Documentos Indexados

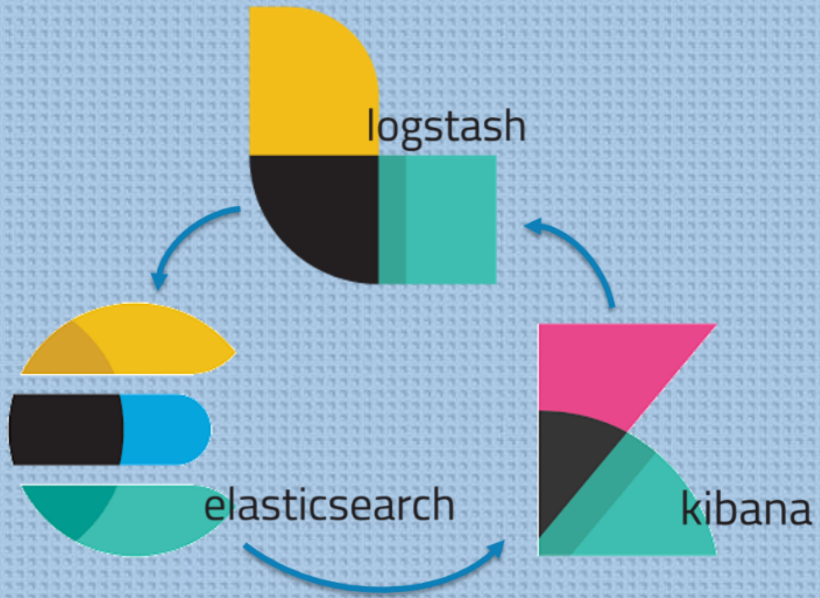
GFT

Roberto López

Curso 2020/2021 - Edición 2

Fecha 17/12/2020

Agenda

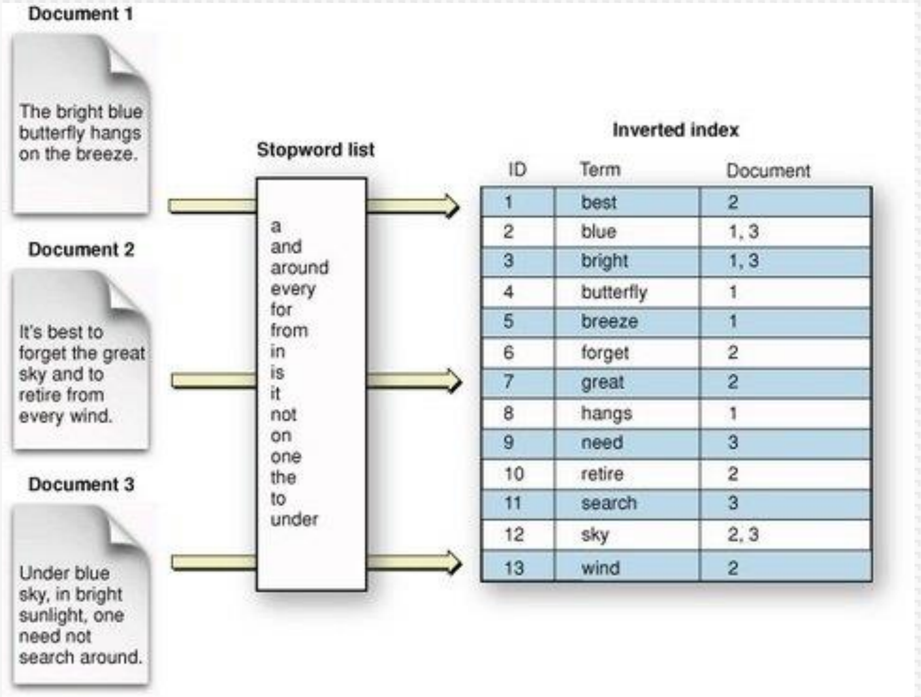


Elastic Stack

ELK Stack

What is elasticsearch?

- Elasticsearch is an inverted index manager of documents:
  - The content of the files are indexed
  - Content is splitted on words or numbers, called terms, that are mapped to their position on the documents
  - Purpose is to allow fast full text searches
- Near real time search platform . You can perform queries and aggregations
- Main cost is on indexing, it can take a second from the time you index a document until it becomes "searchable"



Document 1: The bright blue butterfly hangs on the breeze.

Document 2: It's best to forget the great sky and to retire from every wind.

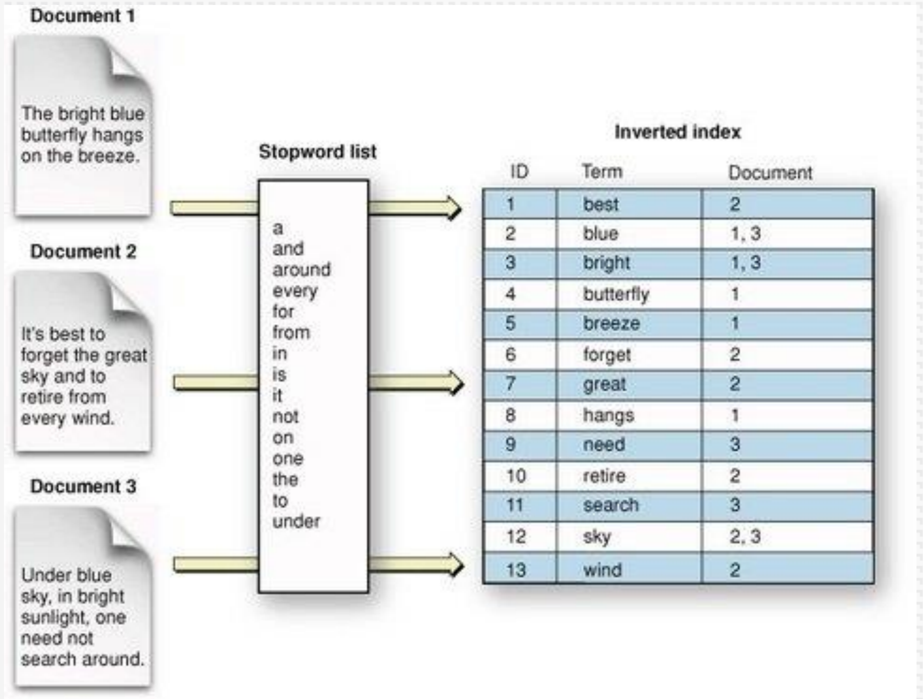
Document 3: Under blue sky, in bright sunlight, one need not search around.

Stopword list: a, and, around, every, for, from, in, is, it, not, on, one, the, to, under

ID	Term	Document
1	best	2
2	blue	1, 3
3	bright	1, 3
4	butterfly	1
5	breeze	1
6	forget	2
7	great	2
8	hangs	1
9	need	3
10	retire	2
11	search	3
12	sky	2, 3
13	wind	2

Key Features

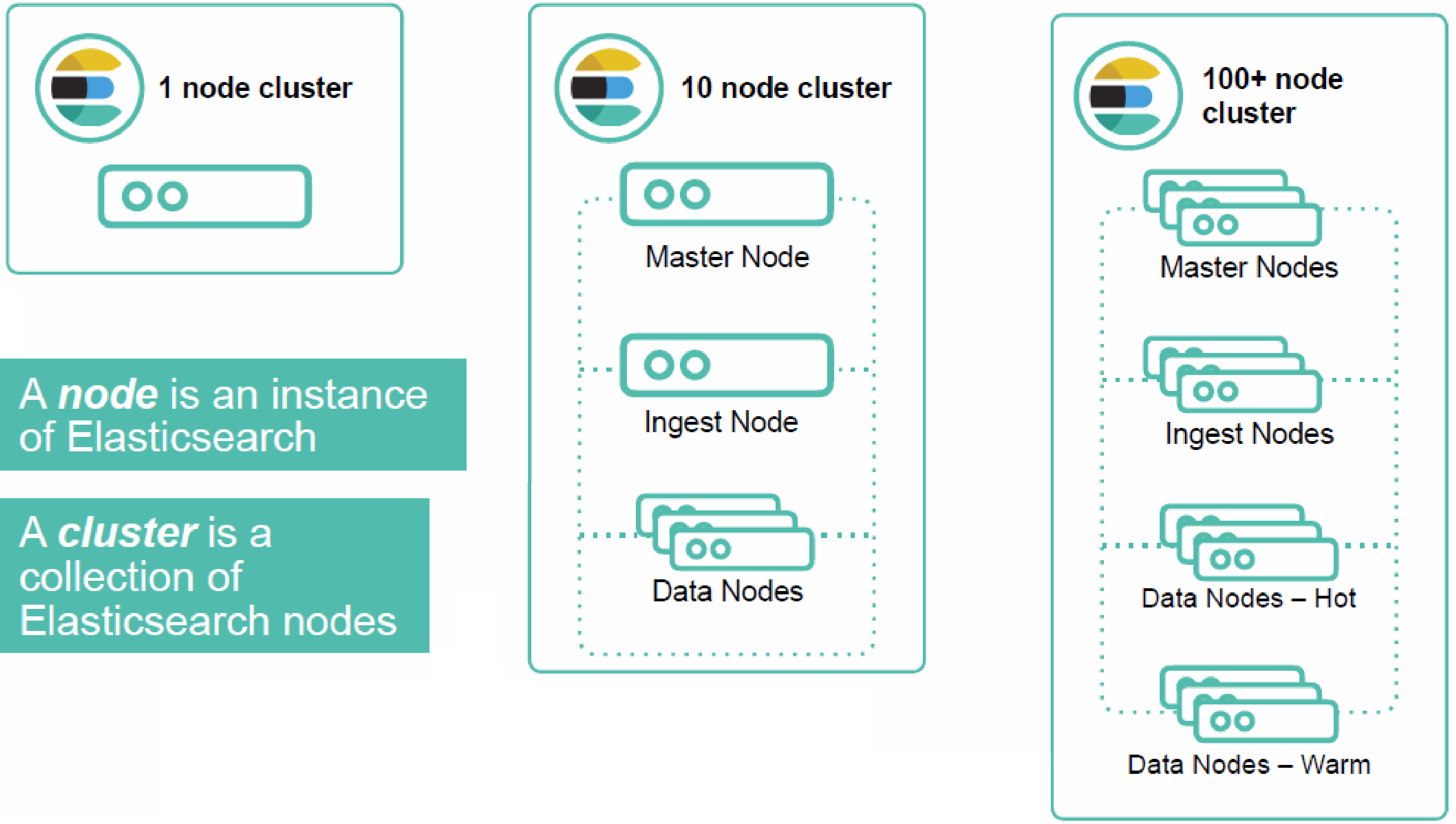
- Elasticsearch is an inverted index manager of documents:
  - The content of the files are indexed
  - Content is splitted on words or numbers, called terms, that are mapped to their position on the documents
  - Purpose is to allow fast full text searches
- Near real time search platform
- Main cost is on indexing, it can take a second from the time you index a document until it becomes "searchable"


Document 1
The bright blue butterfly hangs on the breeze.
Document 2
It's best to forget the great sky and to retire from every wind.
Document 3
Under blue sky, in bright sunlight, one need not search around.
Stopword list
a
and
around
every
for
from
in
is
it
not
on
one
the
to
under
Inverted index

ID	Term	Document
1	best	2
2	blue	1, 3
3	bright	1, 3
4	butterfly	1
5	breeze	1
6	forget	2
7	great	2
8	hangs	1
9	need	3
10	retire	2
11	search	3
12	sky	2, 3
13	wind	2

Scalability and High Avalilability

Is distributed and scales horizontally to hundreds of servers and petabytes of structured and unstructured data



A ***node*** is an instance of Elasticsearch

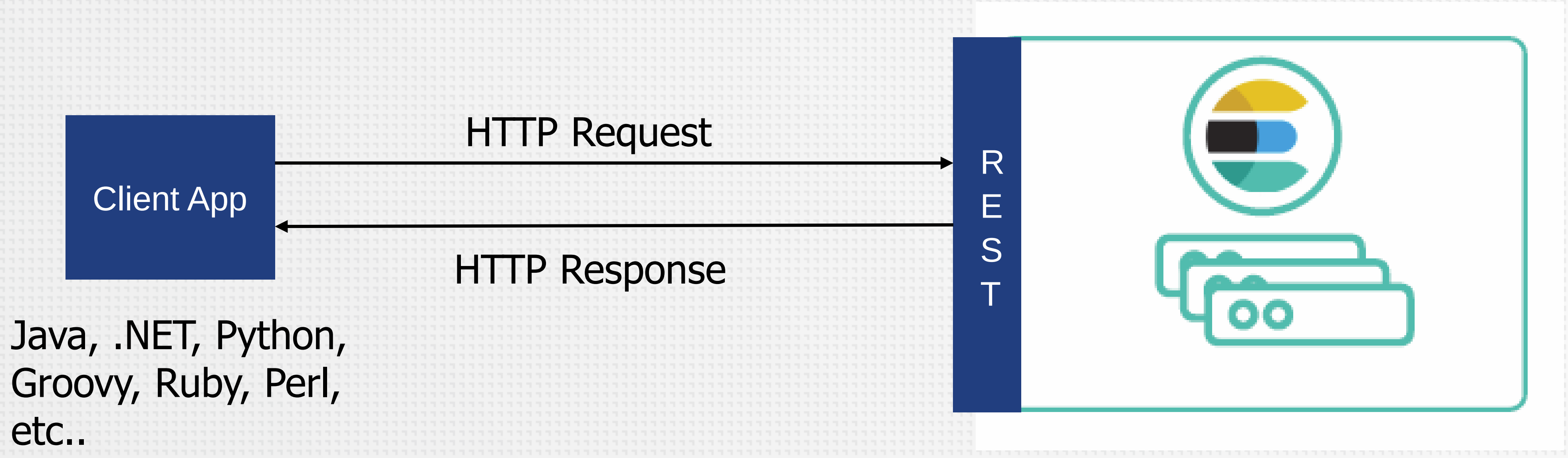
A ***cluster*** is a collection of Elasticsearch nodes

EDEM Escuela de Empresarios

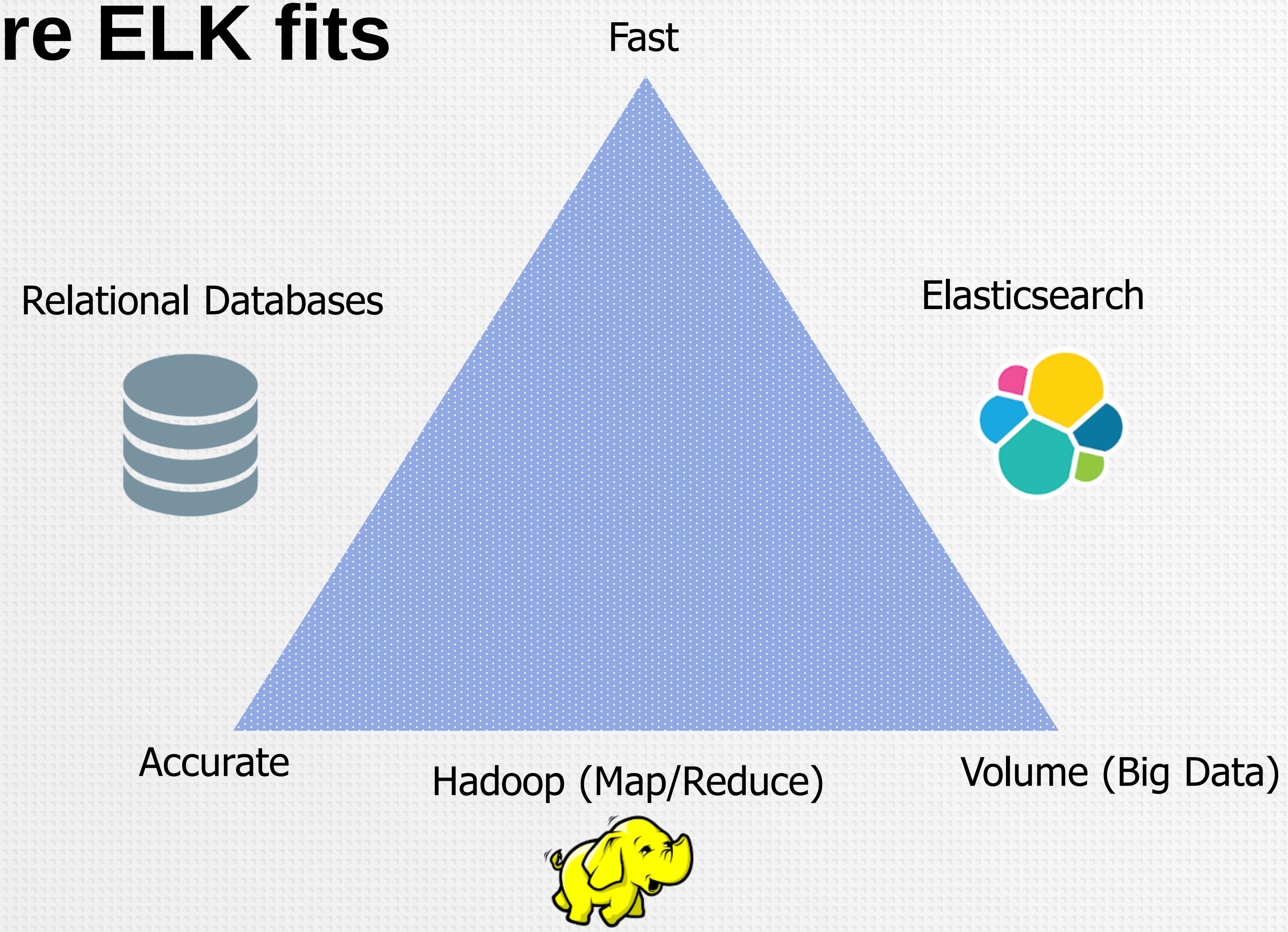
GFT

Developer friendly

- Provides RESTful JSON APIs for communicating with a cluster over HTTP
  - Allows client applications to be written in any language



Where ELK fits



Examples of use cases

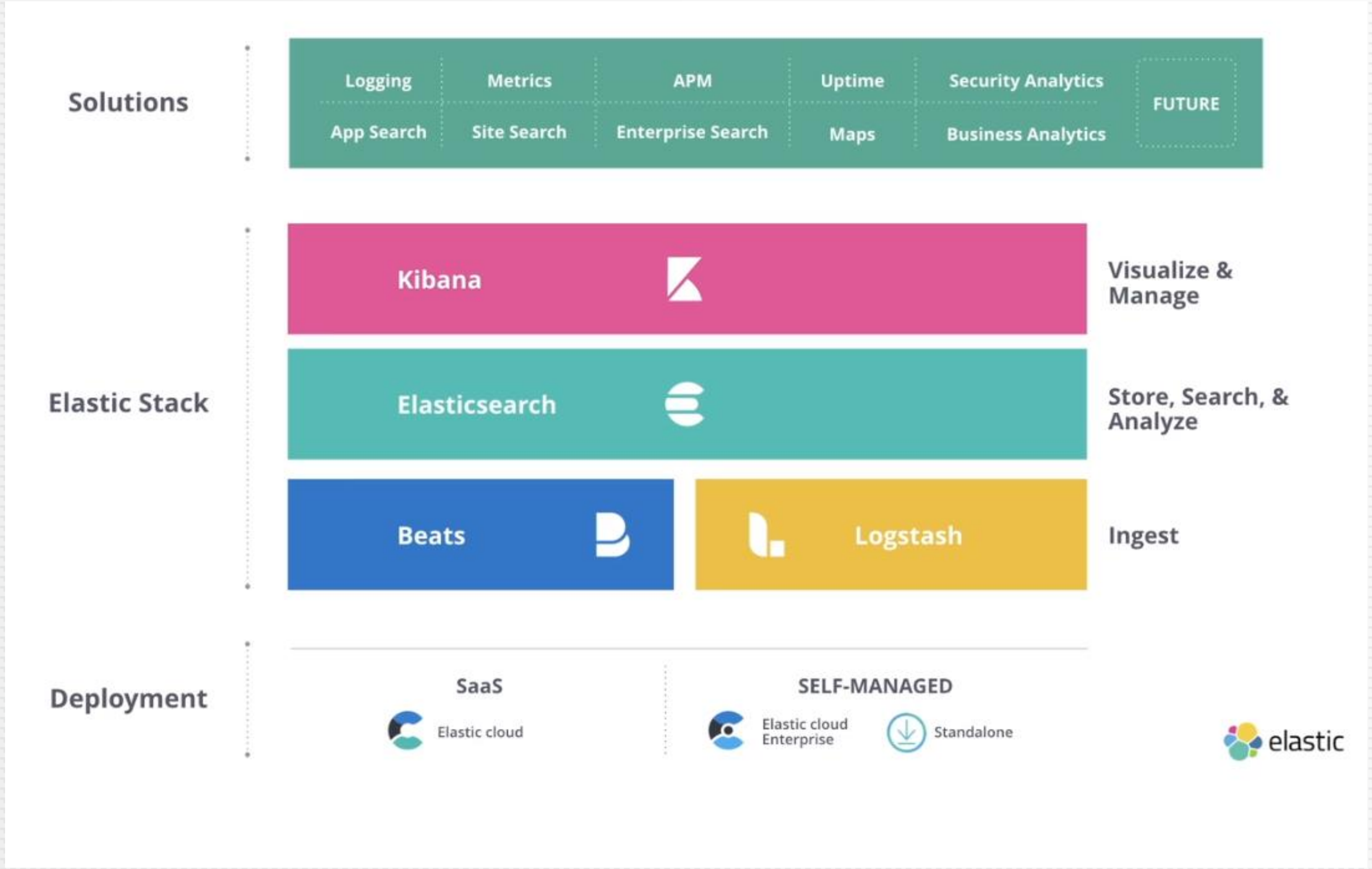
- Wikipedia uses Elasticsearch to provide full-text search with highlighted search snippets, and *search-as-you-type* and *did-you-mean* suggestions.

- *The Guardian* uses it to combine visitor logs with social -network data to provide real-time feedback to its editors about the public's response to new articles.

- Stack Overflow combines full-text search with geolocation queries and uses *more-like-this* to find related questions and answers.

- GitHub uses Elasticsearch to query 130 billion lines of code.

Elastic Ecosystem



Example of projects

ELK Stack

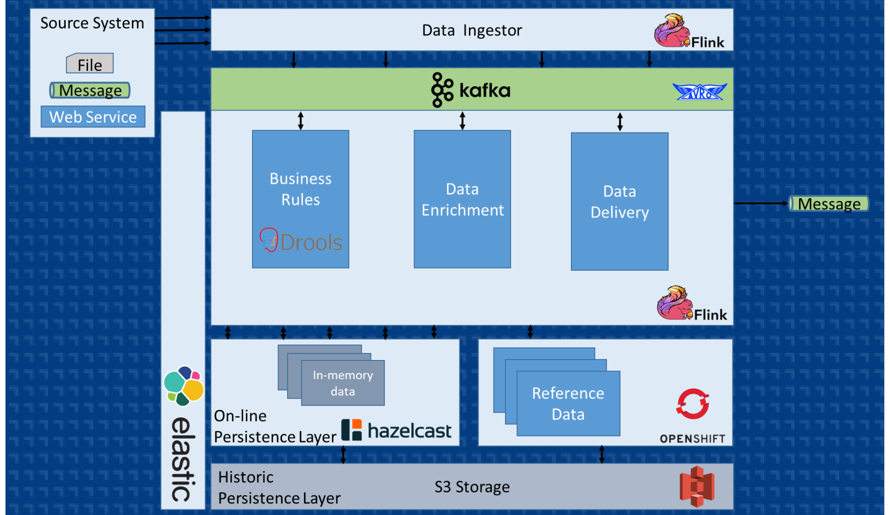
Elastic Stack

EDEM Escuela de Empresarios

GFT

Cluster on-line in a Global Bank

Success story



DATA STREAMING

CHALLENGE

Design an implement an Strategic Architecture for Accounting Integration layer to allow all investment banking products follow same accounting approach:

- Implement a common accounting rules cross products and source systems
- Ingestion layer on Kafka, implementing a Kappa architecture that allows to ingest and process streaming and batch sources
- Real time process at microsecond scale thanks to Flink technology.
- Big data architecture horizontally scalable.

ENGAGEMENT

Design and implement a solution based on Kafka, Flink, Hazelcasat, Microservices, Avro:

- Kafka is used as the message broker and ingestion layer
- Flink is used as the process engine to do the accounting an enrich the data required for downstream system.
- Hazelcast is is used for the data persistence in memory allowing low latencies. Drools as the accounting rules framework.
- Microservices layer is in charge of providing more data required to enrich the transactions.
- Avo is used to define the data schema of the application.

BENEFIT

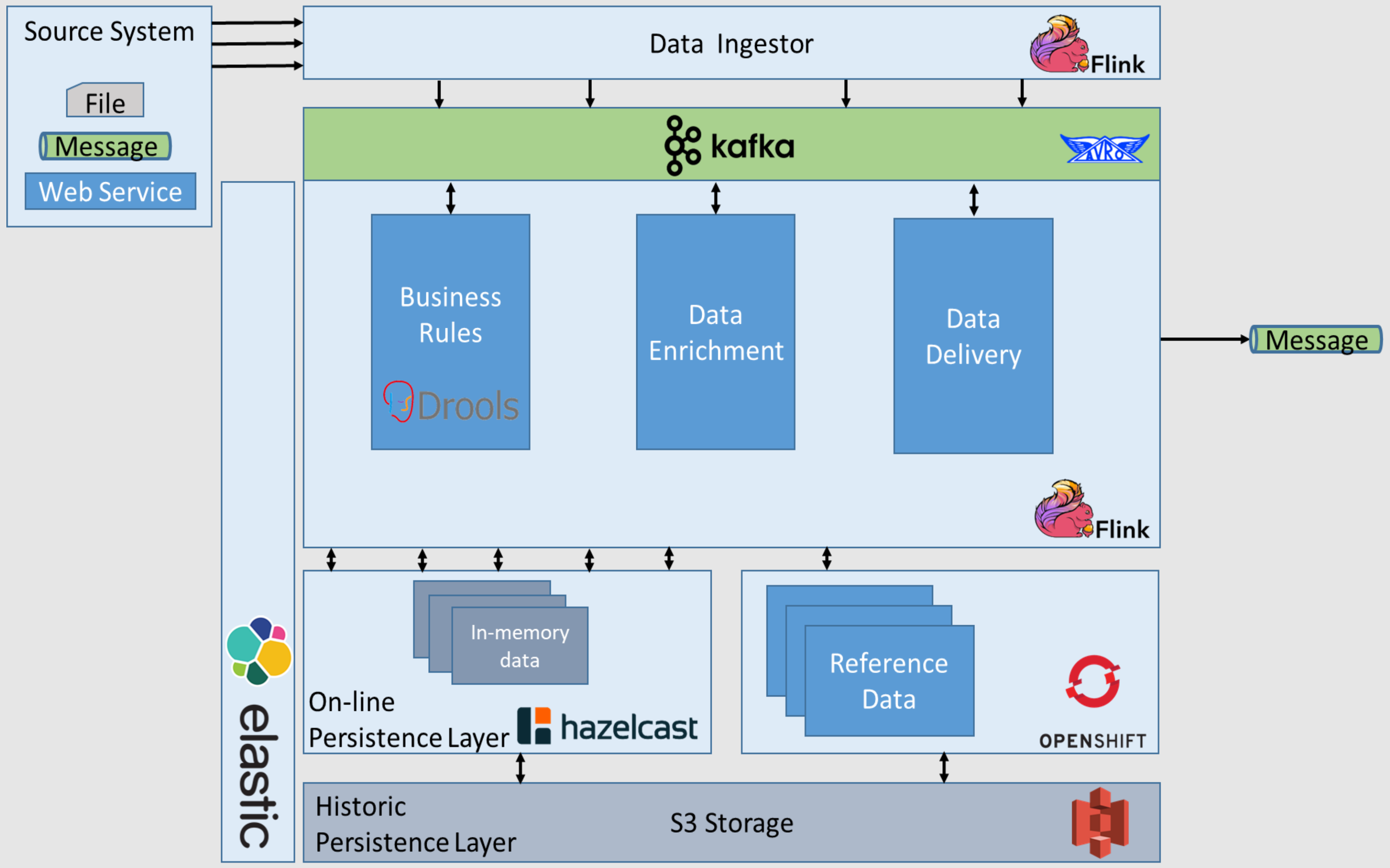
Single accounting schema for all investment products transparent and visible across the board

- New accounting piece fully reutilizable for other products and source systems.
- Own data schema that is independant upstream and downstream data schemas.
- Fully horizontally scalable solution
- Real-time accounting and alerts detection allowing to solve the issues as soon as happens, not waiting to a batch process when maybe is too late.

Elastic Stack

EDEM
Escuela de Empresarios

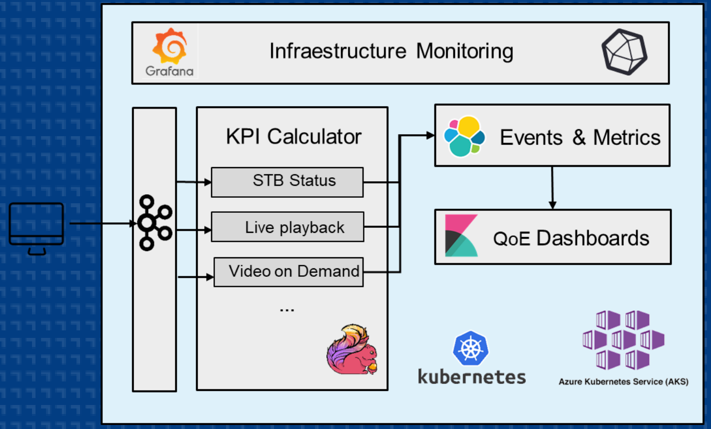
GFT



Quality of Experience of a Video Streaming platform

Success story

Monitor on real-time user experience and quality of service on a video streaming platform.



INTERNET OF THINGS

BIG DATA STREAMING

CHALLENGE

Develop a Quality of Experience Metrics calculator of real-time user events of a video streaming platform with high number of users.

- Video Streaming platform with million users spread around the world and huge volume (4TB of data per day and peaks of 400.000 events per second on live events)
- Provide a real-time engine to calculate KPI Metrics.
- Provide a real-time analytics platform to monitor Quality of Service.
-

ENGAGEMENT

Develop a real-time KPI calculation engine

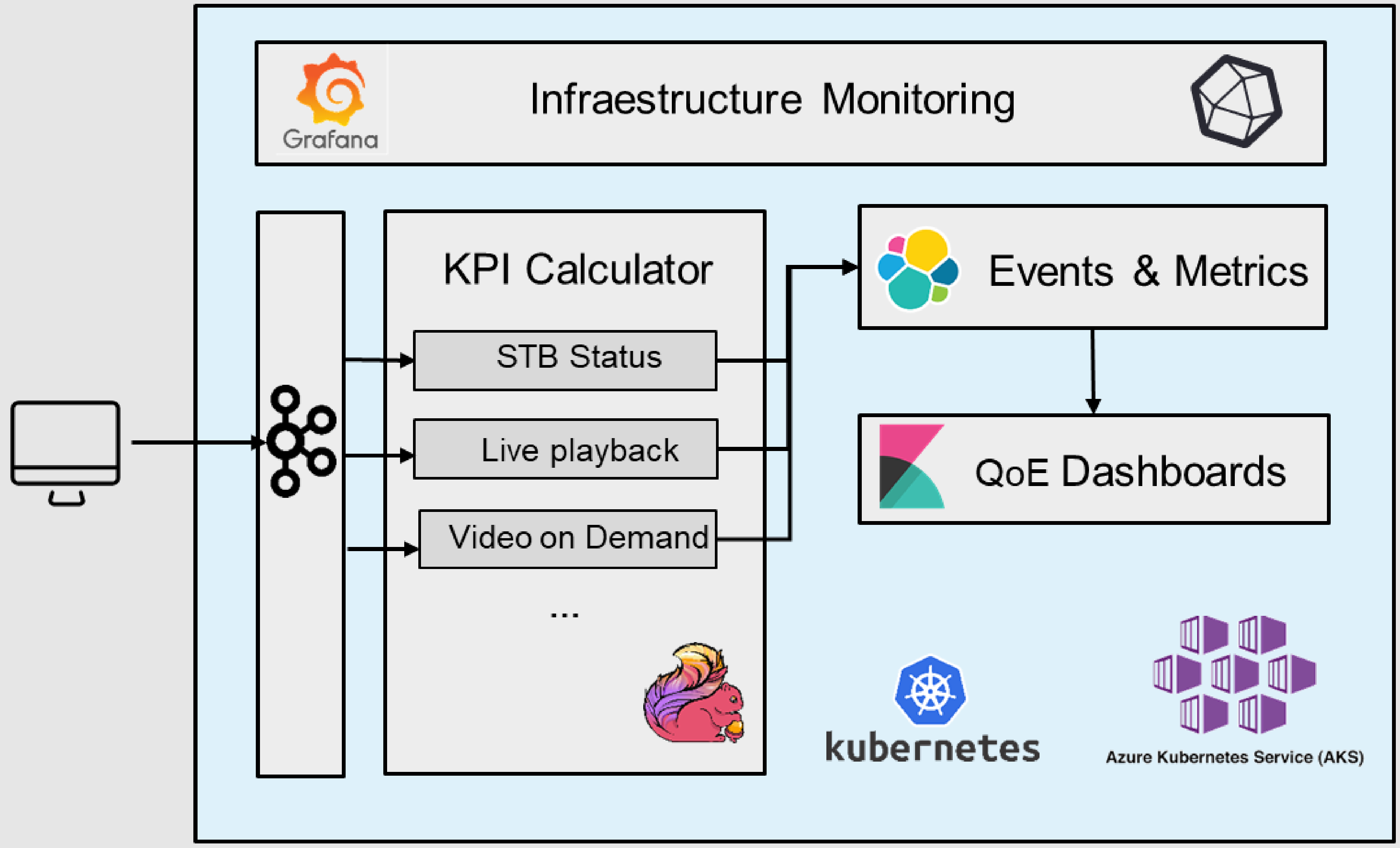
- Use Flink to build a KPI calculation engine because it's low latency and functionality to window aggregation.
- Fine tuning of those jobs to address million of users with several million of events per day.
- Integrate this process engine with kafka broker infrastructure where Set of Box are sending real-time events.
- Deploy on a Kubernetes cluster using AKS.

BENEFIT

Obtain quality of service and monitor user experience on real-time

- React to bad quality of experience indicators on real time and mitigate any usability issue.
- Detect infrastructure/streaming delivery issues to enhance the platform
- Perform analytics over quality of experience in order to define strategy on product evolution

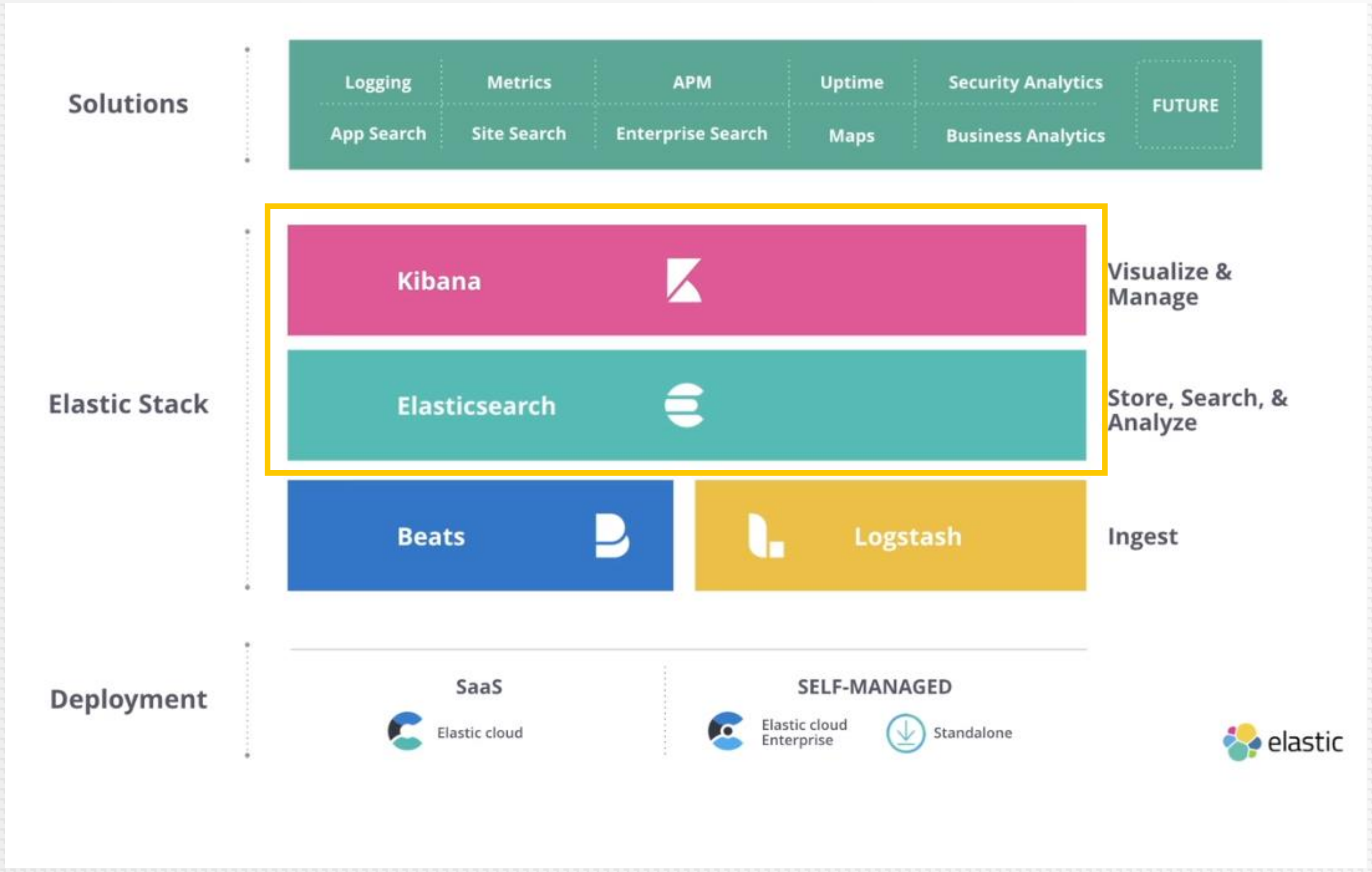
Elastic Stack



EDEM Escuela de Empresarios

GFT

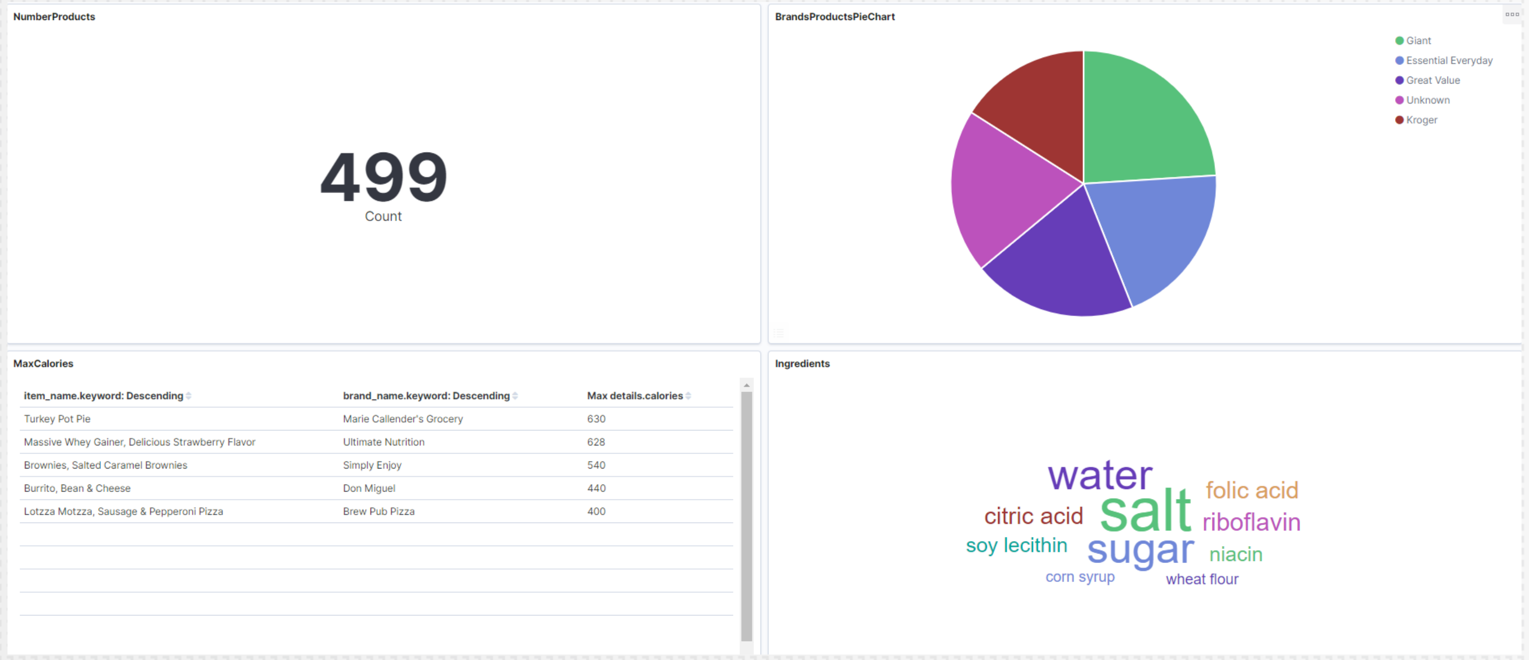
Demo



Demo

Using nutrition dataset already loaded on elasticsearch (AlmacenamientoProcesamiento\docker\data\nutrition\nutrition.csv) build a dashboard in Kibana with the following:

- Total number of products of the dataset.
- Pie chart with top5 brands containing products.
- Table with top 5 calorie products.
- TagCloud with ingredients more common.


NumberProducts

499
Count

BrandsProductsPieChart

- ● Giant
- ● Essential Everyday
- ● Great Value
- ● Unknown
- ● Kroger

MaxCalories

item_name.keyword: Descending	brand_name.keyword: Descending	Max details.calories
Turkey Pot Pie	Marie Callender's Grocery	630
Massive Whey Gainer, Delicious Strawberry Flavor	Ultimate Nutrition	628
Brownies, Salted Caramel Brownies	Simply Enjoy	540
Burrito, Bean & Cheese	Don Miguel	440
Lotzza Motzza, Sausage & Pepperoni Pizza	Brew Pub Pizza	400

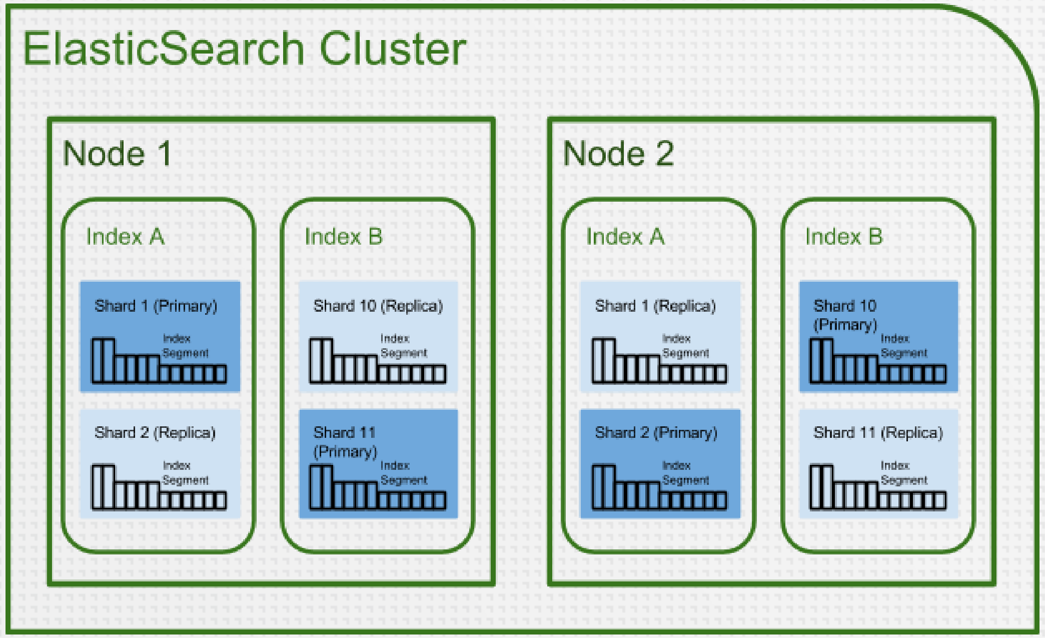
Ingredients

Elasticsearch

ELK Stack

Components of Elasticsearch

- The main components of Elasticsearch are:
  - ***Node***: an instance of Elasticsearch
  - ***Cluster***: one or more nodes that work together to serve the same data and API requests
  - ***Document***: a piece of data that you want to search
  - ***Index***: a collection of documents that have somewhat similar characteristics

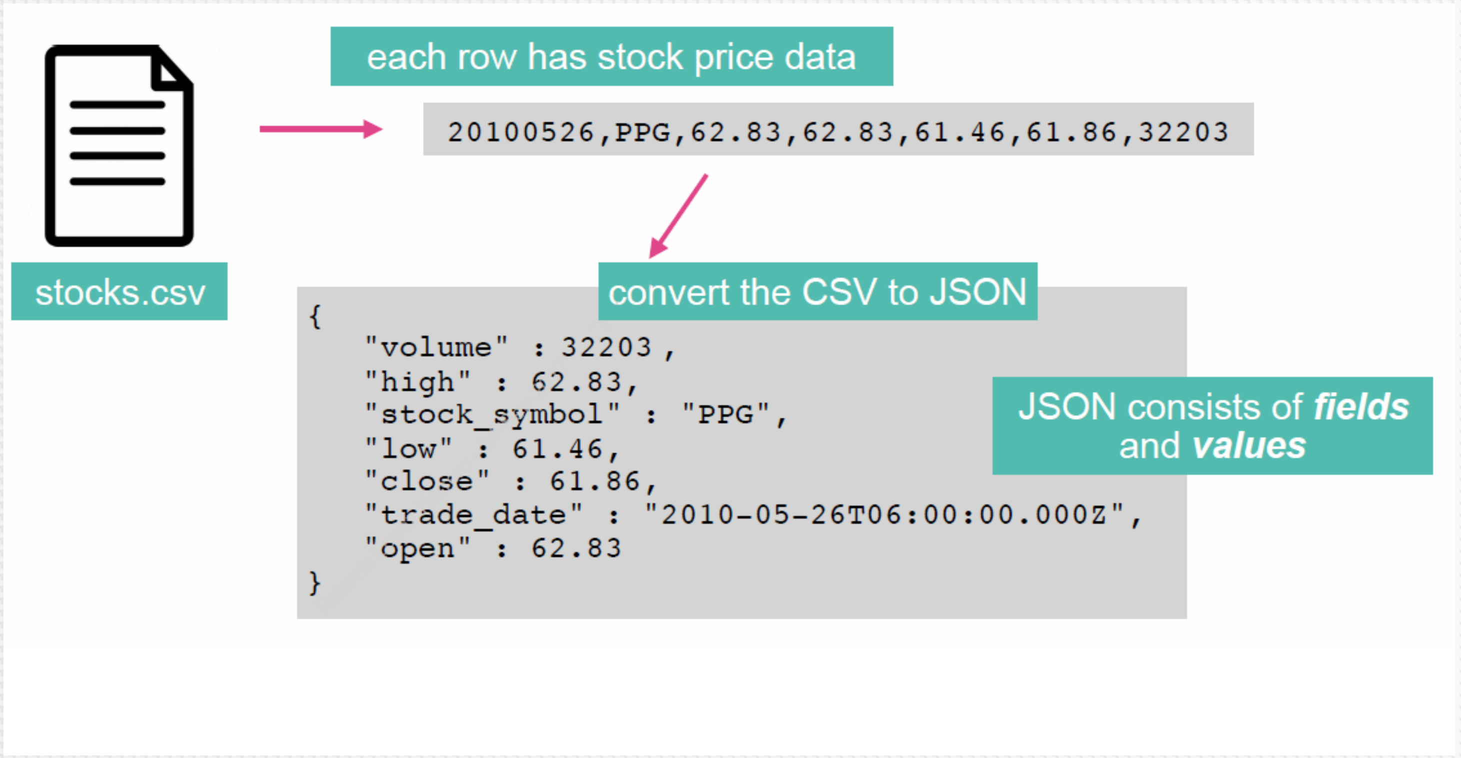


Documents

- A ***document*** can be any text or numeric data you want to search and/or analyze
- For example:
  - An entry in a log file
  - A comment made by a customer on your website,
  - The weather details from a weather station at a specific moment
- Specifically, a ***document*** is a top-level object that is
  - Serialized into JSON and stored in Elasticsearch
- Every document has a ***unique ID***
  - Which either you provide
  - Or Elasticsearch generates for you

Documents

- **Documents must be Json Objects**
- Suppose you have some stock prices in a CSV format
  - The data needs to be converted to JSON
  - Each row of data represents a single document


stocks.csv
each row has stock price data
convert the CSV to JSON
JSON consists of *fields* and *values*

```
20100526,PPG,62.83,62.83,61.46,61.86,32203
```

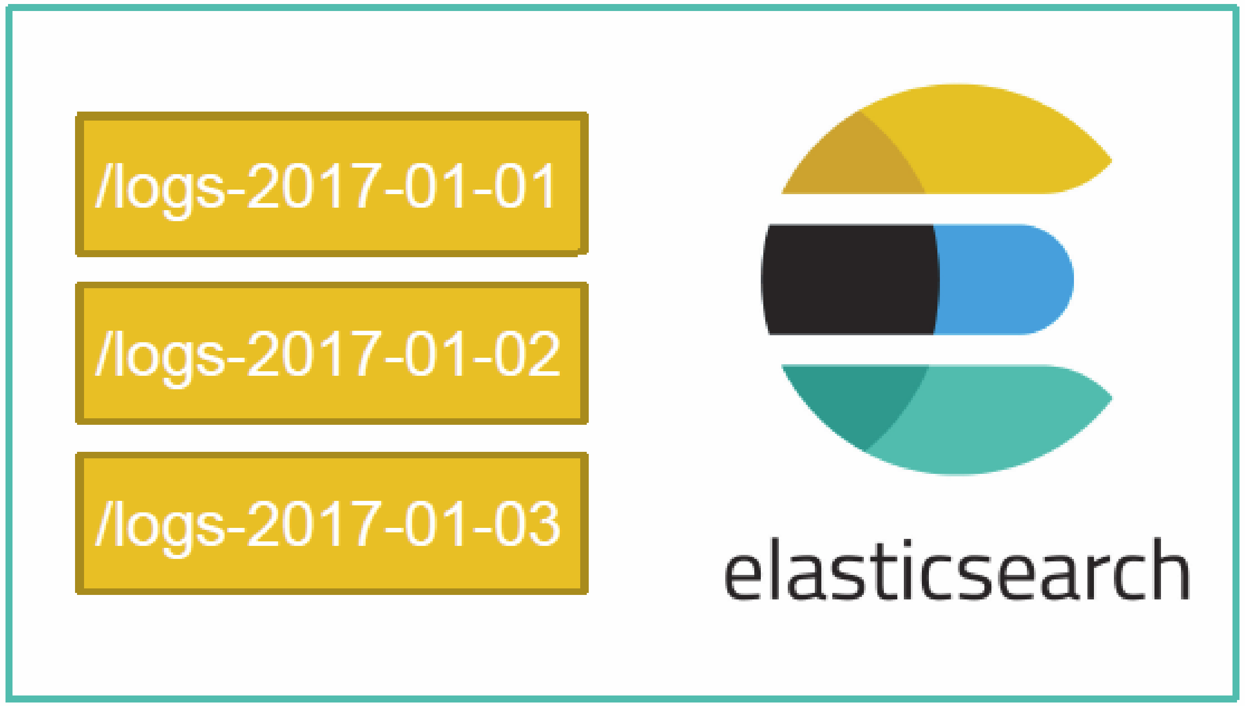
```
{
   "volume" : 32203 ,
   "high" : 62.83,
   "stock_symbol" : "PPG",
   "low" : 61.46,
   "close" : 61.86,
   "trade_date" : "2010-05-26T06:00:00.000Z",
   "open" : 62.83
}
```

Indexes

- An ***index*** in Elasticsearch is a ***logical*** way of grouping data:
  - An index contains ***mappings*** that define the documents' field names and data types of the index
  - A ***logical namespace*** that maps to where its contents are stored in the cluster
- There are two different concepts in this definition:
  - An index has some type of data schema mechanism
  - An index has some type of mechanism to distribute data across a cluster

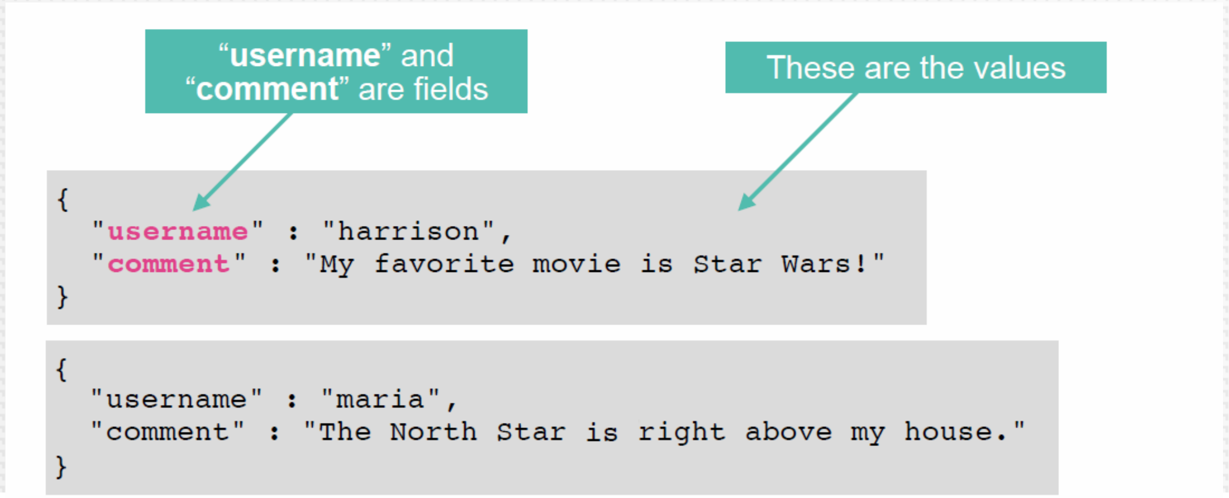
Examples of indexes

- Indexes are fairly lightweight in Elasticsearch, so it is not unusual to create lots of them
- For example, you might define a new index everyday for searching logs
  - Using multiple indices allows you to organize your data ***logically***:
  - **logs-2017-01-01**
  - **logs-2017-01-02**
  - **logs-2017-01-03**



Indexing a document

- *Index (noun)*
  - An *index* is like a *database* in a traditional relational database. It is the place to store related documents.
- *Index (verb)*
  - *To index a document* is to store a document in an *index (noun)* so that it can be retrieved and queried. It is much like the INSERT keyword in SQL except that, if the document already exists, the new document would replace the old.
- *Inverted index*
  - By default, every field in a document is *indexed* (has an inverted index) and thus is searchable. A field without an inverted index is not searchable.



```
{
  "username" : "harrison",
  "comment" : "My favorite movie is Star Wars!"
}
```

```
{
  "username" : "maria",
  "comment" : "The North Star is right above my house."
}
```

Creating an index

- Clients communicate with a cluster using Elasticsearch’s REST APIs
  - There is a collection of ***Document APIs*** for working with indexes and documents
- An index is defined using the **Create Index API**, which can be accomplished with a simple **PUT** command:
  - “200 OK” response if successful

```
curl -XPUT 'http://localhost:9200/my_index' -H "Content-Type: application/json"
```

Indexing a Document

- The **Index API** is used to index a document
  - Use a **PUT** again, but this time send the document in the
- Body of the PUT request:
  - Notice you specify a document unique ID
  - “201 Created” response if successful

```
curl -XPUT 'http://localhost:9200/my_index/1' -i -H
"Content-Type: application/json" -d '
{
        "username":“Alice",
        “comment":“I love to see the stars at night”
}'
```

Dynamic index creation

- Elasticsearch will create the index for you during the indexing of a document, if the index is not already created:
  - We could have simply executed the command on the previous slide (and skipped that first "**PUT my_index**" command)

my_index is created if it does not exist

If the id already exists, the document is reindexed

```
curl -XPUT 'http://localhost:9200/my_index/doc/1' -i -H
"Content-Type: application/json" -d '
{
        "username":"Alice",
        "comment":"I love to see the stars at night"
}'
```

Index a document without specifying a ID

- You can leave off the id and let Elasticsearch generate one for you:
  - But notice that only works with **POST**, not **PUT**
- **Caveat: Not recommended if you need to search or reindex based on id.**

```
curl -XPOST 'http://localhost:9200/my_index/doc/' -i -H
"Content-Type: application/json" -d '
{
        "username" : "Maria",
        "comment" : "My favorite painting is Starry Night"
}'
```

```
HTTP/1.1 201 Created
Location: /my_index/doc/
AVnWIzyrUxRN673zt1An
content-type: application/json;
charset=UTF-8
content-length: 220
{
"_index" : "my_index",
"_type" : "doc",
"_id" : "AVnWIzyrUxRN673zt1An",
"_version" : 1,
"result" : "created",
"_shards" : {
"total" : 2,
"successful" : 1,
"failed" : 0
},
"created"
```

201 response if successful

Retrieving a Document

- Use **GET** to retrieve an indexed document
  - Returns 200 if the document is found
  - Returns a 404 error if the document is not found

```
curl -XGET 'http://localhost:9200/my_index/1' -H
"Content-Type: application/json"
```

```
{
  "_index": "my_index",
  "_type": "doc",
  "_id": "1",
  "_version": 1,
  "found": true,
  "_source": {
        "username": "Alice",
        "comment": "I love to see the stars at night"
  }
}
```

Searching data

- The simplest search is a **match_all**, which simply matches all documents in the index (or indices) being searched:

Use **GET** or **POST**

Use **GET** or **POST** Invoke the _**search** endpoint

```
curl -XGET 'http://localhost:9200/my_index/_search' -H
"Content-Type: application/json" -d '
{
"query": {
"match_all": {}
}
}'
```

Add a “**query**” clause

match_all” matches all documents

Searching data

```
{
"took": 2,
"timed_out": False,
"_shards": {
        "total": 1,
        "successful": 1,
        "failed": 0,
        "skipped": 0
},
"hits": {
"total": 2,
"max_score": 1,
"hits": [
{
"_index": "my_index",
"_type": "doc",
"_id": "AVlBSnbcmM0CN4-9FrtP",
"_score": 1,
"_source": {
        "username": "maria",
        "comment": "The North Star is right above my house."
}
```

took = the number of milliseconds it took to process the query

total = the number of documents found

hits = array containing the documents that “hit” the search criteria

_id = the document’s unique ID

EDEM
Escuela de Empresarios

GFT

Searching data

- If you want a search that matches all documents, you can simply send a request to the **_search** endpoint
  - And omit the “**query**” body
  - Same results as running a **match_all**
  - By default, Elasticsearch returns 10 results

```
curl -XGET 'http://localhost:9200/my_index/_search' -H
"Content-Type: application/json"
```

CRUD Operations

Index	`PUT my_index/doc/4`
	`{`
	`"username" : "kimchy",`
	`"comment" : "I like search!"`
	`}`

Create	`PUT my_index/doc/4/_create`
	`{`
	`"username" : "kimchy",`
	`"comment" : "I like search!"`
	`}`

Read	`GET my_index/doc/4`

Update	`POST my_index/doc/4/_update`
	`{`
	`"doc" : {`
	`"comment" : "I love search!!"`
	`}`
	`}`

Delete	`DELETE my_index/doc/4`

Differences between Index & Create

- The _**create** operation can explicitly be used when indexing a document
  - If the **id** already exists, the operation fails

```
PUT my_index/doc/5/_create
{
"username" : "ricardo",
"comment" : "I love search too"
}
```

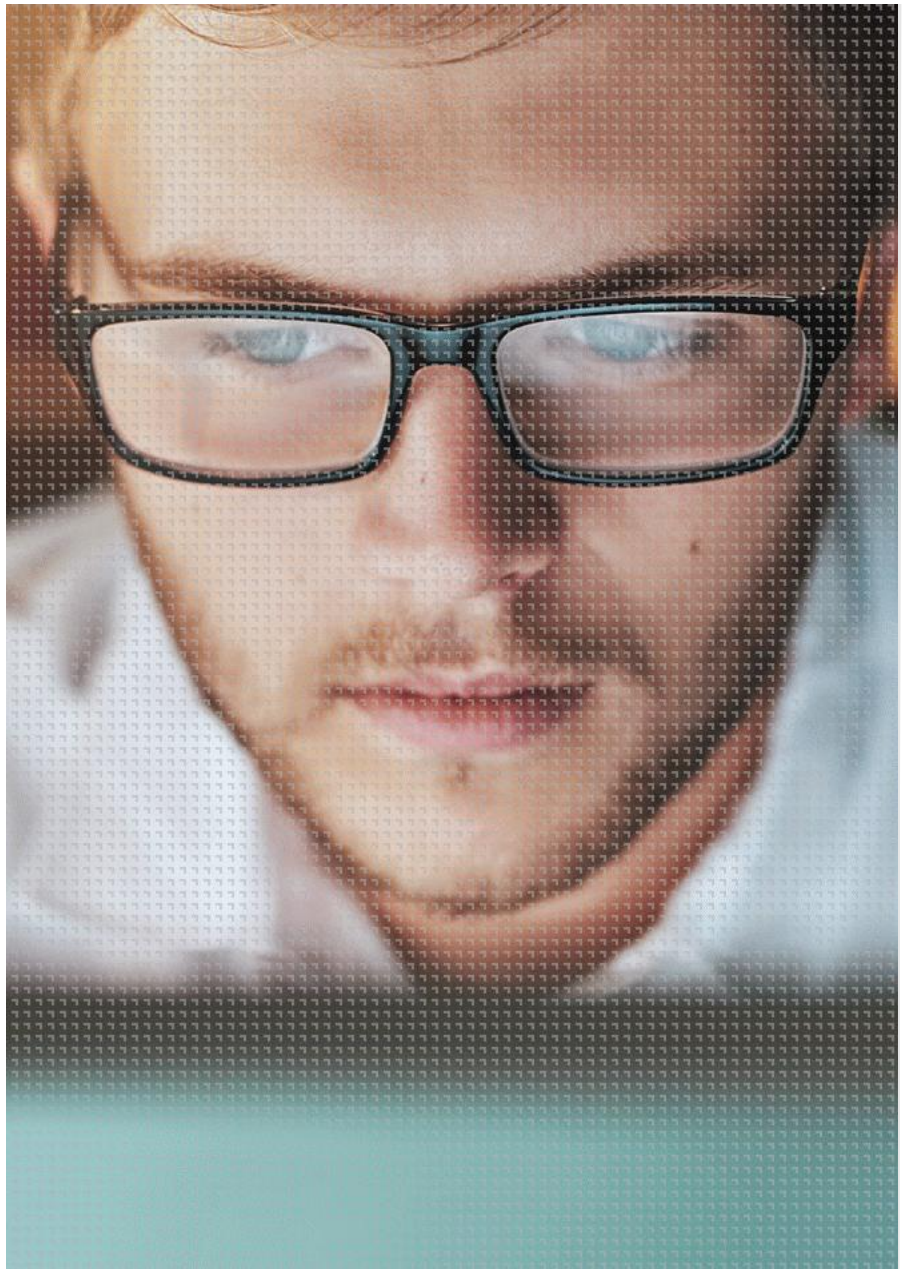
Fails if id=5 is already indexed

```
PUT my_index/doc/6
{
"username" : "maria",
"comment" : "It's a great day for
search"
}
```

Indexes a new document if id=6 is not already indexed. Otherwise, the current doc is
deleted and this new one is indexed.

Query error messages

Issue	Reason	Action
Unable to connect	Networking issue or cluster down	Retry until connection is available (round robin across nodes)
5xx error	Internal server error in Elasticsearch	Retry until successful (using round robin)
4xx error	Invalid JSON or incorrect request structure	Fix bad request before retrying
429 error	Elasticsearch is too busy	Retry (ideally with linear or exponential backoff)
Connection unexpectedly closed	Node died or network issue	Retry (with risk of creating a duplicate document)



Exercise – Play with Elasticsearch

In this exercise you will learn basic usage of Elasticsearch

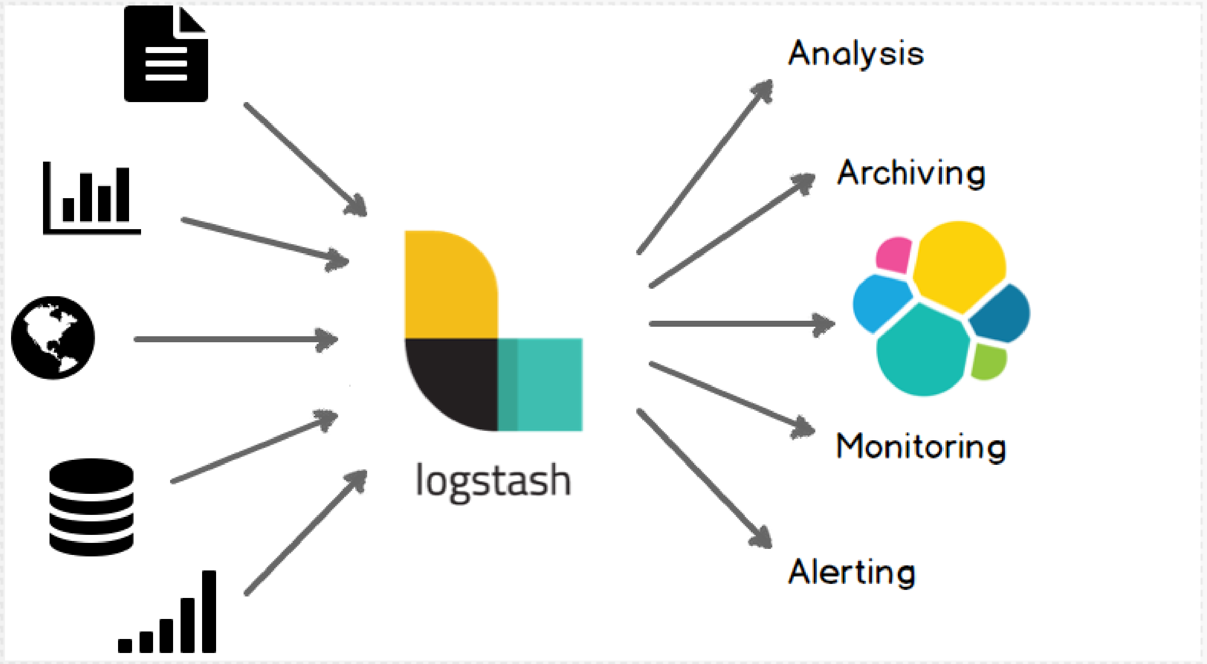
- Go to Exercises doc -> http://bit.ly/EDEMAPExercises
- Perform Exercise 1

Logstash

ELK Stack

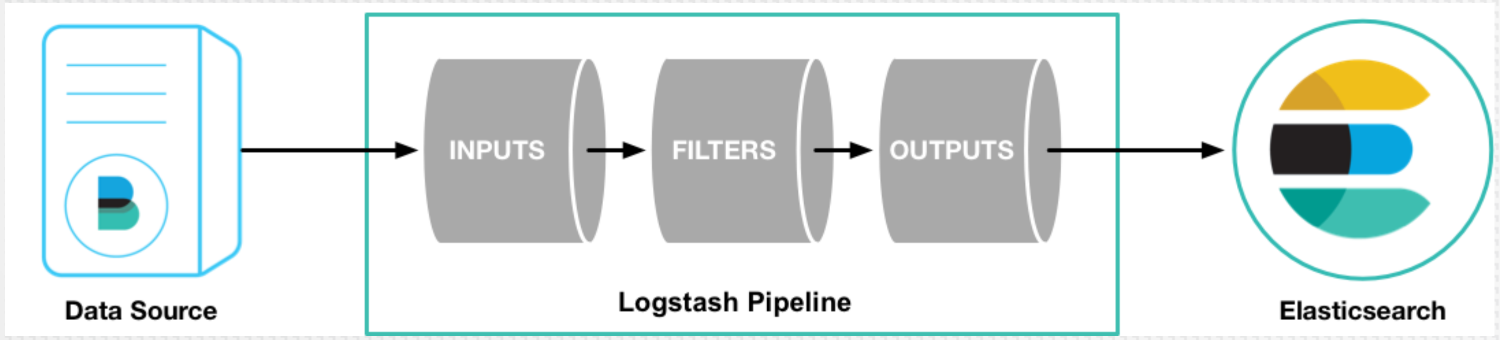
Logstash introduction

- It started as the ingestion tool to feed log data into on elasticsearch.
- Now is quite more than this, allowing to ingest lot of different events from different sources (file, streams, database records, iot, etc..) .
- Furthermore, it allows to collect, parse, and enrich the data before indexing it on Elasticsearch.



Logstash functionality

- Can ingest different events formats (Apache web logs, log4j, syslog, windows event logs, etc..)
- Apart of log gathering you can apply ETL logic (transformation, filtering, enrichment)
- You define a pipeline to orchestrate different input plugins apply ETL and generate output to index.



Configuring a logstash pipeline

- Pipelines are specified on conf files following logstash format.
- It has 3 stages that defines inputs, tranformations/filters and output
- Inputs generate events, filters modify them and outputs sends elsewhere.

```
# The # character at the beginning of a line indicates a comment.
Use
# comments to describe your configuration.
input {
}
# The filter part of this file is commented out to indicate that
it is
# optional.
# filter {
#
# }
output {
}
```

Plugins Configuration

- Configuration of a plugin consists of a plugin name following of a block of properties for it.
- There are Input Plugins, Output Plugins, Filter Plugins and Codec Plugins
- Every plugin property has a type depending plugin requirement (Array, List, Boolean, bytes, String, hash, URI, path, etc..)

```
input {
  file {
    path => "/var/log/messages"
    type => "syslog"
  }

  file {
    path => "/var/log/apache/access.log"
    type => "apache"
  }
}
```

Events management

- All events have properties that allows plugins to access to the data.
- The syntax to access a field is [fieldname]. To refer to a nested field, you specify the full path to that field: [top-level field][nested field].
- You can specify conditionals like in other programming languages (if, else if, else)

```
{
  "agent": "Mozilla/5.0 (compatible; MSIE 9.0)",
  "ip": "192.168.24.44",
  "request": "/index.html"
  "response": {
    "status": 200,
    "bytes": 52353
  },
  "ua": {
    "os": "Windows 7"
  }
}
```

```
filter {
  if [action] == "login" {
    mutate { remove_field => "secret" }
  }
}
```

Events management

- There is a special field called @metadata that will not be part of the events on output stage.
- Purpose of this field is to store internal data that you need to use between plugins.
- For debugging purposes you can see them on the output:

  stdout { codec => rubydebug { metadata => true } }

```
input { stdin { } }

filter {
  mutate { add_field => { "show" => "This data will be in the output" } }
  mutate { add_field => { "[@metadata][test]" => "Hello" } }
  mutate { add_field => { "[@metadata][no_show]" => "This data will not be in the output"
} }
}

output {
  if [@metadata][test] == "Hello" {
    stdout { codec => rubydebug }
  }
}
```

EDEM
Escuela de Empresarios

GFT

Events management

- You can set environment variables references in the configuration by ${var}

```
input {
  tcp {
    port => "${TCP_PORT}"
  }
}
```

- If you need to manage multiline events (java stacktrace, multiple lines aggregations, etc.) you can use multiline codec using regex patterns

```
input {
  stdin {
    codec => multiline {
      pattern => "^\s"
      what => "previous"
    }
  }
}
```

Launching pipelines

- First step before launching a pipeline is to verify that the configuration is correct:

```
bin/logstash -f first-pipeline.conf --config.test_and_exit
```

- If the configuration is correct to start a pipeline next command is used:
  - The --config.reload.automatic option enables automatic config reloading so that you don’t have to stop and restart Logstash every time you modify the configuration file.

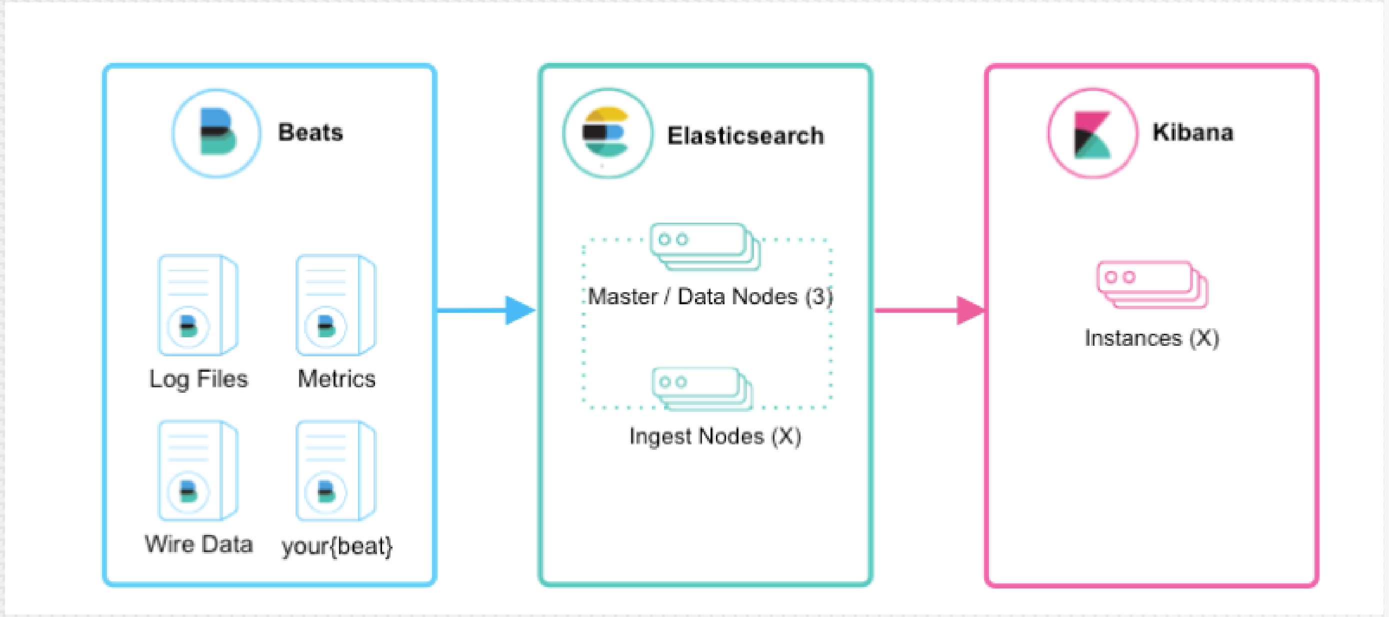
```
bin/logstash -f first-pipeline.conf --config.reload.automatic
```

- If you need to launch multiple pipelines on same process you can edit a configuration file called pipelines.yml placed on path.settings folder:

```
- pipeline.id: my-pipeline_1
  path.config: "/etc/path/to/p1.config"
  pipeline.workers: 3
- pipeline.id: my-other-pipeline
  path.config: "/etc/different/path/p2.cfg"
  queue.type: persisted
```

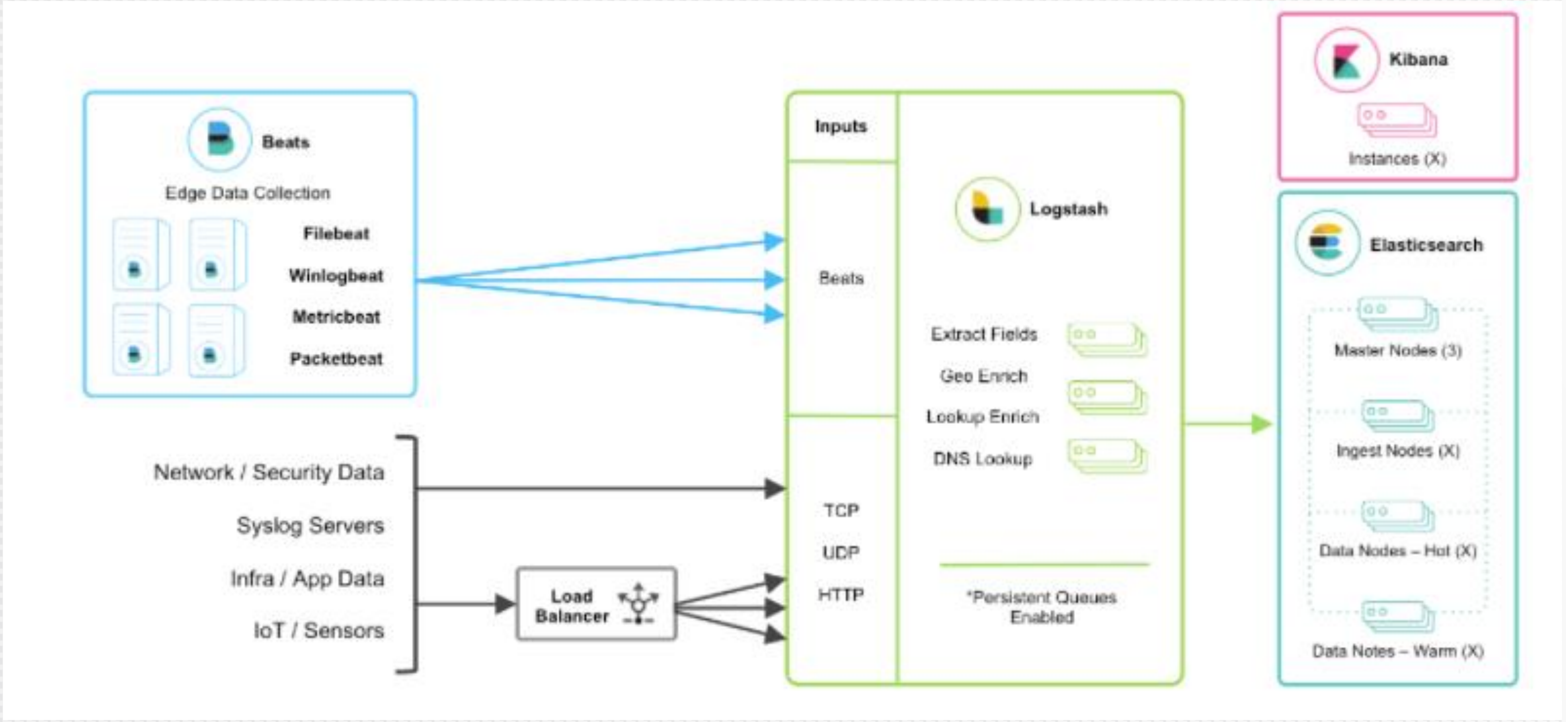
Ingestion architectures

- If you simply want to gather logs/mectics you can avoid using Logstash.
- In this architecture, beats modules ship data directly to Elasticseach.
- Filebeat plugins used for logs gathering.
- For metrics Metricbeat modules can be used.



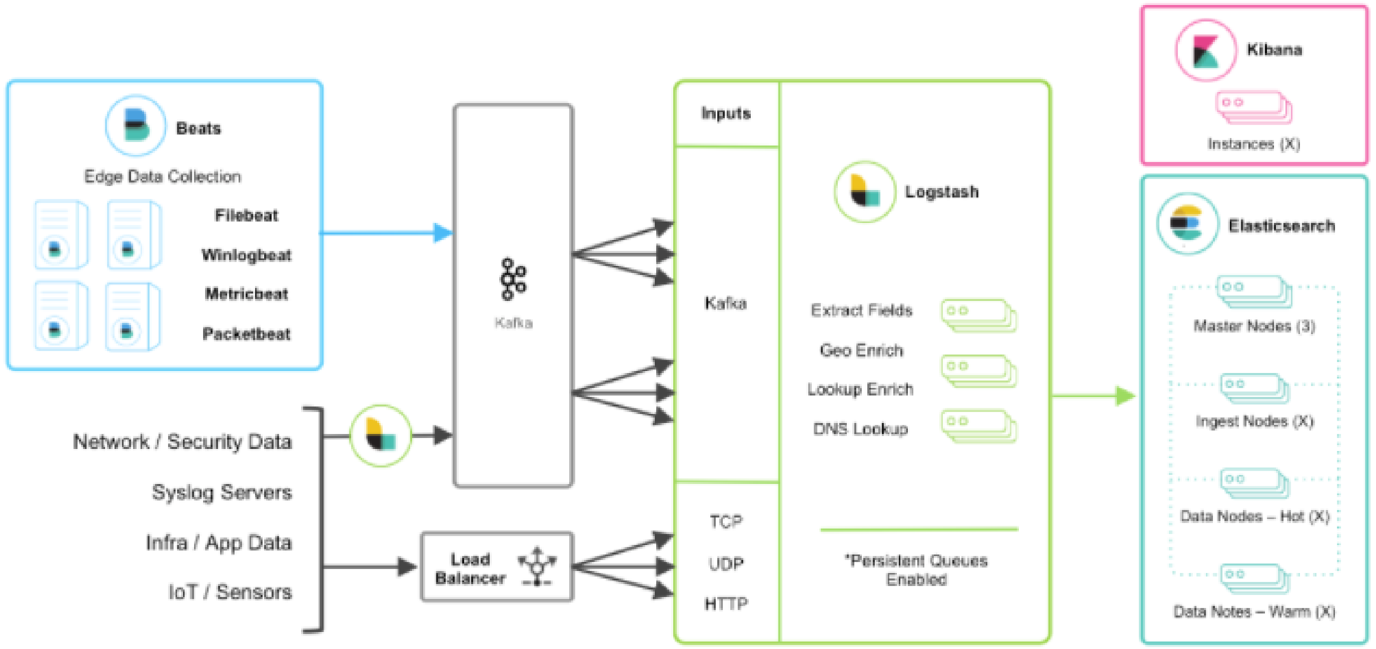
Ingestion architectures

- Logstash enhance basic ingestion process on following aspects:
  - Increase scalability though ingestion buffering.
  - Ingest from other data sources like databases, S3 or messaging queues (kafka)
  - Emit data to multiple destinations, not just elastic: S3, HDFS, files, etc.
  - Compose ETL ingestion pipelines.



Logstash architecture features

- Horizontally scalable, having multiple nodes running same pipeline.
- High availability having two Logstash nodes.
- Designed to be resilient on following ways:
  - When used with Filebeat, the ingest flows provides at-least-one delivery functionality.
  - Includes persistent queues to communicate different pipeline stages to support node failures.
- Provides TCP, UDP and HTTP to feed data into Elastic.
- Can integrate with present kafka platform to use it as a data hub.

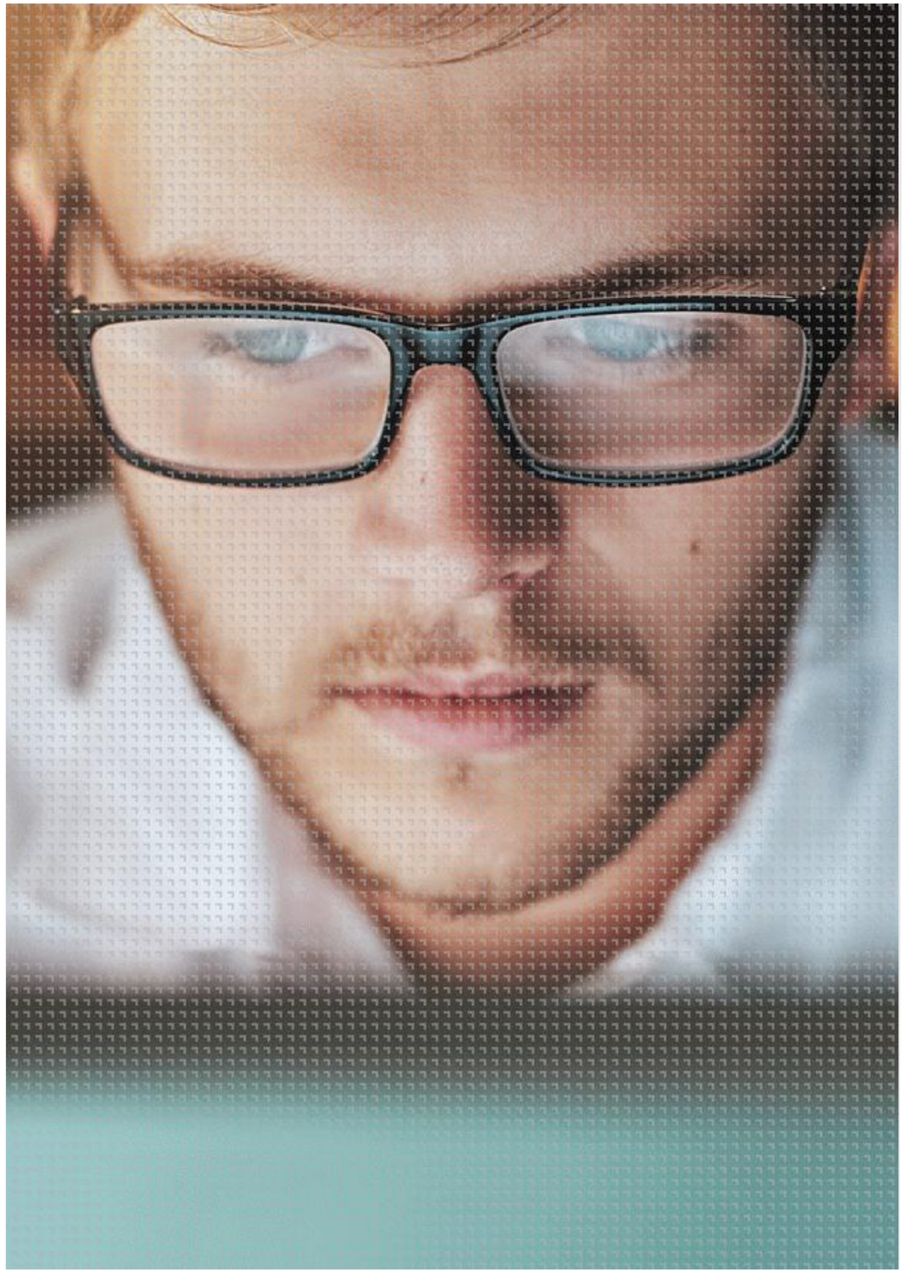


Monitoring Logstash

- There is an API rest that you can use to get status of:
  - Node Info (node status)
  - Plugins info (installed plugins)
  - Node stats
  - Hot Threads API

```
curl -XGET 'localhost:9600/_node/pipelines?pretty'
```

```
{
  "pipelines" : {
    "test" : {
      "workers" : 1,
      "batch_size" : 1,
      "batch_delay" : 5,
      "config_reload_automatic" : false,
      "config_reload_interval" : 3
    },
    "test2" : {
      "workers" : 8,
      "batch_size" : 125,
      "batch_delay" : 5,
      "config_reload_automatic" : false,
      "config_reload_interval" : 3
    }
  }
```



Exercise – Play with Elasticsearch

In this exercise you will learn Logstash basics implementing several pipelines and working with several plugins.

- Go to Exercises doc -> http://bit.ly/EDEMAPExercises
- Perform Exercise 2

Kibana

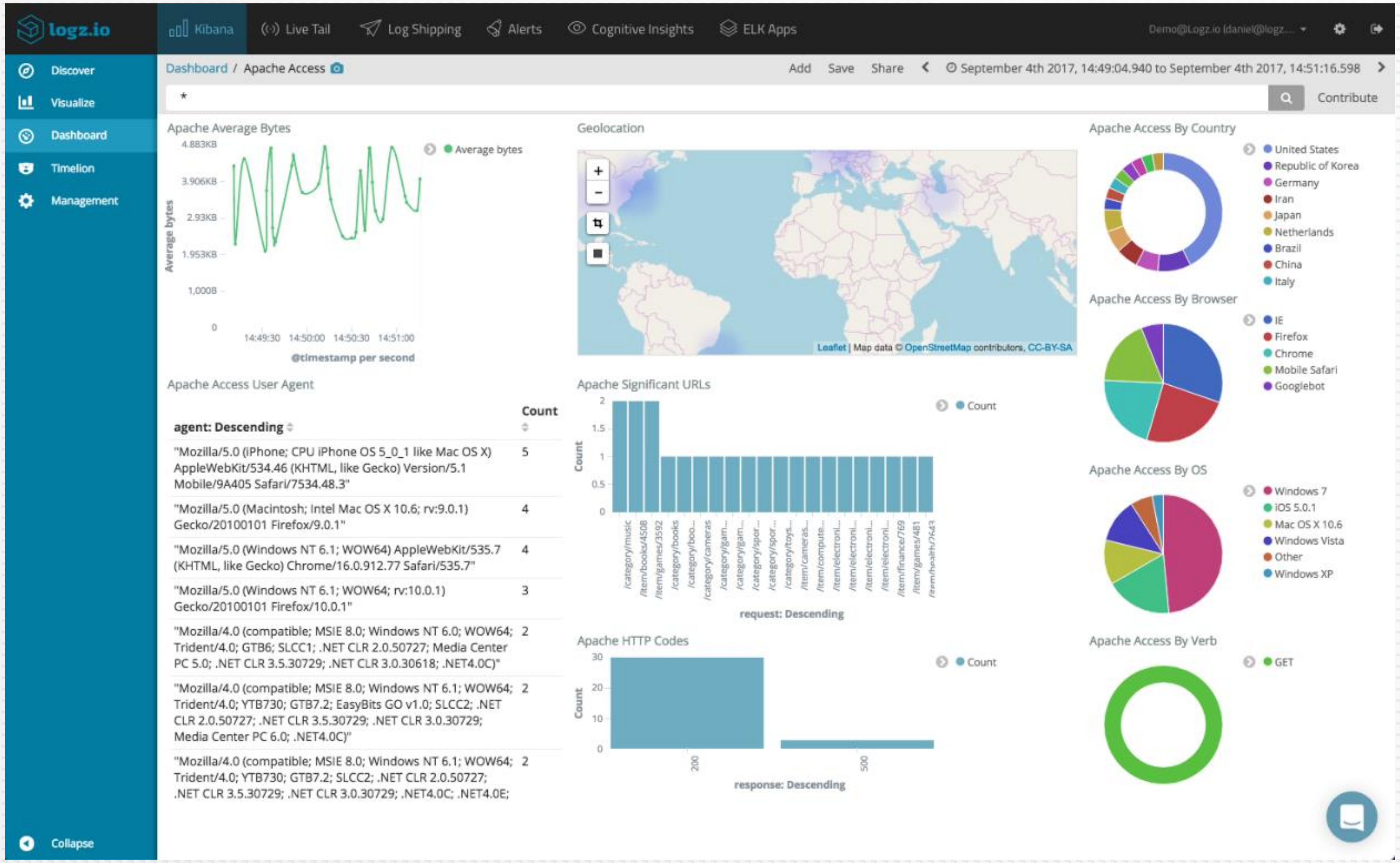
ELK Stack

EDEM Escuela de Empresarios

GFT

Kibana introduction

- Open source platform for data analytics an visualization over elasticsearch data.
- Allows the user to create dynamic dashboards that display any changes on the data on real time manner.
- Has lot of visualization options (time series, bar charts, pie charts, geo maps, etc…)



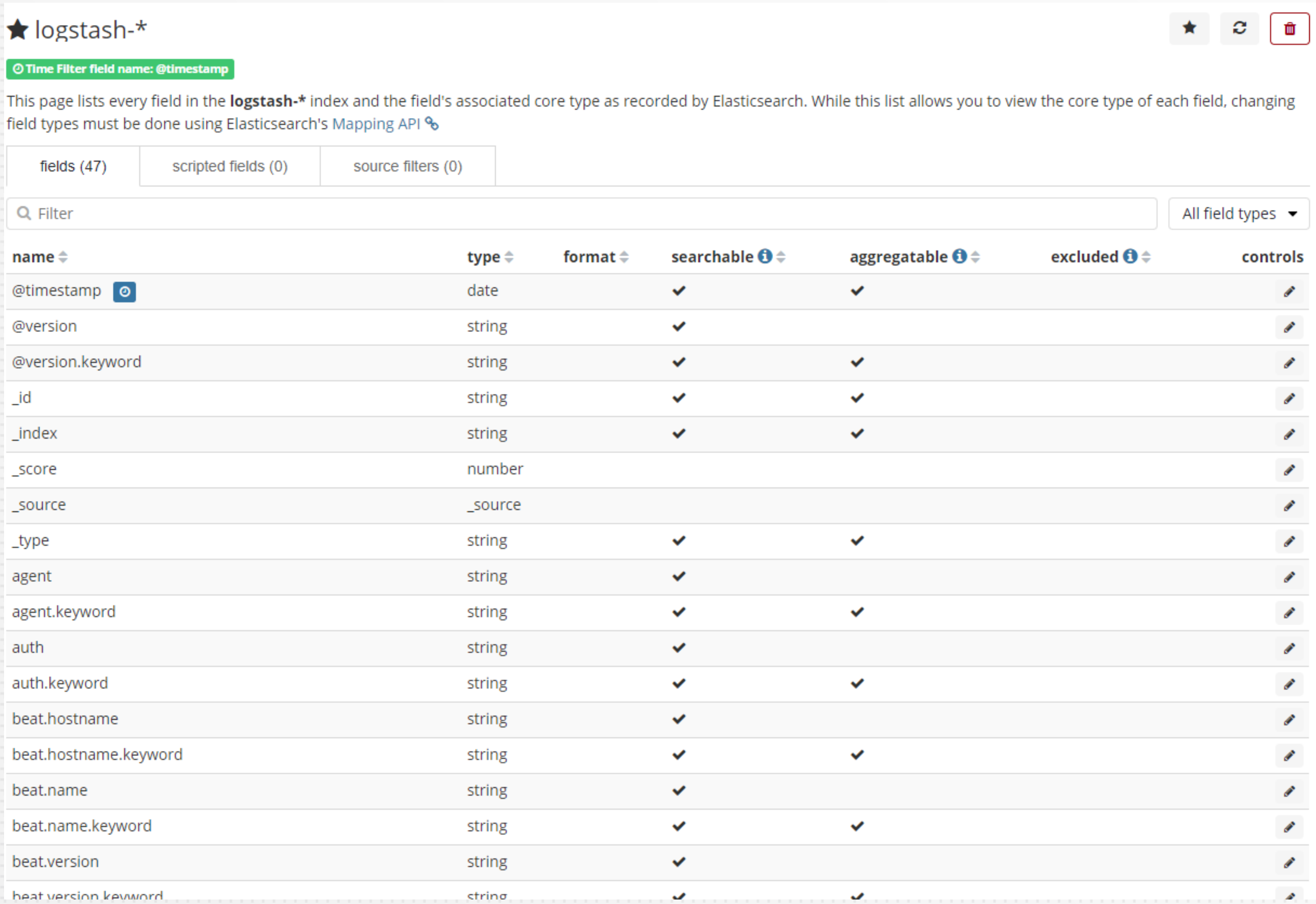
Apache Access User Agent

agent: Descending	Count
"Mozilla/5.0 (iPhone; CPU iPhone OS 5_0_1 like Mac OS X) AppleWebKit/534.46 (KHTML, like Gecko) Version/5.1 Mobile/9A405 Safari/7534.48.3"	5
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10.6; rv:9.0.1) Gecko/20100101 Firefox/9.0.1"	4
"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.7 (KHTML, like Gecko) Chrome/16.0.912.77 Safari/535.7"	4
"Mozilla/5.0 (Windows NT 6.1; WOW64; rv:10.0.1) Gecko/20100101 Firefox/10.0.1"	3
"Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.0; WOW64; Trident/4.0; GTB6; SLCC1; .NET CLR 2.0.50727; Media Center PC 5.0; .NET CLR 3.5.30729; .NET CLR 3.0.30618; .NET4.0C)"	2
"Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; YTB730; GTB7.2; EasyBits GO v1.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C)"	2
"Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; YTB730; GTB7.2; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; .NET4.0C; .NET4.0E;	2

Index management

- To connect to elastic data first thing you have to do is to create an index pattern:
  - You can use * as a wildcard in your index pattern (logstash-*)
  - If your index contains a timestamp, you can define it as a time-based events index for time comparisons.

- When you add an index, Kibana scans the data and provides a list of the documents fields and inferred types.



★ logstash-*

Time Filter field name: @timestamp

This page lists every field in the **logstash-*** index and the field's associated core type as recorded by Elasticsearch. While this list allows you to view the core type of each field, changing field types must be done using Elasticsearch's Mapping API

fields (47) | scripted fields (0) | source filters (0)

Filter | All field types

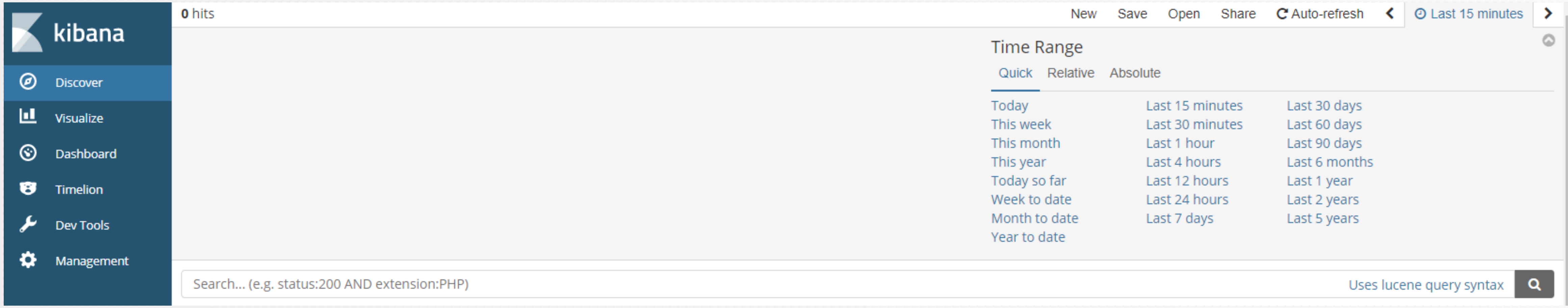
name	type	format	searchable	aggregatable	excluded	controls
@timestamp	date		✓	✓		✎
@version	string		✓			✎
@version.keyword	string		✓	✓		✎
_id	string		✓	✓		✎
_index	string		✓	✓		✎
_score	number					✎
_source	_source					✎
_type	string		✓	✓		✎
agent	string		✓			✎
agent.keyword	string		✓	✓		✎
auth	string		✓			✎
auth.keyword	string		✓	✓		✎
beat.hostname	string		✓			✎
beat.hostname.keyword	string		✓	✓		✎
beat.name	string		✓			✎
beat.name.keyword	string		✓	✓		✎
beat.version	string		✓			✎
beat.version.keyword	string		✓	✓		✎

Fields display formatting

String	Applies to string, date, geographic, numeric fields types	Convert to lowercase, uppercase, title case, short dots transformation
URL	Applies to string, date, numeric fields types	By means of templates a URL/URI is generated using field value.
Date	Applies to string, date fields types	Formats using a date pattern (Ex. YYYY-MM-DD)
Bytes/Number /Percentage	Applies to numeric fields type	Applies a Numeral.js format pattern (Ex. 0,0.[000]b)
Duration	Applies to numeric fields type	Converts to human readable specifying units of the field (seconds, hours, etc..)
Color	Applies to numeric, string fields types	Defining a range it associates a got color and Background color.

Data inspection

- Provides a query tool to perform your data insights over elastic indexes
- You can define a time range filter if your index contains time-based events.
- You can filter also the fields to be displayed based on documents schema (Json)
- Based on Lucene Query syntax.



Today	Last 15 minutes	Last 30 days
This week	Last 30 minutes	Last 60 days
This month	Last 1 hour	Last 90 days
This year	Last 4 hours	Last 6 months
Today so far	Last 12 hours	Last 1 year
Week to date	Last 24 hours	Last 2 years
Month to date	Last 7 days	Last 5 years
Year to date		

Search... (e.g. status:200 AND extension:PHP)

Uses lucene query syntax

Lucene Query Syntax

- Just putting a text string it perform as free text search over whole documents fields. Wildcards ?/* allowed.
- To query over a explicit value just put name of the field and the value following way: `response:200`
- To query over a range, syntax is with square brackets for inclusive and with curly brackets for exclusive : `response: [200 TO 500]` `response: {200 TO 500}`
- Ranges with one side unbounded: `response:>200`
- Where a field has not null value: `_exists_:verb`
- You can use regular expressions: `Ip:\b\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}\b`

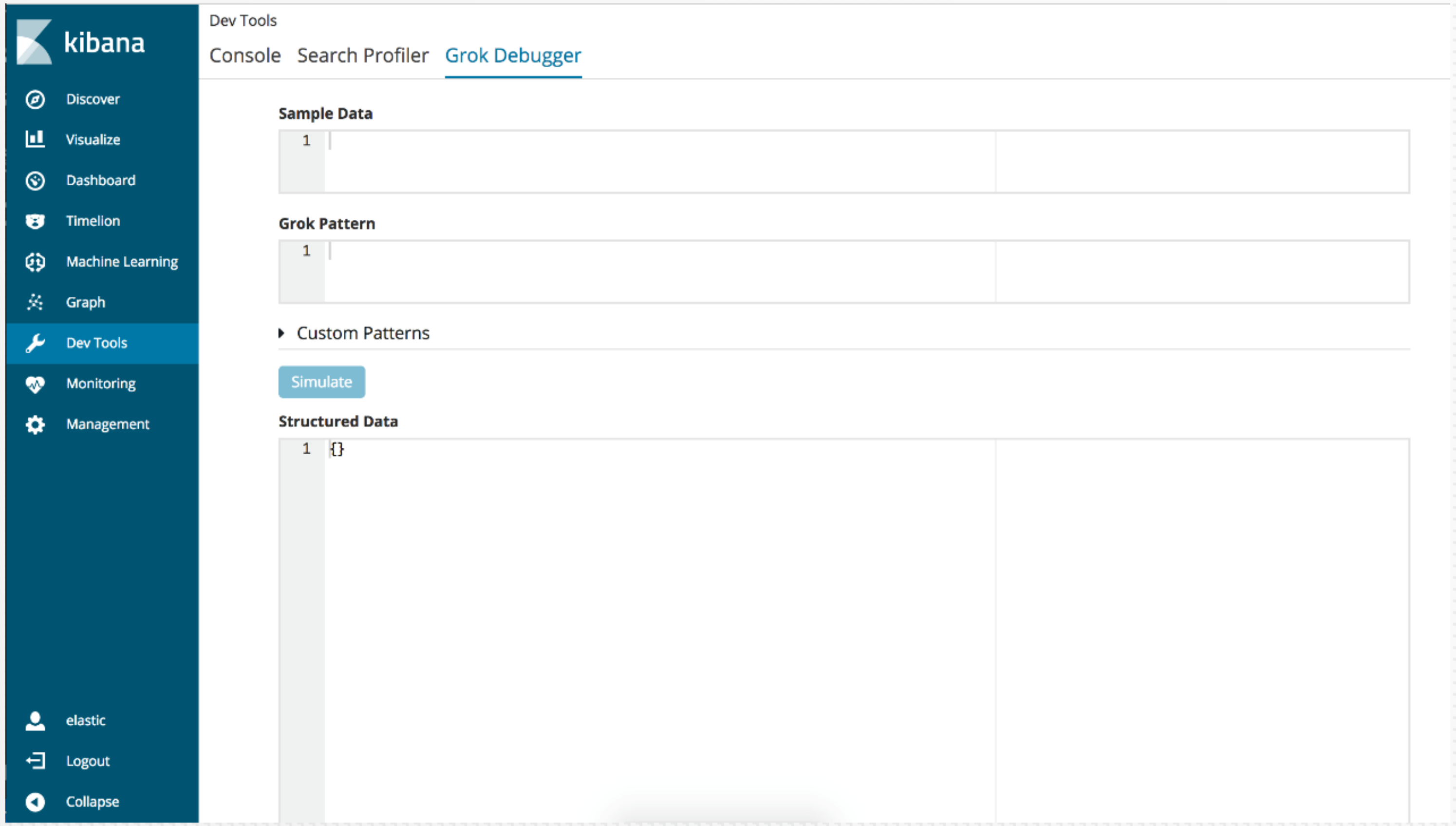
Lucene Query Syntax

- Search of terms similarities using fuzziness concept (Levenshtein distance): `quikc~`
- You can make more relevant one term than other using boosting: `Apache^2 Exception`
- For several fields query you can combine them using standard AND, OR and NOT operators

```
response: [200 TO 500] AND (verb:GET OR verb:POST)
```

Dev Tools

- Provide a console UI to interact with RESP API of Elasticsearch.
- You have a query profiler tool where you can see how the data will be accessed (shards, indexes, etc) (X-pack)
- Transformation debugger for Grok processor (X-pack).
  - Using a sample data and a Grok pattern it generates the result on the fly.

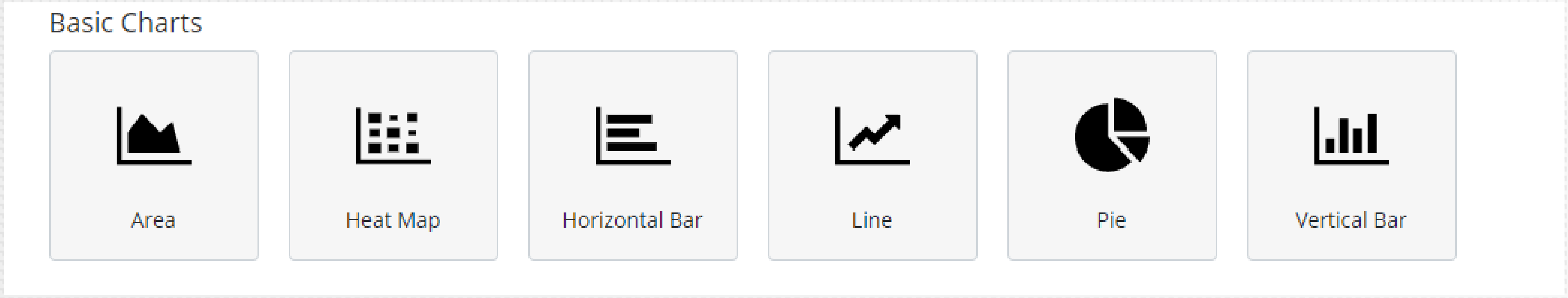


EDEM
Escuela de Empresarios

GFT

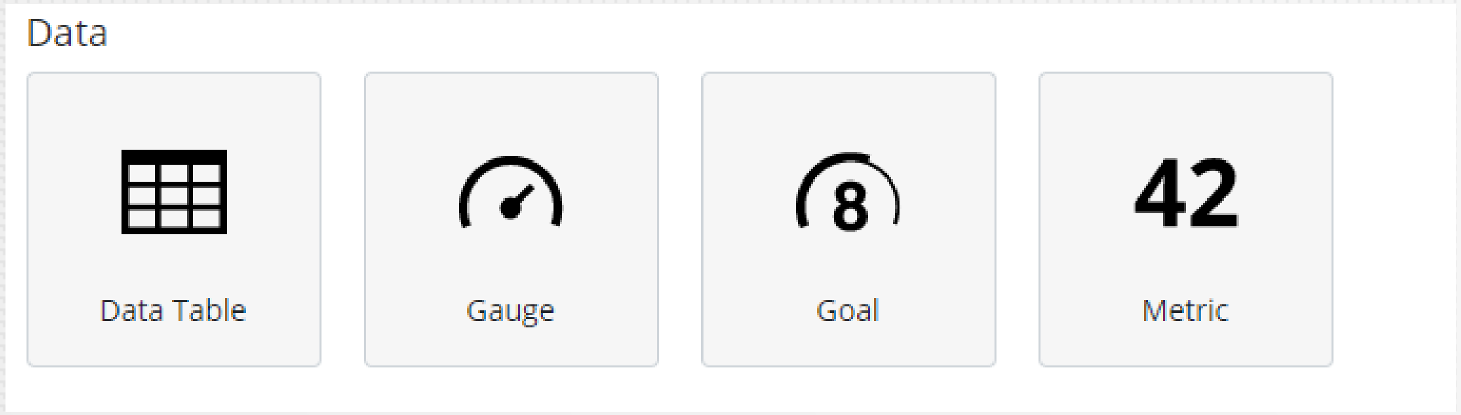
Visualization Basic Charts

Line, Area and Bar charts	Compare different series on X/Y charts
Heat maps	Matrix representation where individual cell values are represented as colors
Pie charts	Pie representation over data distribution



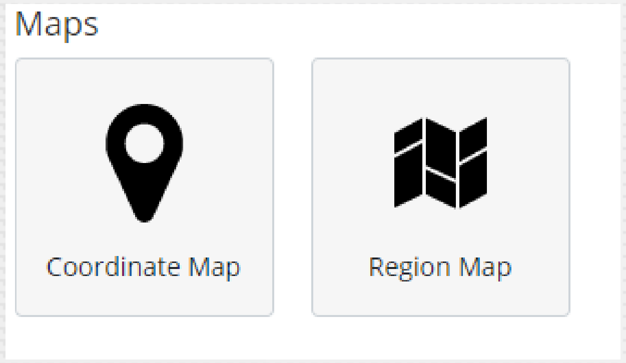
Visualization of Data

Data table	Display raw data of a composed aggregation
Metric	Display a metric value
Goal and Gauge	Display a Gauge or Goal



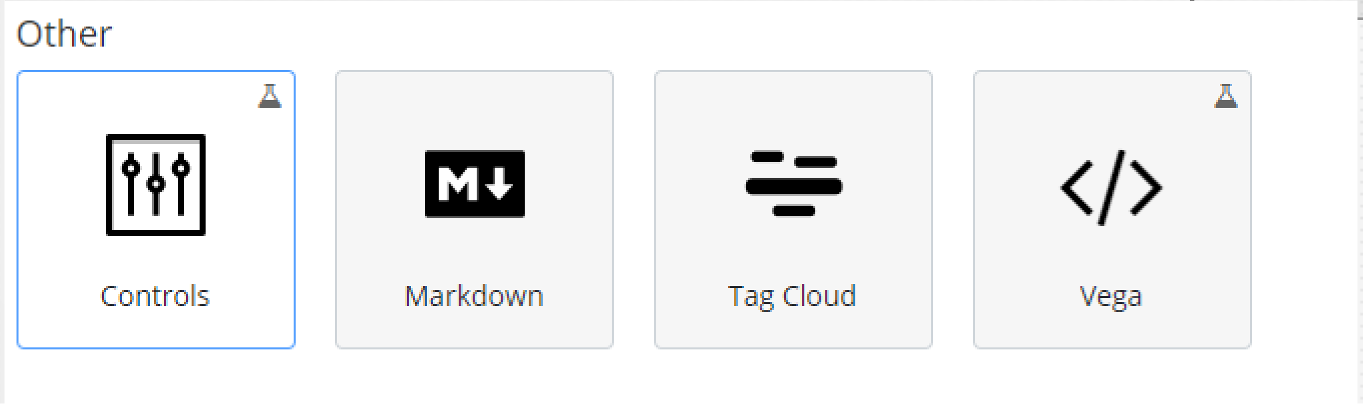
Visualization of Maps

Coordinate Map	Display results of a data aggregation with geographic locations
Metric	Region maps where each region color intensity corresponds to a metric's value



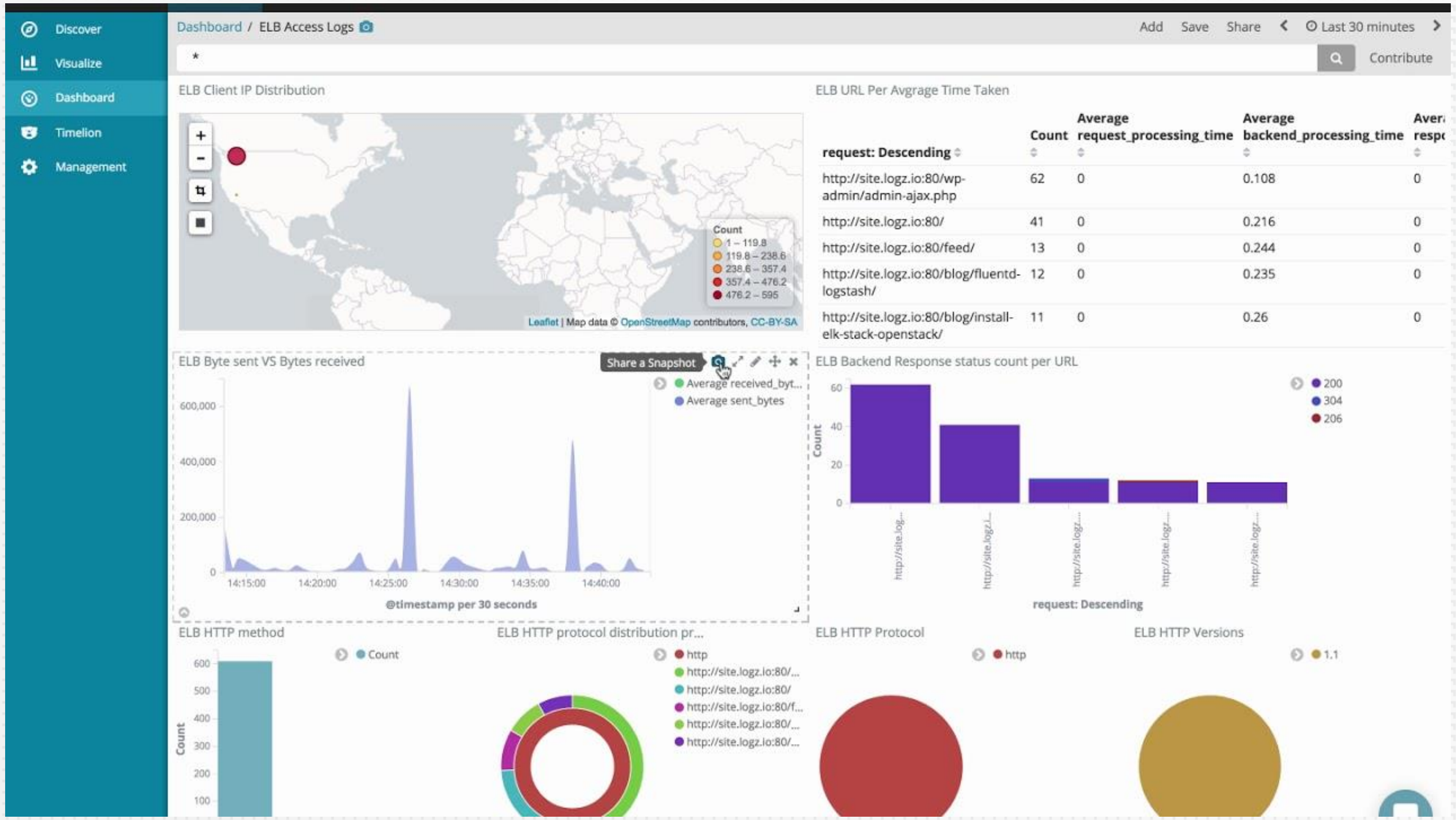
Advanced Visualization

Tag Cloud	Display words as a cloud in which the size of the word correspond to its importance
Markdown widget	Display free-form information or instructions.
Vega graph (Exp)	Support for user-defined graphs based on a declarative language, external data sources, images, and user-defined interactivity.
Controls (Exp)	Interactive controls for dashboard manipulation



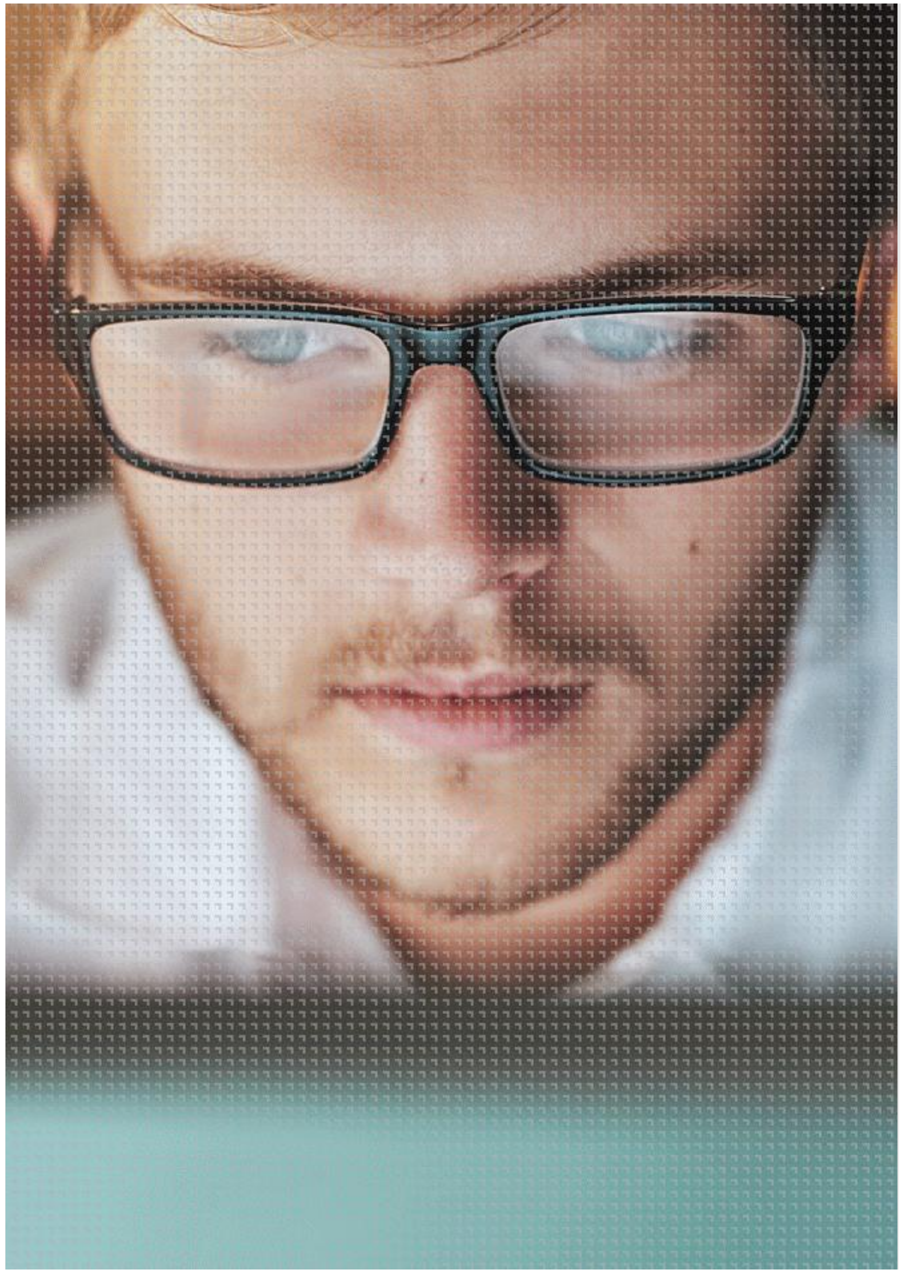
Dashboards

- You can build dashboards to integrate all visualization panes together
- Filters are applied to all visualization graphs on same time
- Also you can select data on one chart and that also affects the rest.



ELB URL Per Avgrage Time Taken

request: Descending	Count	Average request_processing_time	Average backend_processing_time	Aver respo
http://site.logz.io:80/wp-admin/admin-ajax.php	62	0	0.108	0
http://site.logz.io:80/	41	0	0.216	0
http://site.logz.io:80/feed/	13	0	0.244	0
http://site.logz.io:80/blog/fluentd-logstash/	12	0	0.235	0
http://site.logz.io:80/blog/install-elk-stack-openstack/	11	0	0.26	0



Exercise – Play with Kibana

In this exercise you will learn basic usage of Kibana

- Go to Exercises doc -> http://bit.ly/EDEMAPExercises
- Perform Exercise 3