SIC

Serviços e Infraestruturas de Computação

ELK Stack

Elastic Search, Logstash and Kibana

Credits

Several slides and figures in this presentation are based on:

- Logstash’s public documentation
- And a presentation from Clemens Düpmeier (KIT/IAI) that is available at the following link: https://indico.scc.kit.edu/event/89/contributions/3960/attachments/1976/2748/ELK-Stack-Grid-KA-School.pptx

Network & Systems Management

Lots of data is continuously generated…

- Lots of users generating logs:
  - Each user generates hundreds of logs
    (email checks, sent emails, Wi-Fi/Eduroam logs, DHCP requests, etc.)
- Lots of systems generating logs:
  - Routers, firewalls, switches, access points, servers….
- Lots of services and applications generating logs:
  - Netflow, syslog, access logs, service logs, audit logs….
- Typically, nobody cares until something goes wrong, even though this data potentially provides early warning of problems, when they are still easily solvable

Why specific tools for log and event analysis?

Log and Event analysis for network, systems and service management

- Log and event collection from multiple sources
- Ingest and correctly parse different log formats
- Large amounts of data
- Various formats of data

- Scalable architecture (from small networks to very large services)
- Expert knowledge is required

How were things done before?

- Set up one or more log servers for receiving logs from servers and network devices
- Basic scripts performed some filtering of the logs
- CronJobs executed periodically to:
  - Compute stats and send out reports/alerts
  - Detect possible abnormal behavior and react accordingly
- Plain text reports or stats trends webpage

How much data is it?

A few typical daily values from a mid-sized university:

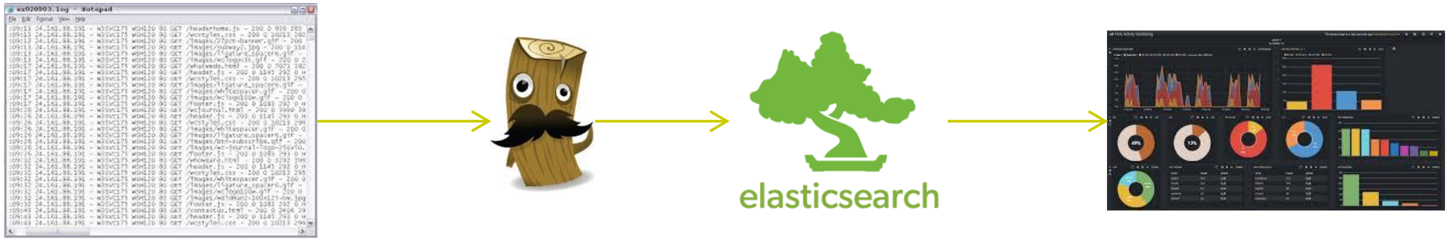
- Routers: 45GB daily just from Netflow
- Wi-Fi: NAT logs – 5TB; Auth logs
- Firewalls
- Mail Server logs:
  - POP3 – 7GB
  - SMTP – 2GB
  - MS Exchange – 150MB
  - Outlook Web App – 8GB
  - MessageTrackingLog – 100MB

…

ELK Stack

Three open-source software products:

- **E**lasticsearch
  Highly scalable search index server
- **L**ogstash
  Collection, enrichment, filtering, and forwarding
- **K**ibana
  Exploration and visualization of data



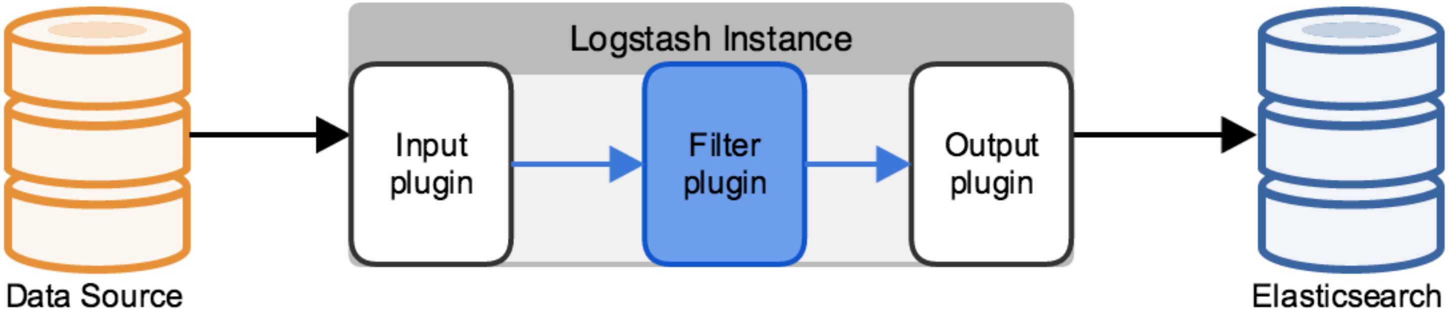
Logs
(from multiple sources)

Logstash

Kibana

Logstash

- Collect, transform, filter and forward data (e.g., log data) from input sources to output sources (e.g., Elasticsearch)
- Implemented in JRuby
- Runs on a JVM (Java Virtual Machine)
- Simple message-based architecture
- Extendable by plugins (*input*, *output*, and *filter* plugins)

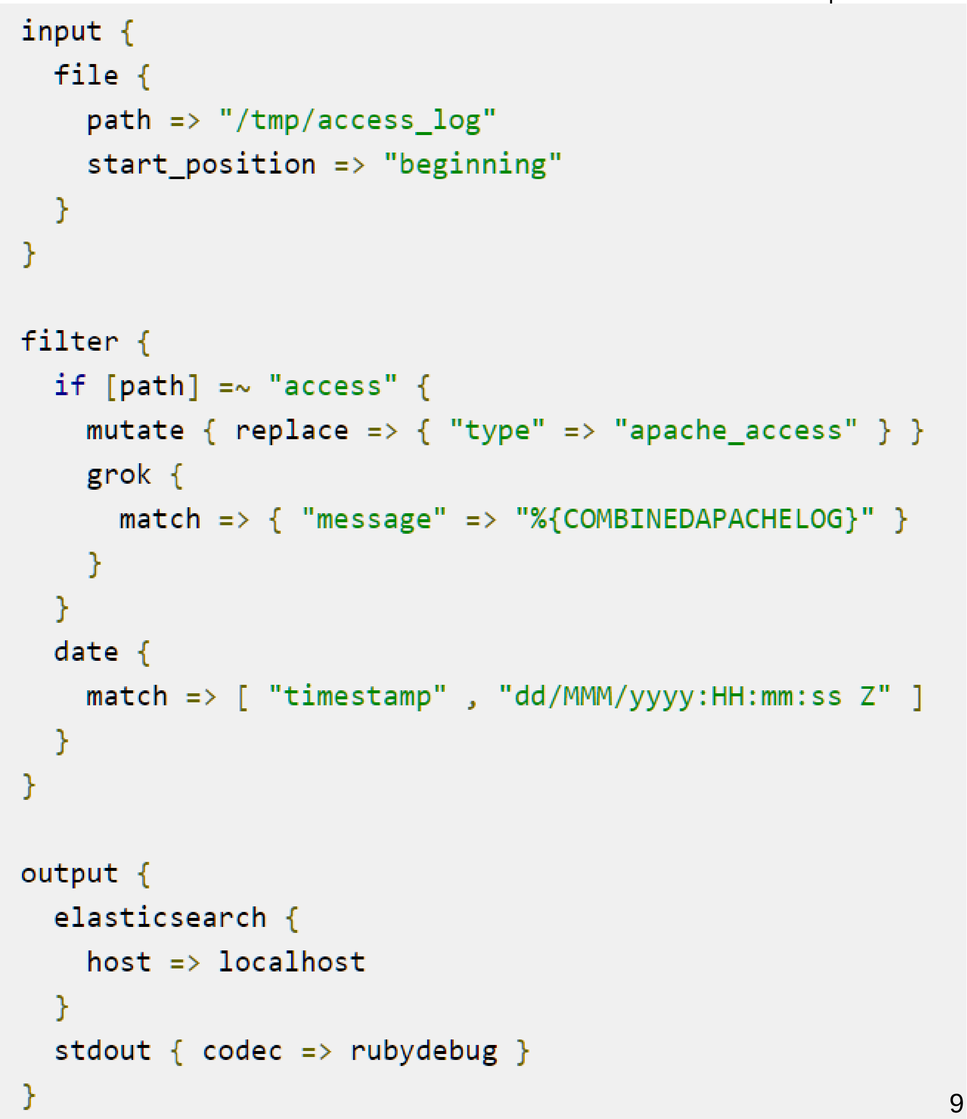


Logstash configuration structure

Multiple inputs of different types.

Conditionally filter and transform data. Several common formats are already supported.

Forward to multiple outputs.



```
input {
  file {
    path => "/tmp/access_log"
    start_position => "beginning"
  }
}

filter {
  if [path] =~ "access" {
    mutate { replace => { "type" => "apache_access" } }
    grok {
      match => { "message" => "%{COMBINEDAPACHELOG}" }
    }
  }
  date {
    match => [ "timestamp" , "dd/MMM/yyyy:HH:mm:ss Z" ]
  }
}

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

Logstash configuration structure
Plugins used in this example

Input plugins: https://www.elastic.co/guide/en/logstash/current/input-plugins.html

- file – streams events from files

Filter plugins: https://www.elastic.co/guide/en/logstash/current/filter-plugins.html

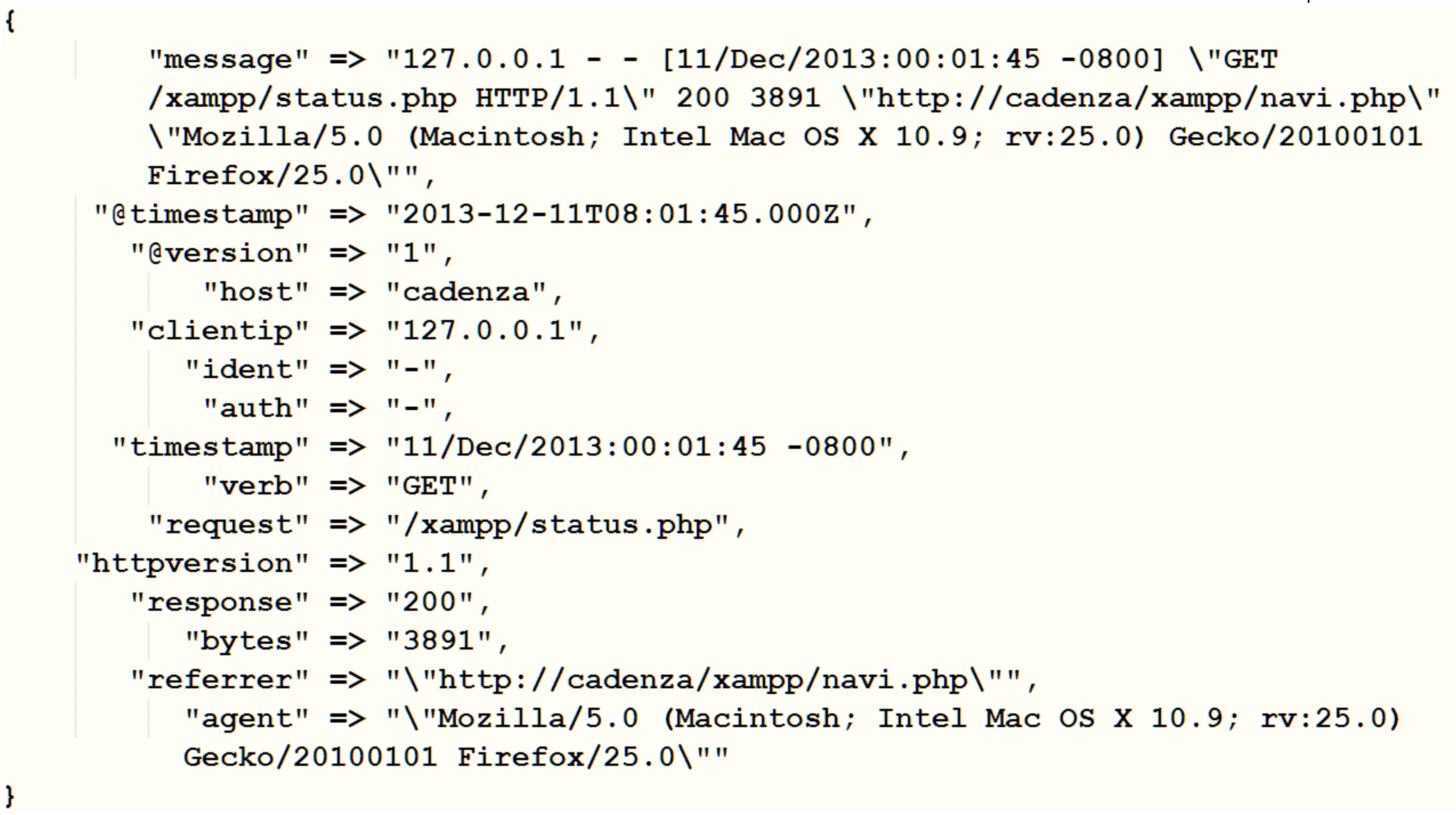
- grok – parses and structures arbitrary text
- date – parses dates from fields to use as the Logstash timestamp for an event
- mutate – performs mutations on fields

Output plugins: https://www.elastic.co/guide/en/logstash/current/output-plugins.html

- elasticsearch – stores logs in Elasticsearch
- stdout – prints events to the standard output

Console output when processing apache log files

Running logstash with *bin/logstash -f logstash.conf*



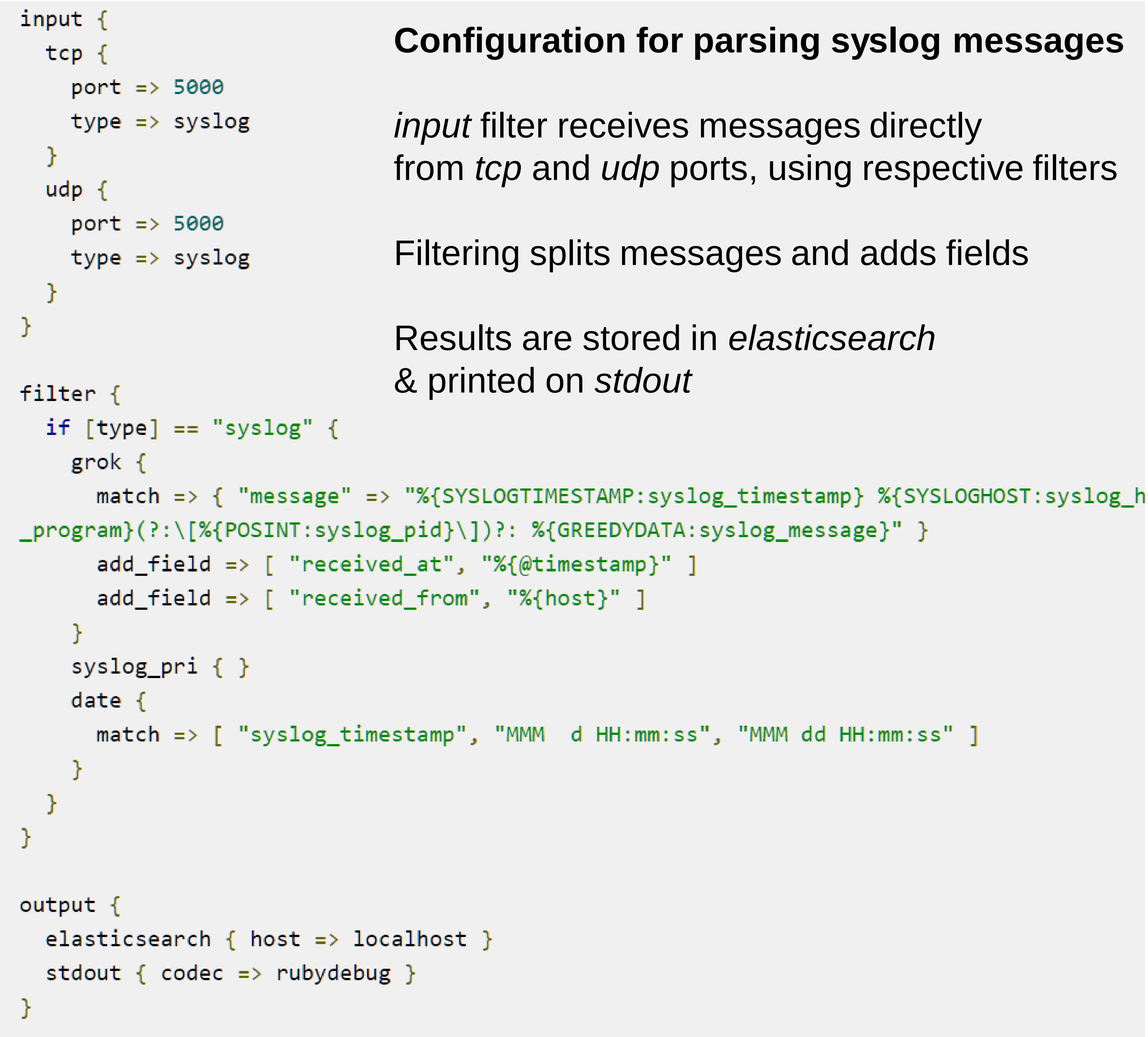
```
{
        "message" => "127.0.0.1 - - [11/Dec/2013:00:01:45 -0800] \"GET
        /xampp/status.php HTTP/1.1\" 200 3891 \"http://cadenza/xampp/navi.php\"
        \"Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:25.0) Gecko/20100101
        Firefox/25.0\"",
     "@timestamp" => "2013-12-11T08:01:45.000Z",
       "@version" => "1",
           "host" => "cadenza",
       "clientip" => "127.0.0.1",
          "ident" => "-",
           "auth" => "-",
      "timestamp" => "11/Dec/2013:00:01:45 -0800",
           "verb" => "GET",
        "request" => "/xampp/status.php",
    "httpversion" => "1.1",
       "response" => "200",
          "bytes" => "3891",
       "referrer" => "\"http://cadenza/xampp/navi.php\"",
          "agent" => "\"Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:25.0)
          Gecko/20100101 Firefox/25.0\""
}
```

Configuration for parsing syslog messages

input filter receives messages directly
from *tcp* and *udp* ports, using respective filters

Filtering splits messages and adds fields

Results are stored in *elasticsearch*
& printed on *stdout*



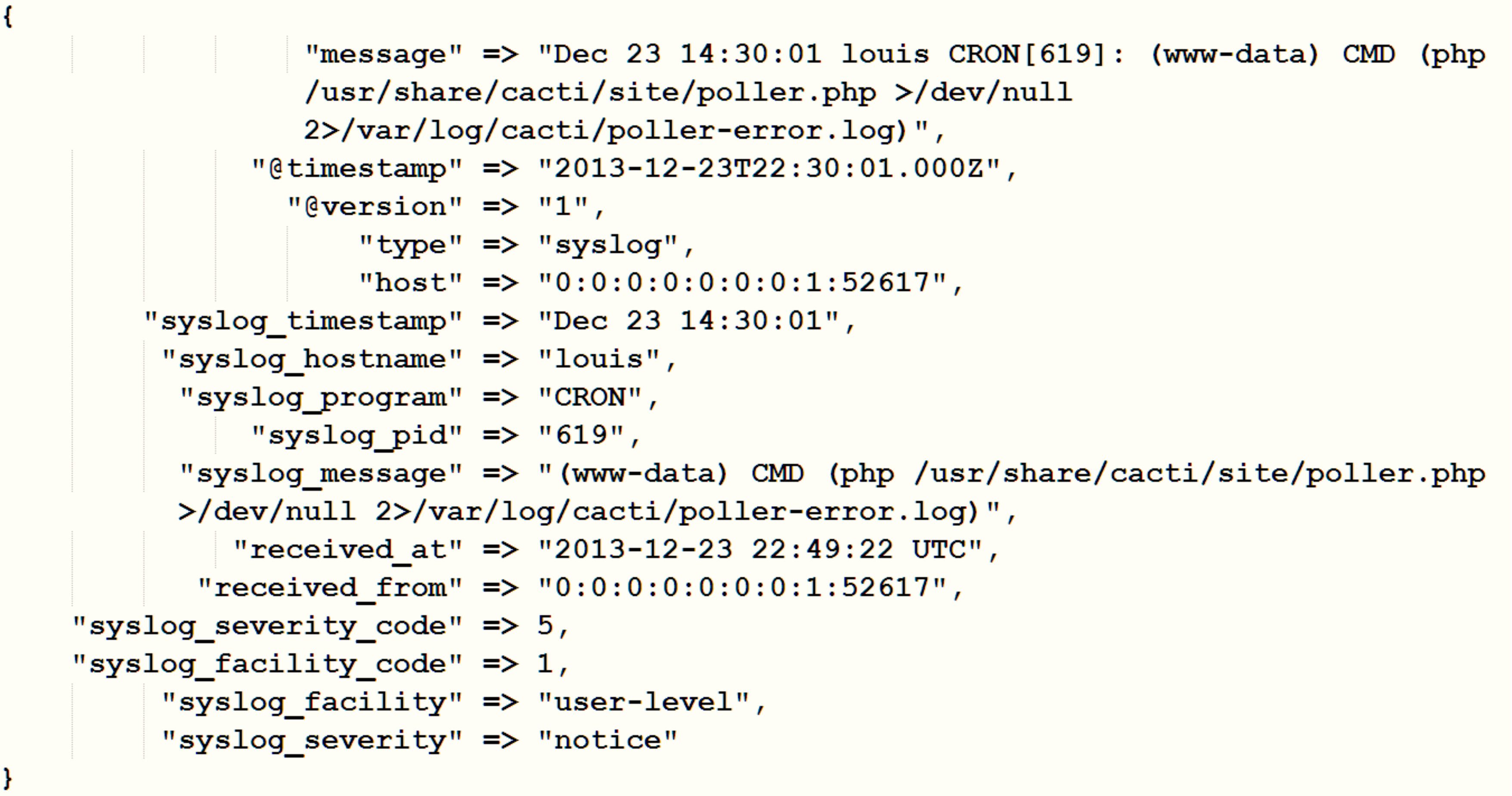
```
input {
  tcp {
    port => 5000
    type => syslog
  }
  udp {
    port => 5000
    type => syslog
  }
}

filter {
  if [type] == "syslog" {
    grok {
      match => { "message" => "%{SYSLOGTIMESTAMP:syslog_timestamp} %{SYSLOGHOST:syslog_h
_program}(?:\[%{POSINT:syslog_pid}\])?: %{GREEDYDATA:syslog_message}" }
      add_field => [ "received_at", "%{@timestamp}" ]
      add_field => [ "received_from", "%{host}" ]
    }
    syslog_pri { }
    date {
      match => [ "syslog_timestamp", "MMM  d HH:mm:ss", "MMM dd HH:mm:ss" ]
    }
  }
}

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

Console output when processing syslog messages

Running logstash with *bin/logstash -f logstash.conf*



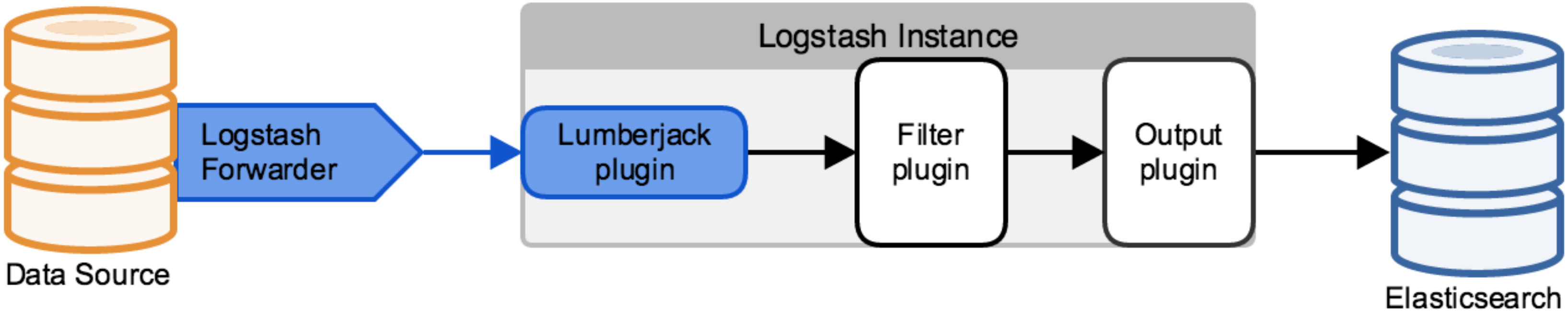
```
{
                 "message" => "Dec 23 14:30:01 louis CRON[619]: (www-data) CMD (php
                 /usr/share/cacti/site/poller.php >/dev/null
                 2>/var/log/cacti/poller-error.log)",
              "@timestamp" => "2013-12-23T22:30:01.000Z",
                "@version" => "1",
                    "type" => "syslog",
                    "host" => "0:0:0:0:0:0:0:1:52617",
        "syslog_timestamp" => "Dec 23 14:30:01",
         "syslog_hostname" => "louis",
          "syslog_program" => "CRON",
              "syslog_pid" => "619",
          "syslog_message" => "(www-data) CMD (php /usr/share/cacti/site/poller.php
          >/dev/null 2>/var/log/cacti/poller-error.log)",
             "received_at" => "2013-12-23 22:49:22 UTC",
           "received_from" => "0:0:0:0:0:0:0:1:52617",
    "syslog_severity_code" => 5,
    "syslog_facility_code" => 1,
         "syslog_facility" => "user-level",
         "syslog_severity" => "notice"
}
```

Examples of available input plugins

https://www.elastic.co/guide/en/logstash/current/input-plugins.html

- *file* – for processing files
- *tcp*, *udp*, *unix* – reading directly from network sockets
- *http* – for processing HTTP POST requests
- *http_poller* – for polling HTTP services as input sources
- *imap* – accessing and processing imap mail
- Plugins to access message queues:
  - *rabbitmq*, *stomp*…
- Plugins to access database systems:
  - *jdbc*, *elasticsearch*…
- To read data from system log services and command-line:
  - *syslog*, *eventlog*, *pipe*, *exec*…

The *lumberjack* input plugin & the logstash forwarder



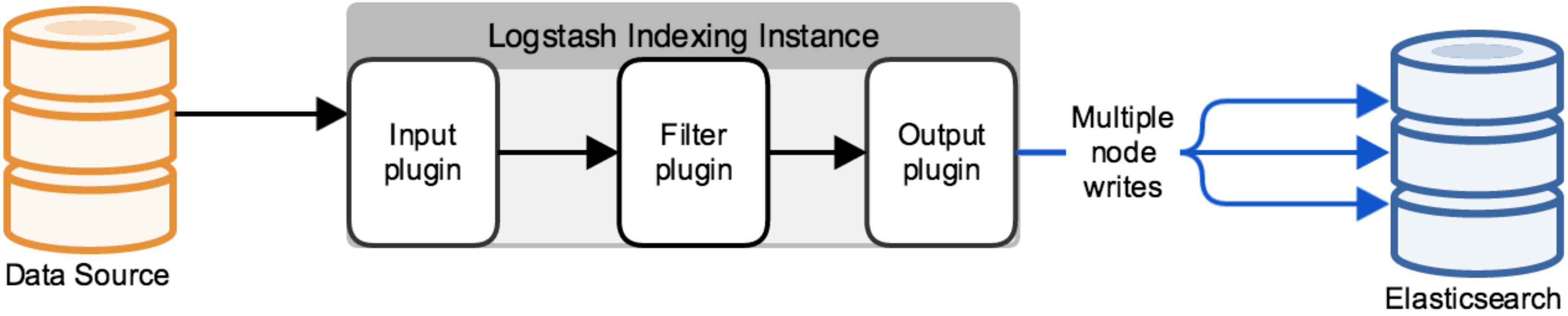
- The *Logstash forwarder* application can be installed on the servers as an “agent”, for sending logs to a Logstash server. This allows to forward local logs/inputs from that server to the *logstash* instance.
- The *lumberjack input plugin*, which receives events using the Lumberjack protocol can then be configured to consume the messages of the *logstash forwarder.*
- The network transfer can be encrypted and authenticated.

Examples of output plugins

https://www.elastic.co/guide/en/logstash/current/output-plugins.html

- *stdout*, *pipe*, *exec* – show output on console, feed to command
- *file* – store output in a file
- *email* – send output as email
- *tcp*, *udp*, *websocket* – send output over network connections
- *http* – send output as HTTP request
- To send output to databases, index server or cloud storage:
  - *elasticsearch*, *solr_http*, *mongodb*, *google_bigquery*, *google_cloud_storage*, *opentsdb*
- To send output to message queues:
  - *rabbitmq*, *stomp*, …
- To forward messages to metrics applications:
  - *graphite*, *graphtastic*, *ganglic*, *metriccatcher*

Writing to multiple nodes



- The *elasticsearch* output plugin may write to multiple nodes
- It will distribute output objects to different nodes (“load balancing”)

(a logstash instance can also be part of an elasticsearch cluster and write data through the cluster protocol)

Examples of filter plugins

https://www.elastic.co/guide/en/logstash/current/filter-plugins.html

- *grok* – parse and structure arbitrary text: best generic option to interpret text as (semi-)structured objects
- To parse different data formats: *csv*, *json*, *xml*…
- *multiline* – collapse multiline messages to one logstash event
- *split* – split multiline messages into several logstash events
- *aggregate* – aggregates separate message lines into one Logstash event
- *mutate* – perform mutations of fields (rename, remove, replace, modify)
- *dns* – lookup DNS entry for IP address
- *geoip* – find geolocation of IP address

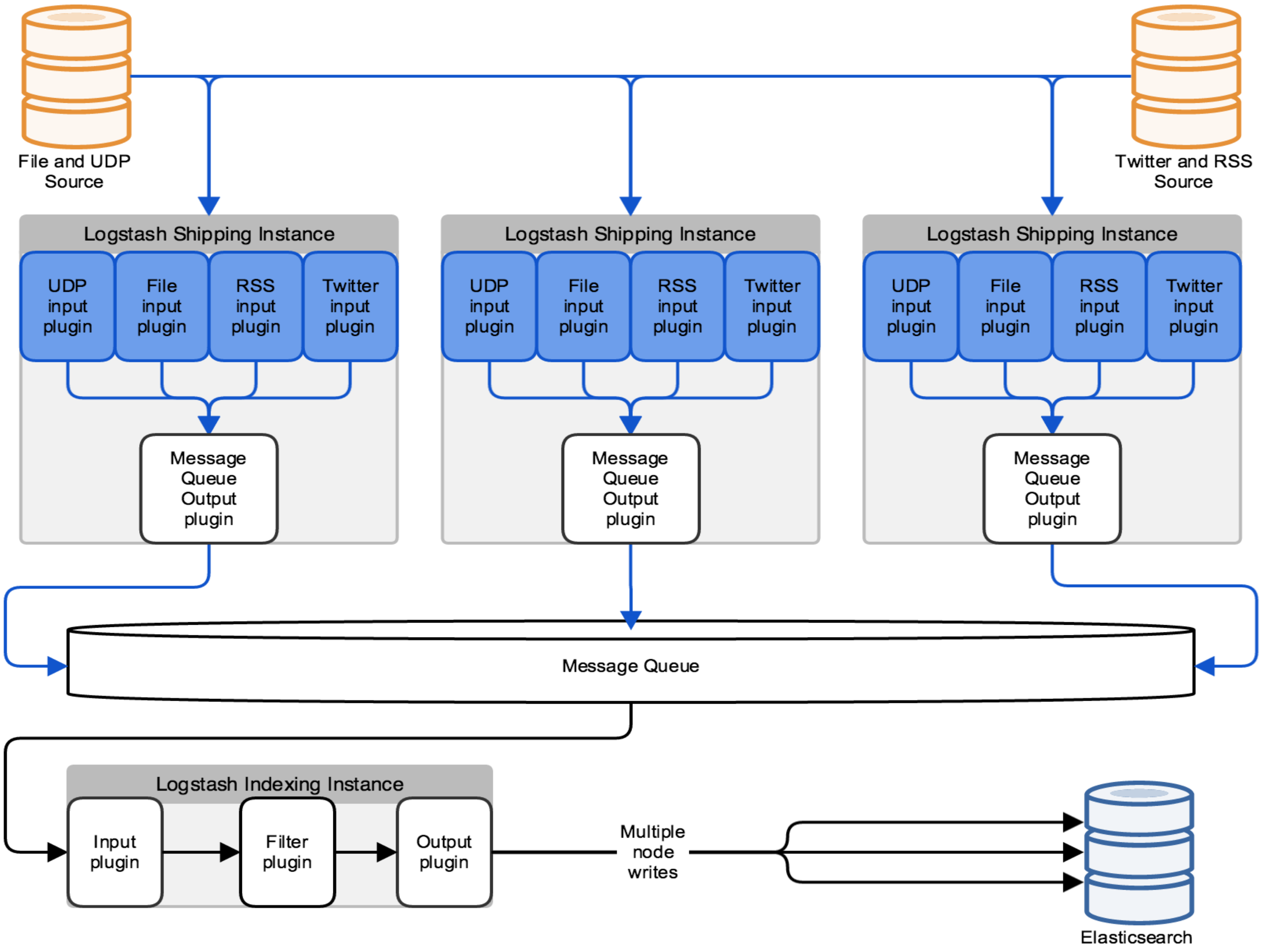
Example with the *grok* plugin

- Input: 55.3.244.1 GET /index.html 15824 0.043
- grok filter:

```
filter {
    grok { match => { "message" => "%{IP:client}
    %{WORD:method} %{URIPATHPARAM:request}
    %{NUMBER:bytes} %{NUMBER:duration}" }
}
```

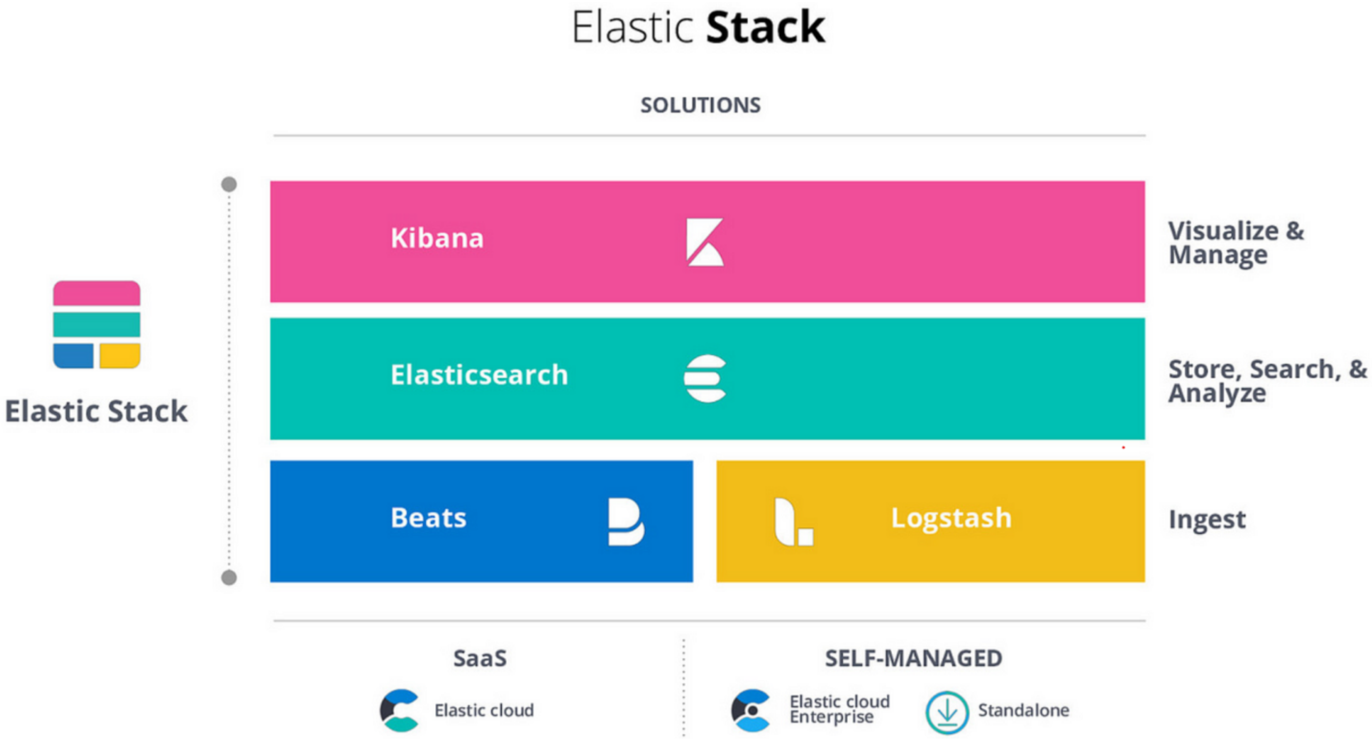
- The output will contain fields such as:
  - client: 55.3.244.1
  - method: GET
  - request: /index.html
  - bytes: 15824
  - duration: 0.043

Scaling and high availability



Beats as a complement (and/or alternative) to Logstash

https://www.elastic.co/beats/



Lightweight data shippers

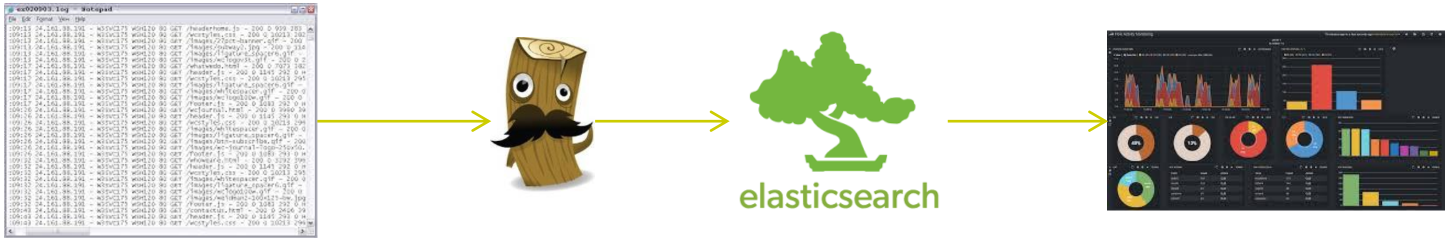
Beats is a free and open platform for single-purpose data shippers.

These data shippers send data from hundreds or thousands of machines and systems to Logstash or Elasticsearch.

ELK Stack (revisited)

Three opensource software products:

- **E**lasticsearch
  Highly scalable search index server
- **L**ogstash
  Collection, enrichment, filtering and forwarding
- **K**ibana
  Exploration and visualization of data



Logs
(from multiple sources)

Logstash

Kibana

Elasticsearch

- Server environment for storing and querying large-scale structured index entries
  - Document-oriented (structured) index entries
  - Combines “full text”-oriented search options- (for text fields) with more precise search options for other types of fields, like date + time fields, geolocation fields, etc.
  - Near real-time search and analysis capabilities
- Provides Restful API as JSON over HTTP

Scalability of Elasticsearch

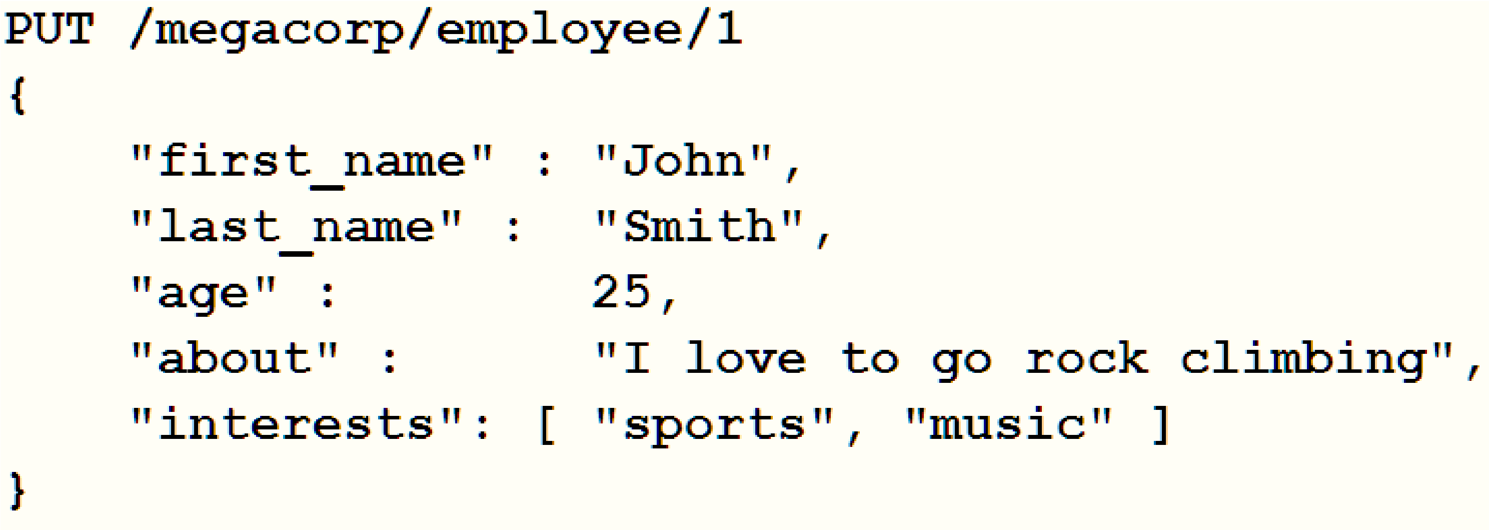
- Elasticsearch can run as one integrated application on multiple nodes of a cluster
- Indexes are stored in instances called “*shards*”, which can be distributed over several nodes
- There a two types of “*shards*”
  - Primary Shards
  - Replicas
- Replicas of “Primary Shards” provide
  - Failure tolerance (and therefore data protection)
  - Faster queries (search faster)

Indexing data with Elasticsearch

- Send JSON documents to server (e.g., use REST API)
  - No schema is necessary, since *ElasticSearch* can automatically determine the type of attributes
  - Nevertheless, it is possible to explicitly specify schema (i.e., the attribute types). Some examples types:
    - *string, byte, short, integer, long, float, double, boolean, date*
- Analysis of text attributes for “full text”-oriented search-
  - Word extraction, reduction of words to their base form (stemming)
  - Stop words
  - Support for multiple languages

Indexing data using the REST API

a simple example

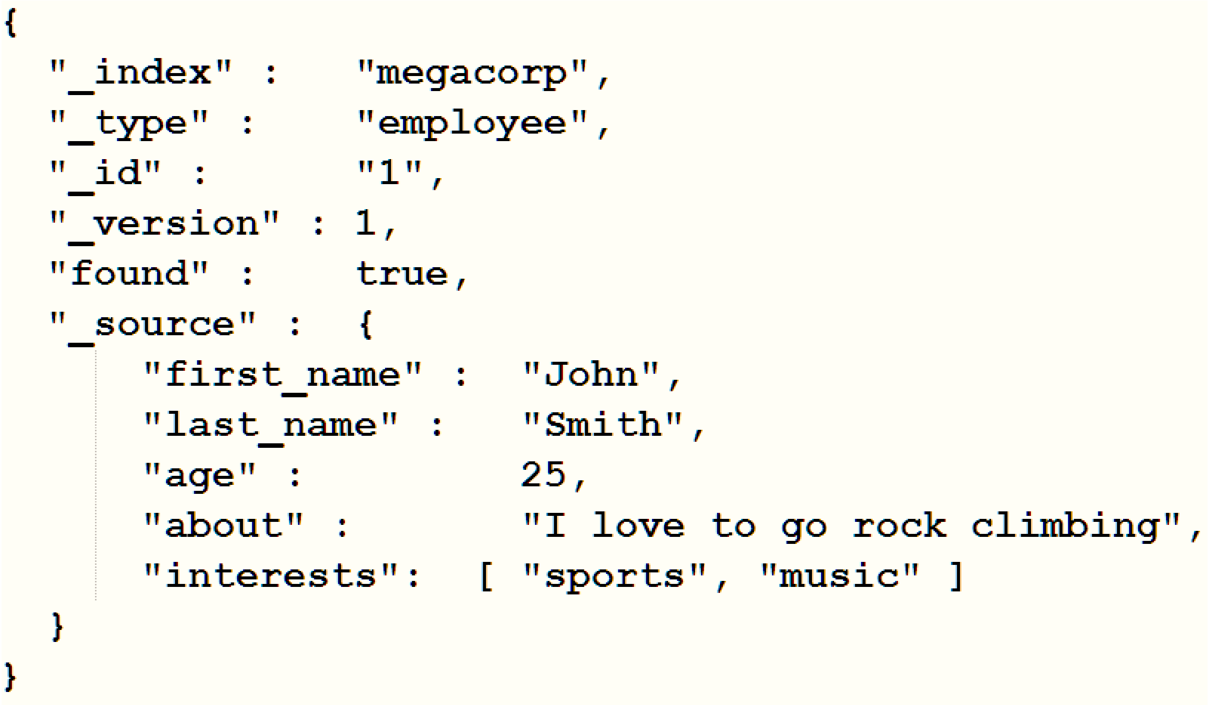


```
PUT /megacorp/employee/1
{
    "first_name" : "John",
    "last_name" :  "Smith",
    "age" :        25,
    "about" :      "I love to go rock climbing",
    "interests": [ "sports", "music" ]
}
```

- PUT request inserts the JSON payload into the index with name "*megacorp*" (company name), as object of type "*employee*"
- Schema for type could have been explicitly defined (at time of index creation or automatically determined)
- Text fields (e.g., "about") will be analyzed if the analyzers are configured for that field
- Request URL specifies the identifier "1" for the index entry

Retrieval of an index entry

GET /megacorp/employee/1



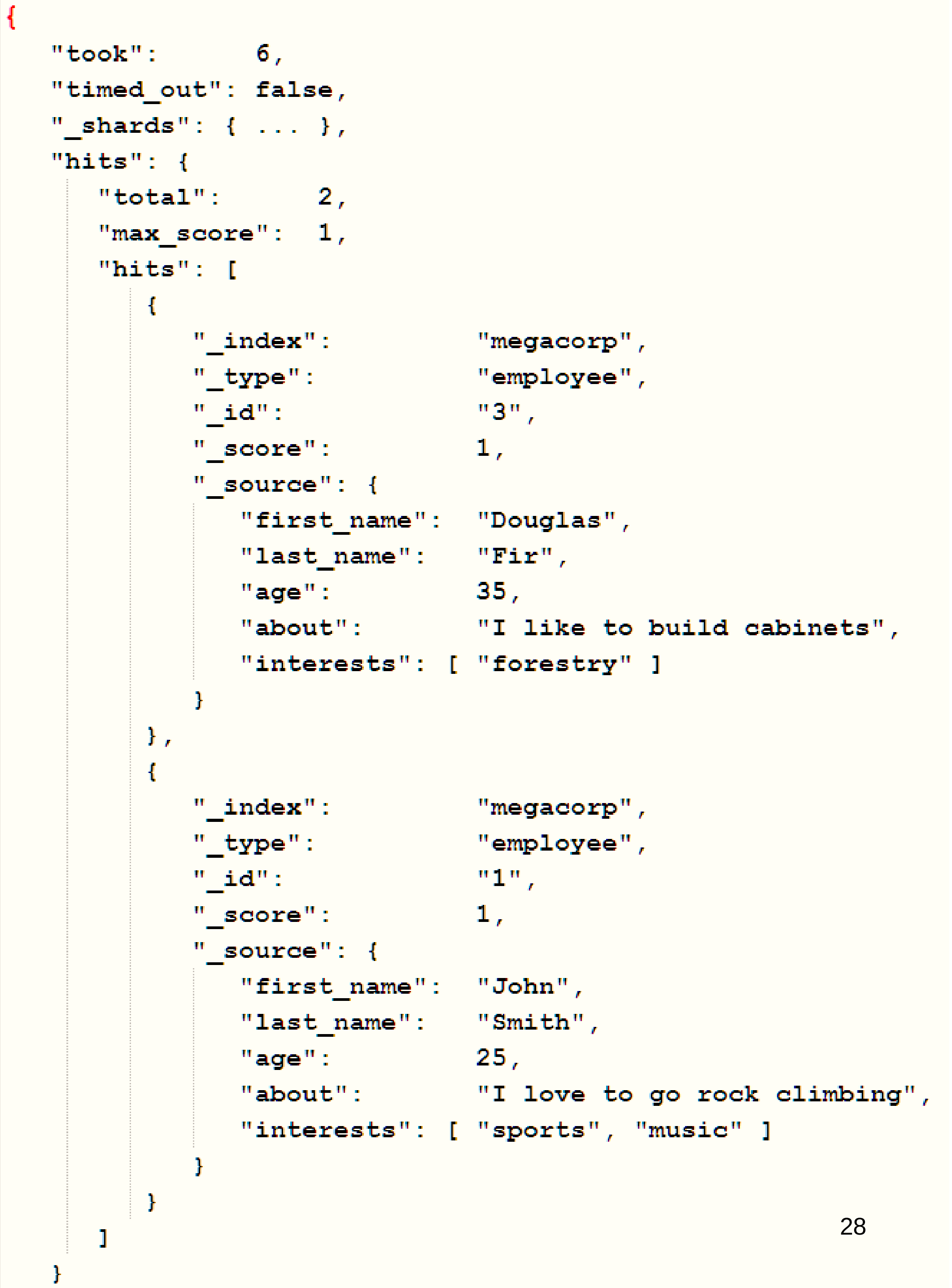
```
{
  "_index" :   "megacorp",
  "_type" :    "employee",
  "_id" :      "1",
  "_version" : 1,
  "found" :    true,
  "_source" :  {
      "first_name" :  "John",
      "last_name" :   "Smith",
      "age" :         25,
      "about" :       "I love to go rock climbing",
      "interests":  [ "sports", "music" ]
  }
}
```

- The “GET” REST API call with “/megacorp/employee/1” will retrieve the entry with id 1 as a JSON object

Simple Query

GET /megacorp/employee/_search

- GET request with "_search" at the end of the URL performs query
- Search results are returned in the JSON response as the "hits" array
- Further metadata specifies the count of search results ("*total*") and *max_score*

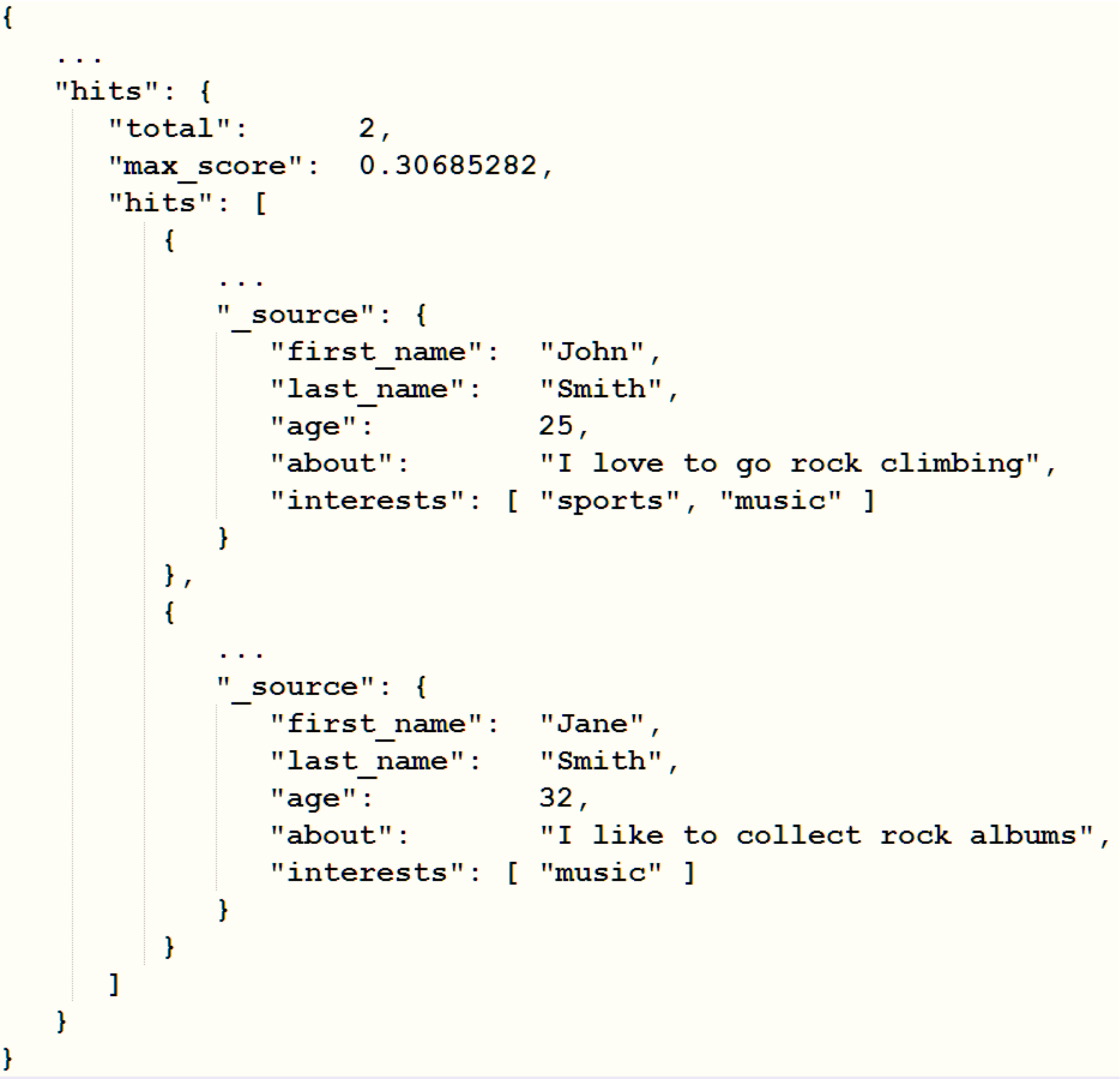


```
{
    "took":      6,
    "timed_out": false,
    "_shards": { ... },
    "hits": {
       "total":      2,
       "max_score":  1,
       "hits": [
          {
             "_index":         "megacorp",
             "_type":          "employee",
             "_id":            "3",
             "_score":         1,
             "_source": {
                "first_name":  "Douglas",
                "last_name":   "Fir",
                "age":         35,
                "about":       "I like to build cabinets",
                "interests": [ "forestry" ]
             }
          },
          {
             "_index":         "megacorp",
             "_type":          "employee",
             "_id":            "1",
             "_score":         1,
             "_source": {
                "first_name":  "John",
                "last_name":   "Smith",
                "age":         25,
                "about":       "I love to go rock climbing",
                "interests": [ "sports", "music" ]
             }
          }
       ]
    }
```

}

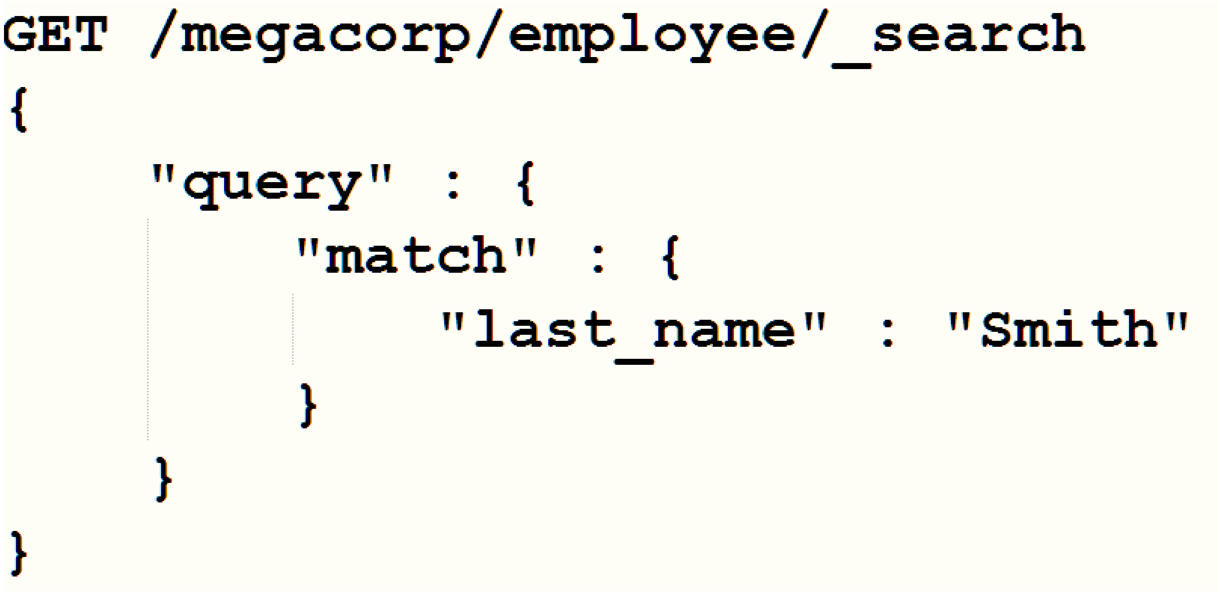
Simple Query with search string

GET /megacorp/employee/_search?q=last_name:Smith



```
{
   ...
   "hits": {
      "total":      2,
      "max_score":  0.30685282,
      "hits": [
         {
            ...
            "_source": {
               "first_name":  "John",
               "last_name":   "Smith",
               "age":         25,
               "about":       "I love to go rock climbing",
               "interests": [ "sports", "music" ]
            }
         },
         {
            ...
            "_source": {
               "first_name":  "Jane",
               "last_name":   "Smith",
               "age":         32,
               "about":       "I like to collect rock albums",
               "interests": [ "music" ]
            }
         }
      ]
   }
}
```

More complex queries with Query DSL



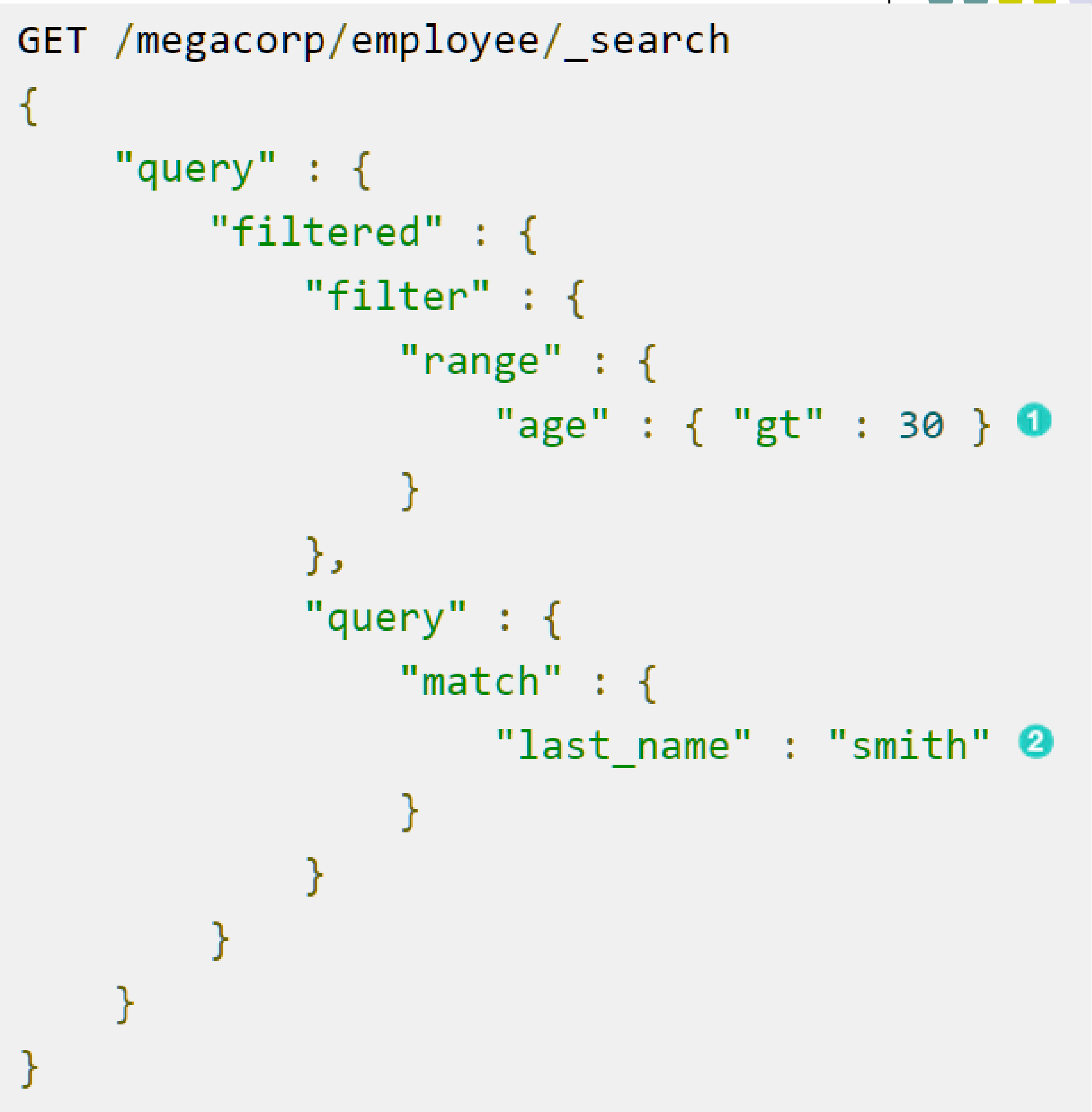
```
GET /megacorp/employee/_search
{
    "query" : {
        "match" : {
            "last_name" : "Smith"
        }
    }
}
```

- Query DSL is a JSON language for more complex queries
- Will be sent as payload, within the search request
- *match* clause has same semantics as in simple query

More complex queries with Query DSL

- Consists of a query and a filter part
- Query part matches all entries with last_name “smith” (2)
- Filter will then only select entries which fulfill the range filter (1)

“age”: {“gt” : 30 }



```
GET /megacorp/employee/_search
{
    "query" : {
        "filtered" : {
            "filter" : {
                "range" : {
                    "age" : { "gt" : 30 } ❶
                }
            },
            "query" : {
                "match" : {
                    "last_name" : "smith" ❷
                }
            }
        }
    }
}
```

Some query possibilities

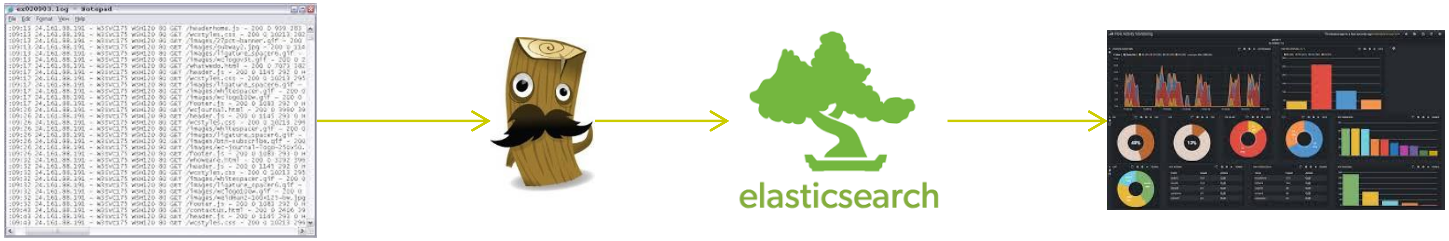
- Combined search on different attributes and indexes
  - Many possibilities for full-text search on attribute values
    - Exact, non-exact, proximity (phrases), partial match
  - Support well-known logical operators (and / or, …)
  - Range queries (for instance date ranges)

  …
- Control relevance and ranking of search results, sort them
  - Boost relevance while indexing
  - Boost or ignore relevance while querying
  - Different possibilities to sort search results otherwise

ELK Stack (revisited)

Three opensource software products:

- **E**lasticsearch
  Highly scalable search index server
- **L**ogstash
  Collection, enrichment, filtering and forwarding
- **K**ibana
  Exploration and visualization of data

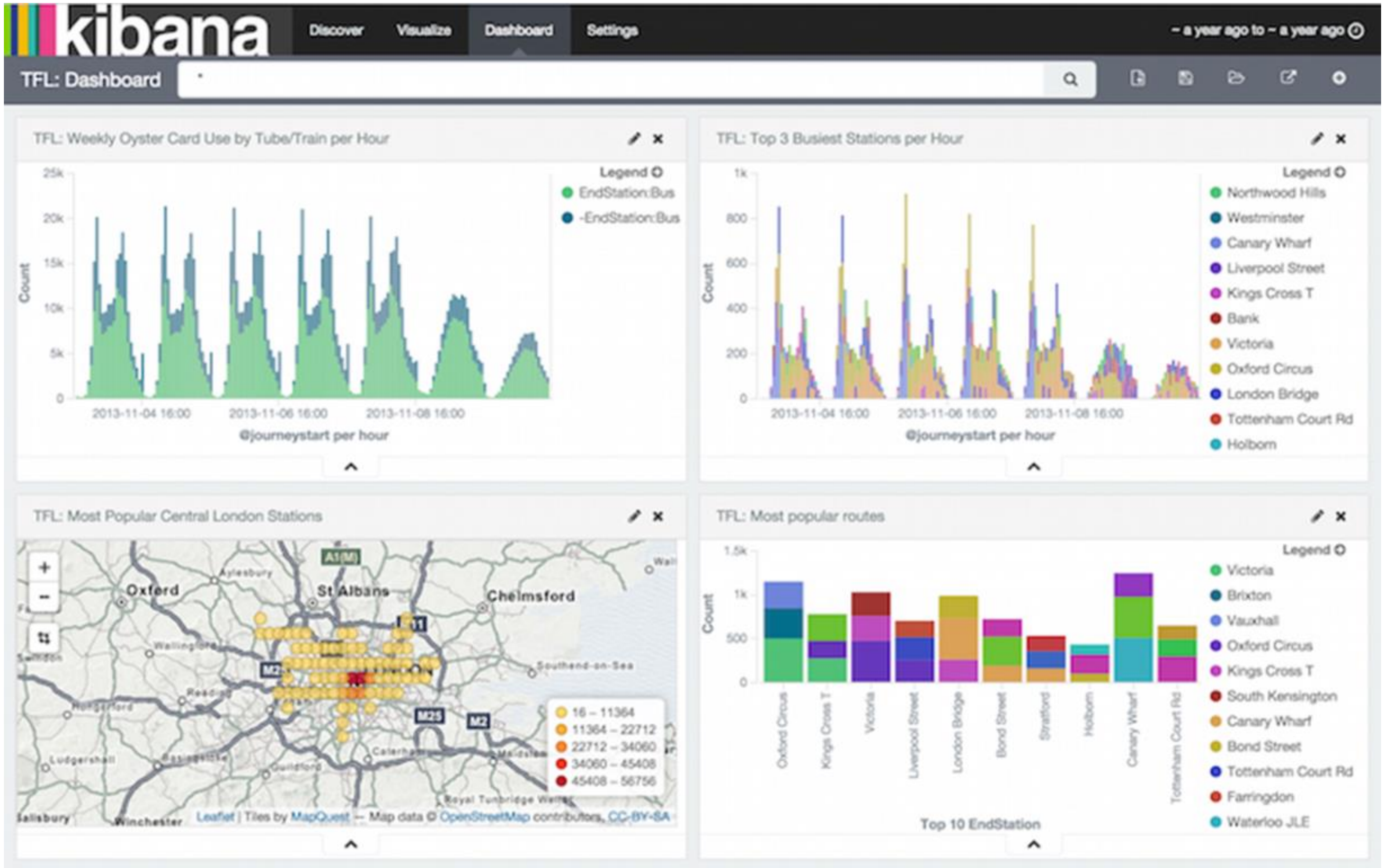


Logs
(from multiple sources)

Logstash

Kibana

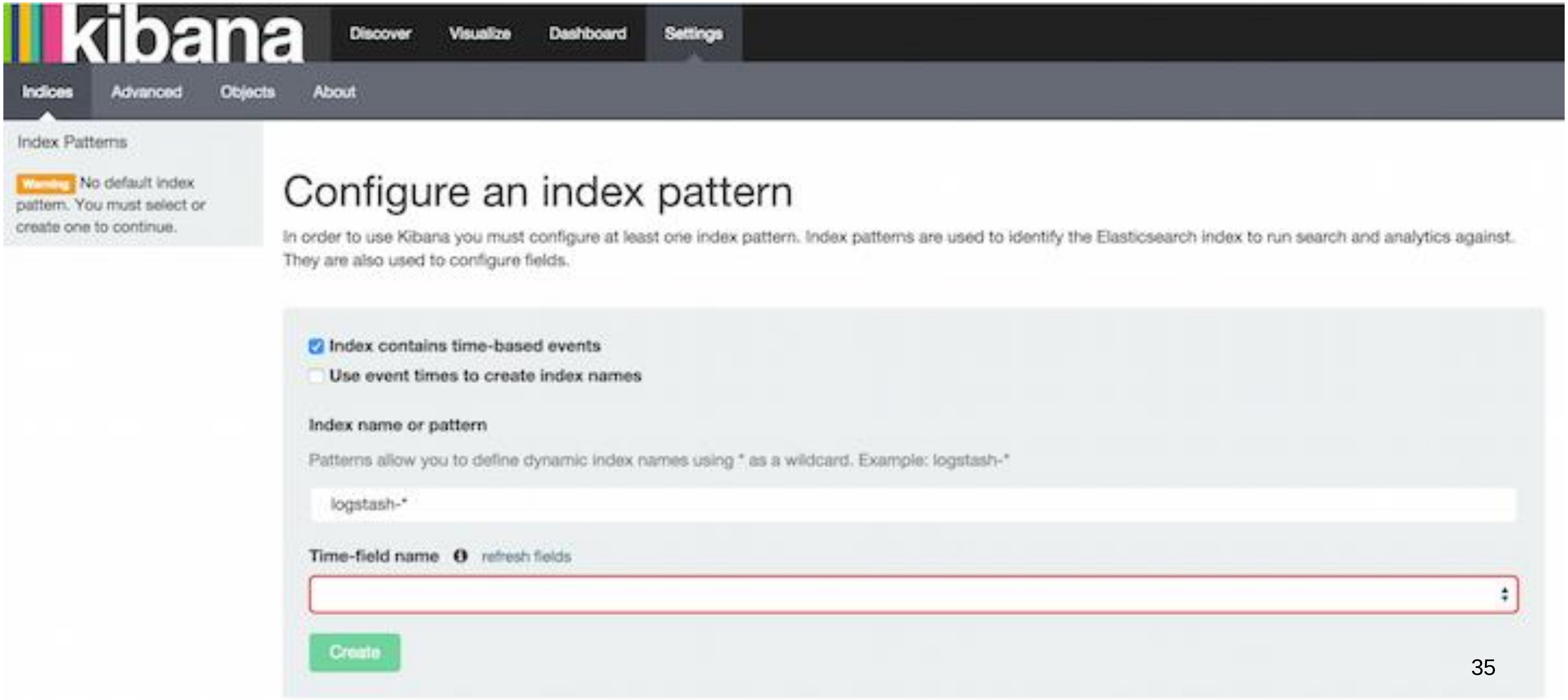
Kibana



- Web-based application for exploring and visualizing data
- Modern browser-based interface (HTML5 + JavaScript)
- Ships with its own web server for easier setup
- Seamless integration with Elasticsearch

Configure Kibana

- After installation first configure Kibana to access the Elasticsearch server(s), by editing the Kibana config file
- Then use the Web UI to configure the indexes to use



Warning No default index pattern. You must select or create one to continue.

Configure an index pattern

In order to use Kibana you must configure at least one index pattern. Index patterns are used to identify the Elasticsearch index to run search and analytics against. They are also used to configure fields.

☑ Index contains time-based events

□ Use event times to create index names

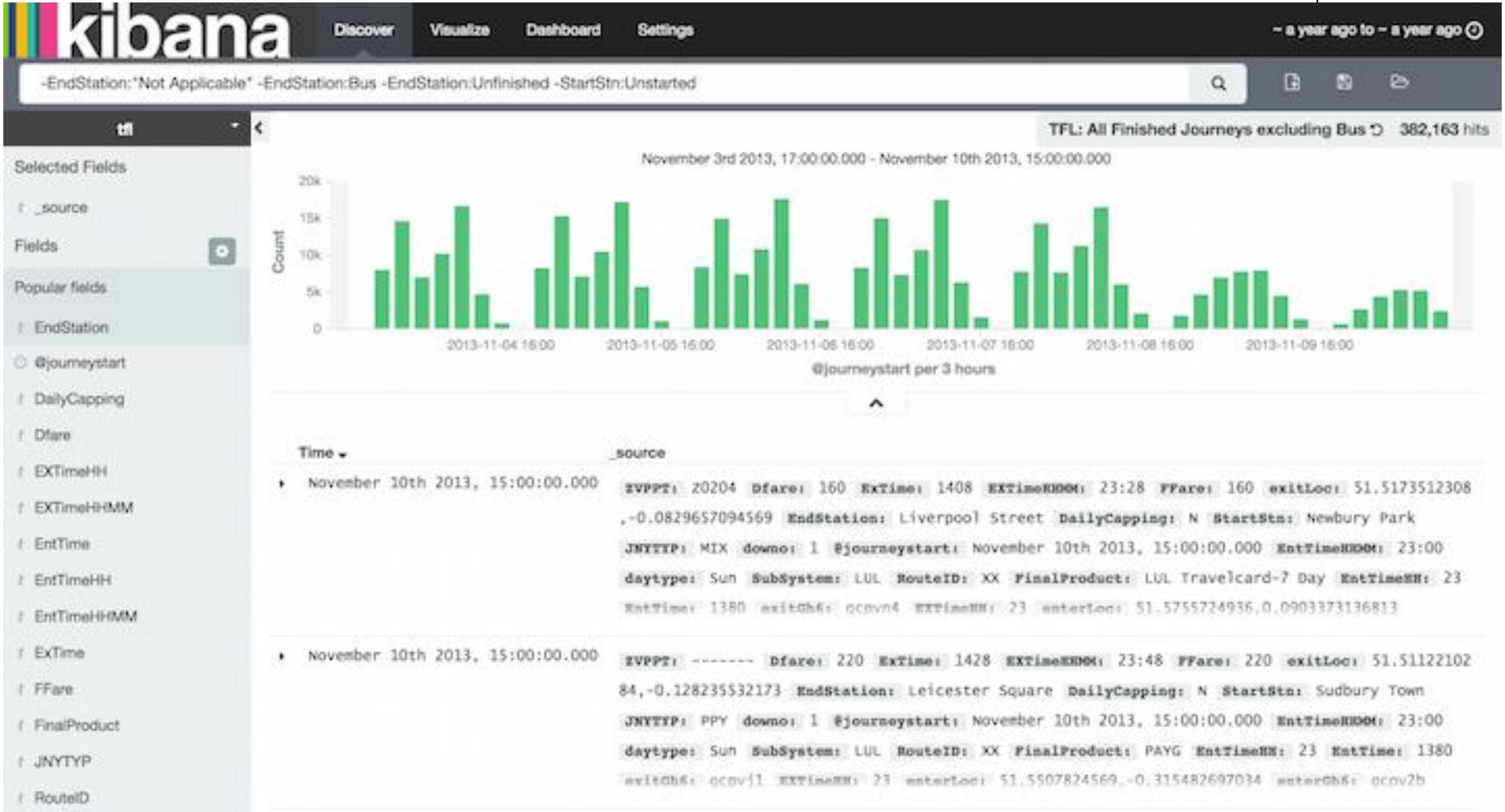
Index name or pattern

Patterns allow you to define dynamic index names using * as a wildcard. Example: logstash-*

logstash-*

Time-field name refresh fields

Discover data



-EndStation:"Not Applicable" -EndStation:Bus -EndStation:Unfinished -StartStn:Unstarted

Selected Fields

- _source

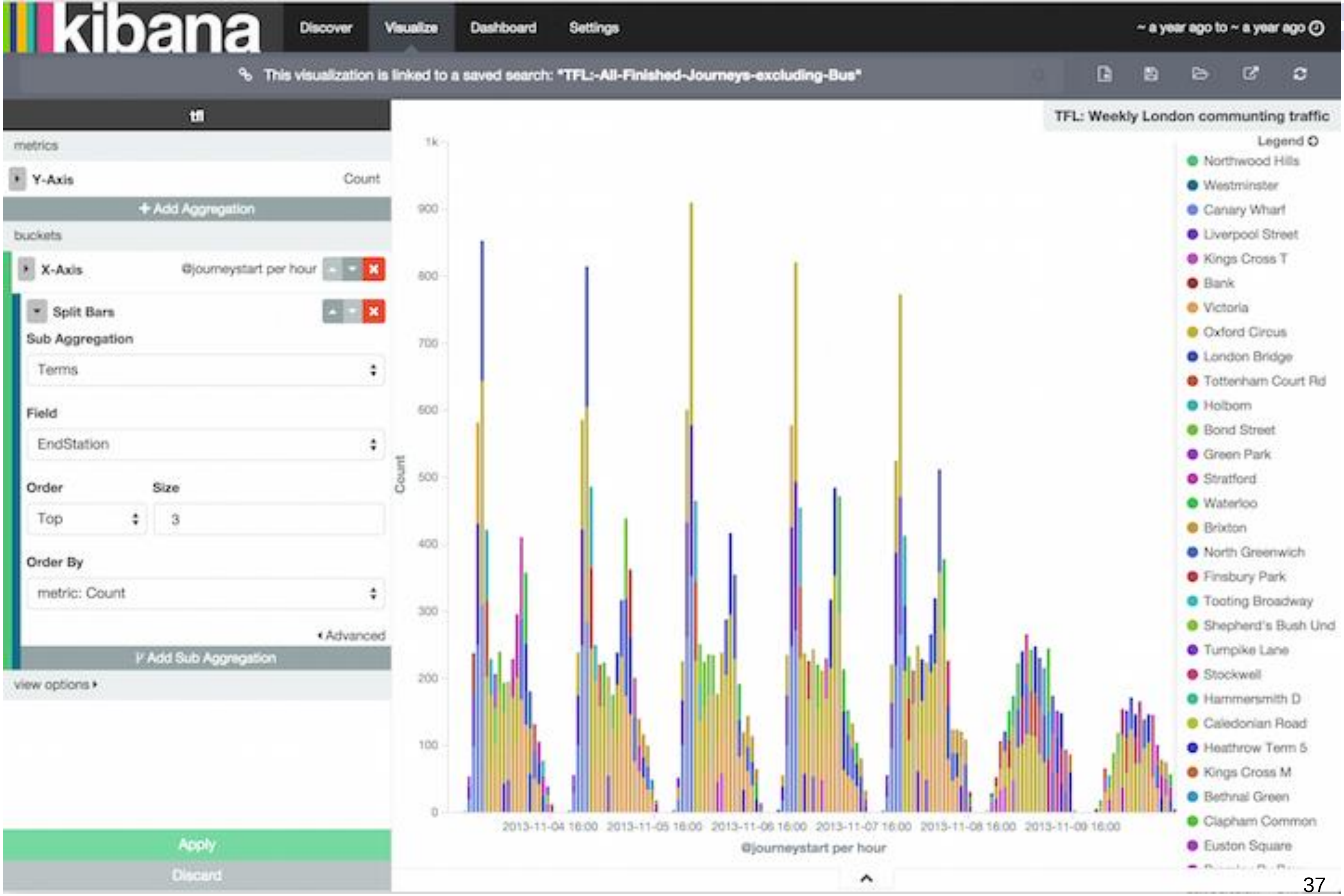
Fields

Popular fields

- EndStation
- @journeystart
- DailyCapping
- Dfare
- EXTimeHH
- EXTimeHHMM
- EntTime
- EntTimeHH
- EntTimeHHMM
- ExTime
- FFare
- FinalProduct
- JNYTYP
- RouteID

Time	_source
November 10th 2013, 15:00:00.000	ZVPPT: Z0204 Dfare: 160 ExTime: 1408 EXTimeHHMM: 23:28 FFare: 160 exitLoc: 51.5173512308,-0.0829657094569 EndStation: Liverpool Street DailyCapping: N StartStn: Newbury Park JNYTYP: MIX downo: 1 @journeystart: November 10th 2013, 15:00:00.000 EntTimeHHMM: 23:00 daytype: Sun SubSystem: LUL RouteID: XX FinalProduct: LUL Travelcard-7 Day EntTimeHH: 23 EntTime: 1380 exitGh6: gcpvn4 EXTimeHH: 23 enterLoc: 51.5755724936,0.090373136813
November 10th 2013, 15:00:00.000	ZVPPT: ------- Dfare: 220 ExTime: 1428 EXTimeHHMM: 23:48 FFare: 220 exitLoc: 51.5112210284,-0.128235532173 EndStation: Leicester Square DailyCapping: N StartStn: Sudbury Town JNYTYP: PPY downo: 1 @journeystart: November 10th 2013, 15:00:00.000 EntTimeHHMM: 23:00 daytype: Sun SubSystem: LUL RouteID: XX FinalProduct: PAYG EntTimeHH: 23 EntTime: 1380 exitGh6: gcpvj1 EXTimeHH: 23 enterLoc: 51.5507824569,-0.315482697034 enterGh6: gcpv2b

Create a visualization



This visualization is linked to a saved search: "TFL:-All-Finished-Journeys-excluding-Bus"

metrics

Y-Axis Count

buckets

X-Axis @journeystart per hour

Split Bars

Sub Aggregation

Terms

Field

EndStation

Order

Top

Size

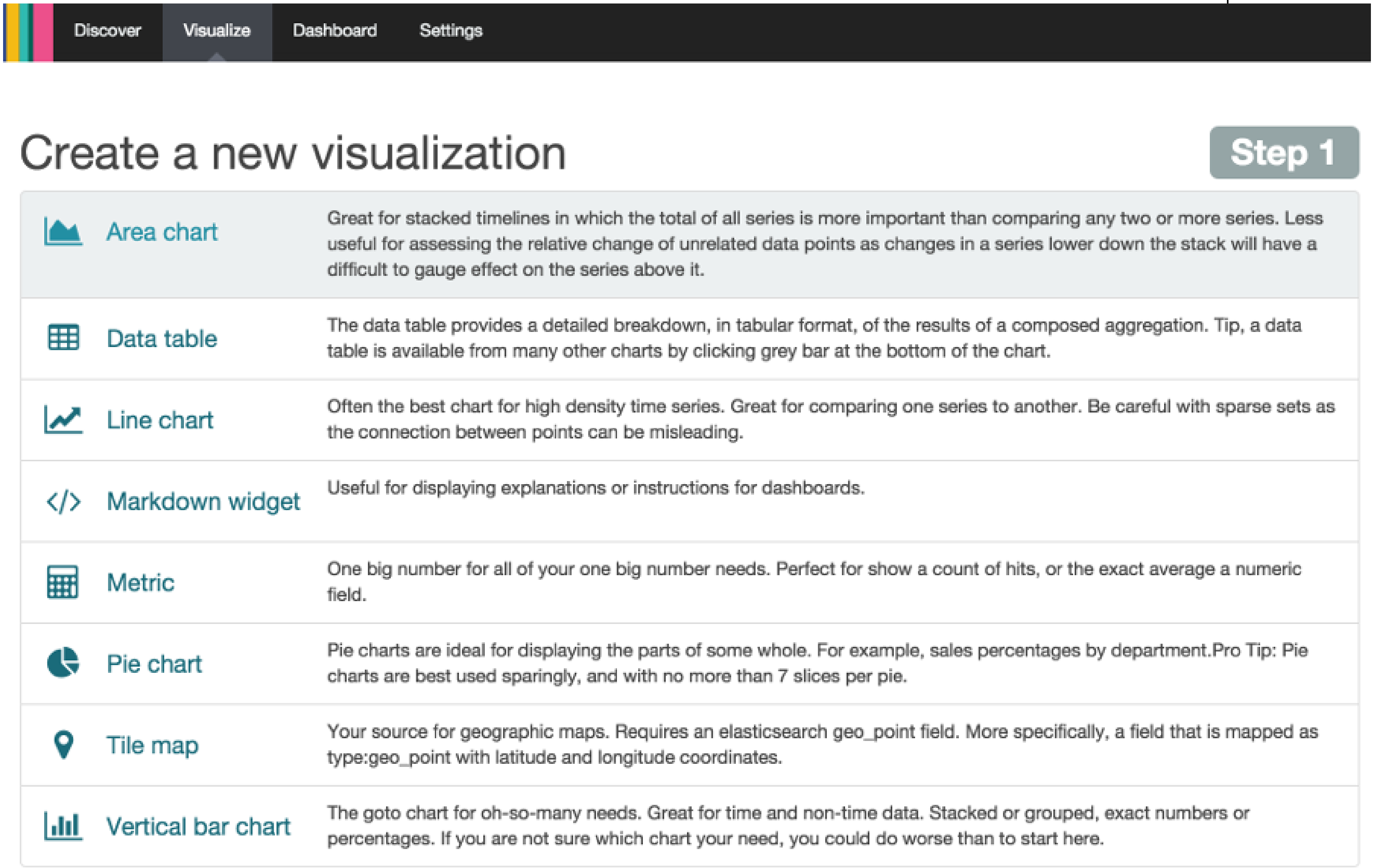
3

Order By

metric: Count

TFL: Weekly London communting traffic

Different types of visualizations

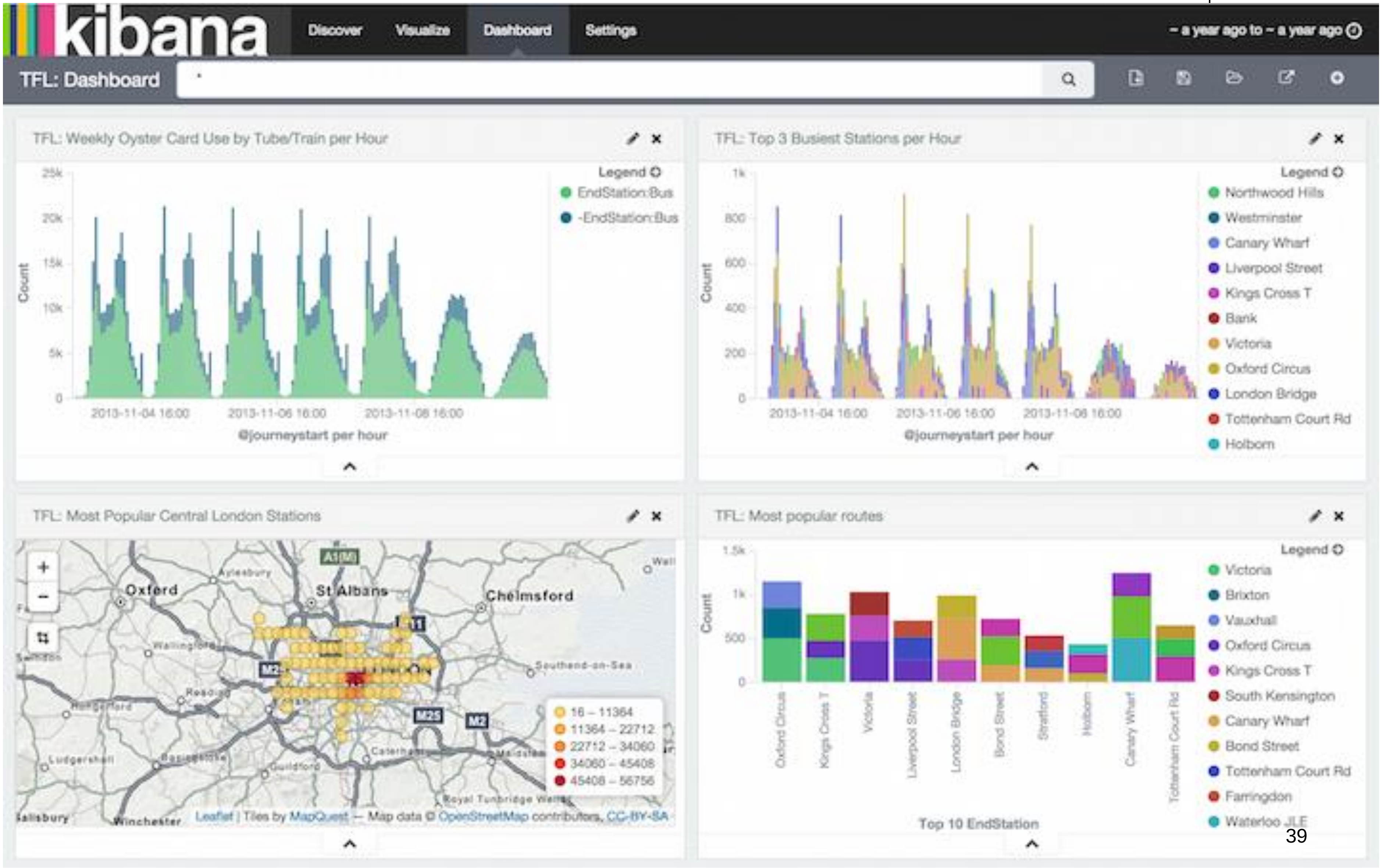


Create a new visualization

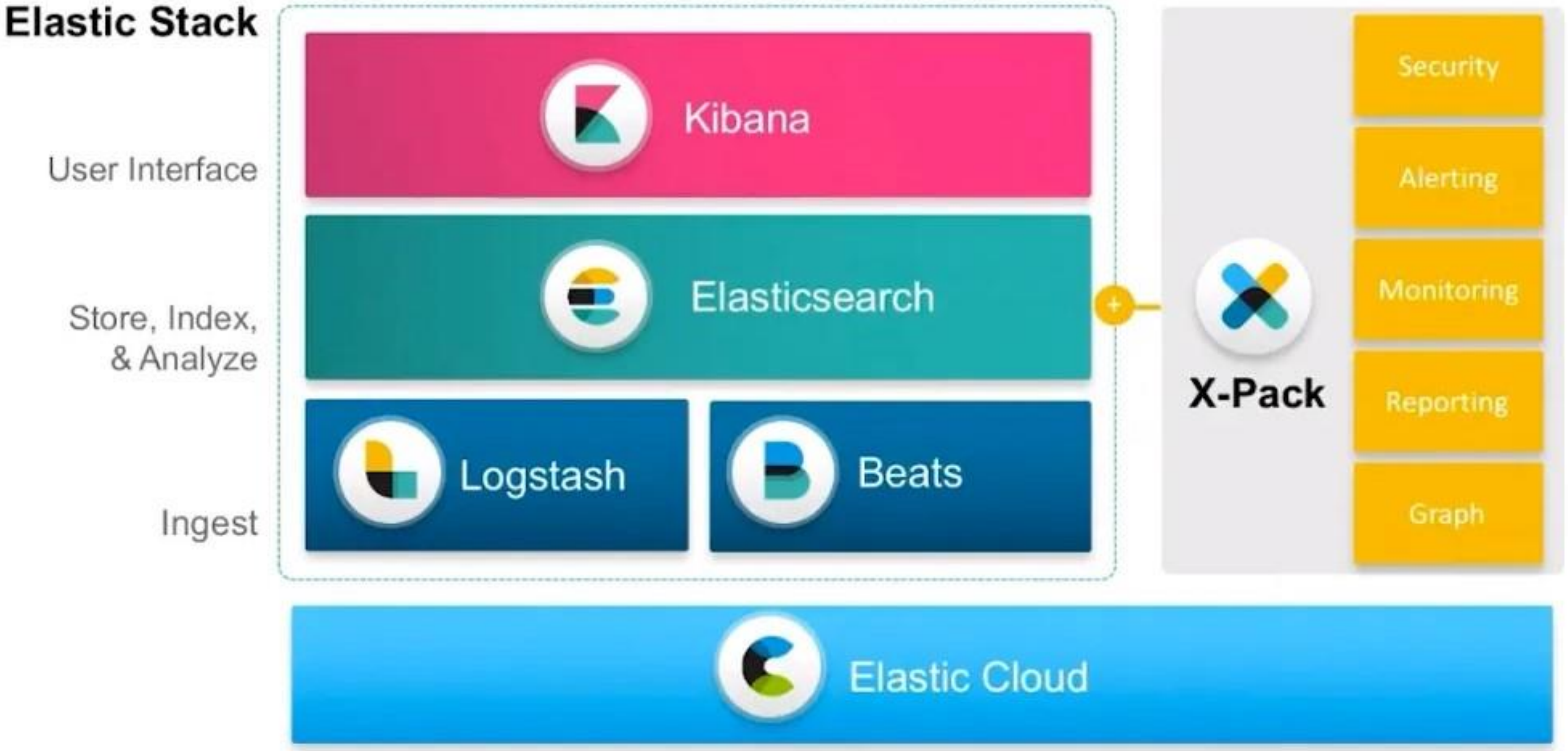
Step 1

Area chart	Great for stacked timelines in which the total of all series is more important than comparing any two or more series. Less useful for assessing the relative change of unrelated data points as changes in a series lower down the stack will have a difficult to gauge effect on the series above it.
Data table	The data table provides a detailed breakdown, in tabular format, of the results of a composed aggregation. Tip, a data table is available from many other charts by clicking grey bar at the bottom of the chart.
Line chart	Often the best chart for high density time series. Great for comparing one series to another. Be careful with sparse sets as the connection between points can be misleading.
Markdown widget	Useful for displaying explanations or instructions for dashboards.
Metric	One big number for all of your one big number needs. Perfect for show a count of hits, or the exact average a numeric field.
Pie chart	Pie charts are ideal for displaying the parts of some whole. For example, sales percentages by department.Pro Tip: Pie charts are best used sparingly, and with no more than 7 slices per pie.
Tile map	Your source for geographic maps. Requires an elasticsearch geo_point field. More specifically, a field that is mapped as type:geo_point with latitude and longitude coordinates.
Vertical bar chart	The goto chart for oh-so-many needs. Great for time and non-time data. Stacked or grouped, exact numbers or percentages. If you are not sure which chart your need, you could do worse than to start here.

Combine visualizations in a Dashboard



Elastic stack. X-Pack, Cloud



Main Use Cases

1. Log data management and analysis

2. Monitoring of systems, networks and applications, including notifying operators of critical events

3. Collecting and analyzing other (massive) data:
   - Business data for business analytics
   - Energy management data / smart grids
   - Environmental data

UC#1: Log data management and analysis

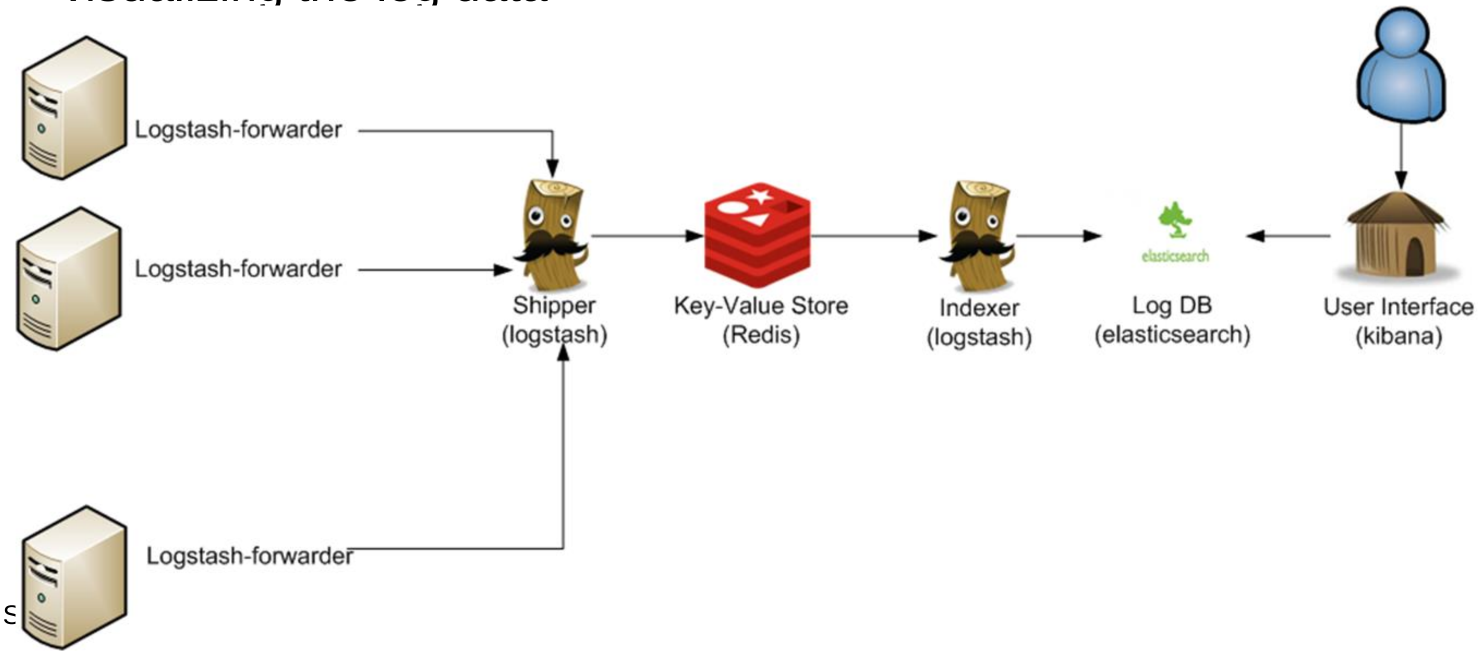
- Many different types of logs:
  - Application logs
  - Operating system logs
  - Network traffic logs from routers
  - ...
- Different goals for analysis:
  - Detect errors at runtime or while testing applications
  - Find and analyze security threats
  - Aggregate statistical data/metrics

UC#1: Problems of log data analysis

- **No centralization**
  - Log data can be dispersed (diff. servers or places in the same server)
- **Accessibility is difficult**
  - Logs can be difficult to find
  - Access to the server/device is often difficult
  - High expertise required to access logs on different platforms
  - Logs can be large and therefore difficult to copy
- **Poor consistency**
  - The structure of log entries is different for each app, system, or device
  - Specific knowledge is necessary for interpreting different log types
  - Variation in formats makes it challenging to search
    (for instance, different types of time formats)

UC#1:
How can the ELK stack help?

- Logstash allows to collect all log entries at a central place, such as Elasticsearch
  - End users do not need to know where the log files are located
  - Big log files are transferred continuously, in smaller chunks
- Log file entries can be transformed into harmonized event objects
- Easy access for end users via Web-based interfaces (e.g., Kibana)
- Elasticsearch / Kibana provides advanced functionality for analyzing and visualizing the log data



UC#2: Monitoring

- The ELK stack supports data monitoring and user alerting:
  - Logstash can check conditions on log file entries and calculate aggregated metrics
  - It may also, conditionally, send notification events to certain output plugins if predetermined monitoring criteria are met. For instance:
    - Forward notification event to email output plugin for notifying users
      (e.g., system operators) about the condition
    - Forward notification event to a dedicated monitoring application
  - Combined with Watcher (another product of Elastic), Elasticsearch can instrument arbitrary Elasticsearch queries, to be automatically performed periodically, to produce alerts and notifications
    - When the watch condition happens, predefined actions can be taken

Some examples of Log analysis:

- Logging and analyzing network traffic
  http://www.networkassassin.com/elk-stack-for-network-operations-reloaded/

- Using ELK to monitor performance
  http://logz.io/blog/elk-monitor-platform-performance/

- How Blueliv uses the ELK Stack for cybersecurity
  https://www.elastic.co/blog/how-blueliv-uses-the-elastic-stack-to-combat-cyber-threats

- Centralized System and Docker logging with ELK Stack
  http://www.javacodegeeks.com/2015/05/centralized-system-and-docker-logging-with-elk-stack.html

Further Reading

- ELK Stack : https://www.elastic.co/elastic-stack/
  - Software download and install
  - Several introductory and advanced training webinars
  - Detailed documentation of the available plugins

- Introductory Guides:
  - https://sematext.com/guides/elk-stack/
  - https://logz.io/learn/complete-guide-elk-stack/