# Data usage statistics and feedback tools

## Deliverable D4.3

## Contents

---

### Document History

| Ver. | Name | Date | Remark |
|---|---|---|---|
| v0.0 | Danilo Giacomi | 08.06.2015 | Initial draft |

### Document Information

- Deliverable Nr Title: D4.3 Data usage statistics and feedback tools
- Lead: Danilo Giacomi (NET7)
- Authors: Danilo Giacomi (NET7)
- Publication Level: Public

### Document Context Information

- Project (Title/Number): Fusepool P3 (609696)
- Work Package / Task: WP4 / T4.3
- Responsible person and project partner: Danilo Giacomi (NET7)

### Quality Assurance / Review

- 1st reviewer: Gabor Remenyi (Geox)
- 2nd reviewer: Carl Blakeley (OGL)

### Official Citations

Fusepool-P3-D4.2

### Copyright

### Acronyms and Abbreviations

| Acronym | Description |
|---|---|
| DoW | Description of Work |
| ELK | Elasticsearch Logstash Kibana |

### Links in this deliverable

| Title | URL |
|---|---|
| FP3 Community website | http://getfp3.com |
| SOD 2015 | http://www.spaghettiopendata.org/page/benvenut-sod15 |
| ELK | https://www.elastic.co/products |
| platform reference implementation | https://github.com/fusepoolP3/p3-platform-reference-implementation |
| logstash forwarder | https://github.com/elastic/logstash-forwarder |
| kibana on the sandbox | sandbox.fusepool.info:8387 |

| Title | URL |
|---|---|
| loggy | https://www.loggly.com/ |
| logentries | https://logentries.com/ |
| splunk | http://www.splunk.com/ |
| piwik | http://piwik.org/docs/log-analytics-tool-how-to/ |
| httpry format string | https://github.com/jbittel/httpry/blob/master/doc/format-string |
| Fusepool P3 Sandbox | sandbox.fusepool.info |

# Executive Summary

This document details the work performed and planned as part of Deliverable D4.3 / Task 4.3 "Data usage statistics and feedback".

# Introduction

The Description of Work for the Fusepool P3 project describes the main task of this deliverable as:

*"T4.3 – Data usage statistics and feedback: tools to assess the usage value of open data and services based on the automatic analysis of logging and auditing data [...] as well as user interfaces for directly posting feedbacks and requests ..."

The T4.3 task is then composed of two sub-tasks: gathering feedbacks from the users and analyse the services logs.

## Feedbacks

To let the end users give us feedbacks on the platform and the project, a feedback facility was added to the FP3 Community website (http://getfp3.com). The Jira issue regarding this task is found at https://fusepool.atlassian.net/browse/FPUSER-63 and was resolved around the end of March 2015.

It was wise to have a feedback facility in place at that time, as we have taken part to some public events where we've disseminated about the project - in particular the Spaghetti Open Data SOD2015 (http://www.spaghettiopendata.org/page/benvenut-sod15) where we've also ran a hackathon session to involve the community and to better share the knowledge about the Fusepool P3 platform.

During the SOD 2015 Hackathon we've also ran a challenge for creating applications using the data processed with the Fusepool P3 platform which has seen the partecipation of some groups of people and which ended up with the release of two applications. More information on this can be found on the Deliverable D4.2.

Feedbacks about the platform are also being requested to the stackeholders identified by the Work Package 6 in the form of a questionnaire sent to them, with instruction on how to perform the needed steps to set up the platform as well as on using it.

## Data usage statistics

To analyse the usage of our platform, and to offer a flexible and visual platform for that, several tools were evaluated, either in the form of commercial services or software or open-source tools.

At first we've tried some services, like Loggy (https://www.loggly.com/) or Logentries (https://logentries.com/), which are extremely good but they are expensive in the long run as they are sold for a significant monthly fee.

We've then scouted the available log analysis softwares, like splunk (http://www.splunk.com/) or piwik (http://piwik.org/docs/log-analytics-tool-how-to/) until we've learned about the ELK stack.

The Elasticsearch-Logstash-Kibana stack (ELK - https://www.elastic.co/products) is a largely used set of tools which not only offers a way to look into collected logs, but also some flexible ways to manage the logs themselves. The ELK stack is based on three core components, each of which offers its services to the stack:

### Elasticsearch

Elasticsearch is a search engine based on the Apache Lucene software, which provides a more useable and concise API, scalability, and operational tools on top of Lucene’s search implementation itself. Apache Lucene is a robust, extremely popular and proven piece of software. Also Apache SOLR - another largely used and well-known search engine - uses it. In the ELK stack Elasticsearch is used to store the logs and to quick retrieve them at visualization time.

### Logstash

Logstash is a data pipeline that helps process logs and other event data from a variety of systems, in the ELK stack it is used to process the logs and send the result of such a process to the Elasticsearch search engine.

Logstash has several plugins to process various type of logs and to interact with different other softwares to which it can send the processed logs. In the ELK stack Logstash is used to send processed logs to Elasticsearch where they are indexed and, later on, retrieved for visualization.

In the Fusepool P3 platform reference implementation we use the logs from the Httpry tool, which we expect containing the IP of the client and the request. From the IP we create a geographical field, using the "grok" filter in logstash, using the following configuration:

```
filter {

   grok {
    add_field => {
      "request"=> "%{request}"
      "client-ip" => "%{client}"
     }

      patterns_dir => "/etc/logstash/patterns"
      pattern => "%{HTTPRY}"


   }

   geoip {
            source => "client-ip"
            target => "geoip"
            add_field => [ "[geoip][coordinates]", "%{[geoip][longitude]}" ]
            add_field => [ "[geoip][coordinates]", "%{[geoip][latitude]}"  ]
         }
   mutate {
           convert => [ "[geoip][coordinates]", "float" ]
        }

}
```

In this configuration we use an external file for the HTTPRY pattern, which is what identify the log message in the log line, in this case HTTPRY is set as following :

```
%{IP:client}\t%{GREEDYDATA:request}
```

as we expect an Httpry log line to just contain an IP address and the request.

Then it creates a geographical field, with longitude and latitude starting from the IP address, in the second part of the file.

With this configuration, when a line like the following is written in the logfile:

```
66.XX.YY.13 /resource?genid=node19llk2qdpx20
```

the conversion generates something similar to the following, which is then used to create the entry in Elasticsearch:

```
{
       "message" => "66.XX.YY.13\t/resource?genid=node19llk2qdpx20",
      "@version" => "1",
    "@timestamp" => "2015-06-29T20:30:47.143Z",
          "type" => "syslog",
          "file" => "/var/log/httpry.log",
          "host" => "8df30498bb29",
        "offset" => "196567",
        "client" => "66.XX.YY.13",
       "request" => [
        [0] "/resource?genid=node19llk2qdpx20",
        [1] "/resource?genid=node19llk2qdpx20"
    ],
     "client-ip" => "66.XX.YY.13",
         "geoip" => {
                      "ip" => "66.XX.YY.13",
           "country_code2" => "US",
           "country_code3" => "USA",
            "country_name" => "United States",
          "continent_code" => "NA",
             "region_name" => "CA",
               "city_name" => "Mountain View",
                "latitude" => 37.385999999999996,
               "longitude" => -122.0838,
                "dma_code" => 807,
               "area_code" => 650,
                "timezone" => "America/Los_Angeles",
        "real_region_name" => "California",
                "location" => [
            [0] -122.0838,
            [1] 37.385999999999996
        ],
             "coordinates" => [
            [0] -122.0838,
            [1] 37.385999999999996
        ]
    }
}
```

All those fields are available in Kibana, and as you can see they contain several geographical information on the client, starting just from the IP address.

### Kibana

Kibana is the third and last element of the ELK stack and works with Elasticsearch from which it retrieves the data. Kibana is the analytics and visualization tool of the ELK stack that lets you search, view, and interact with data stored in Elasticsearch indices.

It offers several aggregation tools, search

### Logstash forwarder and Httpry

Another piece of software is needed to make the whole logging system work, which takes the logs and hands them to the ELK stack. Such software is logstash forwarder (https://github.com/elastic/logstash-forwarder), a linux daemon programmed to scan for changes on selected logs files and configured to send all new lines to the Logstash in the ELK stack. Logstash forwarder has been configured to work with another commonly available logging system, Httpry (https://github.com/jbittel/httpry), which is able to collect and log all the accesses to the server it's run on. Httpry can be ran specifying the ports of the request to log, the fields to log, and the file into witch the logs will be saved. By using the httpry tool, and with the flexibility of logstash in analyse the log entries, we were able to collect and log in a single place all the access to the various services the Fusepool P3 Platform runs. In fact, at the time of this writing, we have twenty entry points to the Fusepool P3 Platform, each of which listening to a different port, and to collect the logs of each service behind those entry points was a critical aspect. Also because we expected each service to use different log notation, or even not log enough information or nothing at all. Httpry, on the contrary, listens to all the requests regardless of which service will the serve each of them, and write a log line about that. Those log lines are stored in a file which is the one watched by the logstash forwarder tool, which has been configured to understand the log message httpry creates, with the configuration we've created for it.

In the platform reference implementation docker image (https://github.com/fusepoolP3/p3-platform-reference-implementation) Httpry has been configured to run as a daemon, listening on the ports of all the services available in the platform itself. It logs the requested URL (relative to the host) as well as the IP address of the client who has performed such request.

The Httpry tool has the ability to log several fields from the incoming request, a list of which is available at https://github.com/jbittel/httpry/blob/master/doc/format-string, with the possibility to log fields either from the HTTP request body or outside of it. At this time we are exploring the tool looking for other interesting fields to log and, in the end, use for our analysis.

### Fusepool P3 platform reference implementation docker image

In the Fusepool P3 platform reference implementation docker image (https://github.com/fusepoolP3/p3-platform-reference-implementation) the ELK stack is distributed as a docker compose multi-container application, setting up and running the three core components, Elasticsearch, logstash and kibana. It has been configured with a ssl certificate/key couple created using "localhost" as the hostname, specifically with the following command:

```
openssl req -x509 -batch -nodes -newkey rsa:2048 -keyout logstash-forwarder.key -out
logstash-forwarder.crt -days 3650 -subj /CN=localhost
```

This created a ssl certificate/key couple valid ten years and to be used on a machine called "localhost"

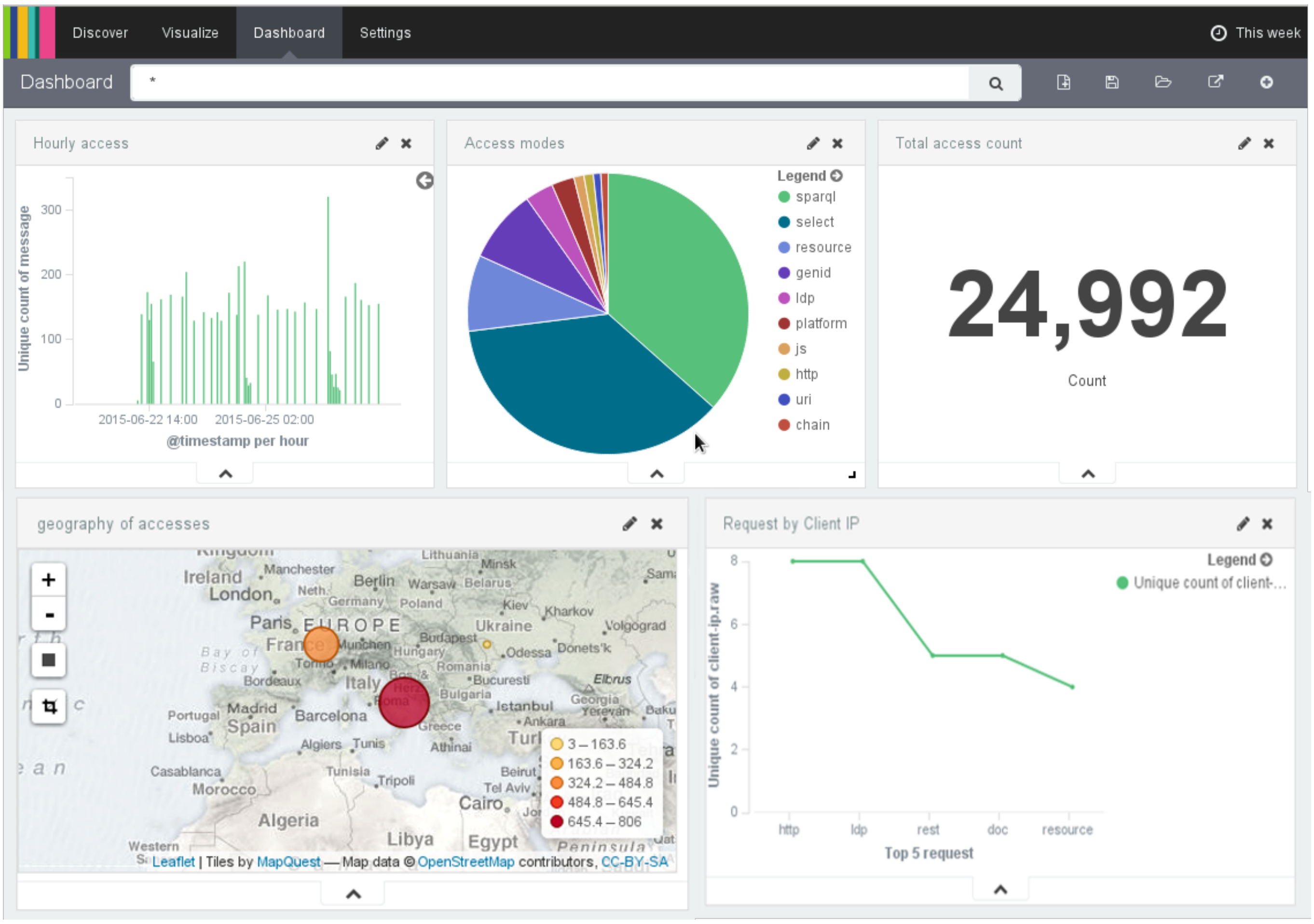
Then, in the platform reference implementation, the Httpry is ran at startup time along with logstash forwarder which has been configured to watch the Httpry log file and send everything written there to the Logstash, in the ELK stack. The startup command for Httpry, as found in the platform reference implementation is the following:

```
httpry -f source-ip,request-uri -d -i eth0 'tcp port 8080 or 8181 or 8151 or 8200 or
8201 or 8202 or 8203 or 8204 or 8205 or 8300 or 8301 or 8302 or 8303 or 8304 or 8305 or
8306 or 8307 or 8308 or 8310 or 8386' -o /var/log/httpry.log
```

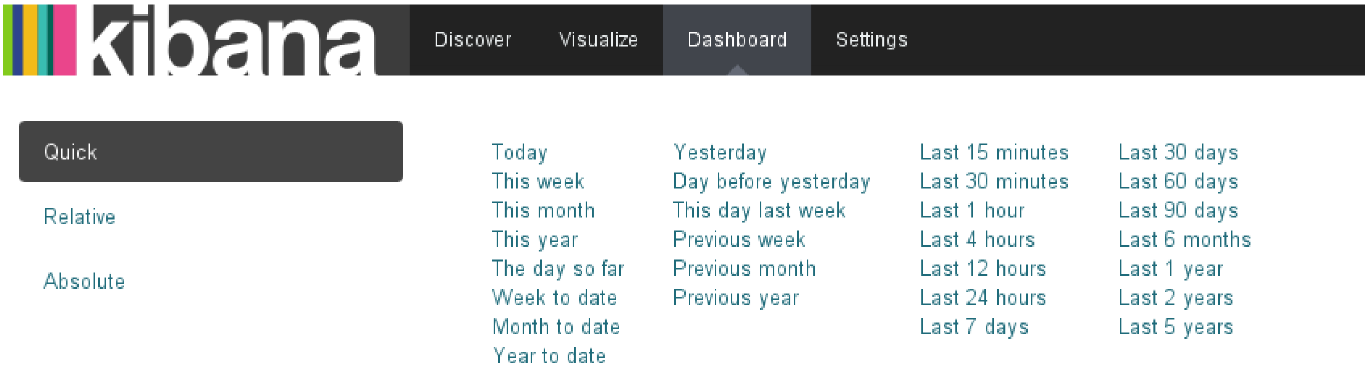
which tells the Httpry service to watch the eth0 interface for accesses on the port listed, logging the request URI and the IP of the user. With those information we can understand which services are accessed and also recognise the geographic zone generating each request, based on the client IP address (the source-ip field).

### Logs analysis example

The logs gathered with the tools just described are visualized using the Kibana tool, which offer several ways to observe the data. The Fusepool sandbox contains all the tools and can be used as an example of what can be achieved with the tools. A starting point can be the Dashboard - http://sandbox.fusepool.info:8387/#/dashboard/Dashboard - which has been configured to show three widgets, as shown in the following images



The image shows a view on the logs collected in the last week, the time frame can be adjusted by clicking on the top-right corner, on the "This week" writing. It opens up a panel with several possibility, clicking on them the widgets will be updated with the logs from the selected time frame. The following image shows them.



The first widget from the left on the top row, "Hourly access", shows the access in the selected time frame, grouped by hour. This, as we'll see, can be adjusted to the needs in the creation of the widget, in the "Visualize" panel.

The second widget on the top row, "Access modes", the pie chart, shows the top ten access modes taken from the logged request. The legend shows the service requested, moving the mouse over them, the related slice of the chart is highlighted. The same happens when the mouse is over a slice, the related legend entry is highlighted.

The last widget on the top row, "Total access count", shows to total access count as a number.

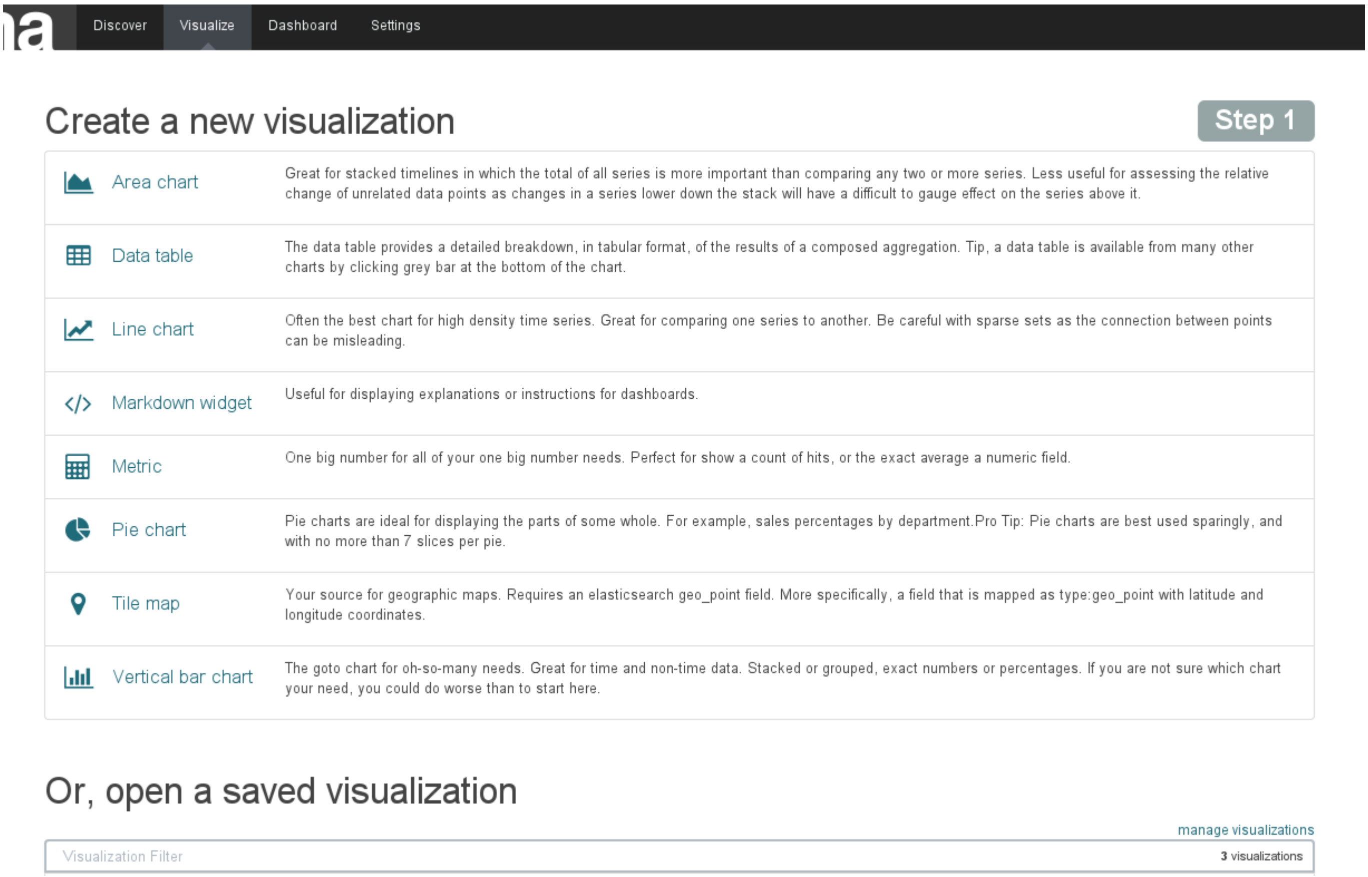
On the second row a map widget, on the left, shows the geographical zone from where each access came. Each point is colored and sized to show how many accesses were coming from that zone.

The last widget, on the right on the bottom row shows the top terms of the requests, with the number of requests for each of them.

These five widgets are just example of what can be done with the tool, but represent interesting facts about the platform and its usage.

By clicking on the "Visualize" tab, in the top menu, it is possible to create new widgets or to view and modify an existing one. In this visualization the widget is larger and can be studied in details.

The following image shows all the available type of visualization, and thus dashboard widgets, the user can choose from



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. |

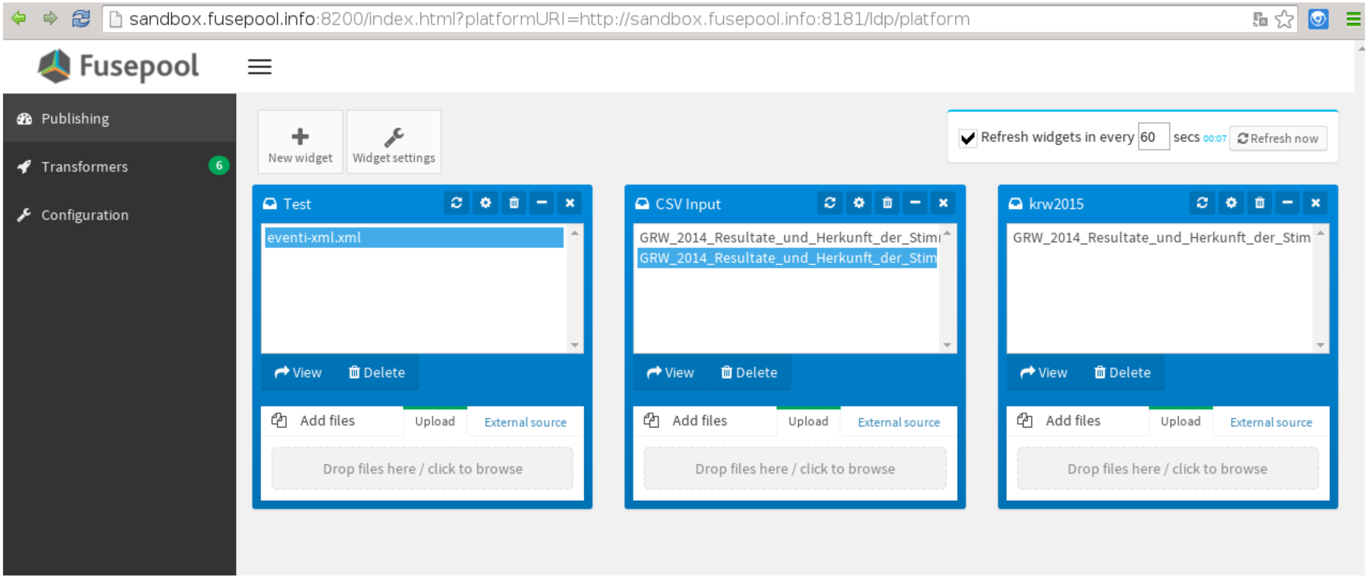
Or, open a saved visualization

manage visualizations

Visualization Filter

3 visualizations

Different searches can also be performed in Kibana, for example by filtering on the content of the requests. For instance, by accessing the Platform Dashboard (http://sandbox.fusepool.info:8200/index.html?platformURI=http://sandbox.fusepool.info:8181/ldp/platform) at the time of this writing, we see there are three widgets used for importing some data. The following image shows the situation at present time.



sandbox.fusepool.info:8200/index.html?platformURI=http://sandbox.fusepool.info:8181/ldp/platform

☑ Refresh widgets in every 60 secs

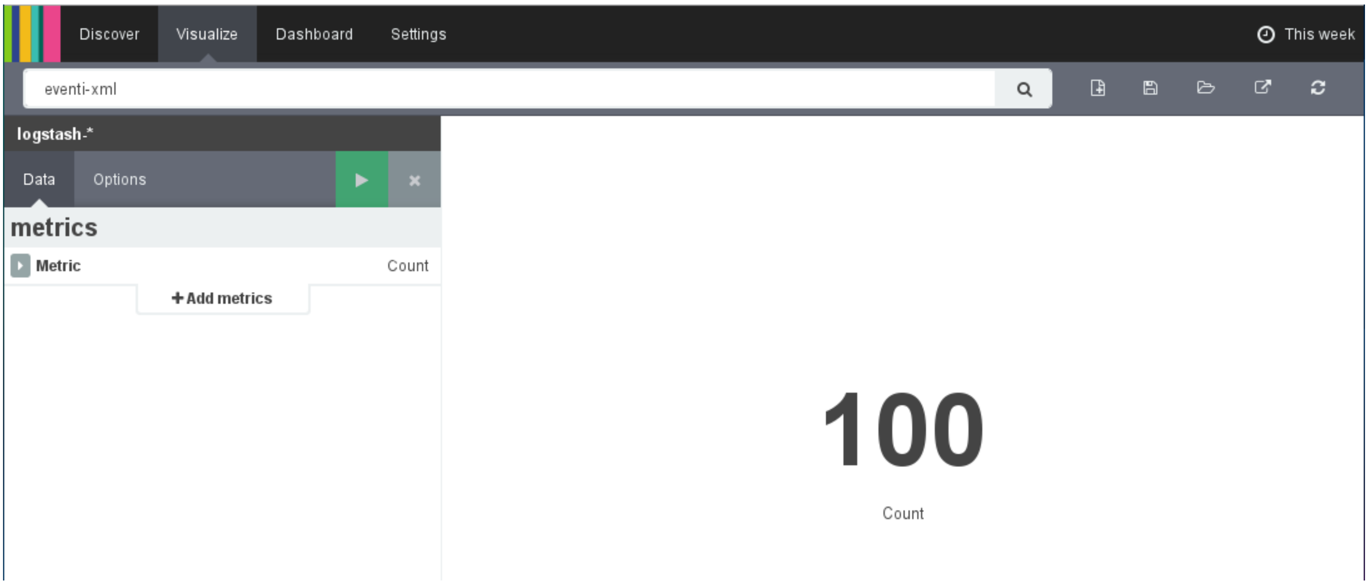
Test: eventi-xml.xml

CSV Input: GRW_2014_Resultate_und_Herkunft_der_Stim, GRW_2014_Resultate_und_Herkunft_der_Stim

krw2015: GRW_2014_Resultate_und_Herkunft_der_Stim

We can see that there is a file called eventi-xml.xml, presumably a file containing a set of events data. Let's create a widget for checking the number of accesses to that file in the past week. From the "Visualize" page we've seen before, we chose the type of visualization we're interested into, for example "Metric" which will show us just the number of matching entries. We choose, in the following page, to create a new search, and we land in a page where a number is already shown: the number of log lines indexed in the selected time frame (in the top-right corner, as we've seen you can select the time frame).

The central textbox, containig just an asterisk is the filter text field, there we'll write the text we want to filter on, in our case we'll write "eventi-xml" and we'll choose "This week" as a time frame, the result is shown in the following image:



eventi-xml

This week

logstash-*

Metric: Count

100
Count

The text filter can be applied to any type of visualization available in Kibana, which in this way offer a powerful way to cross the filters and compare analysed data.

Lastly a visualization prepared as we've just seen can be saved for later re-use, or even for adding it to the dashboard.

### Conclusion and Future Work

What has been shown so far in this document represent some of the possibilities tools offer, identified as interesting for the analysis of the platform usage. The configurations of the Httpry software and the logstash tranformations are still in progress as we will need to fine tune the logged fields and the visualization widgets in order to show intersting facts.

We already have enough information to start understanding the potentials of the tools and to learn how the platform is being accessed and thus how our data are being used. The way all these data are visually shown offers an easy-to-understand layer which can be analysed to different kind of people, from technicians looking for studying the statistic of the accesses over the day hours, to data owners for monitoring how their data are used, who are using them (by showing the geographical zone from where the resources came from) etc.

All the tools discussed in the present document were added to the platform reference implementation only recently, the project partners haven't had enough time at this to study them and to give suggestions about what could be interesting to have, which fields would add valuable information, what visualizations may expresses interesting information and so on so forth.

In the upcoming months we'll let the partners experiment with it and once they've learned what you can get from those tools, we expect to further work on the fine tuning of the configurations to add additional fields, custom transformations and, in the end, to expose more interesing information.