

```
PUT users/_settings
{
  "number_of_replicas": 0
}
```

Nastavení indexů je možné měnit také skrz grafické rozhraní - v Kibaně pod **Stack Management** > **Data** > **Index Management** stačí zvolit daný index a nastavení změnit:

The screenshot shows the Kibana Index Management interface. The left sidebar has a 'Management' section with 'Index Management' selected. The main panel shows the 'users' index selected. A red arrow points to the 'Edit settings' link in the top right. Another red arrow points to the 'users' index name in the list. A third red arrow points to the 'index.number\_of\_replicas' field in the JSON settings editor, which is set to '1'.

## Bulk API - hromadné operace

Operace s dokumenty nemusíme provádět po jednom requestu na dokument, je možné využít `_bulk` api, které umí v rámci jednoho requestu zpracovat několik operací. Slouží k tomu endpoint `_bulk`, před kterým je uveden název indexu. Vstupní data jsou zadávána tak, že na každém řádku je jedna požadovaná operace nebo dokument.

```
POST _bulk
{"index": {"_index": "users", "_id": "5"}}
{"name": "John Brown", "age": 35}
{"create": {"_index": "users"}}
{"name": "Sophia Miller"}
```

Kromě ukládání dokumentů je možné je hromadně upravovat a mazat:

```
POST _bulk
{"update": {"_index": "users", "_id": "5"}}
{"doc": {"age": 37}}
{"delete": {"_index": "users", "_id": "6"}}
```

Jde o efektivní způsob, jak do Elasticsearch uložit větší množství dat, protože se minimalizuje počet navazování spojení po síti. Pokud některá dílčí operace skončí chybou, neznamená to ještě chybu celého requestu - v response se dozvíme, kolik dílčích operací skončilo (ne)úspěšně.

## Modifikace a mazání více dokumentů

Dalším způsobem modifikace dokumentů je využití **Update by query API**, které vyžaduje zapsaný skript popisující logiku změny jednotlivého dokumentu:

```
POST users/_update_by_query
{
  "query": {
    "exists": {
      "field": "age"
    }
  },
  "script" : "ctx._source.age += 2",
}
```

Ke smazání některých dokumentů v rámci indexu lze využít **Delete by query API**. Zde je nutné uvést query, která určí, které dokumenty mají být smazány:

```
POST users/_delete_by_query
{
  "query": {
    "term": {
      "age": 37
    }
  }
}
```

## Reindexace

Index obsahuje několik konfigurací, které není možné po jeho vytvoření změnit. Problém nastává i ve chvíli, kdy do indexu uložíme dokument, jehož pole se uloží jako jeden datový typ (například text), ale později zjistíme, že by měl mít jiný datový typ (například datum) a již není možné jej změnit.

Pokud chceme takovou úpravu provést v rámci Elasticsearch, je nutné nejprve ručně vytvořit nový index se správným nastavením a následně zkopírovat pomocí reindexace data z původního indexu do nového. Pro tuto operaci je k dispozici endpoint `_reindex`:

```
POST _reindex
{
  "source": {
    "index": "users"
  },
  "dest": {
    "index": "users-updated"
  }
}
```

Tento endpoint budeme využívat při změnách v nastavení indexu souvisejících s vyhledáváním. Je možné jej využít i ve specifitějších případech, kdy chceme přeindexovat jen část dokumentů, nebo naopak dostat dokumenty z více indexů do jednoho. V neposlední řadě je možné jej použít pro přenos dat mezi různými clusterly, například ze staging prostředí do lokálníhoho.

Při reindexaci se pouze přenáší data, nepřenáší se nastavení indexu jako je počet shardů, replik, nebo mapping. Pokud tedy chceme, aby měl nově vytvářený index nějaké jiné nastavení, než výchozí, je třeba nejprve vytvořit index s požadovaným nastavením a následně do něj data reindexovat.

## Aliases

V praxi dochází ke změnám v nastavení indexu a následné reindexaci, než ale tyto operace proběhnou, nemůžeme si často dovolit mít index jen částečně funkční, obsahující neúplná data. Pro tento účel je možné využít aliasu, kdy lze jednomu indexu přiřadit alternativní název, pod kterým je dostupný.

Předpokládejme, že data jsou uložena v indexu `user_v1`, ale došlo k změně v nastavení indexu a je tak třeba vytvořit nový index `user_v2` a nakopírovat do něj stávající data. Můžeme si vytvořit alias, který `user`, který bude směřovat vždy na aktuální index:

```
// vytvoření indexu
PUT user_v1

// přidání aliasu
POST _aliases
{
  "actions": [
    {"add": {"index": "user_v1", "alias": "user"}}
  ]
}

// vytvoření nového indexu
PUT user_v2

// překlopení aliasu z starého indexu na nový
POST _aliases
{
  "actions": [
    {"remove": {"index": "user_v1", "alias": "user"}},
    {"add": {"index": "user_v2", "alias": "user"}}
  ]
}
```

```
]
}

// později můžeme nepoužívaný index smazat
DELETE user_v1
```

Tato operace je atomická, takže není třeba se obávat, že by byla data nějakou dobu nedostupná.

## Práce s více indexy

Při vyhledávání v Elasticsearch můžeme specifikovat, ve kterých indexech se má vyhledávat. Dosud jsme vždy uváděli právě jeden index, ale můžeme jich vyjmenovat více (a oddělit názvy čárkou):

```
GET user,customer,subscription/_search
```

Dále můžeme použít znak `*`, který zastoupí libovolné znaky v libovolném počtu. Již jsme jej využili při vytváření sablony, stejně jej můžeme využít pro vyhledávání napříč všemi indexy v Elasticsearch:

```
GET */_search
```

Často se znak `*` používá u indexů, které mají společný prefix. Předpokládejme indexy s objednávkami, přičemž každý rok se vytváří nový index - `order-2018`, `order-2019`, `order-2020`. Pokud bychom chtěli vyhledávat ve všech těchto indexech, bylo by to možné takto:

```
GET order-*/_search
```

Toho lze využít i při vytváření index pattern v Kibaně:

Elastic

Stack Management / Index patterns / Create index pattern

**Ingest** ⓘ

Ingest Node Pipelines

**Data** ⓘ

Index Management

Index Lifecycle Policies

Snapshot and Restore

Rollup Jobs

Transforms

Remote Clusters

**Alerts and Insights** ⓘ

Alerts and Actions

Reporting

**Kibana** ⓘ

[Index Patterns](#)

Saved Objects

Spaces

Advanced Settings

**Stack** ⓘ

License Management

## Create index pattern

An index pattern can match a single source, for example, `filebeat-4-3-22` , or **multiple** data sources, `filebeat-*` .  
[Read documentation](#)

### Step 1 of 2: Define an index pattern

**Index pattern name**

`order-*`

Use an asterisk (\*) to match multiple indices. Spaces and the characters `\`, `/`, `?`, `"`, `<`, `>`, `|` are not allowed.

☐ Include system and hidden indices

✓ Your index pattern matches 3 sources.

<code>order-2018</code>	Index
<code>order-2019</code>	Index
<code>order-2020</code>	Index

Rows per page: 10 ▾

## Smazání indexu

Smazáním indexu dojde také ke smazání všech dokumentů a veškerého nastavení, které obsahuje. V requestu stačí uvést název indexu:

```
DELETE user
```

Alternativně je možné smazat index prostřednictvím grafického rozhraní Kibany:

Elastic

Stack Management / Index Management

**Index Management**

Indices Data Streams Index Templates Component Templates

Update your Elasticsearch indices individually or in bulk. [Learn more.](#)

Include rollout indices Include hidden indices

Manage index

☒ Name

☒ user

Rows per page: 10

**INDEX OPTIONS**

- Show index settings
- Show index mapping
- Show index stats
- Edit index settings
- Close index
- Force merge index
- Refresh index
- Clear index cache
- Flush index
- Freeze index
- Delete index
- Add lifecycle policy

Status	Primaries	Replicas	Docs count	Storage size	Data stream
open	1	0	1	3.9kb	

Reload indices

## Úkol: Indexy, aliasy, reindexace

1. Vytvořte dokument v indexu `manufacturer` s ID `1` a atributy:

- `name`: Apple
- `revenue`: 274515
- `country`: United States

2. Vytvořte další dokument v indexu `manufacturer` - ID tentokrát nechte vygenerovat Elasticsearch:

- `name`: Toyota Group
- `revenue`: 256721
- `country`: Japan

Nyní si představte, že se výrazně zvýšilo vytížení indexu a je nutné mu nastavit více shardů.

3. Vytvořte nový prázdný index `manufacturer-2` s nastavením:

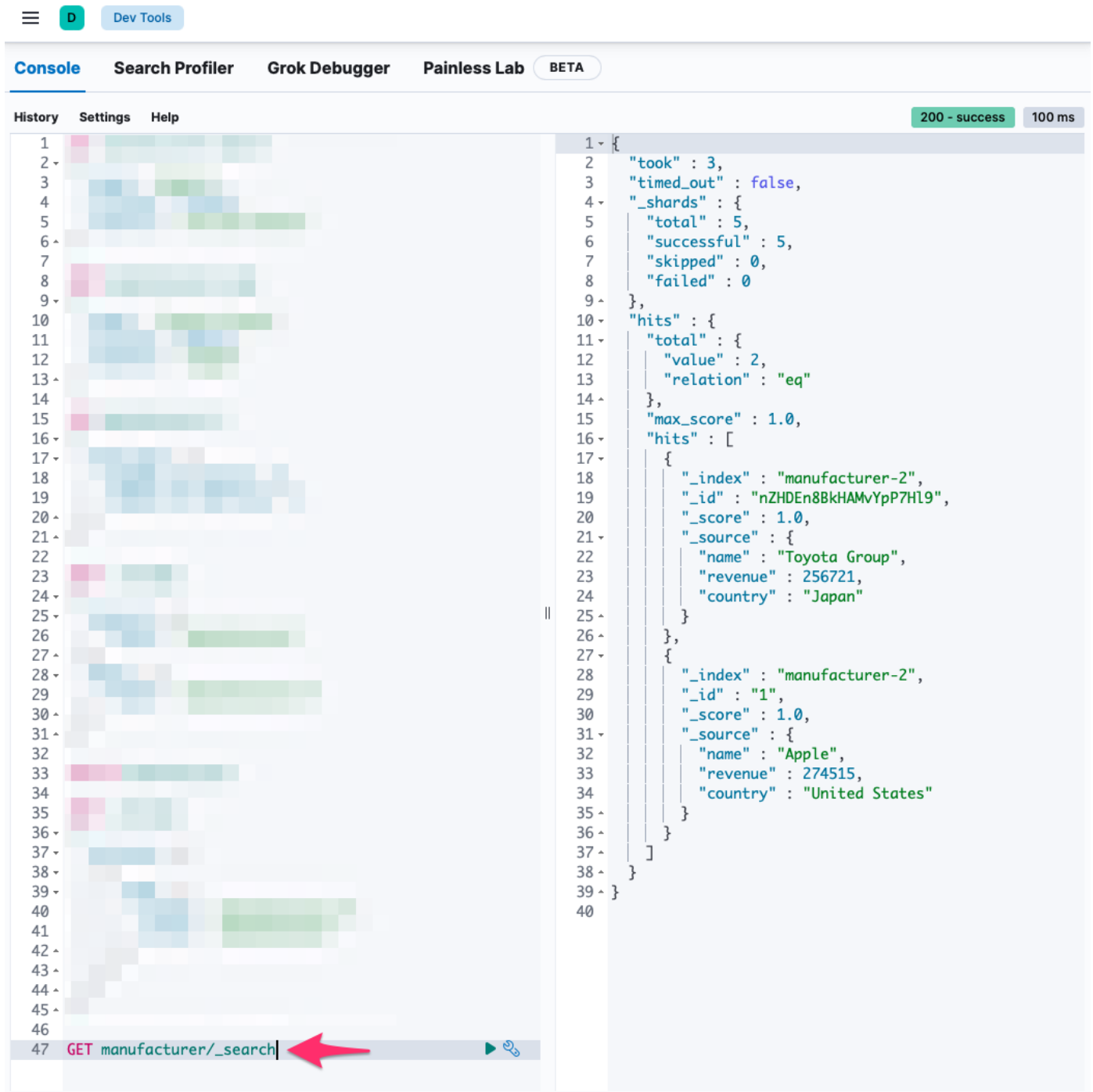
- Počet primary shards (`number_of_shards`): 5
- Počet replica shards `number_of_replicas`: 1

4. Reindexujte data z indexu `manufacturer` do `manufacturer-2`

5. Smažte index `manufacturer`

6. Vytvořte alias `manufacturer`, který směřuje do indexu `manufacturer-2`

7. Vypište všechny dokumenty v `manufacturer`:



The screenshot shows the Elasticsearch DevTools interface. The top navigation bar includes "Console", "Search Profiler", "Grok Debugger", "Painless Lab", and a "BETA" badge. The "Console" tab is active, showing a "History" list on the left and a "Settings" / "Help" section on the right. The main console area displays a successful GET request to `manufacturer/_search` at line 47. The response is a JSON object with the following structure:

```
1 {
2   "took" : 3,
3   "timed_out" : false,
4   "_shards" : {
5     "total" : 5,
6     "successful" : 5,
7     "skipped" : 0,
8     "failed" : 0
9   },
10  "hits" : {
11    "total" : {
12      "value" : 2,
13      "relation" : "eq"
14    },
15    "max_score" : 1.0,
16    "hits" : [
17      {
18        "_index" : "manufacturer-2",
19        "_id" : "nZHDEn8BkHAMvYpP7Hl9",
20        "_score" : 1.0,
21        "_source" : {
22          "name" : "Toyota Group",
23          "revenue" : 256721,
24          "country" : "Japan"
25        }
26      },
27      {
28        "_index" : "manufacturer-2",
29        "_id" : "1",
30        "_score" : 1.0,
31        "_source" : {
32          "name" : "Apple",
33          "revenue" : 274515,
34          "country" : "United States"
35        }
36      }
37    ]
38  }
39 }
```

A red arrow points to the command `GET manufacturer/_search` in the console history.