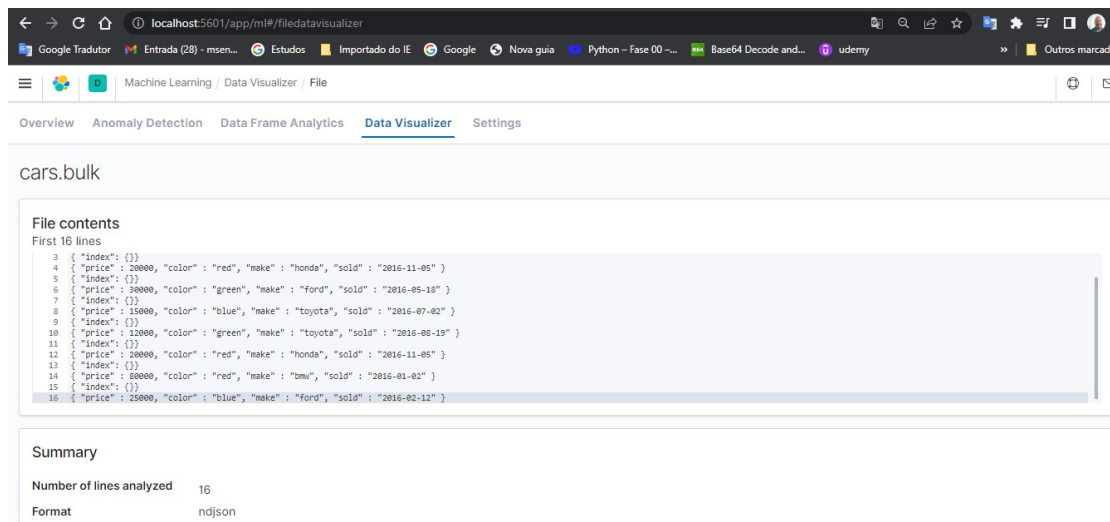


1º Clicar em upload data from log file



2º Clique em Select drag or drop a file

3º Selecione o arquivo desejado.Resultado após importação



Dica: Só com esse arquivo é possível verificar como será a importação.

3º Clique em import:

localhost:5601/app/ml#/filedatavisualizer

Machine Learning / Data Visualizer / File

Overview Anomaly Detection Data Frame Analytics **Data Visualizer** Settings

cars.bulk

File contents

First 16 lines

```
1 { "index": {} }
2 { "price": 10000, "color": "red", "make": "honda", "sold": "2016-10-28" }
3 { "index": {} }
4 { "price": 20000, "color": "red", "make": "honda", "sold": "2016-11-05" }
5 { "index": {} }
6 { "price": 30000, "color": "green", "make": "ford", "sold": "2016-05-18" }
7 { "index": {} }
8 { "price": 15000, "color": "blue", "make": "toyota", "sold": "2016-07-02" }
9 { "index": {} }
10 { "price": 12000, "color": "green", "make": "toyota", "sold": "2016-08-19" }
11 { "index": {} }
12 { "price": 20000, "color": "red", "make": "honda", "sold": "2016-11-05" }
13 { "index": {} }
14 { "price": 80000, "color": "red", "make": "bmw", "sold": "2016-01-02" }
```

Summary

Number of lines analyzed 16

Format ndjson

Import Cancel

Dica : O create index pattern é sempre que vc deseja usar o dashboard do kibana, sempre que for monitorar.

Existirá caso que não precisa, evita demanda de recurso gráfico.

Machine Learning / Data Visualizer / File

Overview Anomaly Detection Data Frame Analytics **Data Visualizer** Settings

cars.bulk

Import data **EXPERIMENTAL**

Simple Advanced

Index name

concessionaria2

☒ Create index pattern

Import

Será mostrado:

File processed Index created Data uploaded Index pa

✓ Import complete

Index	concessionaria2
Index pattern	concessionaria2
Documents ingested	16

View index in Discover Open in Data Visualizer Index Management Index Pattern Management C

2. Executar as consultas

Contar o número de documentos de cada um dos novos índices

Resposta

```
1 GET concessionaria2/_count
2
3
4
5
6
7
8
9
10
11
```

```
1 {
2   "count" : 16,
3   "_shards" : {
4     "total" : 1,
5     "successful" : 1,
6     "skipped" : 0,
7     "failed" : 0
8   }
9 }
10
```

200 - OK 34 ms

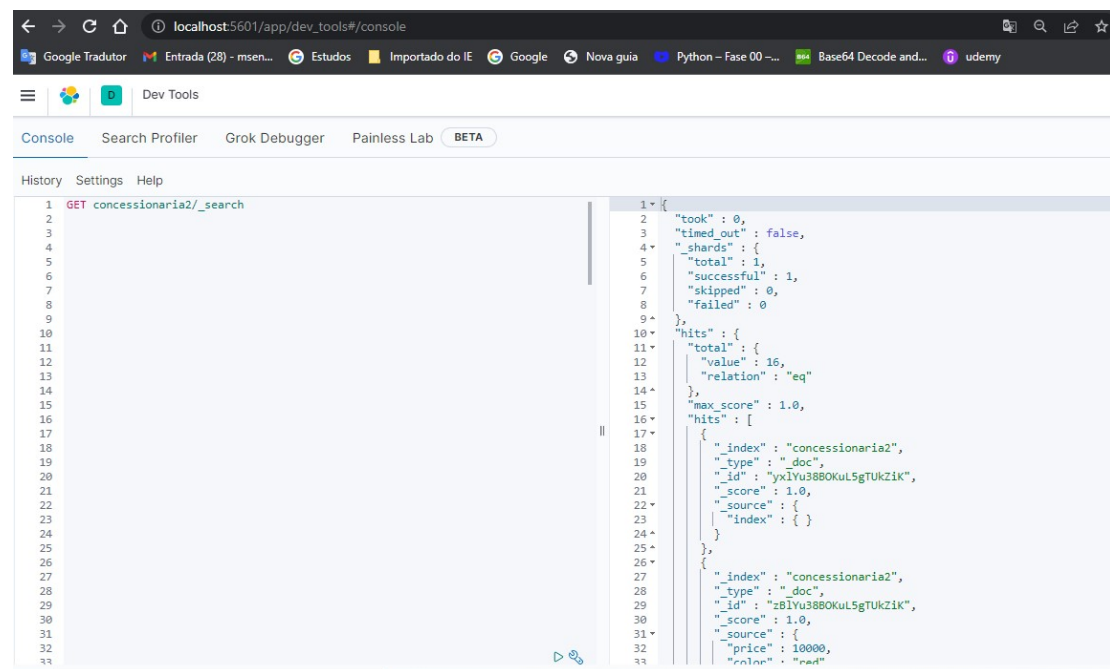
Populacao:

```
1 GET populacao/_count
2
3
4
5
6
7
8
9
10
11
```

```
1 {
2   "count" : 319,
3   "_shards" : {
4     "total" : 1,
5     "successful" : 1,
6     "skipped" : 0,
7     "failed" : 0
8   }
9 }
10
```

Mostrar todos os documentos de cada um dos novos índices (Links para um site externo.)

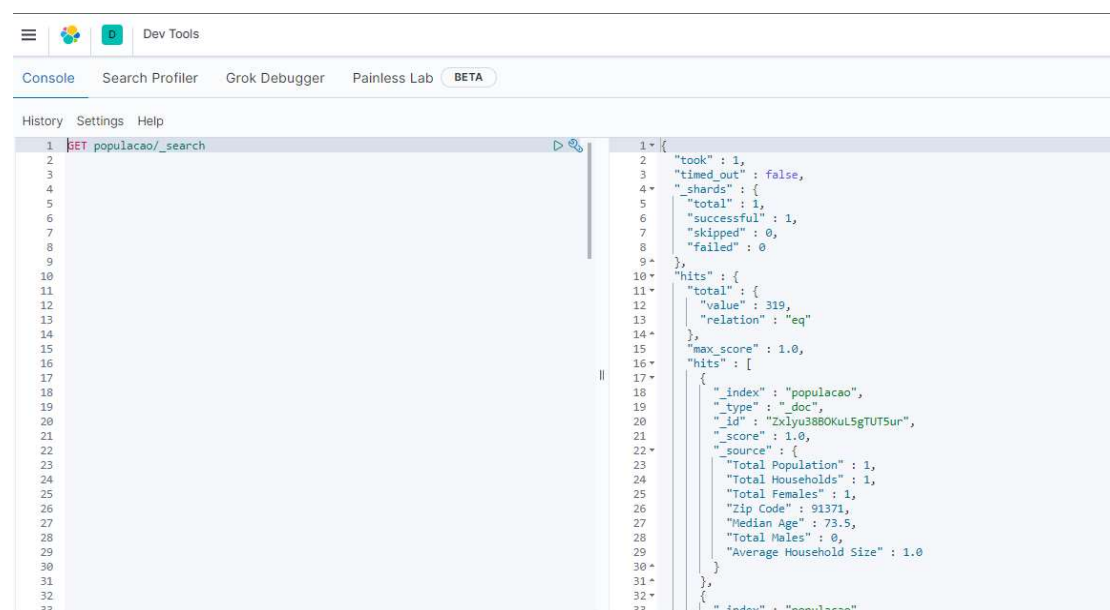
Concessionaria



The screenshot shows a web browser at the URL `localhost:5601/app/dev_tools#/console`. The interface includes a top navigation bar with various tools and a main area with tabs for 'Console', 'Search Profiler', 'Grok Debugger', and 'Painless Lab'. The 'Console' tab is active, displaying a REST client interface. On the left, a GET request is defined for `concessionaria2/_search`. On the right, the JSON response is shown, indicating a successful search with two hits. The response structure is as follows:

```
1 {
2   "took": 0,
3   "timed_out": false,
4   "shards": {
5     "total": 1,
6     "successful": 1,
7     "skipped": 0,
8     "failed": 0
9   },
10  "hits": {
11    "total": {
12      "value": 16,
13      "relation": "eq"
14    },
15    "max_score": 1.0,
16    "hits": [
17      {
18        "index": "concessionaria2",
19        "type": "_doc",
20        "id": "yx1Yu388OKuL5gTUKzIk",
21        "score": 1.0,
22        "_source": {
23          "index": { }
24        }
25      },
26      {
27        "index": "concessionaria2",
28        "type": "_doc",
29        "id": "z81Yu388OKuL5gTUKzIk",
30        "score": 1.0,
31        "_source": {
32          "price": 10000,
33          "color": "red"
34        }
35      }
36    ]
37  }
38 }
```

População



The screenshot shows the same web browser interface as above, but with a GET request to `populacao/_search`. The JSON response shows a successful search with one hit. The response structure is as follows:

```
1 {
2   "took": 1,
3   "timed_out": false,
4   "shards": {
5     "total": 1,
6     "successful": 1,
7     "skipped": 0,
8     "failed": 0
9   },
10  "hits": {
11    "total": {
12      "value": 319,
13      "relation": "eq"
14    },
15    "max_score": 1.0,
16    "hits": [
17      {
18        "index": "populacao",
19        "type": "_doc",
20        "id": "Zx1Yu388OKuL5gTUT5ur",
21        "score": 1.0,
22        "_source": {
23          "Total Population": 1,
24          "Total Households": 1,
25          "Total Females": 1,
26          "Zip Code": 91371,
27          "Median Age": 73.5,
28          "Total Males": 0,
29          "Average Household Size": 1.0
30        }
31      }
32    ]
33  }
34 }
```

3. Clicar no botão de Enviar Tarefa, e enviar o texto: ok