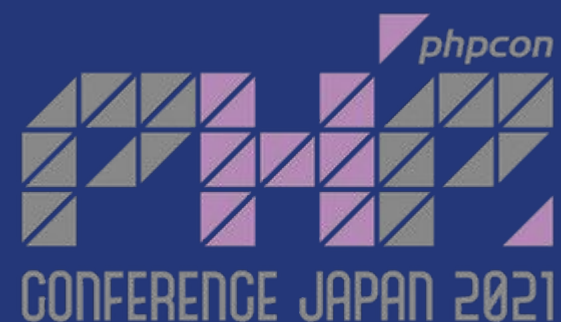




Programming Elasticsearch with PHP

Enrico Zimuel, *Principal Software Engineer*

Oct 3, 2021



Summary

- Introduction to Elasticsearch
- Elasticsearch and PHP
- Connect to Elasticsearch
- Index, Bulk, Search
- Fuzzy search, Aggregation
- Schema on read (from 7.12)
- Async communication
- Future work



php

Elasticsearch

- Elasticsearch is a **distributed, free and open search and analytics engine** for all types of data
- Elasticsearch **scale by design** and manage any size of data
- **Very fast:** near real-time search
- **Wide range of search features:** filter, aggregate, analyze, order any type of information
- Elasticsearch is **document oriented** (JSON), it stores entire objects or documents
- A collection of documents is called an **index**, a table in SQL

REST API

- You can interact with Elasticsearch using **REST APIs**, no client or shell tool

```
$ curl -X GET http://localhost:9200
```

```
{
  "name" : "12b27ad95a8b",
  "cluster_name" : "docker-cluster",
  "cluster_uuid" : "yz2VKxzORYCQUjXz0MerxQ",
  "version" : {
    "number" : "7.12.1",
    "build_flavor" : "default",
    "build_type" : "docker",
    "build_hash" : "3186837...b7",
    "build_date" : "2021-04-20T20:56:39.040728659Z",
    "build_snapshot" : false,
    "lucene_version" : "8.8.0",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

Install and run Elasticsearch

- The easiest way to install Elasticsearch is to use a Docker image. A list of all published Docker images and tags is available at www.docker.elastic.co

```
$ docker pull docker.elastic.co/elasticsearch/elasticsearch:7.14.1
```

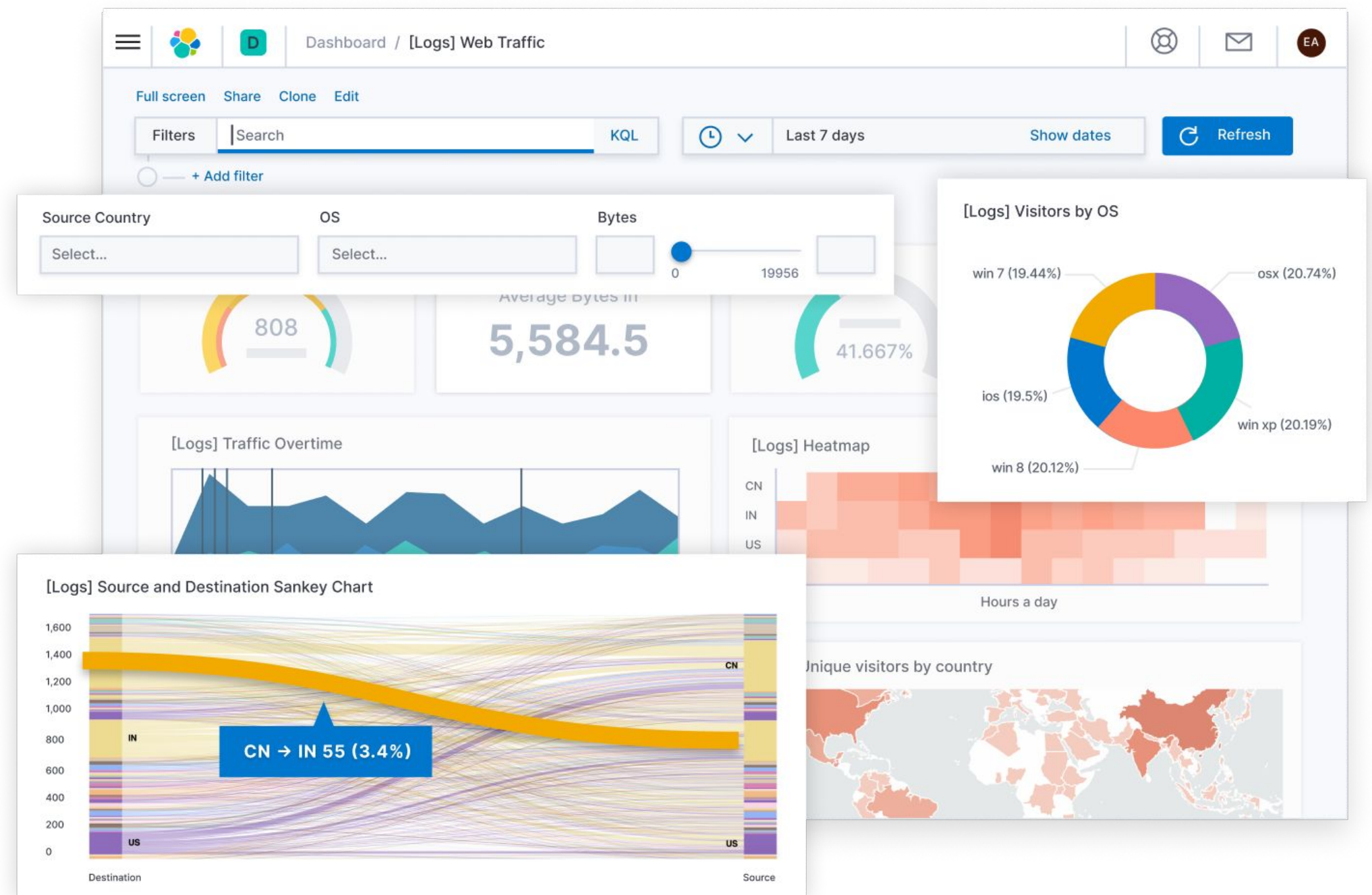
- Start a single-node cluster (**localhost:9200**)

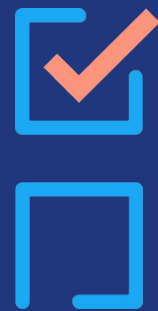
```
$ docker run -p 9200:9200 -p 9300:9300 -e "discovery.type=single-node"  
docker.elastic.co/elasticsearch/elasticsearch:7.14.1
```



Elastic Cloud

- You can run Elasticsearch on [Elastic Cloud](#), the SaaS solution offered by Elastic
- Start a **free 14-day trial**, no credit card required





Elasticsearch & PHP



Elasticsearch-php

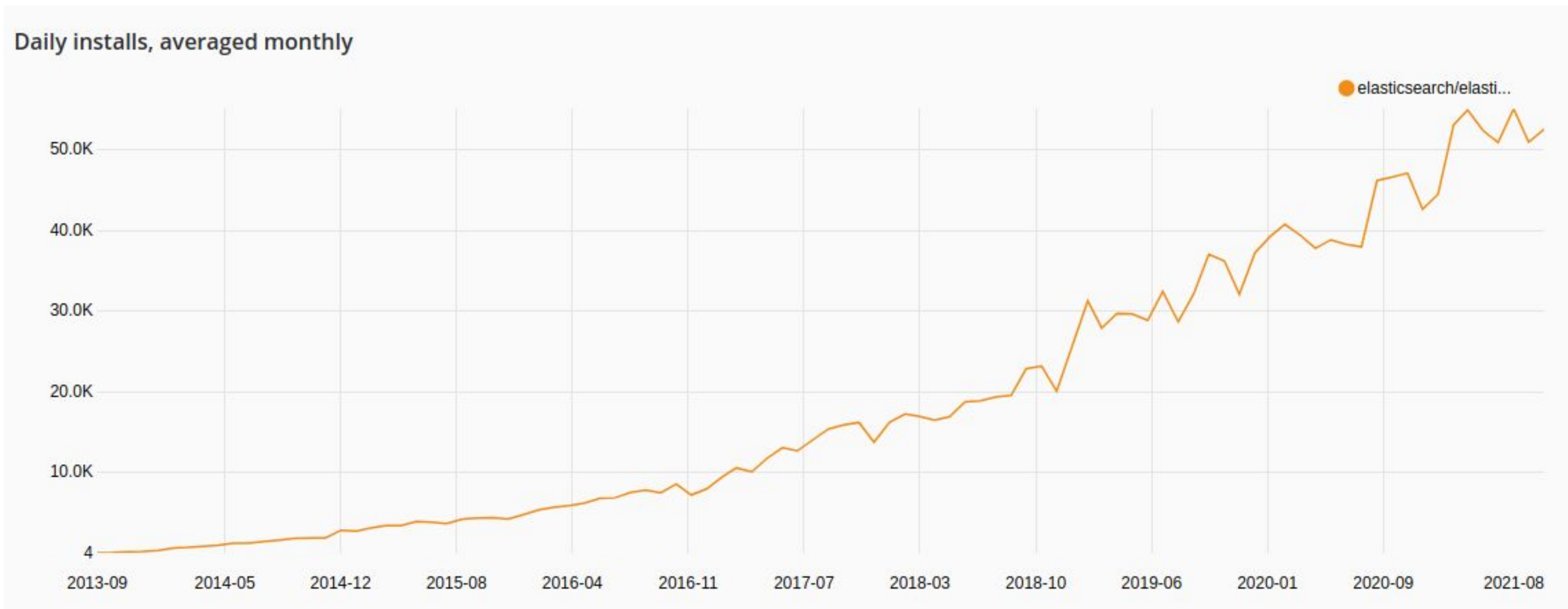
- Official PHP client for Elasticsearch: [elastic/elasticsearch-php](https://github.com/elastic/elasticsearch-php)
- Updated and released with the Elastic stack version
- Use **connection pool** for cluster configuration
- Exposes the **Elasticsearch endpoints** (≈ 400 APIs) using functions of a Client class
- Each function returns the **body of HTTP response** from Elasticsearch or a **boolean value** (for HEAD operations, eg. [Index exists](#))

Elasticsearch-php (2)

- The body is deserialized from JSON using a **Serializer interface** (associative array as default)
- In case of HTTP errors (4xx, 5xx) the PHP client throws an **ElasticsearchException** (eg. Missing404Exception)
- All the endpoints are generated using the [REST API specification](#) of Elasticsearch
- The PHP client for elasticsearch is tested using:
 - Unit tests
 - Integration tests ($\approx 2,400$ tests)

Install statistics

- Total install using composer (packagist.org): 54M+



Endpoints

- All the Elasticsearch API are exposed via functions

Elasticsearch API	PHP function
Index: PUT /<target>/_doc/<_id>	<code>\$client->index(\$params)</code>
Bulk: POST /_bulk	<code>\$client->bulk(\$params)</code>
Update: POST /<index>/_update/<_id>	<code>\$client->update(\$params)</code>
Delete: DELETE /<index>/_doc/<_id>	<code>\$client->delete(\$params)</code>
Search: POST /<target>/_search	<code>\$client->search(\$params)</code>
Cluster Stats: GET /_cluster/stats	<code>\$client->cluster()->stats()</code>

Parameters

- The API parameters are specified using the **\$params** array

```
/**
 * $params['id']           = (string) Document ID
 * $params['index']       = (string) The name of the index (Required)
 * $params['wait_for_active_shards'] = (string) Sets the number of shard copies...
 * $params['op_type']     = (enum) Explicit operation type. Defaults to...
 * $params['refresh']     = (enum) If `true` then refresh the affected shards...
 * $params['routing']     = (string) Specific routing value
 * $params['timeout']     = (time) Explicit operation timeout
 * $params['version']     = (number) Explicit version number for concurrency...
 * $params['version_type'] = (enum) Specific version type (Options = internal...
 * $params['if_seq_no']   = (number) perform the index operation if the last...
 * $params['if_primary_term'] = (number) only perform the index operation if...
 * $params['pipeline']    = (string) The pipeline id to preprocess incoming...
 * $params['require_alias'] = (boolean) When true, requires destination to be an...
 * $params['body']        = (array) The document (Required)
 *
 * @param array $params Associative array of parameters
 * @return array
 * @see https://www.elastic.co/guide/en/elasticsearch/reference/master/docs-index\_.html
 */
public function index(array $params = []) { /* ... */ }
```

Install elasticsearch-php

- Install using [composer](#) (latest stable version):

```
$ composer require elasticsearch/elasticsearch
```

- Or add the following require in **composer.json**:

```
{  
  "require": {  
    "elasticsearch/elasticsearch": "^7.14"  
  }  
}
```

Connect to Elasticsearch

- Connect to **localhost:9200** and call the [Info API](#)

```
use Elasticsearch\ClientBuilder;

$client = ClientBuilder::create()
    ->setHosts(['localhost:9200'])
    ->build();

$result = $client->info();
var_dump($result);
```

```
array(5) {
  'name' => string(12) "cea89f5abf6e"
  'cluster_name' => string(14) "docker-cluster"
  'cluster_uuid' => string(22) "Np1b...qbVi5kQ"
  'version' =>
  array(9) {
    'number' => string(6) "7.10.0"
    'build_flavor' => string(3) "oss"
    'build_type' => string(6) "docker"
    'build_hash' => string(40) "51e9d..96"
    'build_date' => string(27) "2020-11-09T21:30:33.964949Z"
    'build_snapshot' => bool(false)
    'lucene_version' => string(5) "8.7.0"
    'minimum_wire_compatibility_version' => string(5) "6.8.0"
    'minimum_index_compatibility_version' => string(11) "6.0.0"
  }
  'tagline' => string(20) "You Know, for Search"
}
```

Connect to a cluster of nodes

- Connect to a cluster and call the [Cluster health API](#):

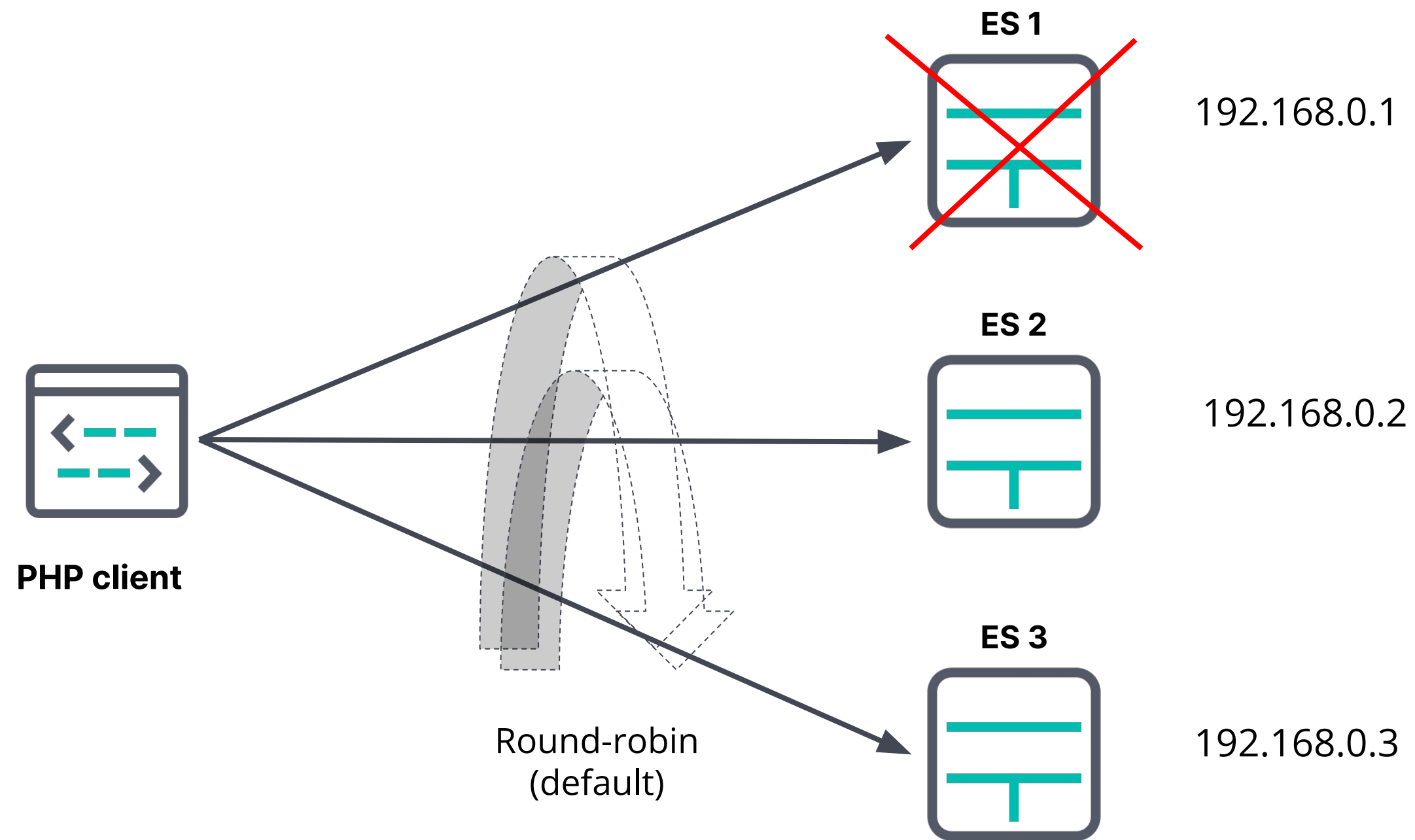
```
use Elasticsearch\ClientBuilder;

$client = ClientBuilder::create()
    ->setHosts([
        '192.168.0.1:9200',
        '192.168.0.2:9200',
        '192.168.0.3:9200'
    ])
    ->build();

$result = $client->cluster()->health();
var_dump($result);
```

```
array(15) {
  'cluster_name' => string(34) "elasticsearch-oss-7-10"
  'status' => string(5) "green"
  'timed_out' => bool(false)
  'number_of_nodes' => int(3)
  'number_of_data_nodes' => int(3)
  'active_primary_shards' => int(0)
  'active_shards' => int(0)
  'relocating_shards' => int(0)
  'initializing_shards' => int(0)
  'unassigned_shards' => int(0)
  'delayed_unassigned_shards' => int(0)
  'number_of_pending_tasks' => int(0)
  'number_of_in_flight_fetch' => int(0)
  'task_max_waiting_in_queue_millis' => int(0)
  'active_shards_percent_as_number' => double(100)
}
```

Connection pool



Selector

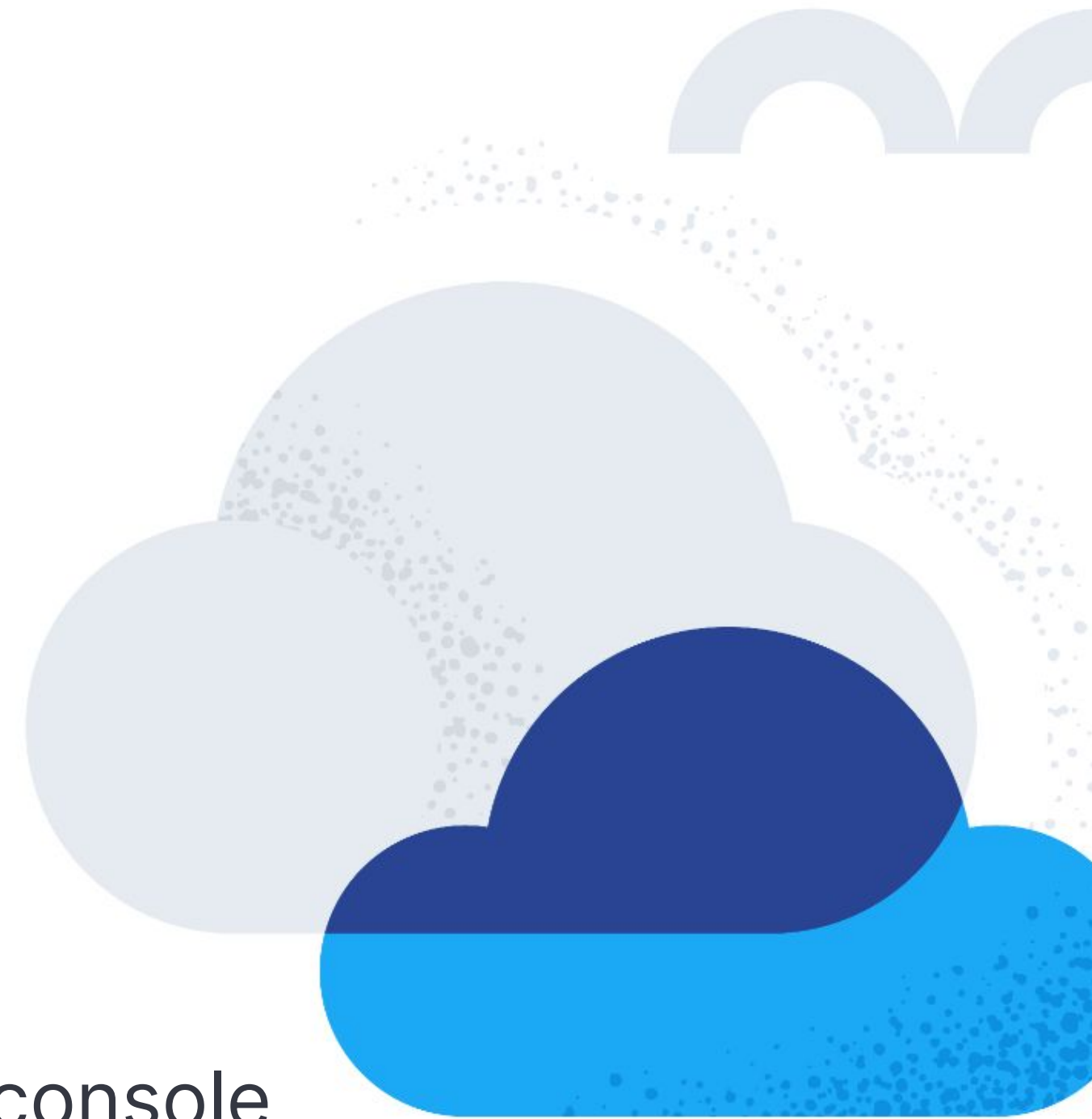
- We provide a [SelectorInterface](#) to implement a custom algorithm for selecting the next node
- We offer the following Selector implementations:
 - **Round-robin** (default): iterate over a set of nodes in circular order;
 - **Sticky Round-robin**: use current connection unless it is dead, otherwise round-robin
 - **Random**: select a random node from the set
- You can set a custom selector implementation using the **ClientBuilder::setSelector()** function

Connect to Elastic Cloud

- You can connect to [Elastic Cloud](#) using **Cloud-id** and **API key**

```
$client = ClientBuilder::create()  
->setElasticCloudId('Cloud-id')  
->setApiKey('<id>', '<key>')  
->build();
```

- **Cloud-id** and **API key** are available in Cloud console
 - **Cloud-id** is an encoded URL of the host
 - **API key** is Base64(<id>:<key>)



JSON vs PHP

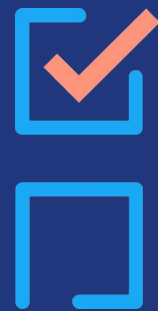
- **JSON** is supported in PHP using the following functions:
 - [json_encode](#) (`$value [, int $flags = 0 [, int $depth = 512]]`) : `string|false`
 - [json_decode](#) (`string $json [, bool|null $associative = NULL [, int $depth = 512 [, int $flags = 0]]]`) : `mixed`
- From PHP 7.3 we can use **JSON_THROW_ON_ERROR** as `$flags` to throw a **JsonException** in case of errors

JSON empty object

- Elasticsearch API uses **empty JSON objects**, for some API
- An empty JSON object `{}` can be expressed in PHP using an empty object, eg. **`new stdClass()`**

```
{
  "query" : {
    "match" : {
      "content" : "foo"
    }
  },
  "highlight" : {
    "fields" : {
      "content" : {}
    }
  }
}
```

```
$params['body'] = [
  'query' => [
    'match' => [
      'content' => 'foo'
    ]
  ],
  'highlight' => [
    'fields' => [
      'content' => new stdClass()
    ]
  ]
];
```



Data management API



Index a document

- When you add documents to Elasticsearch, you **index** JSON

```
use Elasticsearch\ClientBuilder;

$client = ClientBuilder::create()
    ->setHosts(['localhost:9200'])
    ->build();

$params = [
    'index' => 'my_index',
    'id'     => 'my_id',
    'body'   => [ 'testField' => 'abc' ]
];

$result = $client->index($params);
var_dump($result);
```

```
array(8) {
  '_index' => string(8) "my_index"
  '_type' => string(4) "_doc"
  '_id' => string(5) "my_id"
  '_version' => int(1)
  'result' => string(7) "created"
  '_shards' => array(3) {
    'total' => int(2)
    'successful' => int(1)
    'failed' => int(0)
  }
  '_seq_no' => int(0)
  '_primary_term' => int(1)
}
```

Bulk indexing

- You can manage **multiple documents** using the [Bulk API](#): index, create, delete, and update actions in a **single request**
- The actions are specified in the request body using [NDJSON](#)

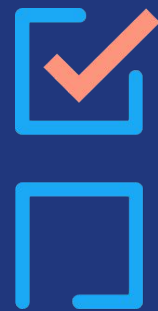
```
for($i=0; $i < 100; $i++) {  
    $params['body'][] = [  
        'index' => [  
            '_index' => 'my_index',  
        ]  
    ];  
    $params['body'][] = [  
        'my_field' => 'my_value',  
        'second_field' => 'some more values'  
    ];  
}  
$result = $client->bulk($params);
```

Missing document

- If the document does not exist returns a **Missing404Exception**

```
use Elasticsearch\ClientBuilder;
use Elasticsearch\Common\Exceptions\Missing404Exception;

$client = ClientBuilder::create()
    ->setHosts(['localhost:9200'])
    ->build();
$params = [
    'index' => 'my_index',
    'id'     => 'unknown_id'
];
try {
    $result = $client->get($params);
} catch (Missing404Exception $e) {
    printf ("Document not found: %s\n", $e->getMessage());
}
```

You Know, for Search!



Searching

- The client gives full access to every query and parameter exposed by the REST API, following the naming scheme as much as possible

```
$params = [  
  'index' => 'my_index',  
  'body' => [  
    'query' => [  
      'match' => [  
        'testField' => 'abc'  
      ]  
    ]  
  ]  
];
```

```
$result = $client->search($params);  
var_dump($result);
```

```
array(4) {  
  'took' => int(1)  
  'timed_out' => bool(false)  
  '_shards' => array(4) {  
    'total' => int(1)  
    'successful' => int(1)  
    'skipped' => int(0)  
    'failed' => int(0)  
  }  
  'hits' => array(3) {  
    'total' => array(2) {  
      'value' => int(1)  
      'relation' => string(2) "eq"  
    }  
    'max_score' => double(0.2876821)  
    'hits' => array(1) {  
      [0] =>  
        array(5) {  
          ...  
        }  
    }  
  }  
  ...  
}
```

RESULTS

Using raw JSON

- You can use a JSON string in the body, and the PHP client will detect it automatically

```
$json = '{
    "query" : {
        "match" : {
            "testField" : "abc"
        }
    }
}';

$params = [
    'index' => 'my_index',
    'body' => $json
];

$result = $client->search($params);
var_dump($result);
```

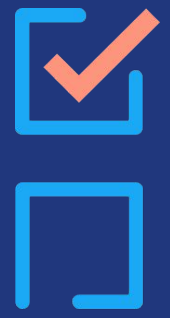
Scrolling

- The scrolling functionality of Elasticsearch is used to **paginate** over many documents (max. 10,000 hits)*
- It is **more efficient** than regular search because it doesn't need to maintain an expensive priority queue ordering the documents
- Scrolling works by maintaining a "**point in time**" snapshot of the index which is then used to page over
- You execute a search request with **scroll enabled**. This returns a "**page**" of documents, and a **scroll_id** which is used to continue paginating through the hits

* = for more than 10'000 we recommend the usage of [scroll search result](#) API

Scrolling example

```
$params = [  
  'scroll' => '30s',  
  'size'   => 50,  
  'index'  => 'my_index',  
  'body'   => [  
    'query' => [ 'match_all' => new \stdClass() ]  
  ]  
];  
$result = $client->search($params);  
while (isset($result['hits']['hits']) && count($result['hits']['hits']) > 0) {  
  // Work on $result['hits']['hits'] array  
  // ...  
  $result = $client->scroll([  
    'body' => [  
      'scroll_id' => $result['_scroll_id'],  
      'scroll'    => '30s'  
    ]  
  ]);  
}
```



Fuzzy Search



Fuzzy search

- Returns documents that contain terms similar to the search term, as measured by a [Levenshtein](#) edit distance
- An **edit distance** is the number of one-character changes needed to turn one term into another
- These changes can include:
 - Changing a character (**b**ox → **f**ox)
 - Removing a character (**b**lack → lack)
 - Inserting a character (sic → sick**k**)
 - Transposing two adjacent characters (**a**ct → **c**at)

Example

```
$params = [  
  'index' => 'my_index',  
  'body' => [  
    'query' => [  
      'fuzzy' => [  
        'doc' => [  
          "value" => "harry"  
        ]  
      ]  
    ]  
  ]  
];  
  
$result = $client->search($params);
```



{ "_id": "doc1",
 "doc": "I will **marry** you because I love you" }



{ "_id": "doc2",
 "doc": "I will live with **harry**" }



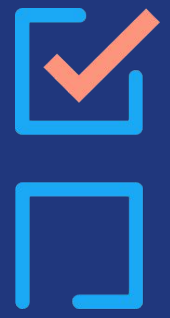
{ "_id": "doc3",
 "doc": "I'm sorry for your loss" }

Lev('harry', 'marry') = 1 in doc1

Lev('harry', 'harry') = 0 in doc2

Lev('harry', 'sorry') = 2 in doc3

where **Lev** is the **Levenshtein** function



Aggregation



Aggregation

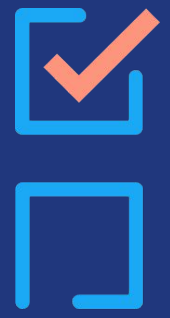
- An **aggregation** summarizes your data as metrics, statistics, or other analytics
- Aggregations help you answer questions like:
 - What's the average load time for my website?
 - Who are my most valuable customers based on transaction volume?
 - What would be considered a large file on my network?
 - How many products are in each product category?

Example

```
$params = [  
  'index' => 'stock-market',  
  'body' => [  
    'aggs' => [  
      'my-agg-name' => [  
        'terms' => [  
          'field' => 'stock'  
        ]  
      ]  
    ]  
  ]  
];  
$result = $client->search($params);  
var_dump($result);
```

```
array(5) {  
  'took' =>  
  int(40)  
  'timed_out' =>  
  bool(false)  
  '_shards' =>  
  array(4) {  
    'total' =>  
    int(1)  
    'successful' =>  
    int(1)  
    'skipped' =>  
    int(0)  
    'failed' =>  
    int(0)  
  }  
  'hits' =>  
  array(3) { ... }  
  'aggregations' =>  
  array(1) {  
    'my-agg-name' =>  
    array(3) {  
      'doc_count_error_upper_bound' =>  
      int(0)  
      'sum_other_doc_count' =>  
      int(606450)  
      'buckets' =>  
      array(10) {  
        ...  
      }  
    }  
  }  
}
```

RESULTS

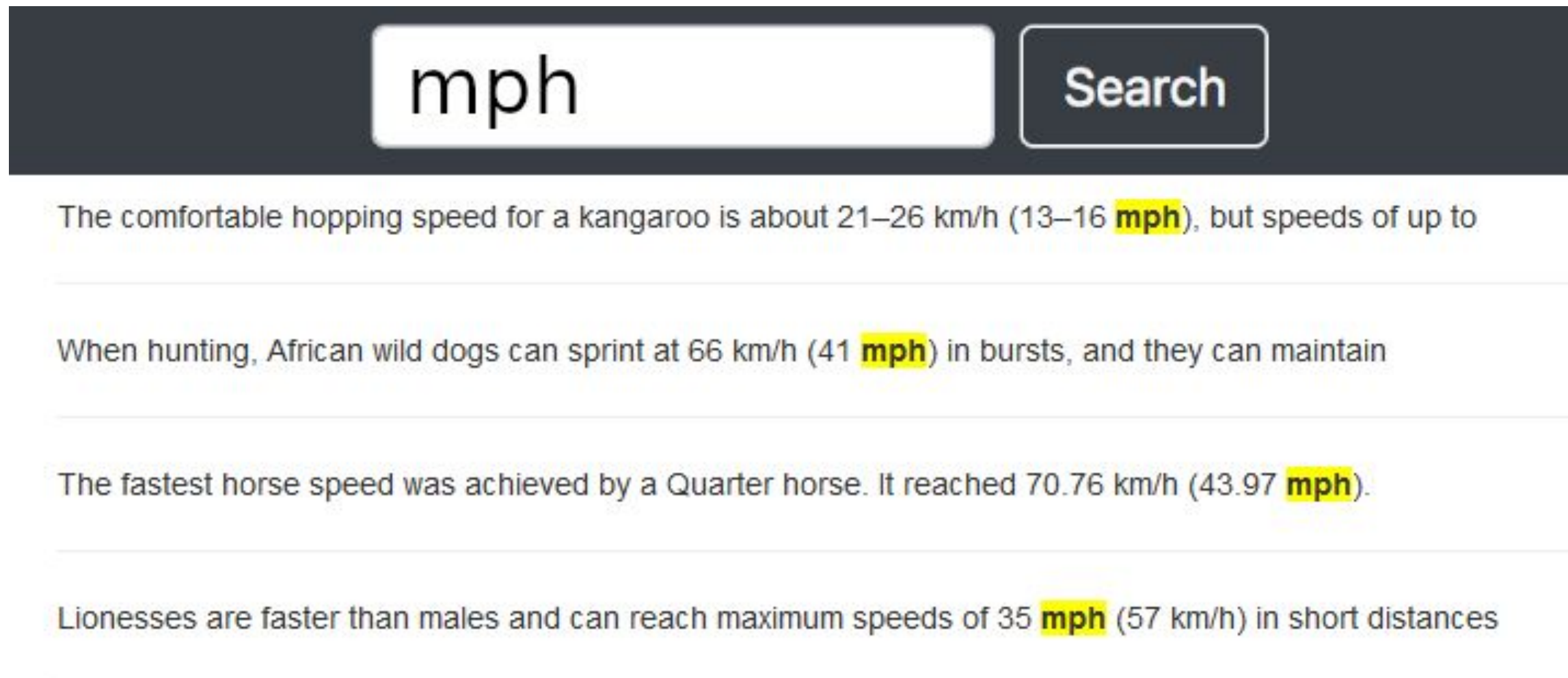


Highlighting



Highlighting

- **Highlighters** enable you to get highlighted snippets from one or more fields in your search results so you can show users where the query matches are



The screenshot shows a search interface with a dark background. A search bar contains the text "mph" and a "Search" button. Below the search bar, four search results are displayed, each with a highlighted snippet containing "mph".

The comfortable hopping speed for a kangaroo is about 21–26 km/h (13–16 **mph**), but speeds of up to

When hunting, African wild dogs can sprint at 66 km/h (41 **mph**) in bursts, and they can maintain

The fastest horse speed was achieved by a Quarter horse. It reached 70.76 km/h (43.97 **mph**).

Lionesses are faster than males and can reach maximum speeds of 35 **mph** (57 km/h) in short distances

Example

```
$params = [  
  'index' => 'stock-demo-v1',  
  'body' => [  
    'query' => [  
      'match' => [  
        'name' => 'AAL'  
      ]  
    ],  
    'highlight' => [  
      'fields' => [  
        'name' => new \stdClass()  
      ]  
    ]  
  ]  
];  
$result = $client->search($params);  
foreach ($result['hits']['hits'] as $res) {  
  print_r($res['highlight']['name']);  
}
```

Array

```
(  
  [0] => The comfortable hooping ...  
  <em>mph</em>) but ...
```

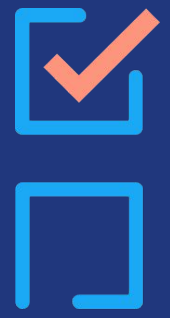
)

Array

```
(  
  [0] => When hunting ... <em>mph</em>) in  
burst ...
```

)

...



Schema on read



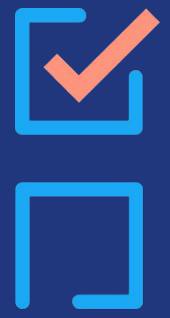
Schema on read

- **Elasticsearch 7.12** introduced the ability to change **schema on read** using runtime fields
- Runtime fields let you define and evaluate fields at **query time**, which opens a wide range of new use cases
- For instance:
 - adapt to a changing log format or fix an index mapping;
 - don't have intimate knowledge of data, you can use runtime fields and define your schema without impacting others

Example

- Create a field with the **average** of high and low stock prices

```
$result = $client->search([
  'index' => 'stock-options',
  'body' => [
    'runtime_mappings' => [
      'average' => [
        'type' => 'double',
        'script' => [
          'source' => "emit((double) (doc['high'].value + doc['low'].value)/2)"
        ]
      ]
    ],
    'fields' => [
      'average'
    ]
  ]
]);
```



Asynchronous calls



Future mode (async)

- The client offers a mode called **future** or **async** mode. This allows batch processing of requests (sent in parallel to the cluster), which can have a dramatic impact on performance and throughput
- PHP is fundamentally single-threaded, however, [libcurl](#) provides a functionality called the "**multi interface**"

Future mode example

```
$params = [  
  'index' => 'test',  
  'id'    => 1,  
  'client' => [  
    'future' => 'lazy'  
  ]  
];  
  
$future = $client->get($params);  
  
$doc = $future['_source']; // This call blocks and forces the future to resolve
```

Future resolution with wait()

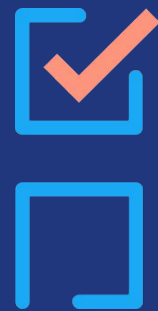
```
$client = ClientBuilder::create()->build();
$futures = [];

for ($i = 0; $i < 1000; $i++) {
    $params = [
        'index' => 'test',
        'id'     => $i,
        'client' => [
            'future' => 'lazy'
        ]
    ];

    $futures[] = $client->get($params); //queue up the request
}

//wait() forces future resolution and will execute the underlying curl batch
$futures[999]->wait();
```

More information about [Future mode](#)



The next elasticsearch-php



Elasticsearch-php 8.x

- A brand new PHP client with **Elasticsearch request and response types** (no more associative array!)
- Use the new features of **PHP 8**: named arguments, union types, constructor property promotion, etc
- Use [PSR](#) standards:
 - [PSR-3](#) for logging
 - [PSR-7](#) for HTTP messages
 - [PSR-17](#) for HTTP factories
 - [PSR-18](#) for HTTP Client
- We will continue to offer **async HTTP call**
- For more information: [elastic/elastic-transport-php](https://www.elastic.co/guide/en/elasticsearch/client/php-api/current/)



Thanks!

For more information:

[Elasticsearch PHP documentation](#)

[Elasticsearch-php github repository](#)