HOGESCHOOL VAN AMSTERDAM

Uforia

Front-end install guide

Uforia

30-6-2014

This document explains how to install and configure the front-end of Uforia

Introduction

This document describes how to install and configure the front-end of Uforia. The guide will cover installation for the three major platforms (Linux, Windows and OS X). Specifically the guide will explain how to install and configure Elasticsearch, Node.js and how to use the Uforia-Browser code.

Uforia stands for 'Universal Forensic Indexer and Analyzer' and is an open-source project hosted at github: https://github.com/uforia/Uforia. Uforia is a modular open-source toolkit capable of scaling to the needs of the user. Through configuration files it allows the user to enable/disable modules as seen fit. The information gathered by Uforia is entered into a database of choice (MySQL and SQLite are currently fully supported with PostgreSQL on the way).

The Uforia-Browser repository is used for development of the front-end. This currently uses Elasticsearch in combination with Node.js, both of which will be further explained in this document.

The Uforia-Browser repository is located here: https://github.com/uforia/Uforia-browser

Testing Environment

Uforia is taking a live-development approach. This means that any changes can immediately be witnessed at http://www.uforia.nl/. The server is a 64-bit Ubuntu installation with the most recent updates downloaded at regular intervals. Elasticsearch is version 1.1.0 and has a single node running on the Uforia server and we have not tested nor accounted for multiple ES nodes at the time of writing.

This document is written under the assumption that the reader is working on a similar system. Windows and OSX have *not* been taken into account when writing this document.

Table of contents

Elasticsearch

Elasticsearch is a real-time distributed search and analytics engine. It allows exploration of data at incredible speed and scale. There have been multiple case studies on Elasticsearch that showcase its utility for the parsing of huge amounts of Data.

Elasticsearch's own documentation has some excellent examples of how Elasticsearch is used by (much) bigger instances:

- Wikipedia uses Elasticsearch to provide full text search with highlighted search snippets, and search-as-you-type and did-you-mean suggestions.
- The Guardian uses Elasticsearch to combine visitor logs with social network data to provide real-time feedback to their editors about the public's response to new articles.
- StackOverflow combines full text search with geolocation queries and uses more-like-thisto find related questions and answers.
- GitHub uses Elasticsearch to query 130 billion lines of code.
- Goldman Sachs uses it to index 5TB of log data every day, and a number of investment banks use it purely to analyze movements in the stock market.

Elasticsearch mappings

A mapping defines how information should be mapped to the search engine. This includes which fields are searchable as well as defining what type of fields they are (strings, numbers, objects).

The Uforia frontend places metadata fields in these mappings allowing the document/file/object to be retrieved in its entirety from the SQL database when necessary**.** At the time of writing a feature to allow users to select which of these fields should and should not be mapped for each individual Uforia module is still in development.

Index creation and mapping API

In order to place a mapping into Elasticsearch an index needs to first be present.

Because the testing enviroment uses the index 'uforia' we'll continue using that in the following examples.

To create an index:

```
curl -XPUT 'http://localhost:9200/uforia/'
```

Because we've been using email for our live demo we'll also use that mapping as an example. There are libraries out there written in several languages that allow the creation of mappings via languages such as Python. A script that will automatically create mappings based on the data already in the SQL databases is still in development but it's possible to create a mapping manually. Like so:

```
curl -XPUT ‘http://localhost:9200/uforia/message_rfc822/_mapping’ -d ‘

            "message_rfc822": {

               "properties": {

                   "Bcc": {"type": "string"},

                   "Body": {"type": "string"},

                   "Cc": {"type": "string"},

                   "Content_Type": {"type": "string"},

                   "Date": {"format": "dateOptionalTime",

                            "type": "date"},

                   "From": {"type": "string"},

                   "Organization": {"type": "string"},

                   "Subject": {"type": "string"},

                   "To": {"type": "string"},

                   "XTo": {"type": "string"},

                   "Xbcc": {"type": "string"},

                   "Xcc": {"type": "string"},

                   "hashid": {"type": "string"}

               }

           },’
```

Every field as a type defined. The core types supported are: string, integer/long, float/double, boolean, date and null.

Uforia and Elasticsearch

Uforia uses Elasticsearch to create so called ‘mappings’ of some of the metadata fields. These mappings are then used to store information related to just these specific fields. As soon as the information is entered it can be queried.

The motivation behind this is that searches are no longer run directly against an SQL database, greatly increasing the return of results. If more data is required by the user then the actual database can always be queried directly for the specific row/file.

Installing Elasticsearch

Elasticsearch v1.x and higher requires Java 7 to be installed on the server, the OpenJDK 7 version can be installed via apt-get like so:

```
sudo apt-get install openjdk-7-jre
```

The Elasticsearch download can be grabbed from here: http://www.elasticsearch.org/download/. In this case we have downloaded a *.deb and that can be installed like so:

```
dpkg -i elasticsearch-*.deb
```

Elasticsearch gets installed in /usr/share/elasticsearch but can be located using `whereis elasticsearch`. This command will also show the configuration directory which by default is `/etc/elasticsearch.`

The version running at the time of writing is 1.1.

Configuring Elasticsearch

Elasticsearch has a variabled called 'ES_HOME' and uses this to refer to its home directory. Elasticsearch has two configuration files that are of import: `elasticsearch.yml` and `logging.yml`

The ES documentation recommends leaving the ES_MIN_MEM and ES_MAX_MEM values at their default and instead setting the ES_HEAP_SIZE value. This will cause the same value to be set to both the min and max values.

The following is recommended. If the values are not present in the files mentioned add them at the bottom. Please note that `ES_HEAP_SIZE` is set to a *minimum* value and can be increased depending on the server RAM available.

```
/etc/security/limits.conf:
elasticsearch - nofile 65535
elasticsearch - memlock unlimited
```

```
/etc/default/elasticsearch:
ES_HEAP_SIZE=512m
MAX_OPEN_FILES=65535
MAX_LOCKED_MEMORY=unlimited
```

```
/etc/elasticsearch/elasticsearch.yml:
bootstrap.mlockall: true
```

Security

Something else to take into consideration is the firewall. It's important to take a moment to check if you're able to connect to ES from the outside while it's running. Configuring a firewall falls oustide the scope of this document but an example command to see if ES is accessible on the uforia machine from the outside is:

```
curl -XGET '145.92.7.231:9200'
```

If this doesn't return a reply then all is well as ES should only be accessible through localhost.

Elasticsearch cURL commands

The following examples are useful for retrieving or deleting information from Elasticsearch and assume 'uforia' is the index Elasticsearch uses. The command 'pp' used in the examples is an alias that will 'prettify' the returned JSON.

```
alias pp='python -mjson.tool'
```

Information about the current running Elasticsearch version:

```
curl -XGET 'localhost:9200'
```

Retrieve all mappings currently in Elasticsearch:

```
curl -XGET 'http://localhost:9200/uforia/_mapping' | pp
```

Retrieve information about all nodes:

```
curl -XGET 'http://localhost:9200/_nodes' | pp
```

Search a mapping:

```
curl -XGET 'http://localhost:9200/uforia/message_rfc822/_search' | pp
```

Delete Uforia Index:

```
curl -XDELETE 'http://localhost:9200/uforia/'
```

Clear cache:

```
curl -XPOST 'http://localhost:9200/uforia/_cache/clear'
```

You could also specify a mapping to only clear cache of that mapping.

Delete every Index:

```
curl -XDELETE 'http://localhost:9200/_all/'
```

Node.js

“Node.js is a software platform for scalable server-side and networking applications. Node.js applications are written in JavaScript. All of the popular server operating systems are supported, including Windows and Linux

Node.js applications are designed to maximize throughput and efficiency, using non-blocking I/O and asynchronous events. Node.js applications run single-threaded, although Node.js uses multiple threads for file and network events. Node.js is commonly used for real time applications due to its asynchronous nature, allowing applications to display information faster for users without the need for refreshing.”

-Wikipedia (http://en.wikipedia.org/wiki/Node.js)

Installing Node.js

It is recommended to let Node.js listen to localhost while having it publicly available via a reverse proxy server. How to achieve this is not described in this document.

Linux

The following steps explain how to install Node.js for Linux using the command line:

1. Install NodeJs using the commando ‘*sudo apt-get install nodejs*’. This will install Node.js on your system.
    a. No further configuration of Node.js is required after installing

Uforia-browser

Tools you'll need

To install Uforia-browser you'll need to have the following software installed:

- Git
- Software that is capable of unpacking a .zip file

Installing and configuring Uforia-browser

Linux

The following steps explain how to get the Uforia interface online for Linux

1. Create a directory where the Uforia-browser files will be hosted
2. Clone the repository in the directory with the command '*git clone https://github.com/uforia/Uforia-browser.git*' or download the latest version with the command '*wget* https://github.com/uforia/Uforia-browser/archive/master.zip' and unpack it in the directory.
3. Open the file 'uforia-browser.js' in the root of the directory with a text editor, in the top part of the file you can configure the server to use the right port, database credentials, etc.

Starting the server

Linux

The following steps explain how to start the Node.js server.

1. Go to the root directory of your Uforia-browser installation
2. Execute the command '*nodejs elasticsearch.js*'
    a. If Node.js crashes after sometime it's most likely due to insufficient memory. You can manually set the amount of memory Node.js can use with the following command '*nodejs –max-old-space-size=8192 elasticsearch.js*'. In this case Node.js can use up to 8GB of memory
3. Go to the correct address in the browser and you should see the Uforia-browser interface

Sources

- https://stackoverflow.com/questions/18132719/how-to-change-elasticsearch-max-size
- http://www.elasticsearch.org/guide/en/elasticsearch/reference/current/setup-configuration.html#setup-configuration-memory
- http://www.elasticsearch.org/guide/en/elasticsearch/guide/current/getting-started.html
- https://github.com/grigorescu/Brownian/wiki/ElasticSearch-Configuration
- http://www.elasticsearch.org/guide/en/elasticsearch/reference/current/setup-service.html