API usage guide

This documentation is a usage guide for the REST API of the picturesafe-search Acceleration Server.

For a complete reference of all API functions see the "API reference".

Table of Contents

1. Overview

The REST API of the picturesafe-search Acceleration Server provides an easy way to create and manage Elasticsearch indices and execute searches.

It abstracts the complexity of Elasticsearch and provides simple but powerful interfaces for the following features:

- Create/rebuild/delete indices
- Add/delete documents
- Fulltext and field searches
- Complex searches
- Pagination, sorting
- Aggregations/facets

One major focus of the API is the search resource, which hides the technical complexity of Elasticsearch queries and focuses more on WHAT to search for than HOW to search for it.

To achieve this, however, some configurations must be specified during index creation, although some technical complexities of Elasticsearch are also hidden there.

2. Getting started

In this section you will learn the following:

- How to execute an API request and how to authorize it
- How to create an index with specific (field) settings
- How to add documents to the index
- How to perform a simple fulltext search
- How to perform a simple field search
- How to perform a combined search
- How to delete an index

2.1. First request

The queries shown in the examples can be executed with tools like 'curl' or 'Postman'. After the first example, this documentation uses a simplified, neutral format.

The simplest request is to query the API and server version of the picturesafe-search Acceleration server. This is the only request (except the login request) that does not require authorization:

Request with curl:

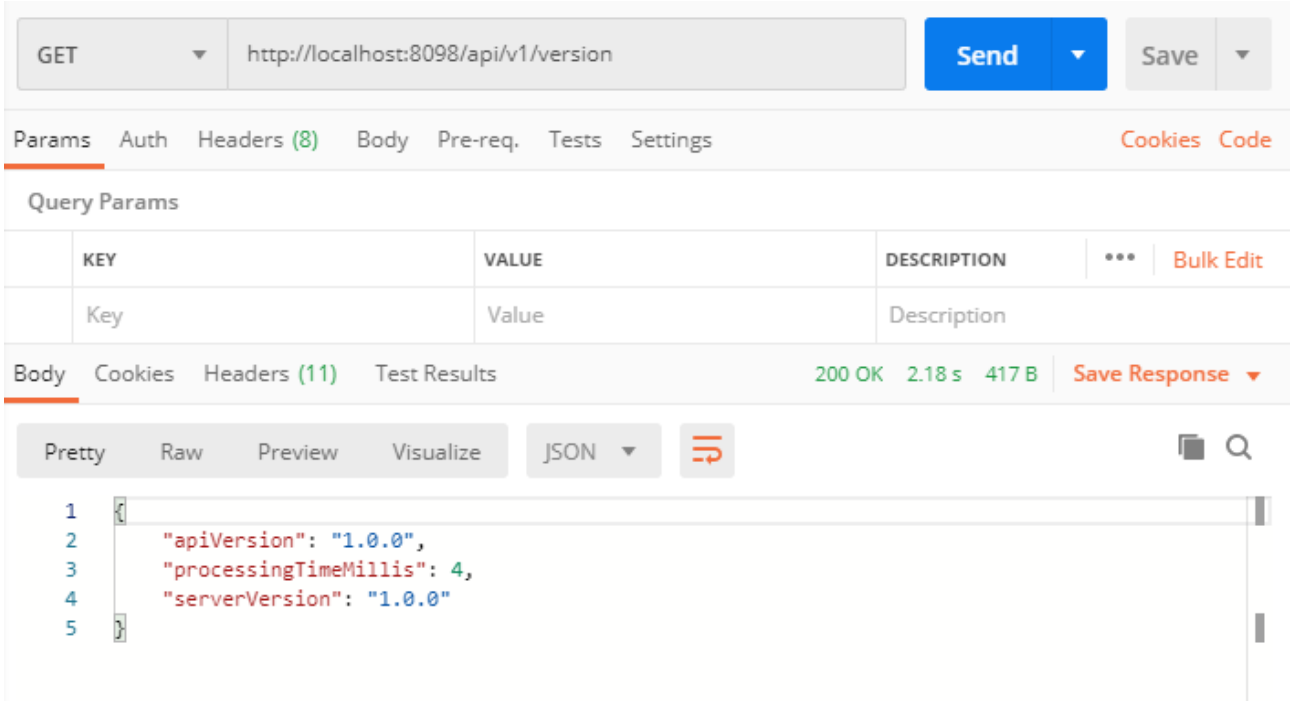
Request

```
curl http://localhost:8080/api/v1/version
```

Response:

```
{
    "apiVersion": "{api-version}",
    "processingTimeMillis": 0,
    "serverVersion": "{server-version}"
}
```

Same request with Postman



If you get an error, check if the picturesafe-search Acceleration Server is started correctly and if the URL used is valid.

2.2. Login and authorization

The REST API uses JSON Web Tokens (JWT) for authorization. You have to provide a valid JWT in the authorization header to access the REST API resources. The login resource provides a JSON Web Token (JWT) based on username and password.

For the login request, you can use your credentials for the picturesafe-search Acceleration Server:

Request

```
POST /api/v1/login
```

```
{
    "username": "myusername",
    "password": "mypassword"
}
```

Response:

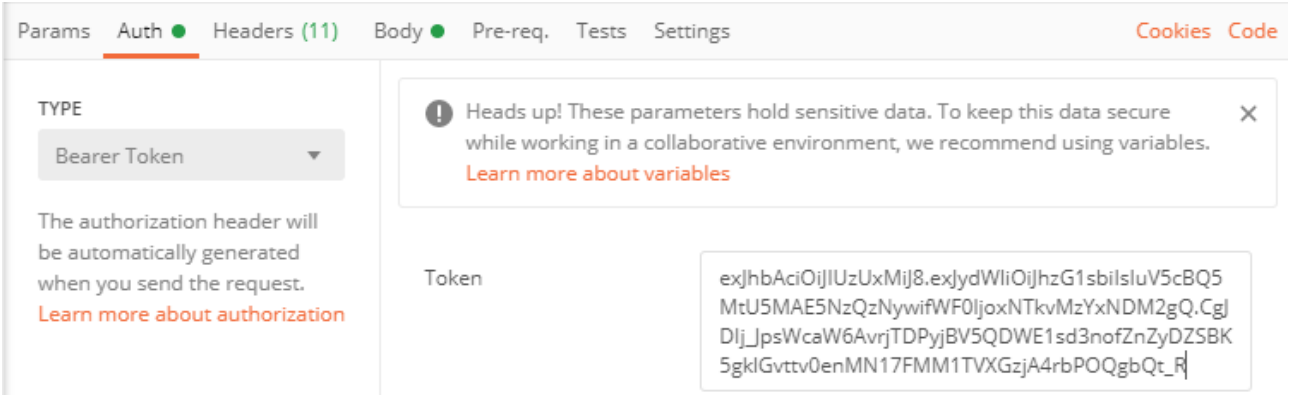
```
{
    "username": "myusername",
    "jwt":
"exJhbAciOiJIUzUxMiJ8.exJydWIiOiJhzG1sbiIsIuV5cBQ5MtU5MAE5NzQzNywifWF0IjoxNTkvMzYxNDM2gQ.CgJDIj_JpsWcaW6AvrjTDPyjBV5QDWE1sd3nofZnZyDZSBK5gklGvttv0enMN17FMM1TVXGzjA4rbPOQgbQt_R"
}
```

For all following requests, you have to provide a valid JWT in the authorization header to access the REST API resources:

Authorization token with curl:

```
curl -H 'Authorization: Bearer <JWT-TOKEN>' ...
```

Authorization token with Postman:



2.3. Create index

The index resource provides the possibility to create an Elasticsearch index with specific settings and to configure fields with specific properties.

Although it is basically possible to create the Elasticsearch index without specific field configurations, certain functions of the picturesafe-search Acceleration Server (such as fulltext search) require a special field configuration.

The following example defines an index with three text fields and a keyword field, where the content of two text fields should be searchable by a fulltext search:

Request

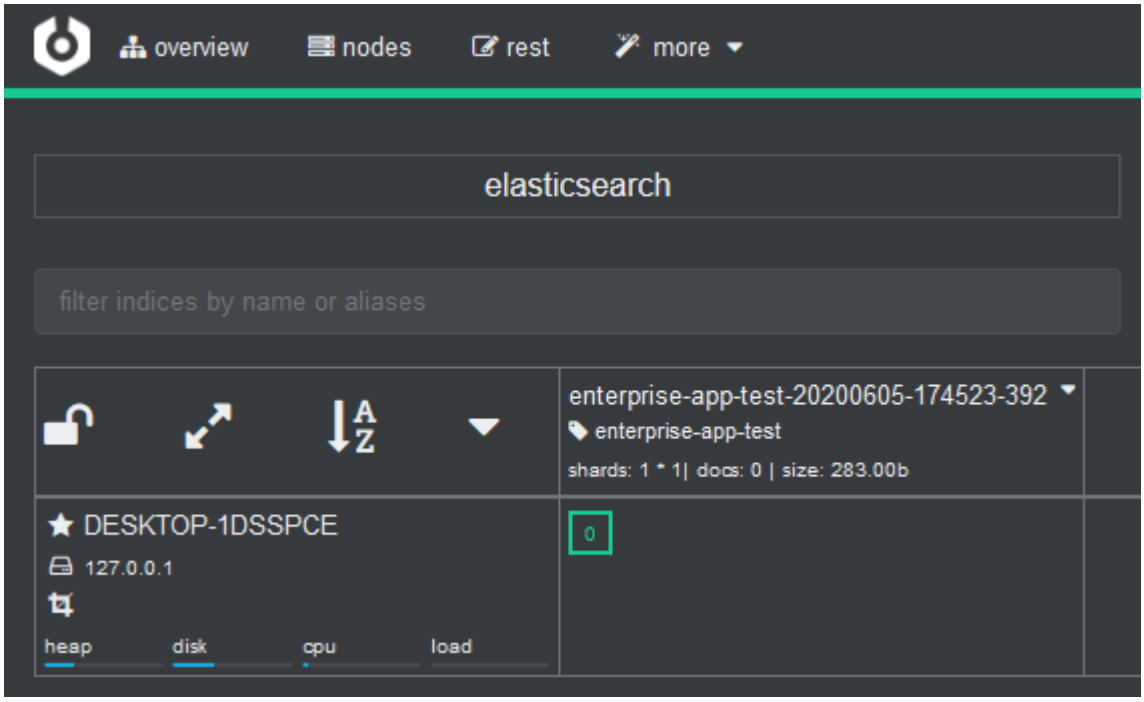
```
POST /api/v1/index/enterprise-app-test/create
```

```
{
    "shards": 1,
    "replicas": 0,
    "hasDefaultFulltext": true,
    "fieldConfigurations": [{
            "name": "title",
            "elasticType": "TEXT",
            "sortable": true,
            "copyTo": ["fulltext"]
        },
        {
            "name": "description",
            "elasticType": "TEXT",
            "copyTo": ["fulltext"]
        },
        {
            "name": "internalRemarks",
            "elasticType": "TEXT"
        },
        {
            "name": "competitor",
            "elasticType": "KEYWORD",
            "aggregatable": true
        }
    ]
}
```

Response:

```
{
    "apiVersion": "{api-version}",
    "processingTimeMillis": 201,
    "indexAlias": "enterprise-app-test",
    "state": "IN_PROGRESS",
    "progress": "CREATE_INDEX",
    "documentsProcessed": null,
    "totalDocuments": null
}
```

With a tool like 'cerebro' you can look at the technical details of the created index:



As you can see, the index was created with the alias name 'enterprise-app-test'.

The index name itself consists of the alias name and a timestamp suffix in the format `-yyyyMMdd-HHmmss-SSS`, for example `enterprise-app-test-20200605-174523-392`.

In API calls, only the index alias is used.

The above request example leads to the following Elasticsearch mapping:

```
{
    "enterprise-app-test-20200605-174523-392": {
        "mappings": {
            "properties": {
                "competitor": {
                    "type": "keyword"
                },
                "description": {
                    "type": "text",
                    "copy_to": [
                        "fulltext"
                    ]
                },
                "fulltext": {
                    "type": "text"
                },
                "internalRemarks": {
                    "type": "text"
                },
                "title": {
                    "type": "text",
                    "fields": {
                        "keyword": {
                            "type": "keyword"
                        }
                    },
                    "copy_to": [
                        "fulltext"
                    ]
                }
            }
        }
    }
}
```


The mapping shows that the three fields `description`, `title` and `internalRemarks` (specified in the create index request) were created as `text` fields.

Due to the request parameter `"hasDefaultFulltext": true` the `text` field `fulltext` was created additionally.

As the field `title` was specified as sortable (`"sortable": true`), an additional `keyword` field was created for it in the mapping.

In addition, the request parameters `"copyTo": ["fulltext"]` provide a corresponding representation in the Elasticsearch mapping, so that the fields `title` and `description` are copied into the field `fulltext`.

The `keyword` field `competitors` was also created, which is later used for aggregations.

The following fields can now be used for adding documents and for searching:

Field name	Type	Fulltext searchable	Sortable	Aggregatable
title	text	yes	yes	no
description	text	yes	no	no
internalRemarks	text	no	no	no
competitors	keyword	no	no	yes

2.4. Add documents

Documents can be added to the Elasticsearch index via the index resource.

The following example adds four documents to the index created above:

Request:

```
PUT /api/v1/index/enterprise-app-test/docs
```

```
[{
        "title": "Chuck Norris Fact #1",
        "description": "Chuck Norris rewrote the Google search engine from scratch.",
        "internalRemarks": "Harder than all the others!",
        "competitor": "Arnold Schwarzenegger"
    },
    {
        "title": "Chuck Norris Fact #2",
        "description": "Chuck Norris doesn't bug hunt, as that signifies a probability
of failure. He goes bug killing.",
        "internalRemarks": "Harder than all the others!",
        "competitor": "Steven Seagal"
    },
    {
        "title": "Chuck Norris Fact #3",
        "description": "He is immutable. If something's going to change, it's going to
have to be the rest of the universe.",
        "internalRemarks": "Harder than all the others!",
        "competitor": "Arnold Schwarzenegger"
    },
    {
        "title": "Steven Seagal Fact #1",
        "description": "He broke Sean Connery's wrist while filming 'Never Say Never
Again'.",
        "internalRemarks": "Tough guy!",
        "competitor": "Arnold Schwarzenegger"
    }
]
```

Response:

```
{
    "apiVersion": "{api-version}",
    "processingTimeMillis": 23,
    "indexAlias": "enterprise-app-test",
    "documentsProcessed": 4
}
```

2.5. Perform search

After you have created the index and added some data, you can perform a search via the search resource.

We start with a fulltext search and then perform a simple field search. Afterwards we combine both

searches to a facet search.

2.5.1. Fulltext search

The following request performs a fulltext search with the query term `Chuck Norris` and sorts the result by `title` in descending order.

Request:

```
POST /api/v1/search/
```

```
{
    "context": {
        "searchType": "INDEX",
        "indexAlias": "enterprise-app-test"
    },
    "query": {
        "queryType": "FULLTEXT",
        "value": "Chuck Norris"
    },
    "maxResults": 10000,
    "pageIndex": 1,
    "pageSize": 10,
    "sortOptions": [{
        "attribute": "title",
        "direction": "DESC"
    }],
    "aggregations": [{
        "aggregationType": "DEFAULT",
        "field": "competitor"
    }]
}
```

Response:

```
{
    "apiVersion": "{api-version}",
    "processingTimeMillis": 9,
    "resultCount": 3,
    "totalHitCount": 3,
    "exactHitCount": true,
    "items": [{
            "id": "8jXflHIB16gFnHxSvRJw",
            "attributes": [{
                    "name": "competitor",
                    "type": "keyword",
                    "value": "Arnold Schwarzenegger"
                },
                {
                    "name": "description",
                    "type": "text",
                    "value": "He is immutable. If something's going to change, it's
going to have to be the rest of the universe."
                },
                {
```

```
                    "name": "title",
                    "type": "text",
                    "value": "Chuck Norris Fact #3"
                },
                {
                    "name": "internalRemarks",
                    "type": "text",
                    "value": "Harder than all the others!"
                }
            ]
        },
        {
            "id": "8TXflHIB16gFnHxSvRJw",
            "attributes": [{
                    "name": "competitor",
                    "type": "keyword",
                    "value": "Steven Seagal"
                },
                {
                    "name": "description",
                    "type": "text",
                    "value": "Chuck Norris doesn't bug hunt, as that signifies a
probability of failure. He goes bug killing."
                },
                {
                    "name": "title",
                    "type": "text",
                    "value": "Chuck Norris Fact #2"
                },
                {
                    "name": "internalRemarks",
                    "type": "text",
                    "value": "Harder than all the others!"
                }
            ]
        },
        {
            "id": "8DXflHIB16gFnHxSvRJw",
            "attributes": [{
                    "name": "competitor",
                    "type": "keyword",
                    "value": "Arnold Schwarzenegger"
                },
                {
                    "name": "description",
                    "type": "text",
                    "value": "Chuck Norris rewrote the Google search engine from
scratch."
                },
                {
```

```
                    "name": "title",
                    "type": "text",
                    "value": "Chuck Norris Fact #1"
                },
                {
                    "name": "internalRemarks",
                    "type": "text",
                    "value": "Harder than all the others!"
                }
            ]
        }
    ],
    "facets": [{
        "name": "competitor",
        "facets": [{
                "facetType": "VALUE",
                "attribute": "competitor",
                "label": "Arnold Schwarzenegger",
                "type": "keyword",
                "value": "Arnold Schwarzenegger",
                "count": 2,
                "from": null,
                "to": null,
                "children": null
            },
            {
                "facetType": "VALUE",
                "attribute": "competitor",
                "label": "Steven Seagal",
                "type": "keyword",
                "value": "Steven Seagal",
                "count": 1,
                "from": null,
                "to": null,
                "children": null
            }
        ]
    }]
}
```


As expected, three hits are returned because three of the four previously added documents contain the query term `Chuck Norris` in the field `title` or `description` (which were copied into the field `fulltext`).

On the other hand, a search with the query term `Harder than all the others` would not return a hit because the field `internalRemarks` was not copied to the field `fulltext`.

Since the `keyword` field `competitor` was marked as `aggregatable` and an aggregation was passed as request parameter, corresponding `facets` were returned in the search result: Two hits contain the value `Arnold Schwarzenegger` in the field `competitor`, one hit contains the value

`Steven Seagal`.

The values of the facets can now be used for a new search.

There are now two possibilities: You want to find all documents where a certain 'competitor' is set (field search), or you want to find all documents of a certain 'competitor' **within** the current search result (facet search).

2.5.2. Field search

The following example shows a simple field search that finds all documents that have the value `Arnold Schwarzenegger` in the field `competitor`:

Since we are only interested in certain fields in the search result for this search, we use the request parameter `resultAttributes` to define the fields to be returned:

Request:

```
POST /api/v1/search/
```

```
{
    "context": {
        "searchType": "INDEX",
        "indexAlias": "enterprise-app-test"
    },
    "query": {
        "queryType": "FIELD",
        "name": "competitor",
        "comparator": "EQ",
        "value": "Arnold Schwarzenegger"
    },
    "maxResults": 10000,
    "pageIndex": 1,
    "pageSize": 10,
    "resultAttributes": [
        "title",
        "competitor"
    ]
}
```

Response:

```
{
    "apiVersion": "{api-version}",
    "processingTimeMillis": 6,
    "resultCount": 3,
    "totalHitCount": 3,
    "exactHitCount": true,
    "items": [{
            "id": "8DXflHIB16gFnHxSvRJw",
            "attributes": [{
                    "name": "competitor",
                    "type": "keyword",
                    "value": "Arnold Schwarzenegger"
                },
                {
                    "name": "title",
                    "type": "text",
                    "value": "Chuck Norris Fact #1"
                }
            ]
        },
        {
            "id": "8jXflHIB16gFnHxSvRJw",
```

```
            "attributes": [{
                    "name": "competitor",
                    "type": "keyword",
                    "value": "Arnold Schwarzenegger"
                },
                {
                    "name": "title",
                    "type": "text",
                    "value": "Chuck Norris Fact #3"
                }
            ]
        },
        {
            "id": "8zXflHIB16gFnHxSvRJw",
            "attributes": [{
                    "name": "competitor",
                    "type": "keyword",
                    "value": "Arnold Schwarzenegger"
                },
                {
                    "name": "title",
                    "type": "text",
                    "value": "Steven Seagal Fact #1"
                }
            ]
        }
    ],
    "facets": []
}
```


2.5.3. Combined search (facet search)

If you want to search for the 'competitor' within the existing search result of the previously performed fulltext search, the field search for the field `competitor` and the previous fulltext search must be combined:

Request:

```
POST /api/v1/search/
```

```
{
    "context": {
        "searchType": "INDEX",
        "indexAlias": "enterprise-app-test"
    },
    "query": {
        "queryType": "COMBINED",
        "operator": "AND",
        "queries": [{
                "queryType": "FULLTEXT",
                "value": "Chuck Norris"
            },
            {
                "queryType": "FIELD",
                "name": "competitor",
                "comparator": "EQ",
                "value": "Arnold Schwarzenegger"
            }
        ]
    },
    "maxResults": 10000,
    "pageIndex": 1,
    "pageSize": 10,
    "resultAttributes": [
        "title",
        "competitor"
    ]
}
```

Response:


```
{
    "apiVersion": "{api-version}",
    "processingTimeMillis": 8,
    "resultCount": 2,
    "totalHitCount": 2,
    "exactHitCount": true,
    "items": [{
            "id": "8DXflHIB16gFnHxSvRJw",
            "attributes": [{
                    "name": "competitor",
                    "type": "keyword",
                    "value": "Arnold Schwarzenegger"
                },
                {
                    "name": "title",
                    "type": "text",
                    "value": "Chuck Norris Fact #1"
                }
            ]
        },
        {
            "id": "8jXflHIB16gFnHxSvRJw",
            "attributes": [{
                    "name": "competitor",
                    "type": "keyword",
                    "value": "Arnold Schwarzenegger"
                },
                {
                    "name": "title",
                    "type": "text",
                    "value": "Chuck Norris Fact #3"
                }
            ]
        }
    ],
    "facets": []
}
```


As you can see, only two hits are returned now, because the search now also contains the fulltext query.

2.6. Delete index

The following request deletes the index and removes the alias:

Request

```
DELETE /api/v1/index/enterprise-app-test
```

Response:


```
{
    "apiVersion": "{api-version}",
    "processingTimeMillis": 440,
    "indexAlias": "enterprise-app-test",
    "state": "DELETED",
    "progress": null,
    "documentsProcessed": null,
    "totalDocuments": null
}
```