Powering Monitoring Analytics with ELK stack

Abdelkader Lahmadi, Frédéric Beck

▶ **To cite this version:**

Abdelkader Lahmadi, Frédéric Beck. Powering Monitoring Analytics with ELK stack. 9th International Conference on Autonomous Infrastructure, Management and Security (AIMS 2015), Jun 2015, Ghent, Belgium. 9th International Conference on Autonomous Infrastructure, Management and Security (AIMS 2015), 2015. hal-01212015

HAL Id: hal-01212015
https://inria.hal.science/hal-01212015

Submitted on 5 Oct 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Powering Monitoring Analytics with ELK stack

Abdelkader Lahmadi, Frédéric Beck

INRIA Nancy Grand Est,
University of Lorraine, France

2015

(compiled on: June 23, 2015)

References

online Tutorials

- Elasticsearch Reference:
  https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html
- Elasticsearch, The Definitive Guide:
  https://www.elastic.co/guide/en/elasticsearch/guide/current/index.html
- Logstash Reference:
  https://www.elastic.co/guide/en/logstash/current/index.html
- Kibana User Guide:
  https://www.elastic.co/guide/en/kibana/current/index.html

Books

- Rafal Kuc, Marek Rogozinski: Mastering Elasticsearch, Second Edition, February 27, 2015

Monitoring data

Machine-generated data

- Nagios logs
- Snort and suricata logs
- Web server and ssh logs
- Honeypots logs
- Network event logs: NetFlow, IPFIX, pcap traces

Snort log

```
[**] [1:2000537:6] ET SCAN NMAP -sS [**] [Classification: Attempted Information Leak]
[Priority: 2] 11/08-11:25:41.773271 10.2.199.239:61562 -> 180.242.137.181:5222 TCP TTL:38
TOS:0x0 ID:9963 IpLen:20 DgmLen:44 ******S* Seq: 0xF7D028A7 Ack: 0x0 Win: 0x800 TcpLen: 24
TCP Options (1) => MSS: 1160
```

Web server log

```
128.1.0.0 - - [30/Apr/1998:22:38:48] "GET /english/venues/Paris/St-Denis/ HTTP/1.1" 404 333
```

Monitoring data: properties

- Variety of data sources: flows, logs, pcap data
- Structure: structured, non structured
- Massive volume: grow at Moore's Law kinds of speeds

Who is looking into them ?

- IT administrators
- Researchers and scientists

Why ?

- Continous monitoring tasks
- Extract useful information and insights: finding meaningful events leading to new discoveries
- Looking for a needle in a haystack: forensics and Incident Handling
- Data search, aggregation, correlation
- Predictive modelling, statistical analysis, anomaly detection

Traditional tools and techniques

Storage

- Text files, binary files
- Relational databases: MySQL, PostgreSQL, MS Access
- New trends: NoSQL databases (Redis, MongoDB, HBase, ...)

```
May 31st 2015, 15:44:06.039     com.whatsapp    10.29.12.166    50048   184.173.179.38  443
May 31st 2015, 15:55:40.268     com.facebook.orca   10.29.12.166    47236   31.13.64.3    443
```

Tools and commands

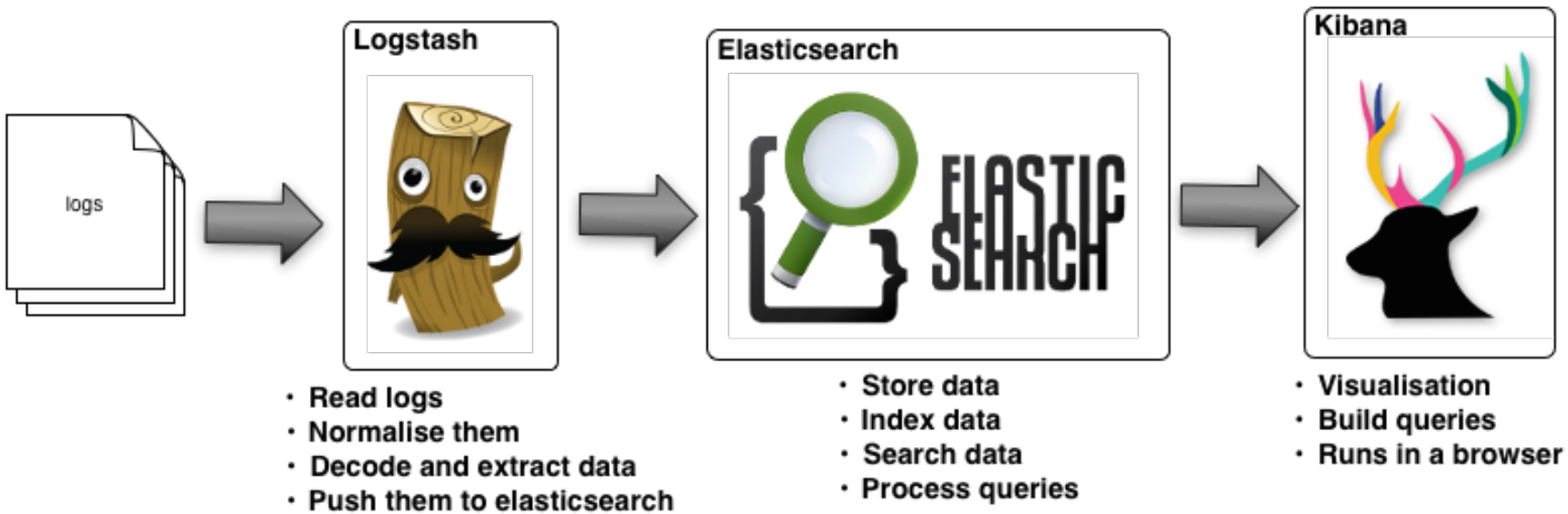
- SQL, Grep, sed, awk, cut, find, gnuplot
- Perl, Python, Shell

```
perl -p -e 's/\t/ /g' raw | cut -d ' ' -f5-
```

ELK Components

Elasticsearch Logstash Kibana

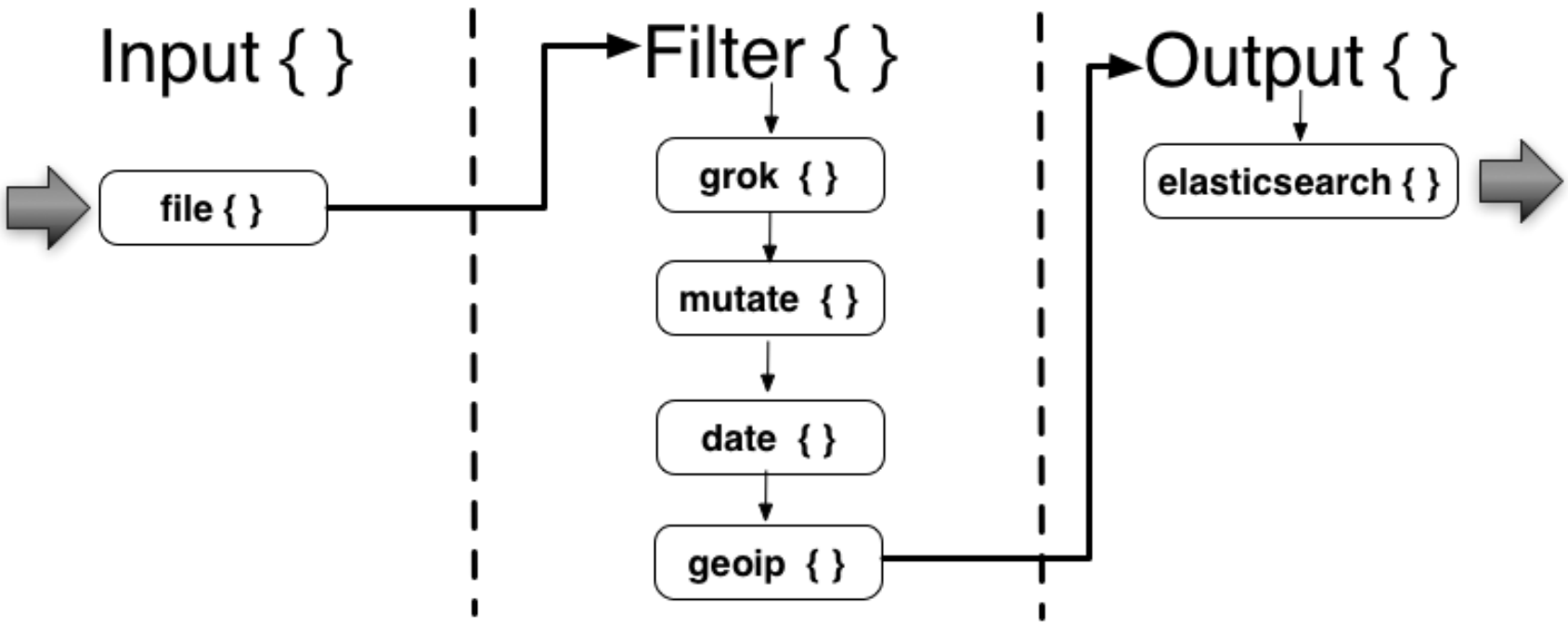
- end to end solution for logging, analytics, search and visualisation



Logstash: architecture

Event processing engine: data pipeline

- Inputs: collect data from variety of sources
- Filters: parse, process and enrich data
- Outputs: push data to a variety of destinations



Logstash: Inputs and codecs

Receive data from files, network, etc

- Network (TCP/UDP), File, syslog, stdin
- Redis, RabbitMQG, irc, Twitter, IMPA, xmpp
- syslog,ganglia, snmptrap, jmx, log4j

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

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

Logstash: Filters

Enrich and process the event data

- grok, date, mutate, elapsed, ruby
- dns, geoip,useragent

```
filter {
  grok {
    match => { "message" => "%{COMBINEDAPACHELOG}" }
  }
  date {
    match => [ "timestamp" , "dd/MMM/yyyy:HH:mm:ss Z" ]
  }
}
```

Logstash: Outputs

Send event data to other systems

- file, stdout, tcp
- elasticsearch, MongoDB, email
- syslog, Ganglia

```
output {
  stdout { codec => rubydebug }
  elasticsearch { host => localhost }
}
```

Elasticsearch: better than Grep!

Search engine

- Built on top of Apache Lucene: full-text-search engine library
- Schema-free
- JSON document oriented: store and REST API
- index, search, sort and filter documents

Distributed search and store

- scaling up (bigger servers) or scaling out (more servers)
- coping with failure: replication

RESTfull API

- store, fetch, alter, and delete data
- Easy backup with snapshot and restore
- No transactions nor rollback

Elasticsearch: glossary

- **Document**: main data unit used during indexing and searching, contains one or more fields
- **Field**: section of the document, a couple of name and value
- **Term**: unit of search representing a word from the text
- **Index**: named collection of documents (Like a database in a relational data base)
- **type**: a class of similar documents (like a table in in a relational data base)
- **Inverted index**: data structure that maps the terms in the index to the documents
- **Shard**: a separate Apache Lucene index. Low level "worker" unit.
- **Replica**: an exact copy of a shard

```
Relational DB => Databases => Tables => Rows      => Columns
Elasticsearch => Indices   => Types  => Documents => Fields
```

Elasticsearch: indexes and documents

Create Index

```
curl -XPUT 'localhost:9200/logs'  -d {
     "settings": {
        "number_of_shards" :   1,
        "number_of_replicas" : 0
    }
}
```

Indexing a document

```
curl -XPUT 'localhost:9200/logs/apachelogs/1'  -d {
          "clientip" => "8.0.0.0",
          "verb" => "GET",
          "request" => "/images/comp_stage2_brc_topr.gif",
          "httpversion" => "1.0",
          "response" => "200",
          "bytes" => "163"
}
```

Retrieving a document

```
curl -XGET 'localhost:9200/logs/apachelogs/1'
```

Elasticsearch: mapping

Define the schema of your documents

- elasticsearch is able to dynamically generate a mapping of fields type
- It is better to provide your own mapping: date fields as dates, numeric fields as numbers string fields as full-text or exact- value strings

Create a mapping

```
curl -XPUT 'localhost:9200/logs' {
    "mappings": {
     "apachelogs" : {
      "properties" : {
        "clientip" : {
          "type" :    "ip"
        },
        "verb" : {
          "type" :   "string"
        },
        "request" : {
          "type" :   "string"
        },
        "httpversion" : {
          "type" :   "string"
        }
        "response" : {
          "type" :   "string"
        }
```

Elasticsearch: Quering

Search Lite

```
curl -XPUT 'localhost:9200/logs/apachelogs/_search'
curl -XPUT 'localhost:9200/logs/apachelogs/_search?q=response:200'
```

Search with Query DSL: match, filter, range

```
curl -XPUT 'localhost:9200/logs/apachelogs/_search' -d {
  "query" : {
    "match" : {
      "response" : "200"
    }
  }
}
curl -XPUT 'localhost:9200/logs/apachelogs/_search' -d {
  "query" : {
    "filtered" : {
        "filter" : {
         "range" : {
        "bytes" : { "gt"  : 1024}
      }
        },
        "query" : {
      "match" : {
        "response" : "200"
      }
        }
```

Elasticsearch: Aggregation

Analyze and summarize data

- How many needles are in the haystack?
- What is the average length of the needles?
- What is the median length of the needles, broken down by manufacturer?
- How many needles were added to the haystack each month?

Buckets

- Collection of documents that meet a criterion

Metrics

- Statistics calculated on the documents in a bucket

SQL equivalent

```
SELECT COUNT(response) FROM table GROUP BY response
```

- COUNT(response) is equivalent to a metric
- GROUP BY response is equivalent to a bucket

Aggregation: buckets

- partioning document based on criteria: by hour, by most-popular terms, by age ranges, by IP ranges

```
curl -XPUT 'localhost:9200/logs/apachelogs/_search?search_type=count
    ' -d {
"aggs" : {
        "responses" : {
            "terms" : {
              "field" : "response"
            }
        }
    }
}
```

- Aggregations are placed under the top-level aggs parameter (the longer aggregations will also work if you prefer that)
- We then name the aggregation whatever we want: responses, in this example
- Finally, we define a single bucket of type terms
- Setting search_type to count avoids executing the fetch phase of the search making the request more efficient

Aggregation: metrics

- metric calculated on those documents in each bucket: min, mean, max, quantiles

```
curl -XPUT 'localhost:9200/logs/apachelogs/_search?search_type=count
    ' -d {
"aggs": {
      "responses": {
         "terms": {
            "field": "response"
         },
         "aggs": {
            "avg_bytes": {
               "avg": {
                  "field": "bytes"
               }
            }
         }
      }
   }
}
```

Aggregation: combining them

An aggregation is a combination of buckets and metrics

- Partition document by client IP (bucket)
- Then partition each client bucket by verbe (bucket)
- Then partition each verbe bucket by response (bucket)
- Finally, calculate the average bytes for each response (metric)
- **Average bytes per <client IP, verbe, response>**

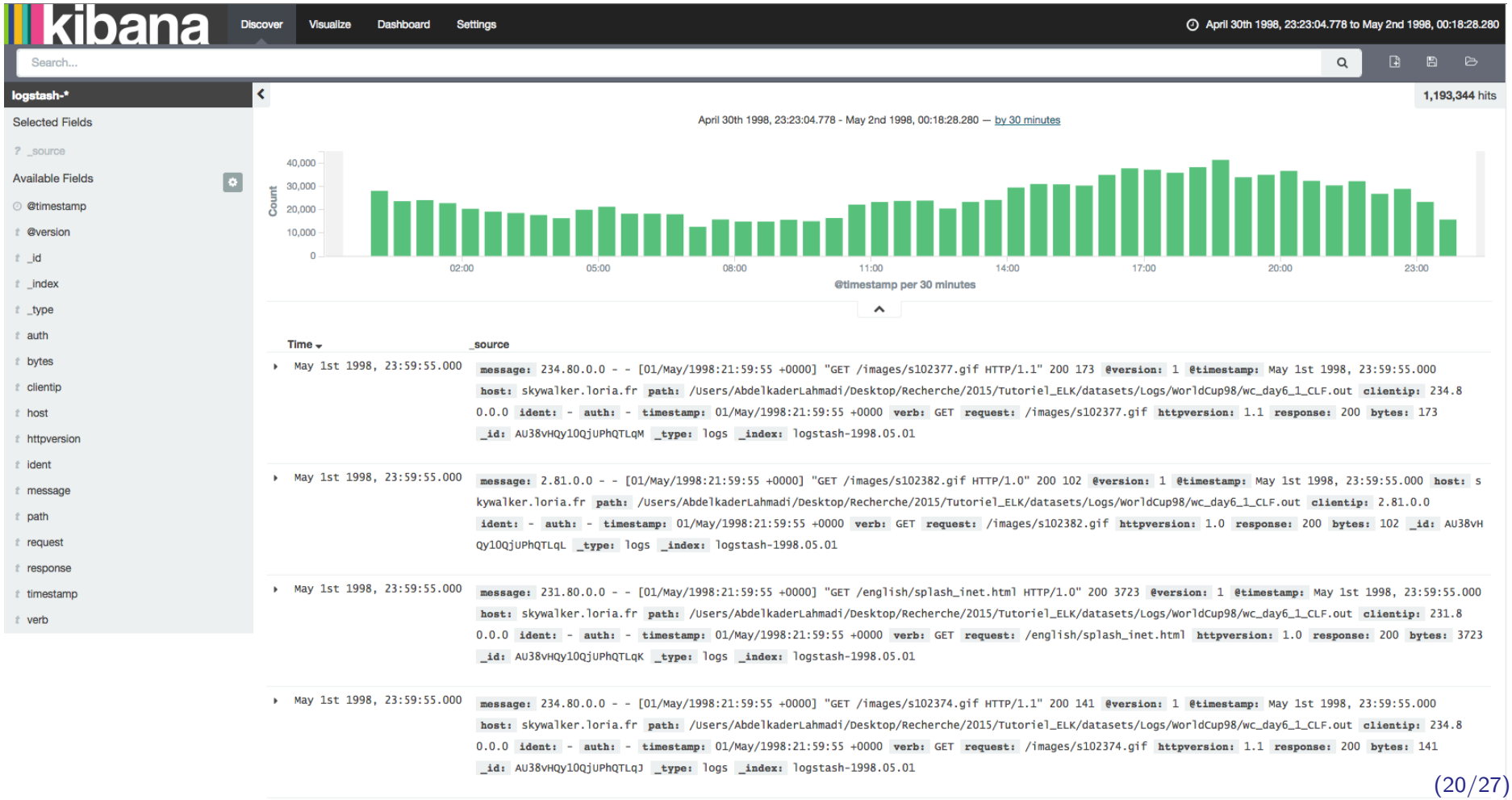
Many types are available

- terms
- range, date_range, ip_range
- histogram, date_histogram
- stats/avg, min, max, sum, percentiles

Kibana: visualisation

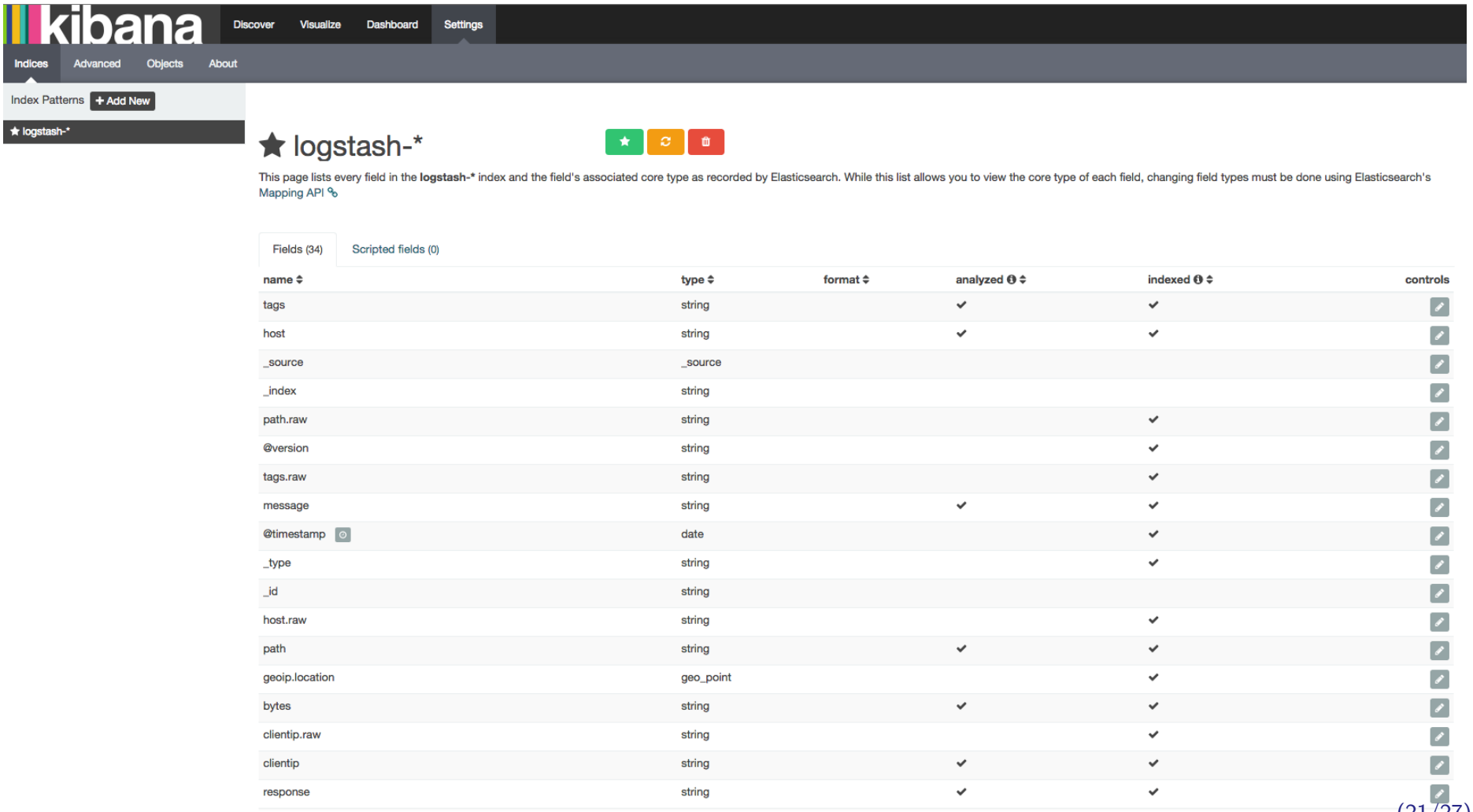
Data visualisation

- browser-based interface
- search, view, and interact with data stored in Elasticsearch indice
- charts, tables, and maps
- dashboards to display changes to Elasticsearch queries in real time



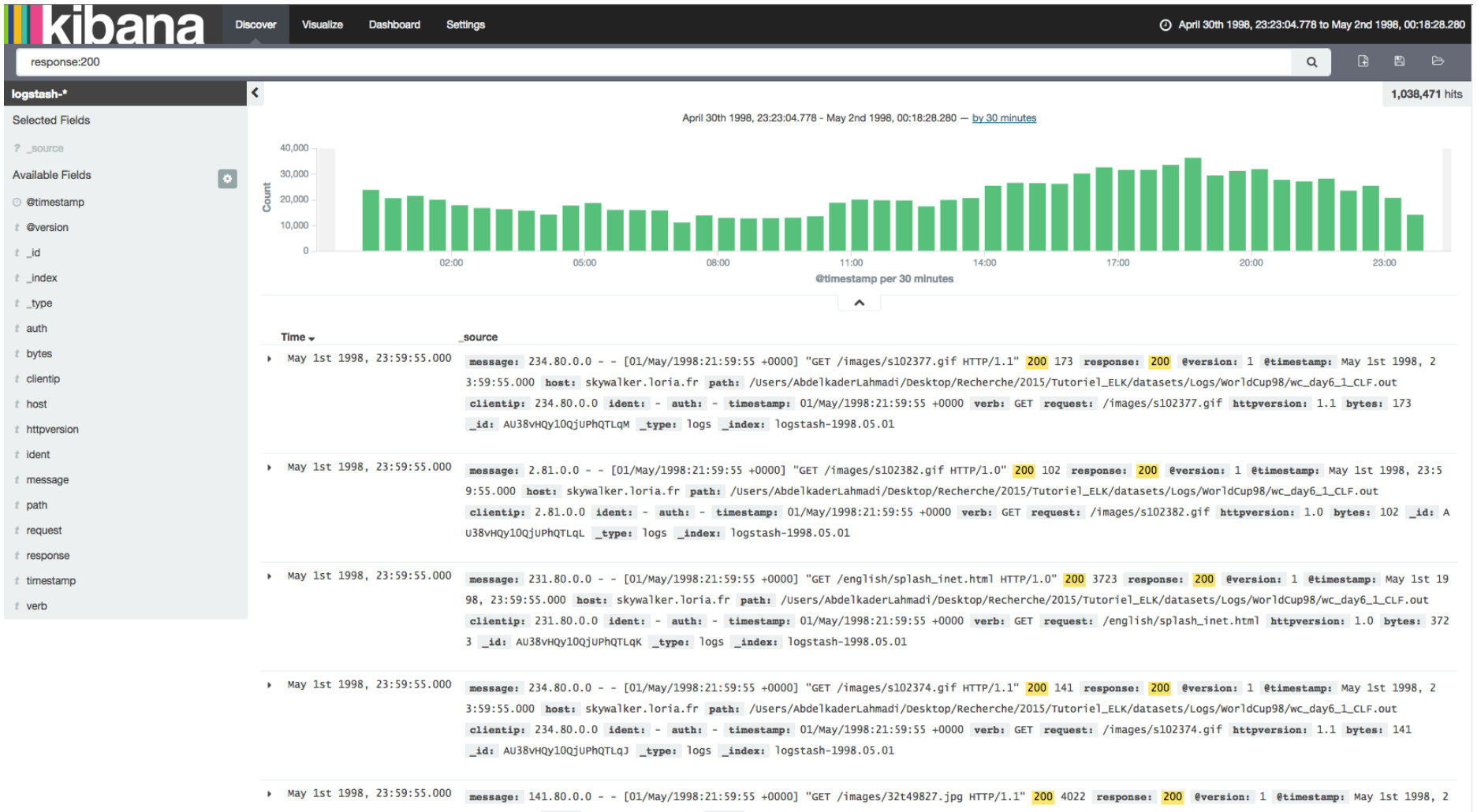
Kibana: setting your index

```
curl 'localhost:9200/_cat/indices?v'
index pri rep docs.count docs.deleted store.size pri.store.size
logstash-1998.04.30   5   1     98434     0     20.6mb         20.6mb
logstash-1998.05.01   5   1     1094910   0     205.8mb       205.8mb
```



Kibana: discover

- Explore you data with access to every document
- Submit search query, filter the search
- Save and Load searches



Kibana: search

Query language

lang:en	to just search inside a field named "lang"
lang:e?	wildcard expression
user.listed_count:[0 to 10]	range search on numeric fields
lang:ens	regular expression search (very slow)

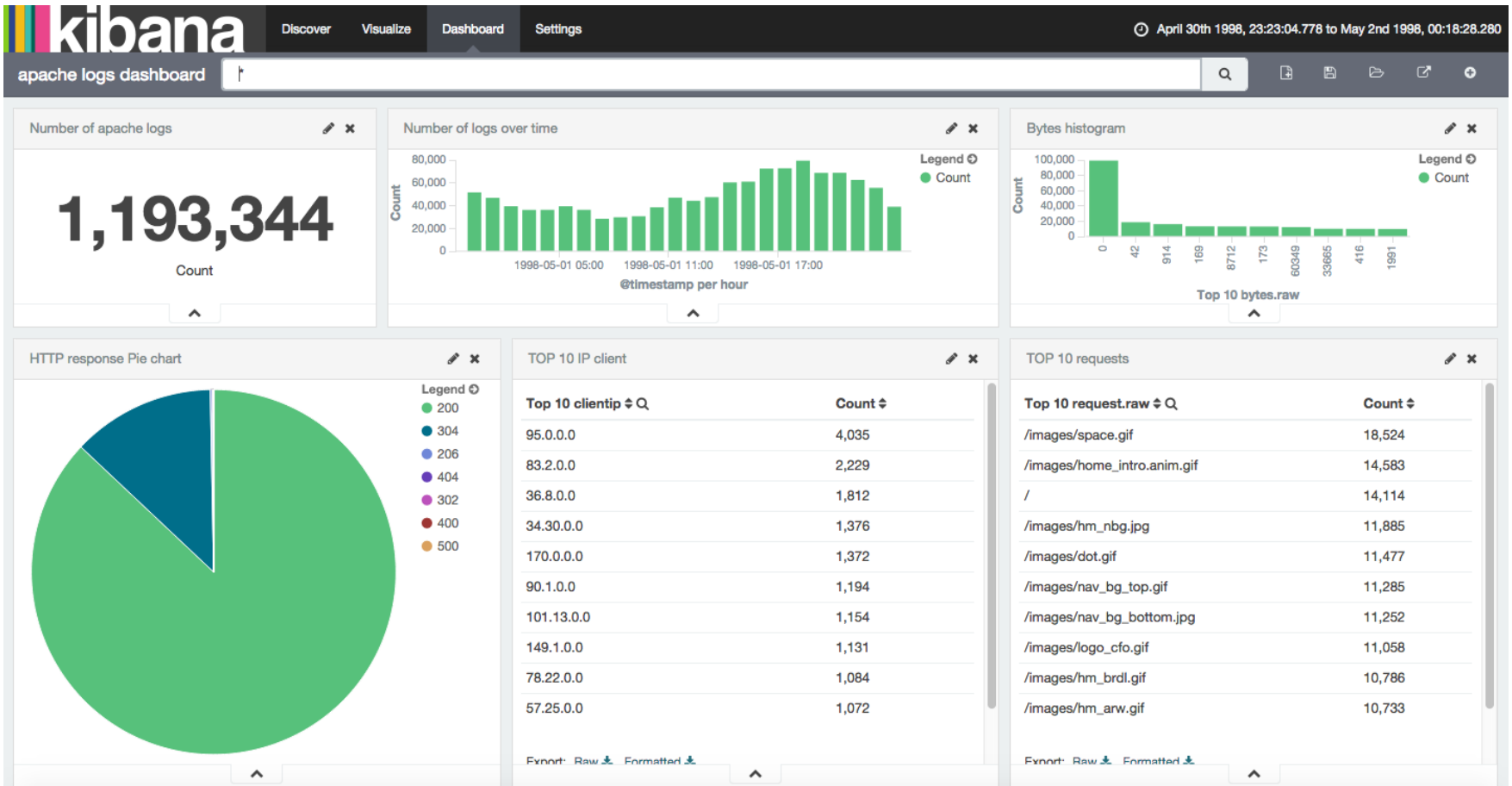
Kibana: visualize

- Area chart: Displays a line chart with filled areas below the lines. The areas can be displayed stacked, overlapped, or some other variations.
- Data table: Displays a table of aggregated data.
- Line chart: Displays aggregated data as lines.
- Markdown widget: Use the Markdown widget to display free-form information or instructions about your dashboard.
- Metric: Displays one the result of a metric aggregation without buckets as a single large number.
- Pie chart: Displays data as a pie with different slices for each bucket or as a donut (depending on your taste).
- Tile map: Displays a map for results of a geohash aggregation.
- Vertical bar chart: A chart with vertical bars for each bucket.

Kibana: dashboard

Displays a set of saved visualisations in groups

- Save, load and share dashboards



What else ?

Elasticsearch is very active and evolving project

- Watchout when using the wildcard !
- Deleting a mapping deletes the data
- Respect procedures to update mappings
- Backup is easy [1], do it!

[1] `https://github.com/taskrabbit/elasticsearch-dump`

Hands-on Lab