Search node

Jesper Ksitensen

Abstract

Search node is an NodeJs frontend to elastic-search that added API-keys and an administration UI. It uses web-sockets to communication with an frontend written in AngularJS.

This document explains how to install search node and extend the AngularJS frontend. This document also includes information about installtion and configuring the Druapl modules.

Contents

1 Introduction

?: Write this section.

1.1 Requirements

The guide assumes that you have an installed linux based server with the following packages installed and at least the versions given.

- nginx 1.4.x
- nodeJS 4.0.x
- supervisor 3.x
- git
- Valid SSL certificates for you domain.

1.2 Conventions

This document uses [] square brackets around the different variables in the configuration templates that needs to be exchanged with the actual values from other parts of the configuration (e.g. API keys).

Her is an explanation of the different key configuration variables.

- [server name]
- [client name]
- [SSL CERT]
- [SSL KEY]
- [CHANGE ME]
- [PASSWORD]
- [SEARCH API KEY]
- [SEARCH INDEX KEY]
- **?** Document placeholders [...]

1.3 Notes

To install the newest version (development version that's not aways stable), you should checkout the development branches in the all the cloned repositories instead of the latest version tag.

The document assumes that you are logged in as the user *deploy*, if this is not the case, you need to change things (e.g. the supervisor run script etc.).

It also assumes that you are installing in */home/www*, which is not the default location on most systems.

2 Search node

This part of the guide explains how to get the search node nodeJS application installed and started. The search node application is a general purpose application to provide a fast search engine through web-socket connections (using elasticsearch). It was developed during the project Aroskanalen but with reusability in mind for other projects.

2.1 Installation

The installation process requires you clone the application, download node modules and define JSON configuration files.

2.1.1 Elasticsearch

Search node used elasticsearch 1.7.x as its search engine and it needs to be installed as well. *Note* that currently the nodeJS module only supports this as the newest version.

First install Java that is used to run elasticsearch.

```
sudo apt-get install openjdk-7-jre -y > /dev/null 2>&1
```

Download and install the search engine.

```
wget
    https://download.elastic.co/elasticsearch/elasticsearch/elasticsearch-1.7.1.deb
sudo dpkg -i elasticsearch-1.7.1.deb
sudo update-rc.d elasticsearch defaults 95 10
```

To enable ICU support (unicode) please install this plugin, which is required for search node to work properly.

```
sudo /usr/share/elasticsearch/bin/plugin -install
    elasticsearch/elasticsearch-analysis-icu/2.5.0
```

For debugging elasticsearch this small administration interface can come handy, but its **optional**.

```
sudo /usr/share/elasticsearch/bin/plugin -install
    mobz/elasticsearch-head
```

2.1.2 Clone

Next clone the git repository for search node and checkout the latest release tag (which can be found using *git tag* command inside the repository after the clone).

```
cd /home/www
git clone git@github.com:search-node/search_node.git
cd search_node
git checkout [v1.x.x]
```

2.1.3 Node packages

Search node uses a plugin architecture that requires installation of libraries from the node package manager (npm). The application comes with an installation script to handle this, simply go to the root of the application and execute the script.

```
cd /home/www/search_node/
./install.sh
```

Note: There are also *upgrade* and *update* scripts, where *update* is used when updating from on version of search node to another to ensure that the right node module versions are used.

2.1.4 Configuration

We need to configure nginx to sit in front of the search node so it is accessible through normal web-ports and also proxy web-socket connections from the frontend client (AngularJS). So we add the upstream connection configuration in a nodejs configuration file for nginx.

```
sudo nano -w /etc/nginx/sites-available/nodejs
```

Append this upstream connection definition to the file.

```
upstream nodejs_search {
  server 127.0.0.1:3010;
}
```

To access the search node UI and allow communication with the search node a virtual host configuration is needed. You need to change the *[server name]* with the actual name of the server.

```
sudo nano -w /etc/nginx/sites-available/search_[server name]
```

```
server {
  listen 80;

  server_name search-[server name];
  rewrite ^ https://$server_name$request_uri? permanent;

  access_log /var/log/nginx/search_access.log;
  error_log /var/log/nginx/search_error.log;
}

# HTTPS server
#
server {
  listen 443;

  server_name search-[server name];

  access_log /var/log/nginx/search_access.log;
  error_log /var/log/nginx/search_error.log;

  location / {
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Host $http_host;

    proxy_buffering off;

    proxy_pass http://nodejs_search/;
    proxy_redirect off;
  }

  location /socket.io/ {
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";

    proxy_pass http://nodejs_search;
  }

  ssl on;
  ssl_certificate /etc/nginx/ssl/[SSL CERT].crt;
  ssl_certificate_key /etc/nginx/ssl/[SSL KEY].key;

  ssl_session_timeout 5m;
  ssl_session_cache shared:SSL:10m;
```

```
  # https://hynek.me/articles/hardening-your-web-servers-ssl-ciphers/
  ssl_prefer_server_ciphers On;
  ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
  ssl_ciphers
      ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128:DH+AES:ECDH+3DES:DH+3DES:RSA+AE
}
```

Enable the configuration by adding a symbolic links for the search node. The nodejs configuration file should be linked during configuration of the middleware. Restart nginx to enable the configuration.

```
cd /etc/nginx/sites-enabled/
sudo ln -s /etc/nginx/sites-available/nodejs
    /etc/nginx/sites-enabled/nodejs
sudo ln -s /etc/nginx/sites-available/search_[server name]
    /etc/nginx/sites-enabled/search_[server name]
sudo service nginx restart
```

Next the application needs to be configured by adding the following content to config.json. Remember to update the administration password and secret.

```
sudo nano -w /home/www/search_node/config.json
```

```
{
  "port": 3010,
  "secret": "[CHANGE ME]",
  "admin": {
    "username": "admin",
    "password": "[PASSWORD]"
  },
  "log": {
    "file": "messages.log",
    "debug": false
  },
  "search": {
    "hosts": [ "localhost:9200" ],
    "mappings": "mappings.json"
  },
  "apikeys": "apikeys.json"
}
```

Before the application can be started the *apikeys.json* and *mappings.json* needs to exist and at least contain an empty JSON object (*{}*).

```
echo '{}' > /home/www/search_node/apikeys.json
echo '{}' > /home/www/search_node/mappings.json
```

The search node needs to be started at boot time which requires a Supervisor run script. Supervisor will also ensure that the node application is restarted, if an error happens and it stops unexpectedly.

```
sudo nano -w /etc/supervisor/conf.d/search_node.conf
```

Supervisor run script for the search node.

```
[program:search-node]
command=node /home/www/search_node/app.js
autostart=true
autorestart=true
environment=NODE_ENV=production
stderr_logfile=/var/log/search-node.err.log
stdout_logfile=/var/log/search-node.out.log
user=deploy
```

```
sudo service supervisor restart
```

As mentioned the search node is not specially created for aroskanalen, so the mappings (configuration for elasticsearch) can be somewhat complex to setup in the UI. To get you started the mapping below can be used as a template for the configuration.

As we need the UI to complete the setup correctly the node application needs to have write access to the files.

```
cd /home/www/search_node/
chmod +w apikeys.json mappings.json
```

Now use the UI (https://search-[server name].aroskanalen.dk)

When you have update the mappings file go back into the UI and select the indexes that you need by edit the API key and select it/them in the edit window. Before a given index can be used you need to activate it in the *indexes* tab. So do that now.

2.2 UI

?: How to use the UI to add more configuration.

and create a new index using the mappings tabs in the UI and add the mappings that you require for a given index. We will go into details about how to configure the mappings correctly in the UI section and in the Drupal section we will show an example for an simple Drupal entity. After the a index have been created (and activated)

After having create an

```
cat /home/www/search_node/mappings.json
```

```
{
  "795359dd2c81fa41af67faa2f9adbd32": {
    "name": "demo",
    "tag": "private",
    "fields": [
      {
        "field": "title",
        "type": "string",
        "language": "da",
        "country": "DK",
        "default_analyzer": "string_index",
        "sort": true,
        "indexable": true
      },
      {
        "field": "name",
        "type": "string",
        "language": "da",
        "country": "DK",
        "default_analyzer": "string_index",
        "sort": true,
        "indexable": true
      }
    ],
    "dates": [
      "created_at",
      "updated_at"
    ]
  }
}
```

Note api-keys “R” vs. “RW” to protect search index.

```
cat apikeys.json
```

```
{
  "795359dd2c81fa41af67faa2f9adbd32": {
    "name": "Test",
    "expire": 300,
    "indexes": [
      "e7df7cd2ca07f4f1ab415d457a6e1c13"
    ],
    "access": "rw"
  }
}
```

3 AngularJS (front end)

The search front-end that users interact with is written in AngularJS and is developed in the search prototype git repository. The script files that should be used in production is the minified versions that are build using a gulp task and is located in the build folder.

The Angular front-end is in fact to applications that communicates with each other, namely search box and search results. It's divided into two applications to create a more flexible front end, that can be placed in different regions on a given page. This also makes it easier to integrate the search into existing CMS frameworks.

The Angular application(s) can use different providers, so it's possible to search via search node or in a JSON data file for faster prototype testing (JSON provider correctly don't support all feature sets yet). One could also create a whole new provider to send search request to third party search engines.

It's build using these parts seen from a top-down view.

- Views (templates)
  - Search box (plus filters)
  - Search results (plus pager and searching spinner)
  - Directives
    * KeyCode (capture keyboard events)
    * Pager
- Controllers
  - Search box
  - Results
- Services
  - Search proxy (sends work to configured provider)
  - Communication (handle events between the to applications)
- Providers
  - Search node
  - JSON

The figure below show how these different components interact with each other and how the date flow is between the two Angular applications. The figure also shows that the configuration is load as an Angular module and is accessible through independency injection (*CONFIG*) throughout the whole framework.

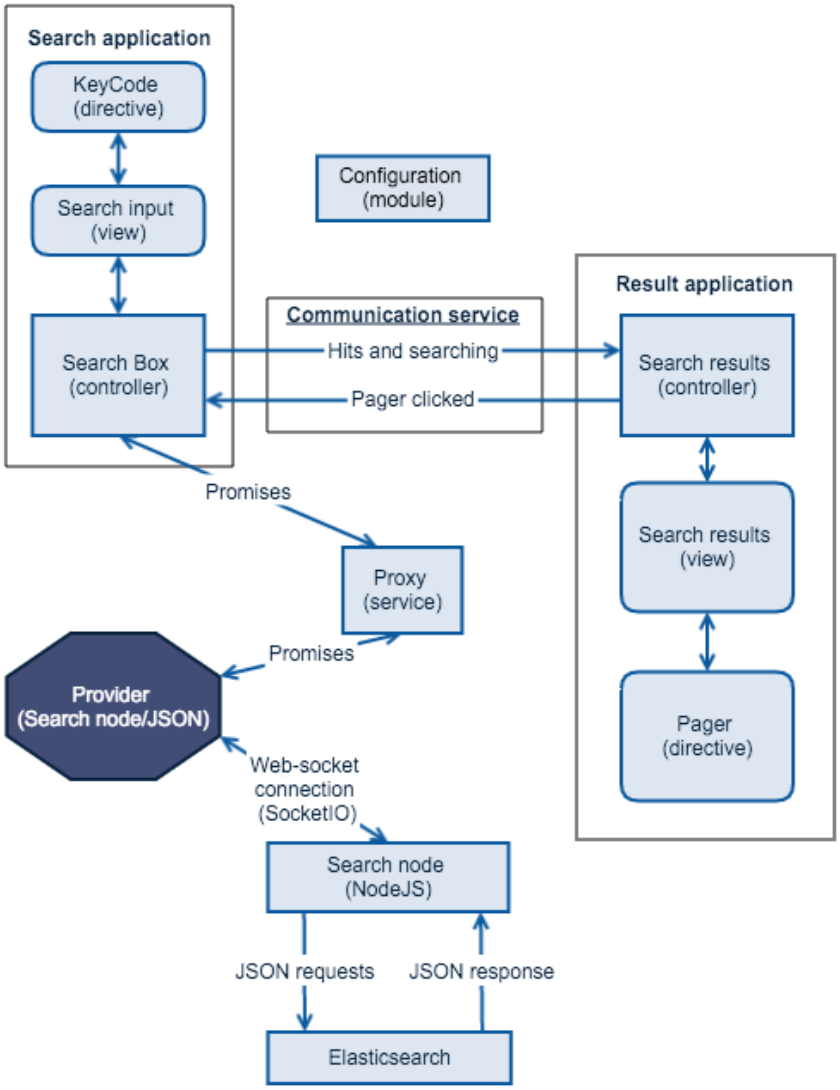

Figure 1: Search node framework

3.1 Configuration

The whole setup is highly configurable and can easily be modified through the configuration (and it's behaviour can be changed by coping, modifying and overriding the contorllers). The configuration is loaded as an Angular module, that can be auto-generated from within a CMS framework or simply typed by hand and added to the page before the application is loaded.

The configuration is defined in JSON and is divided into three parts.

- Basic configuration
  - Id (required) - Unique id for this instanced (used in cache id's for localstorage).
  - initialQueryText (optiona) - Search that will be executed on page load.
- Templates (all required)
  - box - The template used to generate the search box and filters area.
  - result - The results listing template and also the searching spinner.
  - pager - The template used by the pager.
- Provider
  - service (required) - The provider to use.
  - All other are provide configuration options.

The example below is for the search node provider and show the different options available for that provider. The search prototype repository contains example configuration for the different providers.

```
angular.module('searchAppConfig', [])
  .constant('CONFIG', {
    'id' : 'Search prototype',
    'initialQueryText': '',
    'templates': {
      'box': '/js/views/search.html',
      'result': '/js/views/result.html',
      'pager': '/js/directive/pager-directive.html'
    },
    'provider': {
      'service': 'searchNodeProvider',
      'host': 'https://search.node.vm',
      'auth': '/auth.php',
      'index': 'e7df7cd2ca07f4f1ab415d457a6e1c13',
      'fields': ['title', 'body:value'],
      'pager': {
        'size': 8,
```

```
      'page': 0
    },
    'cacheExpire': 30,
    'intervals': ['created'],
    'filters': [
      {
        'field': 'field_level',
        'name': 'Level',
        'type': 'and',
        'terms': {
          "First": {
            'value': 'First'
          },
          "Second": {
            'value': 'Second'
          },
          "Third": {
            'value': 'Third'
          },
          "Fourth": {
            'value': 'Fourth'
          },
        }
      },
      {
        'field': 'field_technology',
        'name': 'Technology',
        'type': 'and',
        'terms': {
          "Angular": {
            'value': 'Angular'
          },
          "Javascript": {
            'value': 'Javascript'
          },
        }
      }
    ]
  }
});
```

?: Above is missing date examples for the search provider.

3.1.1 Loading the applications.

It's important to load the different parts in the right order as shown in the example below.

```
<!-- Load angular  -->
<script src="angular.min.js"></script>

<!-- Load the application configuration -->
<script src="config.js"></script>

<!-- Load the app -->
<script src="assets.min.js"></script>
<script src="search.min.js"></script>
```

3.1.2 Bootstrap process and override

?: Explain how the to applications are bootstrapped and how to override the controllers.

```
<div id="searchBoxApp" data-ng-strict-di
    data-ng-controller="boxController">
  <span data-ng-include="template">
    JavaScript have not been loaded.
  </span>
</div>
```

```
<div id="searchResultApp" data-ng-strict-di
    data-ng-controller="resultController" >
  <span data-ng-include="template">
    JavaScript have not been loaded.
  </span>
</div>
```

3.2 Development (search prototype)

?: Why the searchpt project and how to contribute to the solution.

3.2.1 Cache + md5

?: What is cached and how is the cache controlled.

3.2.2 Build process

?: How to build a new version of the minified files.

```
gulp build --production
```

3.2.3 Vagrant

?: The vagrant used to do the development and what it sets up.

4 Drupal modules

?: Explain the modules and what they require and the idea.

- search_api_search_node
- search_node_page
- search_node_test_settings

4.1 Installation

AngularJS in libraries folder (angular/angular.min.js) -> 1.4.x

?: Who to install the modules.

4.2 Panels vs. blocks

?: How to use the panes or blocks provide by the modules.

4.3 Configuration

?: How to configure search API to use search node. **?**: How to configure the blocks/panes

4.4 Override search controller

?: How to overrride the contoller (in drupal) and link to ulf as an example.

search-node-page-search-box.tpl.php

```
<div id="searchBoxApp" data-ng-strict-di
    data-ng-controller="UlfBoxController">
  <span data-ng-include="template">
    JavaScript have not been loaded.
  </span>
</div>
```

5 Roadmap

- Type-ahead
- Or filters (currently one “and” filters are implemented)
- Sorting
- Filter on non taxonomy fields

6 Examples

This section lists sites that are using search node.

- ULF i Aarhus - http://ulfiaarhus.dk/dagtilbud

fig_caption: yes