# Architecting a Search Feature

(in Rails)

Chris Cressman

https://chriscressman.com

# What is the healthiest diet?

Find out what the latest science is saying about your favorite foods to help you make the healthiest choices for you and your family

Watch our free videos on more than 2,000 health and nutrition topics with new videos and articles uploaded every day

Find Out More | Start Watching

or go straight to our latest video

**Michael Greger, M.D. FACLM**
Founder, NutritionFacts.org

Dive into our free Evidence-Based Eating Guide and see just how easy it is to make healthful lifestyle changes

FIND OUT MORE

“Elastic makes the power of search—the ability to instantly find relevant information and insights from large amounts of data—available for a **diverse set of applications and use cases**.”

“Developers **build on top of the Elastic Stack** to apply the power of search to their data and **solve business problems**.”

elastic

Products Learn Company Pricing

Contact Try Free Login

## ELASTIC FOR

**App Search**
Add search to your app

**APM**
Monitor your apps

**SIEM**
Analyze security events

**Site Search**
Add search to your website

**Logs**
Centralize, analyze logs

**Maps**
Explore geo data

**Enterprise Search**
Search all of your content

**Infrastructure**
Centralize, analyze metrics

**Uptime**
Monitor availability

View all products

## ELASTIC (ELK) STACK

**Elasticsearch**
Store, search, analyze

+

**Kibana**
Visualize, navigate, share

## SAAS

**Elastic Cloud**
SaaS offerings

## ON-PREM

**Download**
Self-managed offerings

**ECE**
Centralized orchestration

Get started with Elasticsearch. Watch video

# Get Sta
# Elastics

Search and analyze

Watch Now

## Deploy Hosted Elasticsearch

Learn More

## Getting Started with Kibana

Watch Now

## Looking for Support?

Contact Us

# Elasticsearch Clients

- Java REST Client [7.3] — other versions
- Java API [7.3] — other versions
- JavaScript API [7.x] — other versions
- Ruby API [7.x] — other versions
- Go API
- .NET API [7.x] — other versions
- PHP API [7.0] — other versions
- Perl API
- Python API
- Community Contributed Clients

## Getting Started Videos


Starting Elasticsearch


Introduction to Kibana


Logstash Starter Guide

Open Source

Programming-Language-Specific

APIs

SDKs

Clients

Libraries

Integrations

# Excellent Reference Documentation

`#bulk(arguments = {}) ⇒ Hash` permalink

> **Note:** The body argument is required and cannot be empty.

Perform multiple index, delete or update operations in a single request.

Supports various different formats of the payload: Array of Strings, Header/Data pairs, or the conveniency “combined” format where data is passed along with the header in a single item in a custom `:data` key.

### Examples:

▸ **Perform three operations in a single request, passing actions and data as an array of hashes**

```
client.bulk body: [
  { index: { _index: 'myindex', _type: 'mytype', _id: 1 } },
  { title: 'foo' },

  { index: { _index: 'myindex', _type: 'mytype', _id: 2 } },
  { title: 'foo' },

  { delete: { _index: 'myindex', _type: 'mytype', _id: 3  } }
]
```

▸ **Perform three operations in a single request, passing data in the `:data` option**

```
client.bulk body: [
  { index:  { _index: 'myindex', _type: 'mytype', _id: 1, data: { title: 'foo' } } },
  { update: { _index: 'myindex', _type: 'mytype', _id: 2, data: { doc: { title: 'foo' } } } },
  { delete: { _index: 'myindex', _type: 'mytype', _id: 3  } }
]
```

But how do you **design** a search feature with these tools?

App Search

Domain Model Integration

Ruby on Rails

Make “Books” Searchable

Public
Domain
Paperbacks

Books • Authors • Genres

romantic horror

Filters ▼ Sort ▼

**Frankenstein**

By **Mary Shelley**

Reprinted October 2018

**Strange Case of Dr Jekyll and Mr Hyde**

By **Robert Louis Stevenson**

Reprinted February 2019

◀ Prev Next ▶

## Communication

- Cluster
- Client

## Indexing

- Search Document Design
- Search Document Serialization
- Index Configuration
- Index Creation
- Initial Indexing
- Systematic Indexing

## Querying

- Search Definition
- Search Results
- Pagination
- Sorting
- Filtering

Solution “Framework”

Language/Tool Agnostic

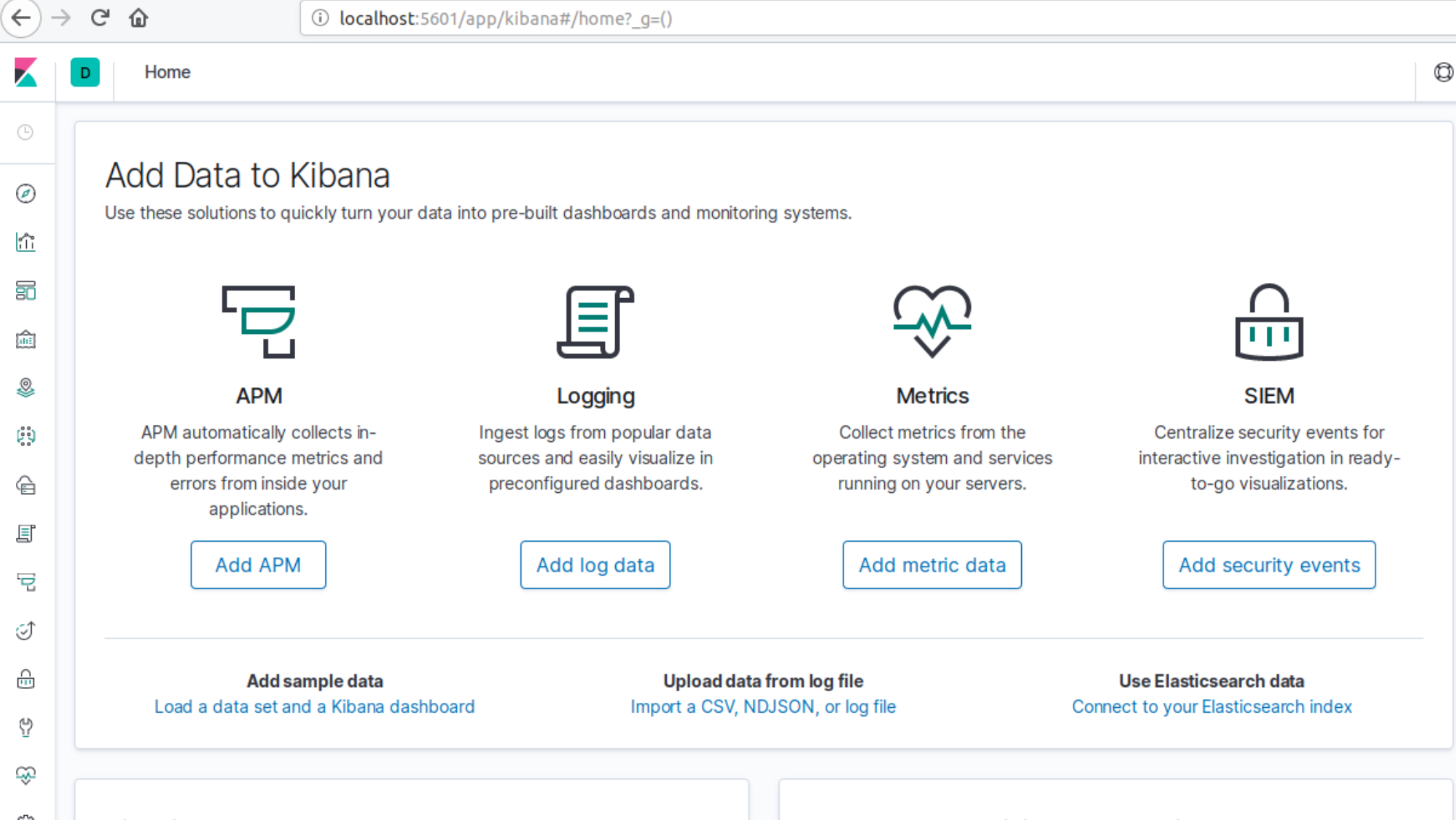
- Communication
  - **Cluster**
  - Client
- Indexing
  - Search Document Design
  - Search Document Serialization
  - Index Configuration
  - Index Creation
  - Initial Indexing
  - Systematic Indexing
- Querying
  - Search Definition
  - Search Results
  - Pagination
  - Sorting
  - Filtering

docker-compose.yml

```
version: '3'
services:
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:7.3.0
    environment:
      - discovery.type=single-node
    volumes:
      - elasticsearch:/usr/share/elasticsearch/data
    ports:
      - 9200:9200
  kibana:
    image: docker.elastic.co/kibana/kibana:7.3.0
    ports:
      - 5601:5601
volumes:
  elasticsearch:
```

```
$ curl localhost:9200

{
  "name" : "c4fabad24d76",
  "cluster_name" : "docker-cluster",
  "cluster_uuid" : "GH_xFJ_aTuSDA9MZTRgMbQ",
  "version" : {
    "number" : "7.3.0",
    "build_flavor" : "default",
    "build_type" : "docker",
    "build_hash" : "de777fa",
    "build_date" : "2019-07-24T18:30:11.767338Z",
    "build_snapshot" : false,
    "lucene_version" : "8.1.0",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```


localhost:5601/app/kibana#/home?_g=()
D
Home
Add Data to Kibana
Use these solutions to quickly turn your data into pre-built dashboards and monitoring systems.
APM
APM automatically collects in-depth performance metrics and errors from inside your applications.
Add APM
Logging
Ingest logs from popular data sources and easily visualize in preconfigured dashboards.
Add log data
Metrics
Collect metrics from the operating system and services running on your servers.
Add metric data
SIEM
Centralize security events for interactive investigation in ready-to-go visualizations.
Add security events
Add sample data
Load a data set and a Kibana dashboard
Upload data from log file
Import a CSV, NDJSON, or log file
Use Elasticsearch data
Connect to your Elasticsearch index

- Communication
  - Cluster
  - **Client**
- Indexing
  - Search Document Design
  - Search Document Serialization
  - Index Configuration
  - Index Creation
  - Initial Indexing
  - Systematic Indexing
- Querying
  - Search Definition
  - Search Results
  - Pagination
  - Sorting
  - Filtering

# REST APIs

edit

Elasticsearch exposes REST APIs that are used by the UI components and can be called directly to configure and access Elasticsearch features.

NOTE

We are working on including more Elasticsearch APIs in this section. Some content might not be included yet.


- API conventions
- cat APIs
- Cluster APIs
- Cross-cluster replication APIs
- Data frame transform APIs
- Document APIs
- Graph Explore API
- Index APIs
- Index lifecycle management APIs
- Ingest APIs
- Info API
- Licensing APIs





+ Elasticsearch Reference: 7.3 (current)
+ Elasticsearch introduction
+ Getting started with Elasticsearch
+ Set up Elasticsearch
+ Upgrade Elasticsearch
+ Aggregations
+ Query DSL
Search across clusters
+ Scripting

Elasticsearch REST APIs

So Any HTTP Client/Library Should Do?

Construct every request

Parse every response

Set up logging/tracing

Manage connections with the nodes in the cluster

Handle retries and errors

# Elasticsearch Clients

- Java REST Client [7.3] — other versions
- Java API [7.3] — other versions
- JavaScript API [7.x] — other versions
- Ruby API [7.x] — other versions
- Go API
- .NET API [7.x] — other versions
- PHP API [7.0] — other versions
- Perl API
- Python API
- Community Contributed Clients

## Getting Started Videos


Starting Elasticsearch


Introduction to Kibana


Logstash Starter Guide

# elasticsearch 7.3.0

Ruby integrations for Elasticsearch (client, API, etc.)

VERSIONS:

**7.3.0** - August 01, 2019 (13 KB)
**7.2.1** - July 25, 2019 (13 KB)
**7.2.0** - June 26, 2019 (13 KB)
**7.1.0** - May 22, 2019 (13 KB)
**7.0.0** - April 30, 2019 (13 KB)

*Show all versions (74 total) →*

RUNTIME DEPENDENCIES (2):

**elasticsearch-api** = 7.3.0

**elasticsearch-transport** = 7.3.0

DEVELOPMENT DEPENDENCIES (16):

**ansi** >= 0

**bundler** >= 0

**cane** >= 0

**elasticsearch-extensions** >= 0

**minitest** >= 0

**minitest-reporters** >= 0

**mocha** >= 0



TOTAL DOWNLOADS
**25,637,931**

FOR THIS VERSION
**141,720**

GEMFILE:

```
gem 'elasticsearch'
```

INSTALL:

```
gem install elastic
```

LICENSE:

**APACHE-2.0**

REQUIRED RUBY VERSION:

```
$ bundle add elasticsearch --version='~> 7.3.0'
```

```
elasticsearch = Elasticsearch::Client.new

cluster_health = elasticsearch.cluster.health

puts JSON.pretty_generate(cluster_health)
{
  "cluster_name": "docker-cluster",
  "status": "green",
  "timed_out": false,
  "number_of_nodes": 1,
  ...
}

elasticsearch.index(...)
elasticsearch.delete(...)
elasticsearch.bulk(...)
elasticsearch.get(...)
elasticsearch.search(...)
```

# REST API

Because this format uses literal `\n`'s as delimiters, please be sure that the JSON actions and sources are not pretty printed. Here is an example of a correct sequence of bulk commands:

```
POST _bulk
{ "index" : { "_index" : "test", "_id" : "1" } }
{ "field1" : "value1" }
{ "delete" : { "_index" : "test", "_id" : "2" } }
{ "create" : { "_index" : "test", "_id" : "3" } }
{ "field1" : "value3" }
{ "update" : {"_id" : "1", "_index" : "test"} }
{ "doc" : {"field2" : "value2"} }
```

Copy as cURL View in Console

The result of this bulk operation is:

```
{
   "took": 30,
   "errors": false,
   "items": [
      {
         "index": {
            "_index": "test",
```

# Ruby API

**#bulk**(arguments = {}) ⇒ Hash permalink

> **Note:** The body argument is required and cannot be empty.

Perform multiple index, delete or update operations in a single request.

Supports various different formats of the payload: Array of Strings, Header/Data pairs, or the conveniency “combined” format where data is passed along with the header in a single item in a custom `:data` key.

**Examples:**

▸ **Perform three operations in a single request, passing actions and data as an array of hashes**

```
client.bulk body: [
  { index: { _index: 'myindex', _type: 'mytype', _id: 1 } },
  { title: 'foo' },

  { index: { _index: 'myindex', _type: 'mytype', _id: 2 } },
  { title: 'foo' },

  { delete: { _index: 'myindex', _type: 'mytype', _id: 3  } }
]
```

▸ **Perform three operations in a single request, passing data in the `:data` option**

```
client.bulk body: [
  { index:  { _index: 'myindex', _type: 'mytype', _id: 1, data: { title: 'foo' } } },
  { update: { _index: 'myindex', _type: 'mytype', _id: 2, data: { doc: { title: 'foo' } } } },
  { delete: { _index: 'myindex', _type: 'mytype', _id: 3  } }
]
```

Idiomatic, Programmatic Access to Your Elasticsearch Cluster

Foundation on Which Deeper Integrations Are Built

Communication
- Cluster
- Client

Indexing
- **Search Document Design**
- Search Document Serialization
- Index Configuration
- Index Creation
- Initial Indexing
- Systematic Indexing

Querying
- Search Definition
- Search Results
- Pagination
- Sorting
- Filtering

The app allows us to **browse** books

# Public Domain Paperbacks

Books • Authors • Genres

Filters ▼ Sort ▼

**Title**

By **Author**

Reprinted Date

**Title**

By **Author**

Reprinted Date

◀ Prev Next ▶

But we want to **search** books

Public
Domain
Paperbacks

Books • Authors • Genres

romantic horror

Filters ▼ Sort ▼

**Frankenstein**

By **Mary Shelley**

Reprinted October 2018

**Strange Case of Dr Jekyll and Mr Hyde**

By **Robert Louis Stevenson**

Reprinted February 2019

◀ Prev Next ▶

Public
Domain
Paperbacks

Books • Authors • Genres

romantic horror

Filters ▼ | Sort ▼

Filters ▼

Author
Foo (3)
Bar (3)

Genre
Foo (3)
Bar (3)

Year Published
Foo (3)
Bar (3)

**Frankenstein**

By **Mary Shelley**

Reprinted October 2018

**Strange Case of Dr Jekyll and Mr Hyde**

By **Robert Louis Stevenson**

Reprinted February 2019

◀ Prev  Next ▶

Sort ▼

**Best Match**

**Date Published**

**Title**

Ruby on Rails (Web application framework)

ActiveRecord (ORM)

## 4 Tables:

books

authors

genres

books_genres

```
ActiveRecord::Schema.define do
  create_table "books" do |t|
    t.string   "name"
    t.string   "image"
    t.text     "description"
    t.string   "isbn"
    t.date     "published_at"
    t.integer  "author_id"
    t.datetime "created_at"
    t.datetime "updated_at"
  end
end
```

```
ActiveRecord::Schema.define do
  create_table "authors" do |t|
    t.string   "name"
    t.string   "image"
    t.text     "description"
    t.datetime "created_at"
    t.datetime "updated_at"
  end
end
```

```
ActiveRecord::Schema.define do
  create_table "genres" do |t|
    t.string   "name"
    t.string   "image"
    t.text     "description"
    t.datetime "created_at"
    t.datetime "updated_at"
  end
end
```

```
ActiveRecord::Schema.define do
  create_table "books_genres" do |t|
    t.integer  "book_id"
    t.integer  "genre_id"
  end
end
```

## 3 Models:

Book

Author

Genre

```ruby
class Book < ApplicationRecord
  belongs_to :author
  has_and_belongs_to_many :genres
end
```

```ruby
class Author < ApplicationRecord
  has_many :books
end
```

```ruby
class Genre < ApplicationRecord
  has_and_belongs_to_many :books
end
```

We want to search **books**

**Views** of a book

## authors

| id | name | image | description | created_at | updated_at |
|---|---|---|---|---|---|
| 2 | Robert Louis Stevenson | Robert-louis-stevenson.jpg | Robert ... | 2019-08-20T19:12:55.322Z | 2019-08-20T19:12:55.322Z |

## books

| id | author_id | name | image | description | isbn | published_at | created_at | updated_at |
|---|---|---|---|---|---|---|---|---|
| 2 | 2 | Strange Case... | dr-jek... | Dr Jekyll ... | 284998401-9 | 2018-01-01 | 2019-08-20T19... | 2019-08-20T19... |

## books_genres

| book_id | genre_id |
|---|---|
| 2 | 1 |
| 2 | 2 |

## genres

| id | name | image | description | create_at | updated_at |
|---|---|---|---|---|---|
| 1 | Horror | horror.jpg | Horror is a genre of spec... | 2019-08-20T19:12:55.345Z | 2019-08-20T19:12:55.345Z |
| 2 | Science Fiction | science-fiction.jpg | Science fiction (Sci-Fi)... | 2019-08-20T19:12:55.351Z | 2019-08-20T19:12:55.351Z |

```
book = Book.find_by(
  name: 'Strange Case of Dr Jekyll and Mr Hyde'
)

book.isbn
"284998401-9"

book.author.id
2

book.author.name
"Robert Louis Stevenson"

book.genres.map(&:id)
[1, 2]

book.genres.map(&:name)
["Horror", "Science Fiction"]
```

What we think of as a book is a **view** that involves **multiple models/tables** and must be constructed

We must **design** an additional **view** of a book that's suitable for searching

A **search document**

## Considerations:

Filters

Query

Sorts

Aggregations

Results

Public
Domain
Paperbacks

Books • Authors • Genres

romantic horror

Filters ▼ | Sort ▼

Filters ▼

Author
Foo (3)
Bar (3)

Genre
Foo (3)
Bar (3)

Year Published
Foo (3)
Bar (3)

**Frankenstein**

By **Mary Shelley**

Reprinted October 2018

**Strange Case of Dr Jekyll and Mr Hyde**

By **Robert Louis Stevenson**

Reprinted February 2019

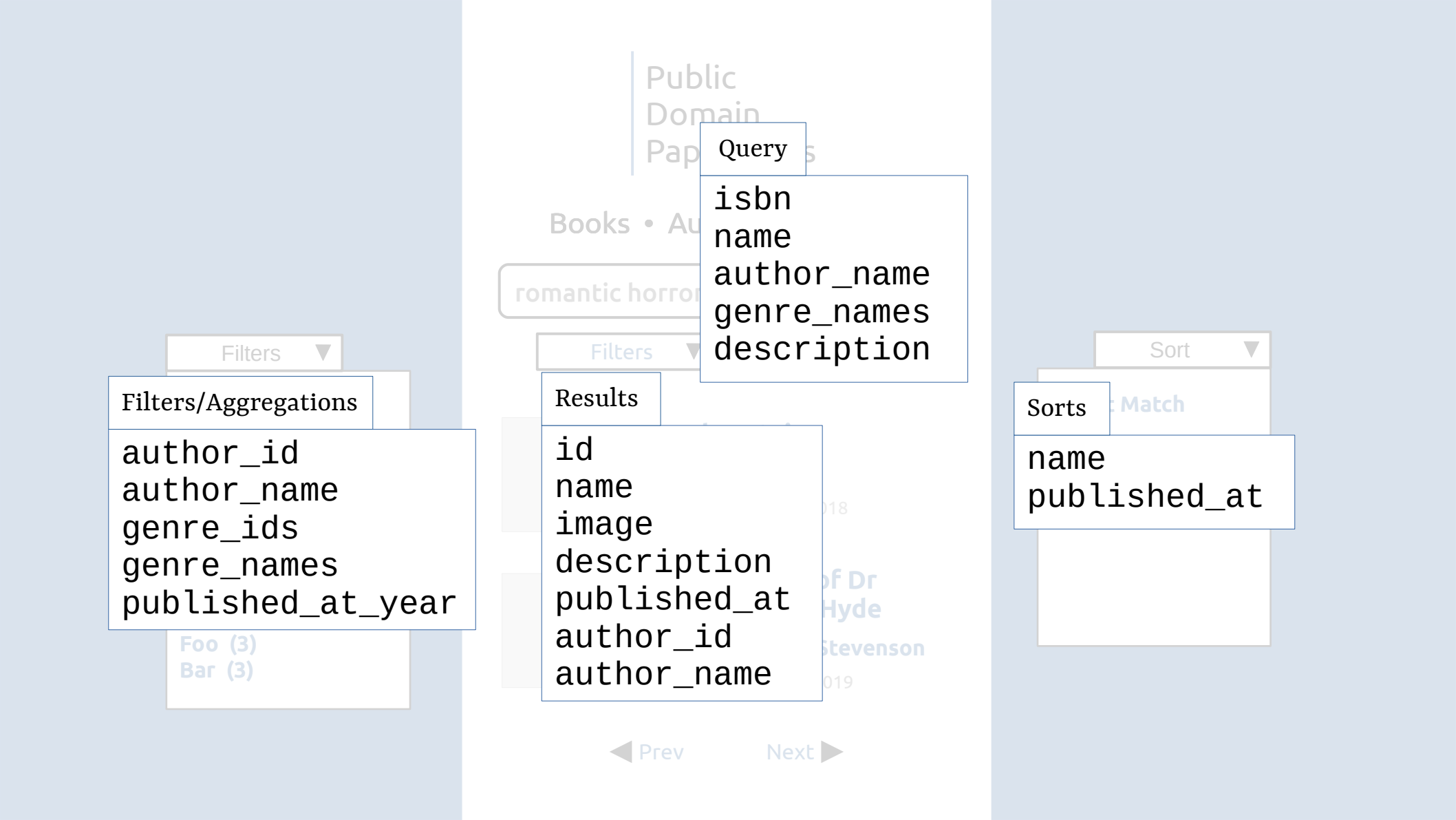
◀ Prev  Next ▶

Sort ▼

**Best Match**

**Date Published**

**Title**


Query
isbn
name
author_name
genre_names
description
Filters/Aggregations
author_id
author_name
genre_ids
genre_names
published_at_year
Results
id
name
image
description
published_at
author_id
author_name
Sorts
name
published_at

id
name
image
description
isbn
published_at
author_id
published_at_year
author_name
genre_ids
genre_names

id
name
image
description
isbn
published_at
author_id
**published_at_year**
**author_name**
**genre_ids**
**genre_names**

```ruby
class Book < ApplicationRecord
  def published_at_year
    published_at.year
  end

  def author_name
    author.name
  end

  def genre_ids
    genres.map(&:id)
  end

  def genre_names
    genres.map(&:name)
  end
end
```

```
book = Book.second

book.id
2

book.name
"Strange Case of Dr Jekyll and Mr Hyde"

book.author_id
2

book.author_name
"Robert Louis Stevenson"

book.genre_ids
[1, 2]

book.genre_names
["Horror", "Science Fiction"]

book.published_at
Mon, 01 Jan 2018

book.published_at_year
2018

book.description
"Strange Case of Dr Jekyll and Mr Hyde is a gothic novella..."

book.image
"dr-jekyll-and-mr-hyde.jpg"

book.isbn
"398742348-X"
```

- Communication
  - Cluster
  - Client
- Indexing
  - Search Document Design
  - **Search Document Serialization**
  - Index Configuration
  - Index Creation
  - Initial Indexing
  - Systematic Indexing
- Querying
  - Search Definition
  - Search Results
  - Pagination
  - Sorting
  - Filtering

# elasticsearch-model 7.0.0.pre

ActiveModel/Record integrations for Elasticsearch.

VERSIONS:

**7.0.0** - August 21, 2019 (68 KB)
**7.0.0.pre** - April 30, 2019 (68 KB)
**6.1.0** - August 12, 2019 (63 KB)
**6.0.0** - September 06, 2018 (59 KB)
**6.0.0.pre** - August 13, 2018 (58 KB)

*Show all versions (23 total) →*

RUNTIME DEPENDENCIES (3):

**activesupport** > 3

**elasticsearch** > 1

**hashie** >= 0

DEVELOPMENT DEPENDENCIES (19):

**activemodel** > 3

**bundler** >= 0

**cane** >= 0

**elasticsearch-extensions** >= 0

**kaminari** >= 0

**minitest** >= 0



TOTAL DOWNLOADS
**8,023,106**

FOR THIS VERSION
**27,395**

GEMFILE:

```
gem 'elasticsearch-
```

INSTALL:

```
gem install elastic
```

LICENSE:

**APACHE 2**

```
$ bundle add elasticsearch-model --version='~> 7.0.0.pre'
```

```ruby
class Book < ApplicationRecord
  include Elasticsearch::Model
end

Book.import
Book.search(...)

Book.__elasticsearch__
Book.__elasticsearch__.client
Book.__elasticsearch__.client.cluster.health

book = Book.first
book.__elasticsearch__.client.cluster.health

book.as_indexed_json
```

```
{
  "id": 2,
  "name": "Strange Case of Dr Jekyll and Mr Hyde",
  "image": "dr-jekyll-and-mr-hyde.jpg",
  "description": "Strange Case of Dr Jekyll and Mr Hyde...",
  "isbn": "284998401-9",
  "published_at": "2018-01-01",
  "author_id": 2,
  "created_at": "2019-08-20T19:12:56.078Z",
  "updated_at": "2019-08-20T19:12:56.078Z"
}
```

```
{
  "id": 2,
  "name": "Strange Case of Dr Jekyll and Mr Hyde",
  "image": "dr-jekyll-and-mr-hyde.jpg",
  "description": "Strange Case of Dr Jekyll and Mr Hyde...",
  "isbn": "284998401-9",
  "published_at": "2018-01-01",
  "author_id": 2,
  "created_at": "2019-08-20T19:12:56.078Z",
  "updated_at": "2019-08-20T19:12:56.078Z"
}
```

```
{
  "id": 2,
  "name": "Strange Case of Dr Jekyll and Mr Hyde",
  "image": "dr-jekyll-and-mr-hyde.jpg",
  "description": "Strange Case of Dr Jekyll and Mr Hyde...",
  "isbn": "284998401-9",
  "published_at": "2018-01-01",
  "author_id": 2,
  "published_at_year": 2018,
  "author_name": "Robert Louis Stevenson",
  "genre_ids": [
    1,
    2
  ],
  "genre_names": [
    "Horror",
    "Science Fiction"
  ]
}
```

```ruby
class Book < ApplicationRecord
  def as_indexed_json(options={})
    as_json(
      except: [
        :created_at,
        :updated_at
      ],
      methods: [
        :published_at_year,
        :author_name,
        :genre_ids,
        :genre_names
      ]
    )
  end
end
```

```
{
  "id": 2,
  "name": "Strange Case of Dr Jekyll and Mr Hyde",
  "image": "dr-jekyll-and-mr-hyde.jpg",
  "description": "Strange Case of Dr Jekyll and Mr Hyde...",
  "isbn": "284998401-9",
  "published_at": "2018-01-01",
  "author_id": 2,
  "published_at_year": 2018,
  "author_name": "Robert Louis Stevenson",
  "genre_ids": [
    1,
    2
  ],
  "genre_names": [
    "Horror",
    "Science Fiction"
  ]
}
```

| id | author_id | name |
|---|---|---|
| 2 | 2 | Strange Case... |

| id | name |
|---|---|
| 2 | Robert Louis Stevenson |

```
book = Book.find(2)

book.name
'Strange Case...'

book.author_name
'Robert Louis Stevenson'
```

```
book.as_indexed_json
'{"name":"Strange Case...","author_name":"Robert Louis..."}'
```

Now we can create search documents

But first we need a place to put them

- Communication
  - Cluster
  - Client
- Indexing
  - Search Document Design
  - Search Document Serialization
  - **Index Configuration**
  - Index Creation
  - Initial Indexing
  - Systematic Indexing
- Querying
  - Search Definition
  - Search Results
  - Pagination
  - Sorting
  - Filtering

**Configure:**

index name

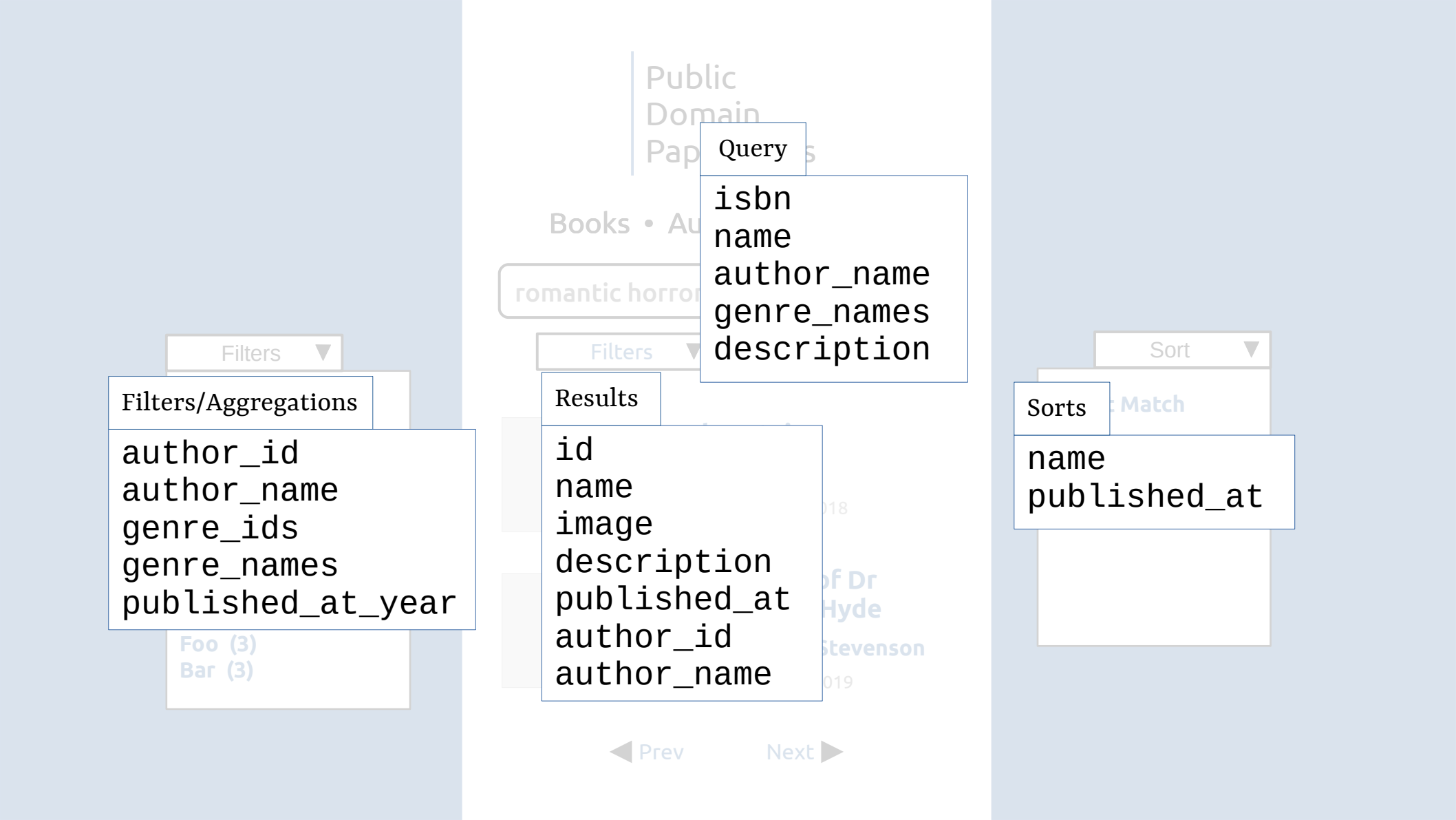
index settings

index mapping

```ruby
class Book < ApplicationRecord

  index_name "#{Rails.env}-books"

  settings number_of_replicas: 0 do
    mapping dynamic: 'false' do
      ...
    end
  end
end
```


Query
isbn
name
author_name
genre_names
description
Filters/Aggregations
author_id
author_name
genre_ids
genre_names
published_at_year
Results
id
name
image
description
published_at
author_id
author_name
Sorts
name
published_at

```
mapping dynamic: 'false' do
  indexes :name, type: 'text' do
    indexes :raw, type: 'keyword'
  end
  indexes :image, type: 'keyword', index: false
  indexes :description, type: 'text'
  indexes :isbn, type: 'keyword'
  indexes :published_at, type: 'date'
  indexes :author_id, type: 'integer'
  indexes :published_at_year, type: 'integer'
  indexes :author_name, type: 'text'
  indexes :genre_ids, type: 'integer'
  indexes :genre_names, type: 'text'
end
```

Communication
- Cluster
- Client

Indexing
- Search Document Design
- Search Document Serialization
- Index Configuration
- **Index Creation**
- Initial Indexing
- Systematic Indexing

Querying
- Search Definition
- Search Results
- Pagination
- Sorting
- Filtering

The create index API allows for providing a mapping definition:

```
PUT test
{
    "settings" : {
        "number_of_shards" : 1
    },
    "mappings" : {
        "properties" : {
            "field1" : { "type" : "text" }
        }
    }
}
```

Copy as cURL View in Console

NOTE

Before 7.0.0, the *mappings* definition used to include a type name. Although specifying types in requests is now deprecated, a type can still be provided if the request parameter include_type_name is set. For more details, please see *Removal of mapping types*.

`#create(arguments = {}) ⇒ Object` permalink

Create an index.

Pass the index `settings` and `mappings` in the `:body` attribute.

**Examples:**

▸ **Create an index with specific settings, custom analyzers and mappings**

```
client.indices.create index: 'test',
                      body: {
                        settings: {
                          index: {
                            number_of_shards: 1,
                            number_of_replicas: 0,
                            'routing.allocation.include.name' => 'node-1'
                          },
                          analysis: {
                            filter: {
                              ngram: {
                                type: 'nGram',
                                min_gram: 3,
                                max_gram: 25
                              }
                            },
                            analyzer: {
                              ngram: {
                                tokenizer: 'whitespace',
                                filter: ['lowercase', 'stop', 'ngram'],
                                type: 'custom'
                              },
                              ngram_search: {
                                tokenizer: 'whitespace',
                                filter: ['lowercase', 'stop'],
                                type: 'custom'
                              }
                            }
                          }
                        },
                        mappings: {
                          properties: {
```

```
$ RAILS_ENV=development bin/rails runner 'Book.create_index!'
$ RAILS_ENV=test bin/rails runner 'Book.create_index!'
```

```
puts Book.
  __elasticsearch__
  .client
  .cat
  .indices(h: 'I')
  .split
  .sort
```

```
.kibana_1
.kibana_task_manager
development-books
test-books
```

Communication
- Cluster
- Client

Indexing
- Search Document Design
- Search Document Serialization
- Index Configuration
- Index Creation
- **Initial Indexing**
- Systematic Indexing

Querying
- Search Definition
- Search Results
- Pagination
- Sorting
- Filtering

```
$ RAILS_ENV=development bin/rails runner 'Book.import'

$ RAILS_ENV=test bin/rails runner 'Book.import'
```

Tables → Object → Search Document →

Elasticsearch Internal Representations

| | _id | name | author_id | author_name | genre_ids | genre_names | published_at_year | isbn |
|---|---|---|---|---|---|---|---|---|
| > | 1 | Frankenstein | 1 | Mary Shelley | 1 | Horror | 2,018 | 788299638-6 |
| > | 2 | Strange Case of Dr Jekyll and Mr Hyde | 2 | Robert Louis Stevenson | 1, 2 | Horror, Science Fiction | 2,018 | 284998401-9 |
| > | 3 | The Time Machine | 3 | H.G. Wells | 2 | Science Fiction | 2,019 | 829471958-7 |
| > | 4 | Treasure Island | 2 | Robert Louis Stevenson | 3 | Adventure | 2,019 | 293231092-5 |

# development-books

Default

This page lists every field in the **development-books** index and the field's associated core type as recorded by Elasticsearch. To change a field type, use the Elasticsearch Mapping API

Fields (16) | Scripted fields (0) | Source filters (0)

Filter | All field types

| Name | Type | Format | Searchable | Aggregatable | Excluded |
|---|---|---|---|---|---|
| _id | string | | ● | ● | |
| _index | string | | ● | ● | |
| _score | number | | | | |
| _source | _source | | | | |
| _type | string | | ● | ● | |
| author_id | number | | ● | ● | |
| author_name | string | | ● | | |
| description | string | | ● | | |
| genre_ids | number | | ● | ● | |
| genre_names | string | | ● | | |
| image | string | | | ● | |
| isbn | string | | ● | ● | |
| name | string | | ● | | |
| name.raw | string | | ● | ● | |
| published_at | date | | ● | ● | |
| published_at_year | number | | ● | ● | |

- Communication
  - Cluster
  - Client
- Indexing
  - Search Document Design
  - Search Document Serialization
  - Index Configuration
  - Index Creation
  - Initial Indexing
  - ~~Systematic Indexing~~ **Let's Search!**
- Querying
  - Search Definition
  - Search Results
  - Pagination
  - Sorting
  - Filtering

- Communication
  - Cluster
  - Client
- Indexing
  - Search Document Design
  - Search Document Serialization
  - Index Configuration
  - Index Creation
  - Initial Indexing
  - Systematic Indexing
- Querying
  - **Search Definition**
  - Search Results
  - Pagination
  - Sorting
  - Filtering

```
Book.search(q, options={})

Book.search(search_definition)
```

# elasticsearch-dsl 0.1.8

A Ruby DSL builder for Elasticsearch

VERSIONS:

**0.1.8** - June 06, 2019 (61.5 KB)
**0.1.7** - March 25, 2019 (60.5 KB)
**0.1.6** - July 29, 2018 (55 KB)
**0.1.5** - January 21, 2017 (54 KB)
**0.1.4** - June 21, 2016 (53.5 KB)

*Show all versions (10 total) →*

DEVELOPMENT DEPENDENCIES (13):

**bundler** >= 0

**cane** >= 0

**elasticsearch** >= 0

**elasticsearch-extensions** >= 0

**minitest** ~> 5

**minitest-reporters** ~> 1

**mocha** >= 0

**pry** >= 0

**rake** ~> 11.1

**shoulda-context** >= 0

**simplecov** >= 0



TOTAL DOWNLOADS
**1,659,117**

FOR THIS VERSION
**50,148**

GEMFILE:

```
gem 'elasticsearch-
```

INSTALL:

```
gem install elastic
```

LICENSE:

**APACHE-2.0**

```
$ bundle add elasticsearch-dsl --version='~> 0.1.8'
```

```
class Book < ApplicationRecord
  def self.search(q, options={})

    search_definition =
      Elasticsearch::DSL::Search.search do
        ...
      end

    __elasticsearch__.search(search_definition)
  end
end
```

```
search_definition =
  Elasticsearch::DSL::Search.search do
    query do
      if q.present?
        multi_match do
          query q
          fields [
            'isbn^10',
            'name^9',
            'author_name^5',
            'genre_names^1',
            'description'
          ]
        end
      else
        match_all
      end
    end
  end
```

```
Book.search(nil)
```

```
{
  "index": "development-books",
  "type": null,
  "body": {
    "query": {
      "match_all": {}
    }
  }
}
```

```
Book.search('romantic horror')
```

```
{
  "index": "development-books",
  "type": null,
  "body": {
    "query": {
      "multi_match": {
        "query": "romantic horror",
        "fields": [
          "isbn^10",
          "name^9",
          "author_name^5",
          "genre_names^1",
          "description"
        ]
      }
    }
  }
}
```

- Communication
  - Cluster
  - Client
- Indexing
  - Search Document Design
  - Search Document Serialization
  - Index Configuration
  - Index Creation
  - Initial Indexing
  - Systematic Indexing
- Querying
  - Search Definition
  - **Search Results**
  - Pagination
  - Sorting
  - Filtering

Public
Domain
Paperbacks

Books • Authors • Genres

romantic horror

Filters ▼ Sort ▼

**Frankenstein**

By **Mary Shelley**

Reprinted October 2018

**Strange Case of Dr Jekyll and Mr Hyde**

By **Robert Louis Stevenson**

Reprinted February 2019

◀ Prev Next ▶

```
Book.search('romantic horror').results.each do |result|
  puts
  puts result.name
  puts result.author_name
End

Frankenstein
Mary Shelley

Strange Case of Dr Jekyll and Mr Hyde
Robert Louis Stevenson
```

```
Book.search('romantic horror').results.class
Elasticsearch::Model::Response::Results

Book.search('romantic horror').records.class
Elasticsearch::Model::Response::Records

Book.search('romantic horror').response.class
Elasticsearch::Model::HashWrapper
```

```
Book.search('romantic horror').results.first.class
Elasticsearch::Model::Response::Result

Book.search('romantic horror').records.first.class
Book

Book.search('romantic horror').results.first.author_name
'Mary Shelley'

Book.search('romantic horror').records.first.author_name
'Mary Shelley'
```

## Results/Records Considerations:

Performance (additional queries)

Staleness

- Communication
  - Cluster
  - Client
- Indexing
  - Search Document Design
  - Search Document Serialization
  - Index Configuration
  - Index Creation
  - Initial Indexing
  - Systematic Indexing
- Querying
  - Search Definition
  - Search Results
  - **Pagination**
  - Sorting
  - Filtering

# kaminari 1.1.1

Kaminari is a Scope & Engine based, clean, powerful, agnostic, customizable and sophisticated paginator for Rails 4+

VERSIONS:

**1.1.1** - October 21, 2017 (21 KB)
**1.1.0** - October 13, 2017 (21 KB)
**1.0.1** - January 19, 2017 (20.5 KB)
**1.0.0** - January 10, 2017 (20 KB)
**1.0.0.rc1** - December 14, 2016 (19.5 KB)

*Show all versions (51 total) →*

RUNTIME DEPENDENCIES (4):

**activesupport** >= 4.1.0

**kaminari-actionview** = 1.1.1

**kaminari-activerecord** = 1.1.1

**kaminari-core** = 1.1.1

DEVELOPMENT DEPENDENCIES (5):

**bundler** >= 1.0.0

**capybara** >= 1.0

**rake** >= 0

**rr** >= 0



TOTAL DOWNLOADS
**45,369,747**

FOR THIS VERSION
**11,225,255**

GEMFILE:

```
gem 'kaminari', '~>
```

INSTALL:

```
gem install kaminar
```

LICENSE:

**MIT**

Gemfile

```
gem "kaminari", "~> 1.1.1"

gem "elasticsearch", "~> 7.3.0"

gem "elasticsearch-model", "~> 7.0.0.pre"

gem "elasticsearch-dsl", "~> 0.1.8"
```

```
Book.search(nil).results.count
4

Book.search(nil).page(1).per(3).results.count
3

Book.search(nil).page(2).per(3).results.count
1

Book.search(nil).page(2).per(3).total_pages
2

Book.search(nil).page(2).per(3).current_page
2

Book.search(nil).page(2).per(3).prev_page
1
```

- Communication
  - Cluster
  - Client
- Indexing
  - Search Document Design
  - Search Document Serialization
  - Index Configuration
  - Index Creation
  - Initial Indexing
  - Systematic Indexing
- Querying
  - Search Definition
  - Search Results
  - Pagination
  - **Sorting**
  - Filtering

Public
Domain
Paperbacks

Books • Authors • Genres

romantic horror

Filters ▼ | Sort ▼

Filters ▼

Author
Foo (3)
Bar (3)

Genre
Foo (3)
Bar (3)

Year Published
Foo (3)
Bar (3)

**Frankenstein**

By **Mary Shelley**

Reprinted October 2018

**Strange Case of Dr Jekyll and Mr Hyde**

By **Robert Louis Stevenson**

Reprinted February 2019

◀ Prev  Next ▶

Sort ▼

**Best Match**

**Date Published**

**Title**

```ruby
class Book < ApplicationRecord
  def self.available_sorts
    {
      best: {
        display_name: 'Best Match',
        field: '_score',
        order: 'desc'
      },
      date: {
        display_name: 'Date Published',
        field: 'published_at',
        order: 'desc'
      },
      title: {
        display_name: 'Title',
        field: 'name.raw',
        order: 'asc'
      }
    }
  end
end
```

```
Book.available_sorts.each do |id, sort|
  puts;
  puts "* #{sort[:display_name]} :#{id}"
  puts "  (#{sort[:field]}, #{sort[:order]})"
End

* Best Match (:best)
  _score, desc

* Date Published (:date)
  published_at, desc

* Title (:title)
  name.raw, asc
```

```
Book.search(nil, sort: :title).results.first.name
"Frankenstein"

Book.search(nil, sort: :date).results.first.name
"The Time Machine"
```

```
class Book < ApplicationRecord
  def self.search(q, options={})
    selected_sort = available_sorts[options.fetch(:sort, :best)]

    search_definition =
      Elasticsearch::DSL::Search.search do
        query do
          ...
        end

        sort do
          by selected_sort[:field], order: selected_sort[:order]
        end
      end

  end
end
```

```
{
  "index": "development-books",
  "type": null,
  "body": {
    "query": {...},
    "sort": [
      {
        "published_at": {
          "order": "desc"
        }
      }
    ]
  }
}
```

- Communication
  - Cluster
  - Client
- Indexing
  - Search Document Design
  - Search Document Serialization
  - Index Configuration
  - Index Creation
  - Initial Indexing
  - Systematic Indexing
- Querying
  - Search Definition
  - Search Results
  - Pagination
  - Sorting
  - **Filtering**

Public
Domain
Paperbacks

Books • Authors • Genres

romantic horror

Filters ▼ | Sort ▼

Filters ▼

Author
Foo (3)
Bar (3)

Genre
Foo (3)
Bar (3)

Year Published
Foo (3)
Bar (3)

**Frankenstein**

By **Mary Shelley**

Reprinted October 2018

**Strange Case of Dr Jekyll and Mr Hyde**

By **Robert Louis Stevenson**

Reprinted February 2019

◀ Prev  Next ▶

Sort ▼

**Best Match**

**Date Published**

**Title**

```
Book.search(nil).response.aggregations.each do |term, values|
  type = term.singularize.titleize
  puts
  values.buckets.each do |bucket|
    puts "#{type} #{bucket['key']} (#{bucket['doc_count']})"
  end
end
```

```
Genre 1 (2)
Genre 2 (2)
Genre 3 (1)
Year 2018 (2)
Year 2019 (2)
Author 2 (2)
Author 1 (1)
Author 3 (1)
```

```
search_definition =
  Elasticsearch::DSL::Search.search do
    query do
      ...
    end

    sort do
      ...
    end

    aggregation :authors do
      terms field: 'author_id'
    end
    aggregation :genres do
      terms field: 'genre_ids'
    end
    aggregation :years do
      terms field: 'published_at_year'
    end
  end
```

```
{
  "index": "development-books",
  "type": null,
  "body": {
    "query": {...},
    "sort": [...],
    "aggregations": {
      "authors": {
        "terms": {
          "field": "author_id"
        }
      },
      "genres": {
        "terms": {
          "field": "genre_ids"
        }
      },
      "years": {
        "terms": {
          "field": "published_at_year"
        }
      }
    }
  }
}
```

```
Book.search(nil).results.count
4

Book.search(nil, author: 2).results.count
2

Book.search(nil, genre: 3).results.count
1

Book.search(nil, year: 2019).results.count
2

Book.search('time travel', year: 2019).results.count
1
```

```ruby
def self.search(q, options={})
  selected_author = options[:author]
  selected_genre = options[:genre]
  selected_year = options[:year]
  selected_sort = available_sorts[options.fetch(:sort, :best)]

  search_definition =
    Elasticsearch::DSL::Search.search do
      query do
        bool do
          filter { term author_id: selected_author } if selected_author
          filter { term genre_ids: selected_genre } if selected_genre
          filter { term published_at_year: selected_year } if selected_year
          must do
            if q.present?
              ...
            else
              match_all
            end
          end
        end
      end
    end

end
```

```
{
  "index": "development-books",
  "type": null,
  "body": {
    "query": {
      "bool": {
        "filter": [
          {
            "term": {
              "author_id": 2
            }
          }
        ],
        "must": [
          {
            "multi_match": {...}
          }
        ]
      }
    },
    "sort": [...],
    "aggregations": {...}
  }
}
```

- Communication
  - Cluster
  - Client
- Indexing
  - Search Document Design
  - Search Document Serialization
  - Index Configuration
  - Index Creation
  - Initial Indexing
  - **Systematic Indexing**
- Querying
  - Search Definition
  - Search Results
  - Pagination
  - Sorting
  - Filtering

```
book_1 = Book.find(1)
search_book_1 = Book.__elasticsearch__.client.get(
                  index: Book.index_name,
                  id: 1
                )

book_1.name
"Frankensteen"

search_book_1.dig('_source', 'name')
"Frankensteen"
```

```
Book
  .find(1)
  .update_attribute(:name, 'Frankenstein')
```

```
book_1 = Book.find(1)
search_book_1 = Book.__elasticsearch__.client.get(
                  index: Book.index_name,
                  id: 1
                )

book_1.name
"Frankenstein"

search_book_1.dig('_source', 'name')
"Frankensteen"
```

# Update, create, or delete a book

```ruby
class Book < ApplicationRecord
  after_save :save_search_book
  after_destroy :destroy_search_book

  def save_search_book(*args)
    SaveSearchBookJob.perform_later(id)
  end

  def destroy_search_book(*args)
    DestroySearchBookJob.perform_later(id)
  end
end
```

```ruby
class SaveSearchBookJob < ApplicationJob
  queue_as :default

  def perform(book_id)
    book = Book.find(book_id)
    book.__elasticsearch__.index_document
  end
end


class DestroySearchBookJob < ApplicationJob
  queue_as :default

  def perform(book_id)
    client = Book.__elasticsearch__.client
    index = Book.index_name
    client.delete(index: index, id: book_id)
  end
end
```

**But:**

You can assign genres to a book without saving the book

You can assign genres to a book from a Genre

```ruby
class Book < ApplicationRecord
  has_and_belongs_to_many :genres,
    after_add: :save_search_book,
    after_remove: :save_search_book

  after_save :save_search_book
  after_destroy :destroy_search_book
end
```

```ruby
class Genre < ApplicationRecord
  has_and_belongs_to_many :books,
    after_add: :save_search_book,
    after_remove: :save_search_book

  def save_search_book(book)
    SaveSearchBookJob.perform_later(book.id)
  end
end
```

And don’t forget Author callbacks
(like I did!)

- Communication
  - Cluster
  - Client
- Indexing
  - Search Document Design
  - Search Document Serialization
  - Index Configuration
  - Index Creation
  - Initial Indexing
  - Systematic Indexing
- Querying
  - Search Definition
  - Search Results
  - Pagination
  - Sorting
  - Filtering

**And Beyond!**

- Improve filtering
  - Conjunctive facets
  - Ranges/sliders
- Improve querying
  - Typo tolerance
  - Spelling correction
  - Bigram matching
  - Phrase matching
- Improve results handling
  - Highlighting
- Improve UX
  - Search as you type, query suggestions, autocomplete
- Dashboards
  - Promote, hide, or otherwise manage results for specific queries
  - Manage synonyms
  - View analytics for searches and clicks
  - Manage relevance (e.g. field boosting)

# Consider Off-the-Shelf Solutions

ELASTIC APP SEARCH

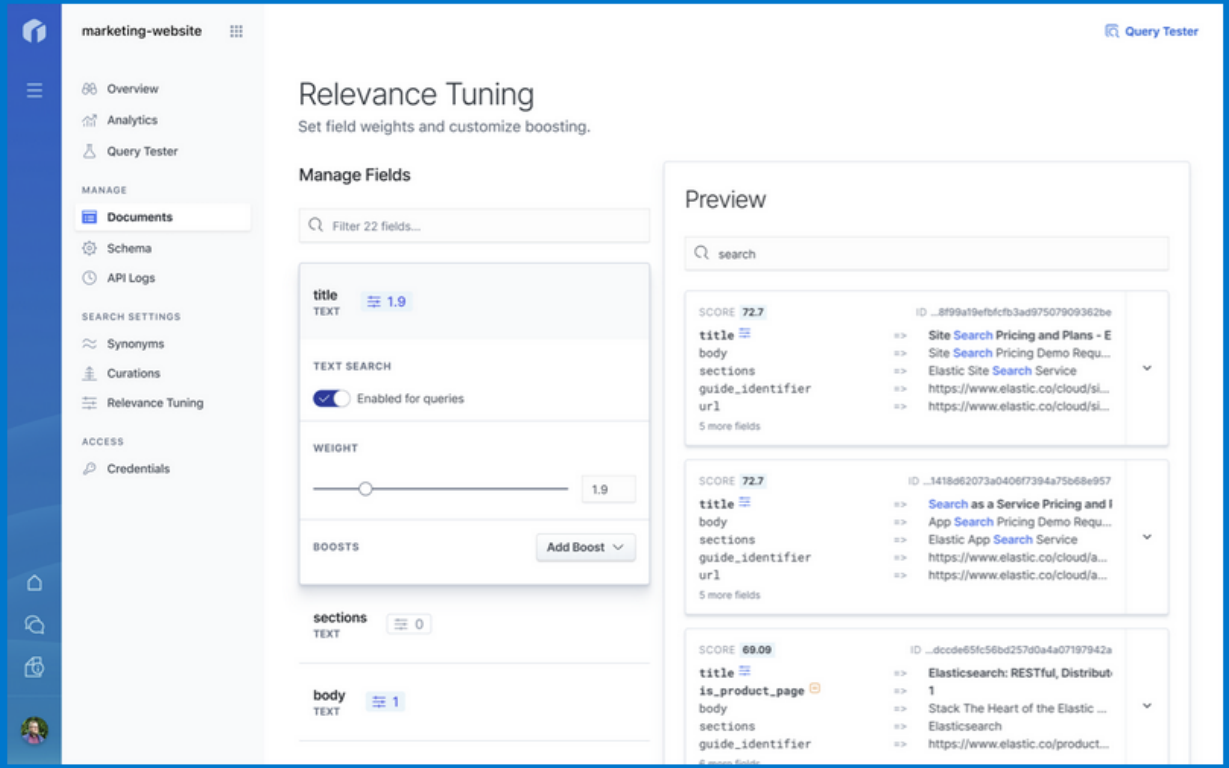
# Advanced search made simple

The curated experience of Elastic App Search brings the focused power of Elasticsearch to a refined set of APIs and intuitive dashboards. Leverage the seamless scalability, tunable relevance controls, thorough documentation, well-maintained clients, and robust analytics to build a leading search experience with ease.

Enter your email

Start Trial

Download the latest version




Looking to build a leading search experience with

Create advanced, user-facing search experiences

Elastic is a search company. Learn how we apply the

But you can build these things yourself

And there are tools to help

But designing them can be hard

I hope this presentation may help

**Framework for app search / domain model integration:**

## Communication

- Cluster
- Client

## Indexing

- Search Document Design
- Search Document Serialization
- Index Configuration
- Index Creation
- Initial Indexing
- Systematic Indexing

## Querying

- Search Definition
- Search Results
- Pagination
- Sorting
- Filtering

# Thank You

Architecting a Search Feature (in Rails)

Chris Cressman

https://chriscressman.com