Elastica-Docs Documentation

Release

Yash Todi

May 29, 2017

Contents:

CHAPTER 1

Introduction

Elastica was in 2010 one of the first PHP clients for elasticsearch and was initially built for elasticsearch v0.16.0 and before. Since then a lot of things have changed in the elasticsearch API. Filter, facets, aggregations and more were added which also added complexity on the client side.

In 2013 elasticsearch decided to unify all its elasticsearch client to have a common base and started to build elasticsearch-php. Since then, both clients are developing in parallel.

CHAPTER 2

Goals

The main objective behind making the documentation is to have a single space where people can at least initiate using the Elastica library.

Elastica is a wonderful PHP client to interact with the Elasticsearch engine which has been around since a long time and gained popularity in the community.

But, unfortunately the library suffers from lack of documentation and tutorials to get people to use it to its full potential.

Note: This is an unofficial document written purely on the basis of my understanding of Elastica and Elasticsearch itself.

CHAPTER 3

Installation

Note: As per the official information, Elastica 5.1.0 is compatible with all elasticsearch 5.x releases. It was tested with elasticsearch v5.1.2.

Download

You can download this project in either zip or tar formats.

The prefered way is to clone Elastica with Git by running:

```
$ git clone git://github.com/ruflin/Elastica
```

Composer

You can also install Elastica by using composer:

```
{
        "require":{
        "ruflin/elastica":"dev-master"
        }
}
```

Include in your project

If you've used composer to install Elastica it's very easy. Assuming your document root index file is in /htdocs/myproject/web/ and composer installed the vendor file in /htdocs/myproject/vendor your index.php has to look like this:

```php
<?php

require_once '../vendor/autoload.php';
```

If you don't use composer in your project you have to include Elastica. Best way to do this is to use PHP spl_autoload_register. This function is automatically called in case you are trying to use a class/interface which hasn't been defined yet.

So let's assume you installed Elastica to `/var/www/Elastica`. When you make an instance of `\Elastica\Client`, the function will check if there is a file with that class in `/var/www/Elastica/Client` and load it.

```php
<?php

function __autoload_elastica ($class) {
    $path = str_replace('\\', '/', substr($class, 1));

    if (file_exists('/var/www/' . $path . '.php')) {
        require_once('/var/www/' . $path . '.php');
    }
}
spl_autoload_register('__autoload_elastica');

//Or using anonymous function PHP 5.3.0>=
spl_autoload_register(function($class){
    if (file_exists('/var/www/' . $class . '.php')) {
            require_once('/var/www/' . $class . '.php');
        }
});
```

CHAPTER 4

Working with Elastica Client Object

Connecting to Elasticsearch

Note: Assuming you have elasticsearch set up and running.

Running a single node

When your single elasticsearch node is running on `localhost:9200`, which is the default, you can simply connect:

```
<?php

$elasticaClient = new \Elastica\Client();
```

Running a cluster

Elasticsearch was built with the cloud / multiple distributed servers in mind. It is quite easy to start a elasticsearch cluster simply by starting multiple instances of elasticsearch on one server or on multiple servers. To start multiple instances of elasticsearch on your local machine, just run the command to start an instance in the elasticsearch folder twice.

One of the goals of the distributed search index is availability. If one server goes down, search results should still be served. But if the client connects to only the server that just went down, no results are returned anymore. Because of this, `\Elastica\Client` supports multiple servers which are accessed in a round robin algorithm. This is the only and also most basic option at the moment. So if we start a node on port 9200 and port 9201 above, we pass the following arguments to `\Elastica\Client` to access both servers:

```
<?php

$elasticClient = new \Elastica\Client(array(
```

```
    'servers' => array(
        array('host' => 'localhost', 'port' => 9200),
        array('host' => 'localhost', 'port' => 9201)
    )
));
```

Note: A callback function can be passed as a second parameter while creating the client object.

Client Configurations

Generally, an elastica client object consists of the following parameters:

Parameter	Default Value	Additional Info
host	`null`	
port	`null`	
path	`null`	
url	`null`	
proxy	`null`	
transport	`null`	
persistent	`true`	
timeout	`null`	
connections	`[]`	
roundRobin	`false`	
log	`false`	Set to true, to enable logging, set a string to log to a specific file
retryOnConflict	`0`	Used while updating a document.
bigintConversion	`false`	Set to true to enable the JSON bigint to string conversion option (see issue #717)
username	`null`	
password	`null`	

Note: You may find an additional `connectionStrategy` parameter which is set to `RoundRobin` or `Simple` depending on the `roundRobin` flag being `true` or `false` respectively.

Lookup the client configuration

The current configuration of an elastica client can be found using the `getConfig()` and `getConfigValue()` methods.

The `getConfig()` method returns the whole config of the elastica client object as an array. Optionally, a key can be passed to fetch specific information.

```
<?php

$elasticClient->getConfig();       // Will return the whole config

$elasticClient->getConfig('host'); // Will return the host

$elasticClient->getConfig('port'); // Will return the port
```

The `getConfigValue()` method compulsorily requires a key parameter which fetches the required information. Optionally, a second parameter can be passed as a default value. The default value is returned in case the key is not found to be present in the config.

```
<?php

$elasticClient->getConfigValue('host');           // Will return the host

$elasticClient->getConfigValue('host', 'foo');    // Will return the host

$elasticClient->getConfigValue('hosting', 'foo'); // Will return 'foo'
```

Set/Overwrite the client configuration

An array can be passed to `setConfig()` method which will add/update new values to the existing client config while retaining the default values.

```
<?php

$elasticClient->setConfig( array( 'port' => '9201' ) ); // Will set the port to 9201␣
↪and retain the default values for the rest
```

However, to set a specific value, it is better to use the `setConfigValue()` method with the `key` and `value` passed as parameters.

```
<?php

$elasticClient->setConfigValue('port', '9201'); // Will set the port to 9201 and␣
↪retain the default values for the rest
```

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Index