ELK stack & log parsing

TOMMASO DIOTALEVI
Ph.D. Student in Physics XXXIV Cycle (Academic Year 2018/19)

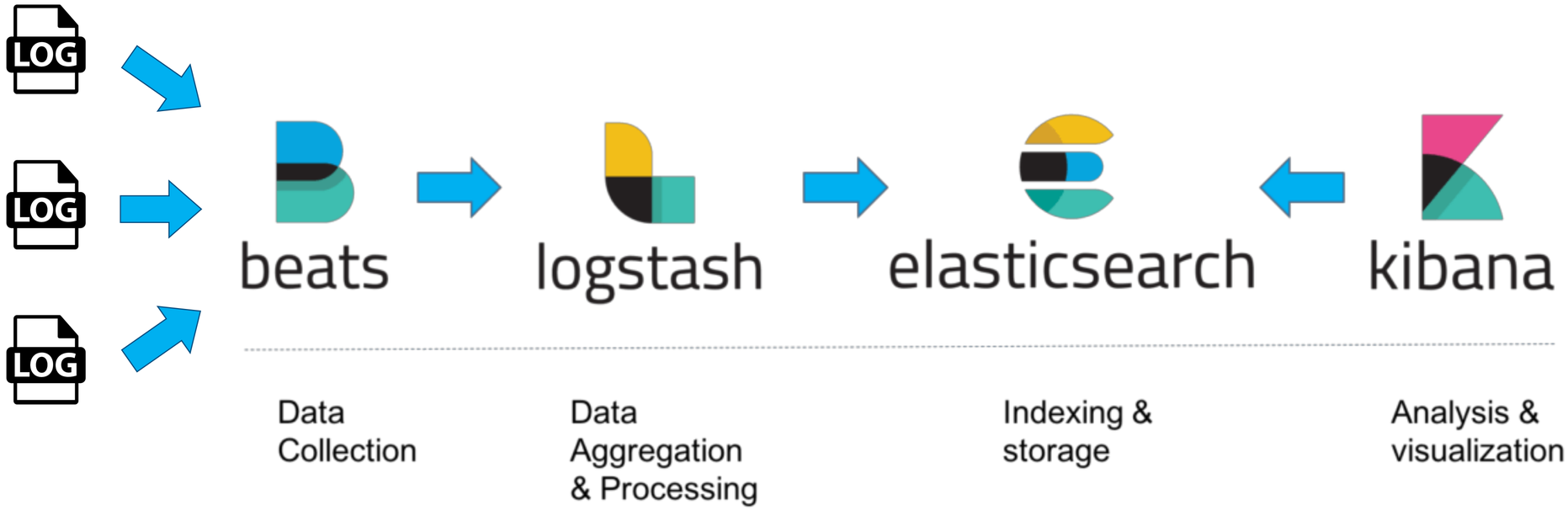
INFN-Bologna and University of Bologna

Goals

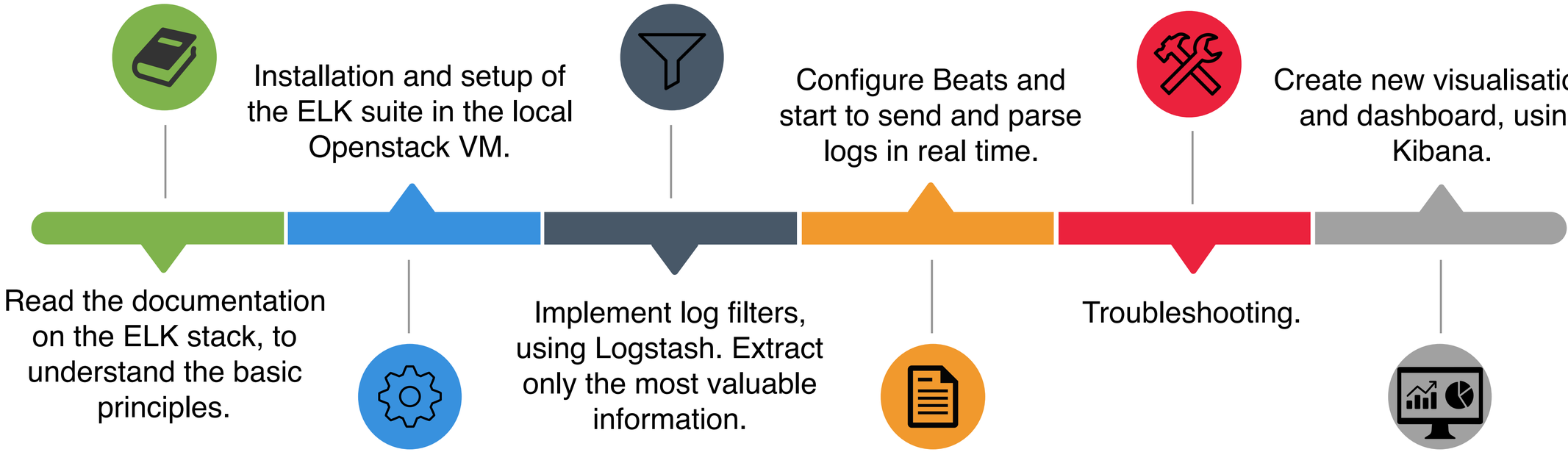
- Collection of logs (StoRM and Gridftp) coming from different machines @CNAF Tier-1.
- Data wrangling the information coming from such logs, using the ELK Stack suite.
- Create new visualisations and dashboards.

This will lay the groundwork for the application of *Machine Learning* algorithms, to make predictions of possible malfunctions and critical failures, improving the overall system maintenance.

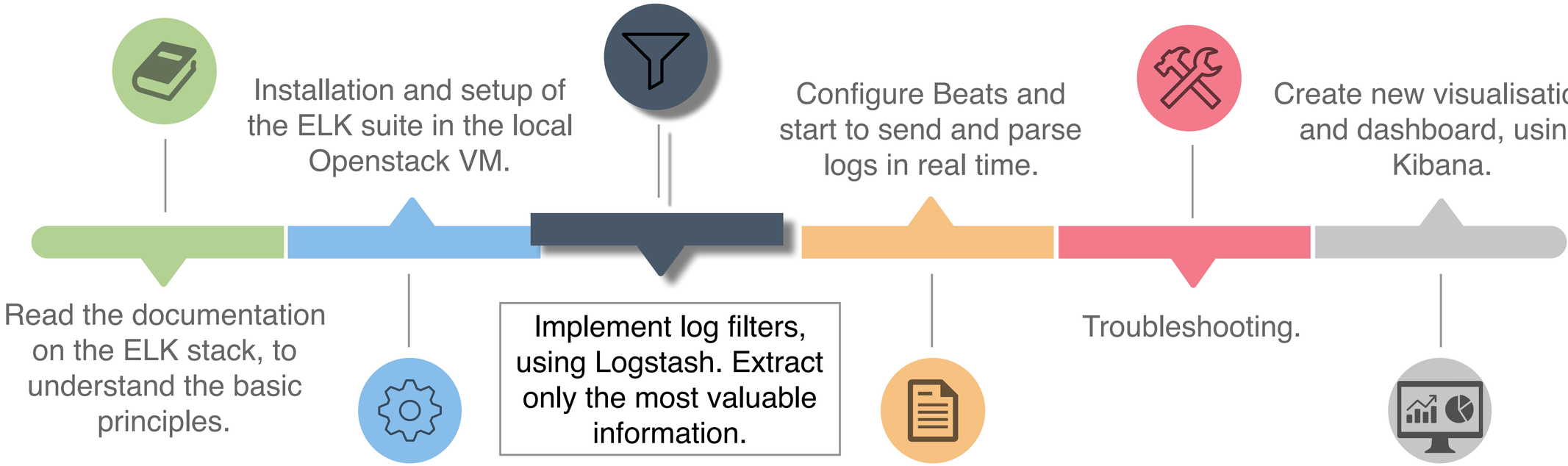
The ELK stack



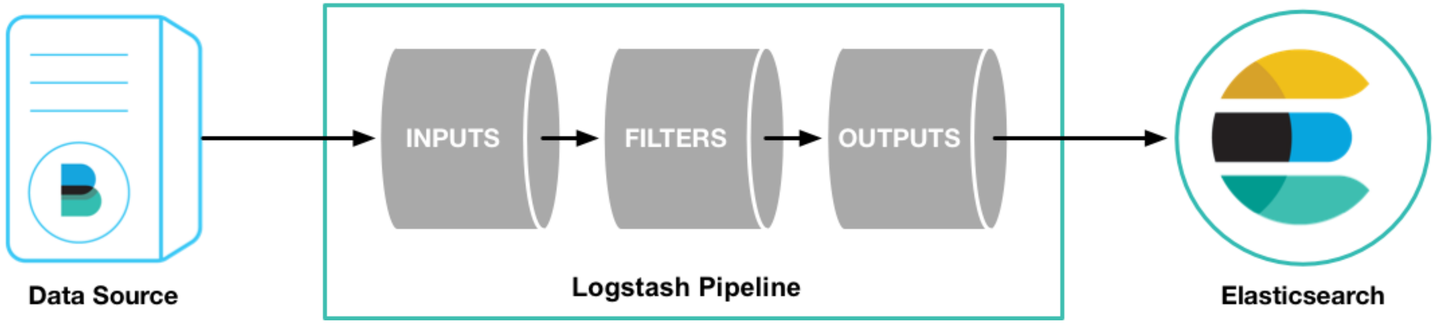
Timeline


Read the documentation on the ELK stack, to understand the basic principles.
Installation and setup of the ELK suite in the local Openstack VM.
Implement log filters, using Logstash. Extract only the most valuable information.
Configure Beats and start to send and parse logs in real time.
Troubleshooting.
Create new visualisations and dashboard, using Kibana.

Timeline


Read the documentation on the ELK stack, to understand the basic principles.
Installation and setup of the ELK suite in the local Openstack VM.
Implement log filters, using Logstash. Extract only the most valuable information.
Configure Beats and start to send and parse logs in real time.
Troubleshooting.
Create new visualisations and dashboard, using Kibana.

Parse logs using Logstash



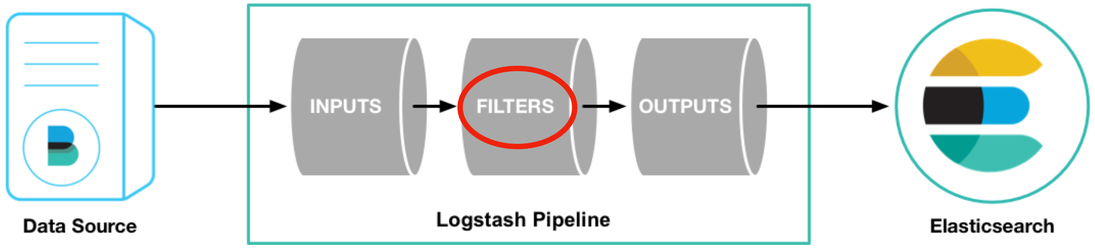
Inside the local cluster, Logstash creates a well defined pipeline.

- Input configuration that collects data from Filebeats in a continuous live-feed streaming.
- Filter configuration required for parsing each event, identify named fields to build a user defined structure.
- Output configuration to route parsed data in a search analytics engine (Elasticsearch).

Parse logs using Logstash

The different choice of filters for a correct parsing of log data was the biggest part of this project.

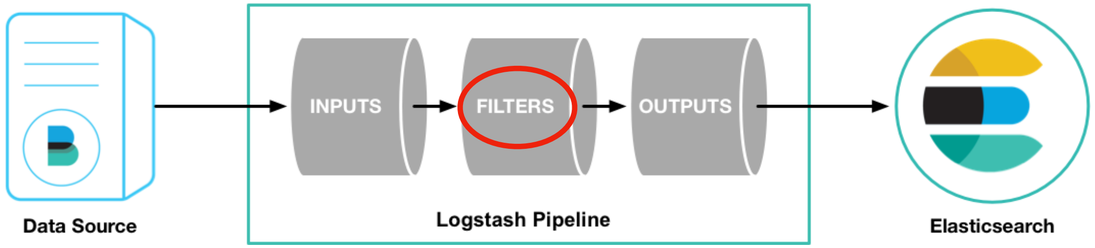
A large amount of them was parsed using the ***grok*** filter.



Parse logs using Logstash

The different choice of filters for a correct parsing of log data was the biggest part of this project.

A large amount of them was parsed using the ***grok*** filter.



A grok filter, based on Regular Expressions, is adopted to match specific portions of log entries by creating a series of pattern defined as follows:

%{SYNTAX:SEMANTIC}

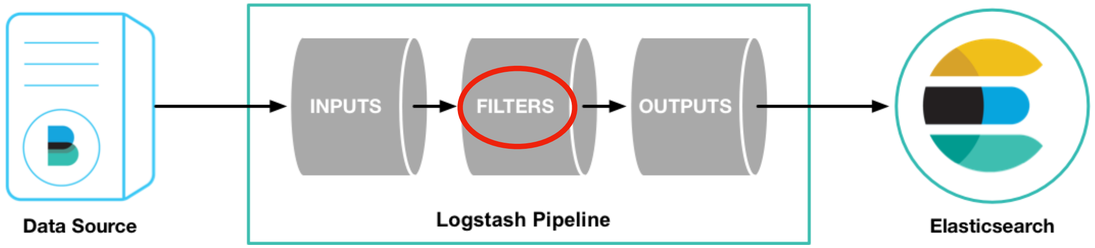
where SYNTAX is the name of the pattern that will match the text, while the SEMANTIC is the identifier of the piece of text being matched.

```
match => { "message" => "%{IP_EMB:clientIP}"}
```

Parse logs using Logstash

The different choice of filters for a correct parsing of log data was the biggest part of this project.

A large amount of them was parsed using the ***grok*** filter.



A grok filter, based on Regular Expressions, is adopted to match specific portions of log entries by creating a series of pattern defined as follows:

%{SYNTAX:SEMANTIC}

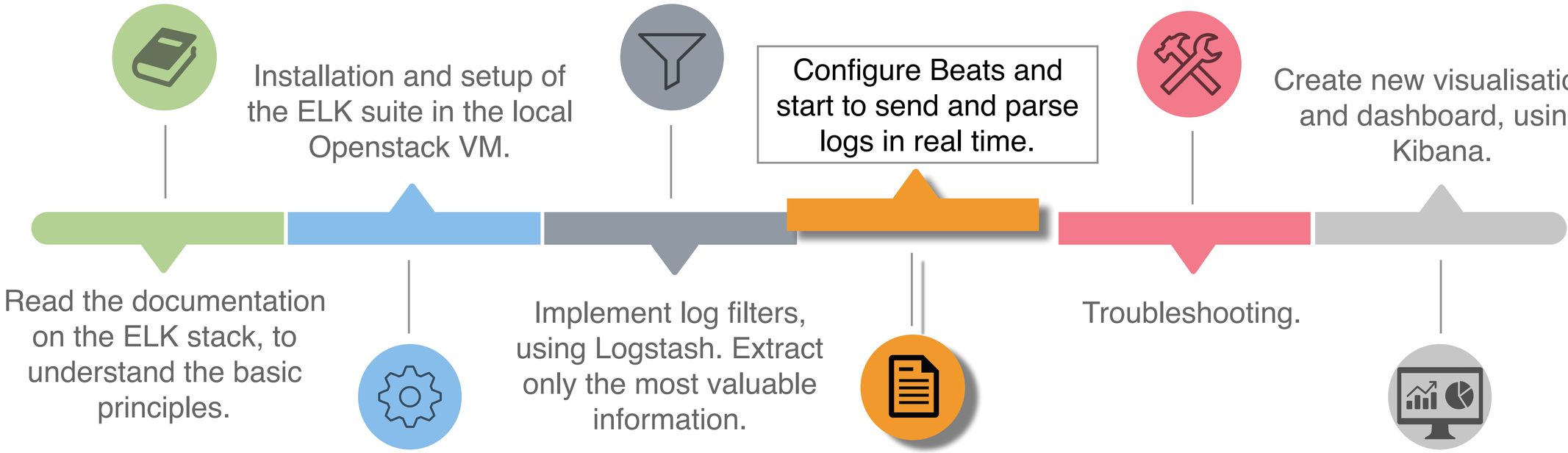
where SYNTAX is the name of the pattern that will match the text, while the SEMANTIC is the identifier of the piece of text being matched.

```
match => { "message" => "%{IP_EMB:clientIP}"}
```

Several patterns are predefined e.g. IP, DATE, TIME. However, ***custom patterns*** are required in order to match every possible scenario. (Such patterns are stored in a specific file).

```
IP_EMB ::(ffff(:0{1,4}){0,1}:){0,1}((25[0-5]|(2[0-4]|1{0,1}[0-9]){0,1}[0-9])\.){3,3}(25[0-5]|(2[0-4]|1{0,1}[0-9]){0,1}[0-9])|%{IP}
```

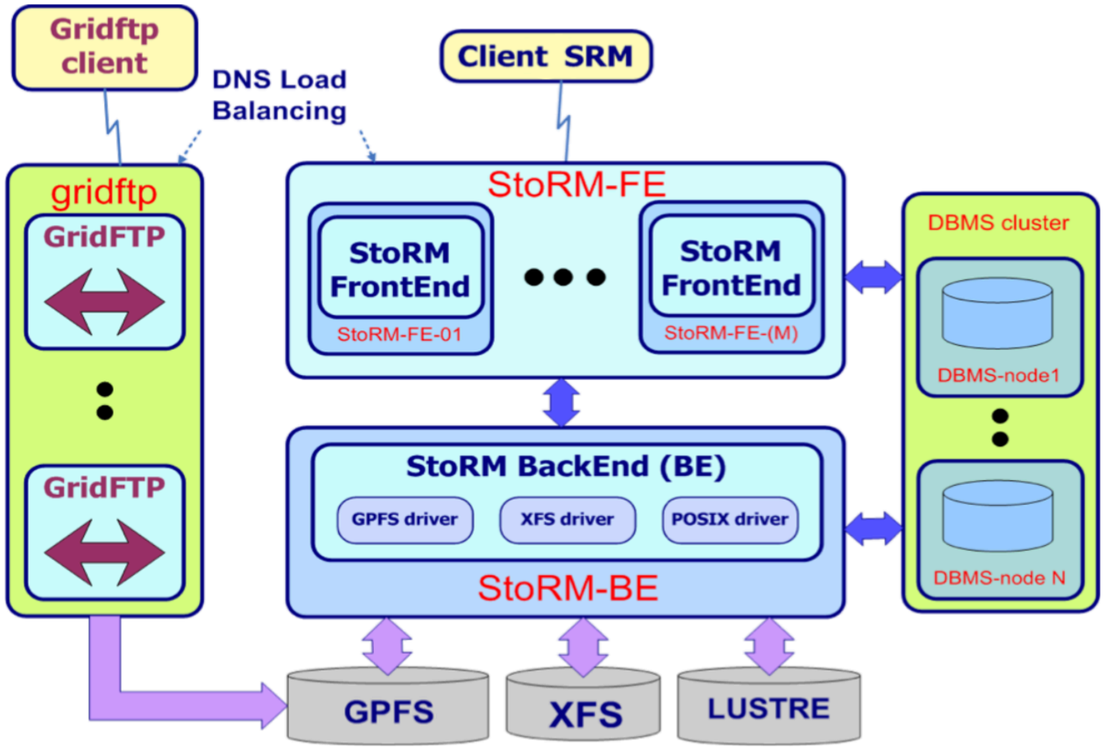
Timeline


Read the documentation on the ELK stack, to understand the basic principles.
Installation and setup of the ELK suite in the local Openstack VM.
Implement log filters, using Logstash. Extract only the most valuable information.
Configure Beats and start to send and parse logs in real time.
Troubleshooting.
Create new visualisations and dashboard, using Kibana.

Types of log parsed

Using Filebeats, several logs were parsed, coming from different machines.

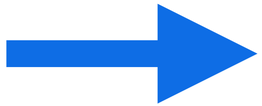
- *storm-atlas*
  - storm-frontend-server.log
  - storm-backend.log
  - heartbeat.log
  - monitoring.log
- *storm-fe-atlas-07*
  - storm-frontend-server.log
  - monitoring.log
- *ds-808 & ds-908*
  - storm-gridftp-session.log



Types of log parsed

Using Filebeats, several logs were parsed, coming from different machines.

- *storm-atlas*
  - storm-frontend-server.log
  - storm-backend.log
  - heartbeat.log
  - monitoring.log
- *storm-fe-atlas-07*
  - storm-frontend-server.log
  - monitoring.log
- *ds-808 & ds-908*
  - storm-gridftp-session.log



All with a different structure and formalism!

Example of parsed log, with new structured information:

🕐	@timestamp	November 15th 2018, 18:25:06.478
t	@version	1
t	_id	gzpnGGcBcvwUa1jlsGXn
t	_index	filebeat-2018.11.15
#	_score	-
t	_type	doc
t	action	srmReleaseFiles
t	beat.hostname	storm-atlas.cr.cnaf.infn.it
t	beat.name	storm-atlas.cr.cnaf.infn.it
t	beat.version	6.4.2
t	clientDN	/DC=ch/DC=cern/OU=Organic Units/OU=Users/CN=atlpilo1/CN=614260/CN=Robot: ATLAS Pilot1
t	host.name	storm-atlas.cr.cnaf.infn.it
t	input.type	log
t	message	18:25:06.478 - INFO [xmlrpc-488926] - srmReleaseFiles: user </DC=ch/DC=cern/OU=Organic Units/OU=Users/CN=atlpilo1/CN=614260/CN=Robot: ATLAS Pilot1> operation on [SURL: srm://storm-fe.cr.cnaf.infn.it/atlas/atlasdatadisk/rucio/data15_13TeV/85/6e/AOD.11227506._001507.pool.root.1] succesfully done with: [status: SRM_SUCCESS: Released]
#	offset	404,176,017
t	prospector.type	log
t	result	SRM_SUCCESS
t	source	/var/log/storm/storm-backend.log
t	status	INFO
t	surl	srm://storm-fe.cr.cnaf.infn.it/atlas/atlasdatadisk/rucio/data15_13TeV/85/6e/AOD.11227506._001507.pool.root.1
t	tags	beats_input_codec_plain_applied, _grokparsefailure
t	timestamp	2018-11-15 18:25:06.478
t	token	xmlrpc-488926

Example of parsed log, with new structured information:

🕒	@timestamp	November 15th 2018, 18:25:06.478
t	@version	1
t	_id	gzpnGGcBcvwUa1jlsGXn
t	_index	filebeat-2018.11.15
#	_score	-
t	_type	doc
t	action	srmReleaseFiles
t	beat.hostname	storm-atlas.cr.cnaf.infn.it
t	beat.name	storm-atlas.cr.cnaf.infn.it
t	beat.version	6.4.2
t	clientDN	/DC=ch/DC=cern/OU=Organic Units/OU=Users/CN=atlpilo1/CN=614260/CN=Robot: ATLAS Pilot1
t	host.name	storm-atlas.cr.cnaf.infn.it
t	input.type	log
t	message	18:25:06.478 - INFO [xmlrpc-488926] - srmReleaseFiles: user </DC=ch/DC=cern/OU=Organic Units/OU=Users/CN=atlpilo1/CN=614260/CN=Robot: ATLAS Pilot1> operation on [SURL: srm://storm-fe.cr.cnaf.infn.it/atlas/atlasdatadisk/rucio/data15_13TeV/85/6e/AOD.11227506._001507.pool.root.1] succesfully done with: [status: SRM_SUCCESS: Released]
#	offset	404,176,017
t	prospector.type	log
t	result	SRM_SUCCESS
t	source	/var/log/storm/storm-backend.log
t	status	INFO
t	surl	srm://storm-fe.cr.cnaf.infn.it/atlas/atlasdatadisk/rucio/data15_13TeV/85/6e/AOD.11227506._001507.pool.root.1
t	tags	beats_input_codec_plain_applied, _grokparsefailure
t	timestamp	2018-11-15 18:25:06.478
t	token	xmlrpc-488926

→ Original message (remains in the log document)

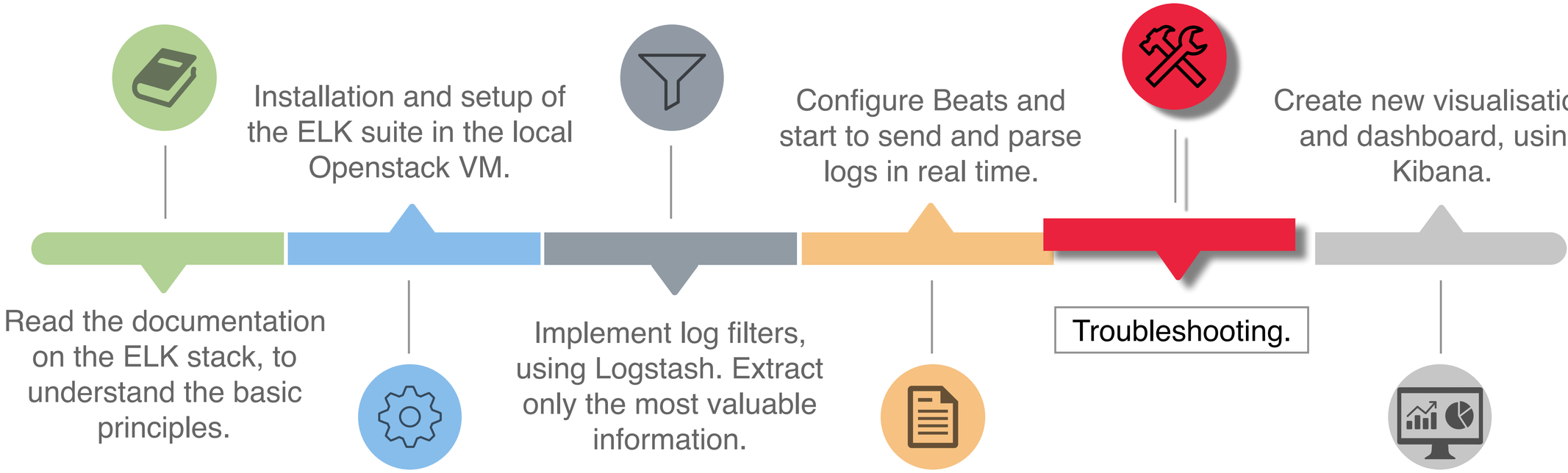
Example of parsed log, with new structured information:

🕒	@timestamp	November 15th 2018, 18:25:06.478
t	@version	1
t	_id	gzpnGGcBcvwUa1jlsGXn
t	_index	filebeat-2018.11.15
#	_score	-
t	_type	doc
t	action	srmReleaseFiles
t	beat.hostname	storm-atlas.cr.cnaf.infn.it
t	beat.name	storm-atlas.cr.cnaf.infn.it
t	beat.version	6.4.2
t	clientDN	/DC=ch/DC=cern/OU=Organic Units/OU=Users/CN=atlpilo1/CN=614260/CN=Robot: ATLAS Pilot1
t	host.name	storm-atlas.cr.cnaf.infn.it
t	input.type	log
t	message	18:25:06.478 - INFO [xmlrpc-488926] - srmReleaseFiles: user </DC=ch/DC=cern/OU=Organic Units/OU=Users/CN=atlpilo1/CN=614260/CN=Robot: ATLAS Pilot1> operation on [SURL: srm://storm-fe.cr.cnaf.infn.it/atlas/atlasdatadisk/rucio/data15_13TeV/85/6e/AOD.11227506._001507.pool.root.1] succesfully done with: [status: SRM_SUCCESS: Released]
#	offset	404,176,017
t	prospector.type	log
t	result	SRM_SUCCESS
t	source	/var/log/storm/storm-backend.log
t	status	INFO
t	surl	srm://storm-fe.cr.cnaf.infn.it/atlas/atlasdatadisk/rucio/data15_13TeV/85/6e/AOD.11227506._001507.pool.root.1
t	tags	beats_input_codec_plain_applied, _grokparsefailure
t	timestamp	2018-11-15 18:25:06.478
t	token	xmlrpc-488926

→ Timestamp, in a date specific format.

→ Original message (remains in the log document)

Timeline


Read the documentation on the ELK stack, to understand the basic principles.
Installation and setup of the ELK suite in the local Openstack VM.
Implement log filters, using Logstash. Extract only the most valuable information.
Configure Beats and start to send and parse logs in real time.
Troubleshooting.
Create new visualisations and dashboard, using Kibana.

Troubleshooting

Multiline Parsing

An error occurred during the parsing of multiline log messages, such as Java Stack traces. In an atypical day of logs, several multiline entires were produced as the result of a debug operation. These messages, were not parsed correctly into Logstash since it was initially configured to consider a single log but rather a line ending with a break-line.

Troubleshooting

Multiline Parsing

An error occurred during the parsing of multiline log messages, such as Java Stack traces. In an atypical day of logs, several multiline entires were produced as the result of a debug operation. These messages, were not parsed correctly into Logstash since it was initially configured to consider a single log but rather a line ending with a break-line.

Truncated Logs

An issue concerning StoRM logs, is the unexpected truncation of certain logs, in order to preserve disk space. Since this problem is not solvable at parse level, the solution adopted was the removal of incomplete information and leave out all the rest.

Troubleshooting

Multiline Parsing

An error occurred during the parsing of multiline log messages, such as Java Stack traces. In an atypical day of logs, several multiline entires were produced as the result of a debug operation. These messages, were not parsed correctly into Logstash since it was initially configured to consider a single log but rather a line ending with a break-line.

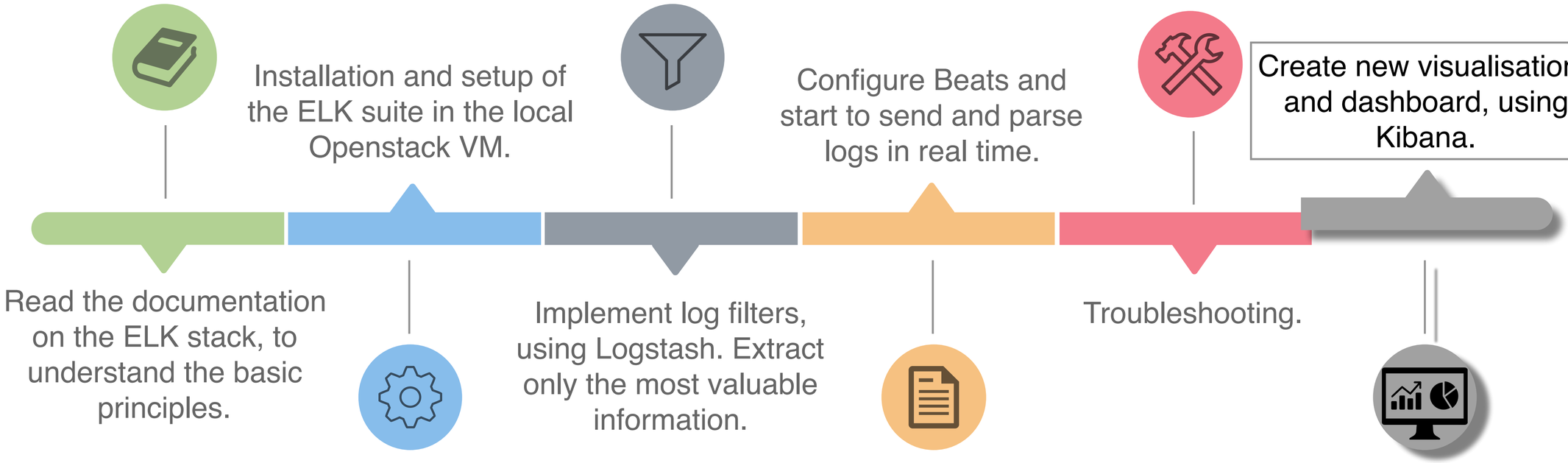
Truncated Logs

An issue concerning StoRM logs, is the unexpected truncation of certain logs, in order to preserve disk space. Since this problem is not solvable at parse level, the solution adopted was the removal of incomplete information and leave out all the rest.

Log sharing

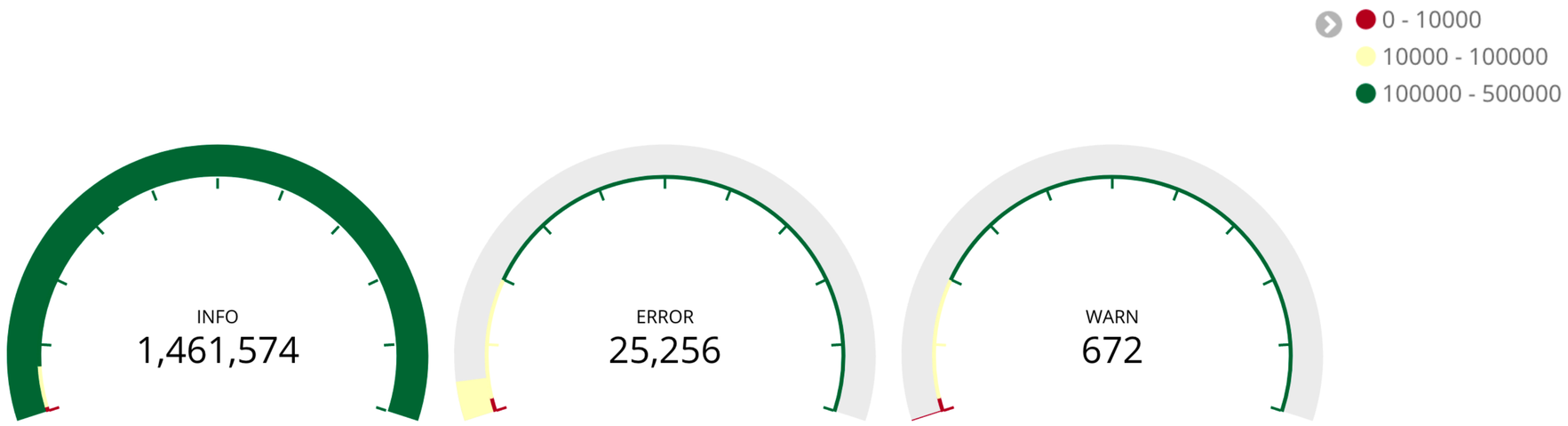
The production of CSV files with information coming live from the machines was important also for others in the work group. In fact, using such flat files, it is possible to load them into high-level classification libraries and start to think at possible learning models. Solved using an external tool specific for such conversion, called *es2csv*.

Timeline


Read the documentation on the ELK stack, to understand the basic principles.
Installation and setup of the ELK suite in the local Openstack VM.
Implement log filters, using Logstash. Extract only the most valuable information.
Configure Beats and start to send and parse logs in real time.
Troubleshooting.
Create new visualisations and dashboard, using Kibana.

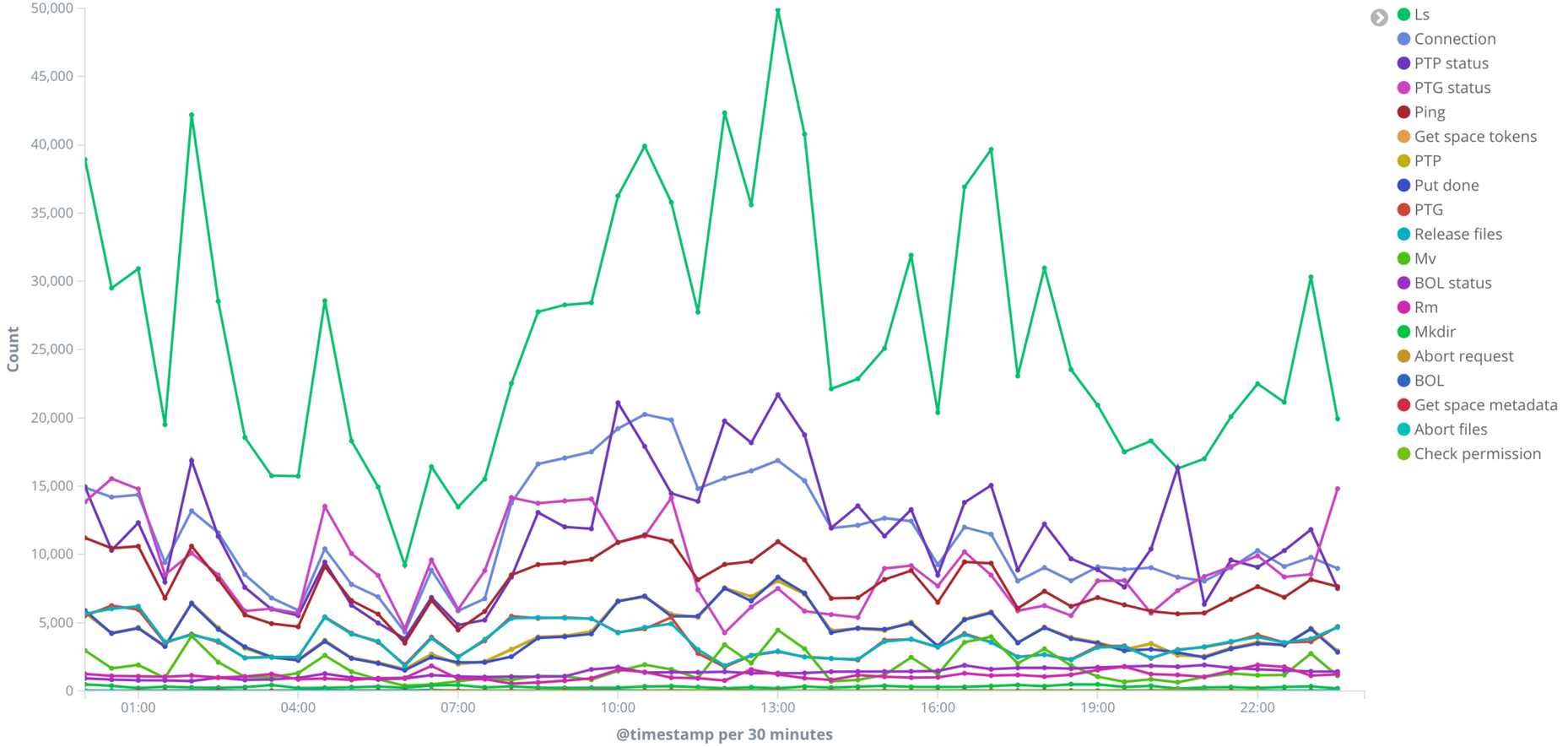
Visualize data

Using Kibana User Interface, it is possible to create new visualisations and collect them to form new dashboards.
Example: 1 day of logs (25th of November 2018 - local time).

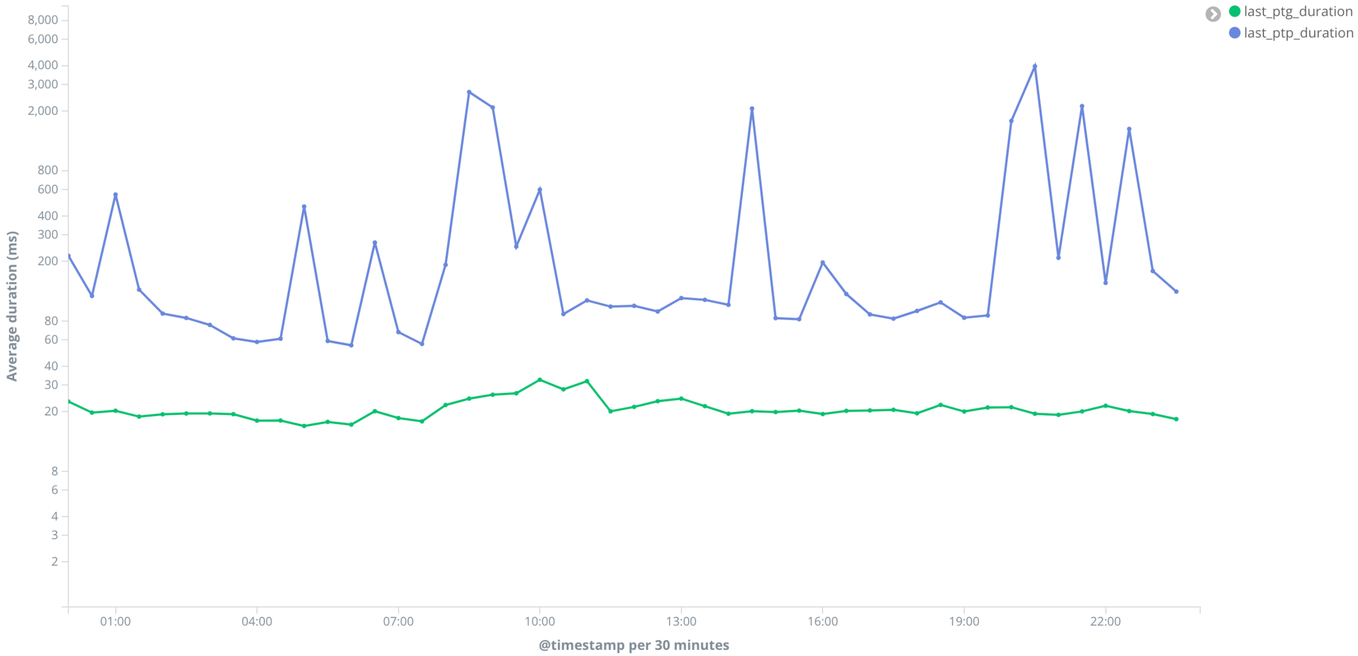


Count of INFO, ERROR and WARN logs from the StoRM Back-End machine.

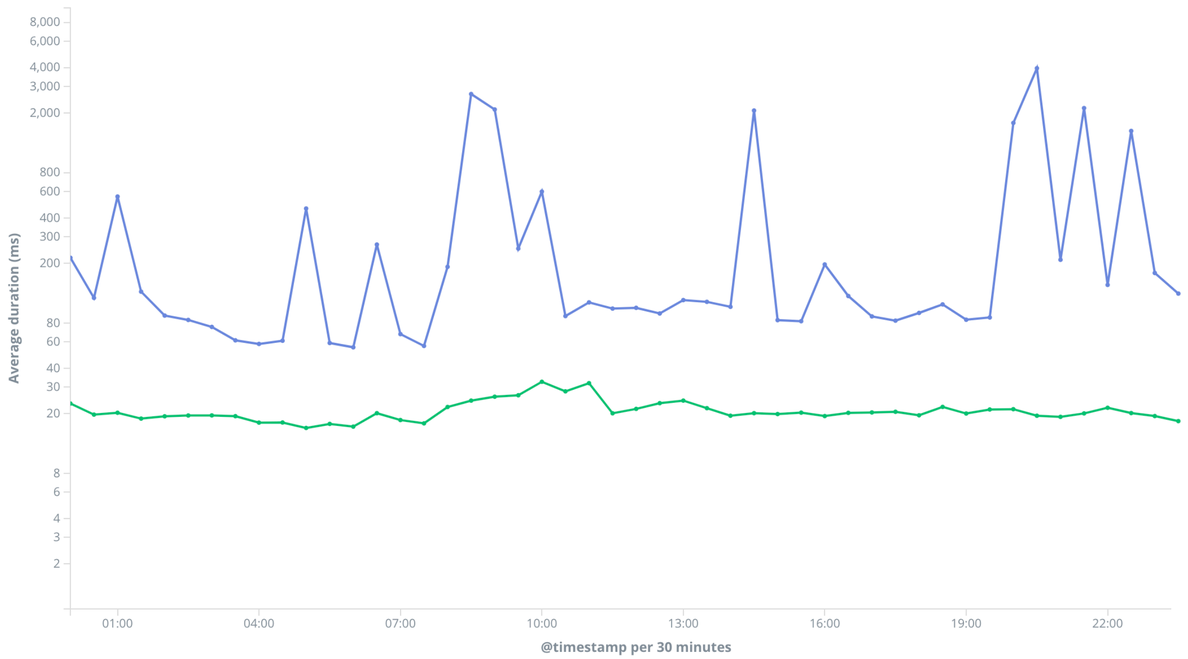
Visualize data



Count of different requests for the StoRM Front-End. (the Back-End similar plot is not shown.)

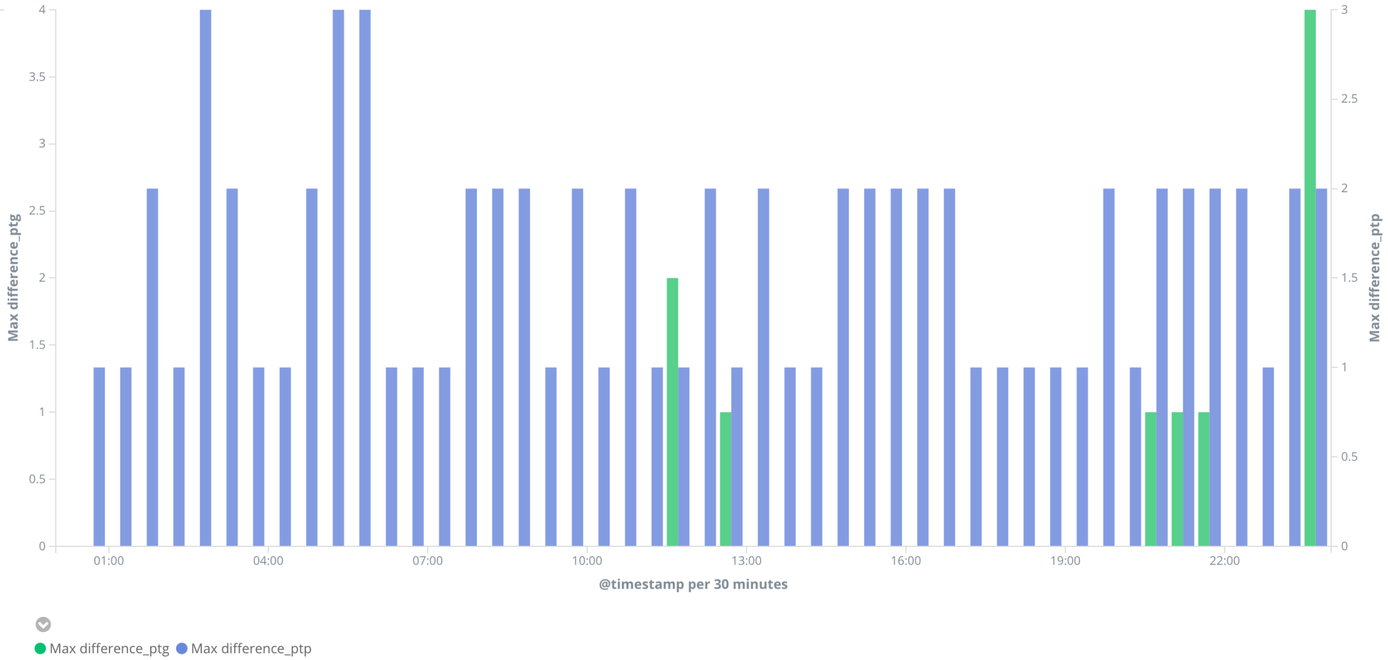


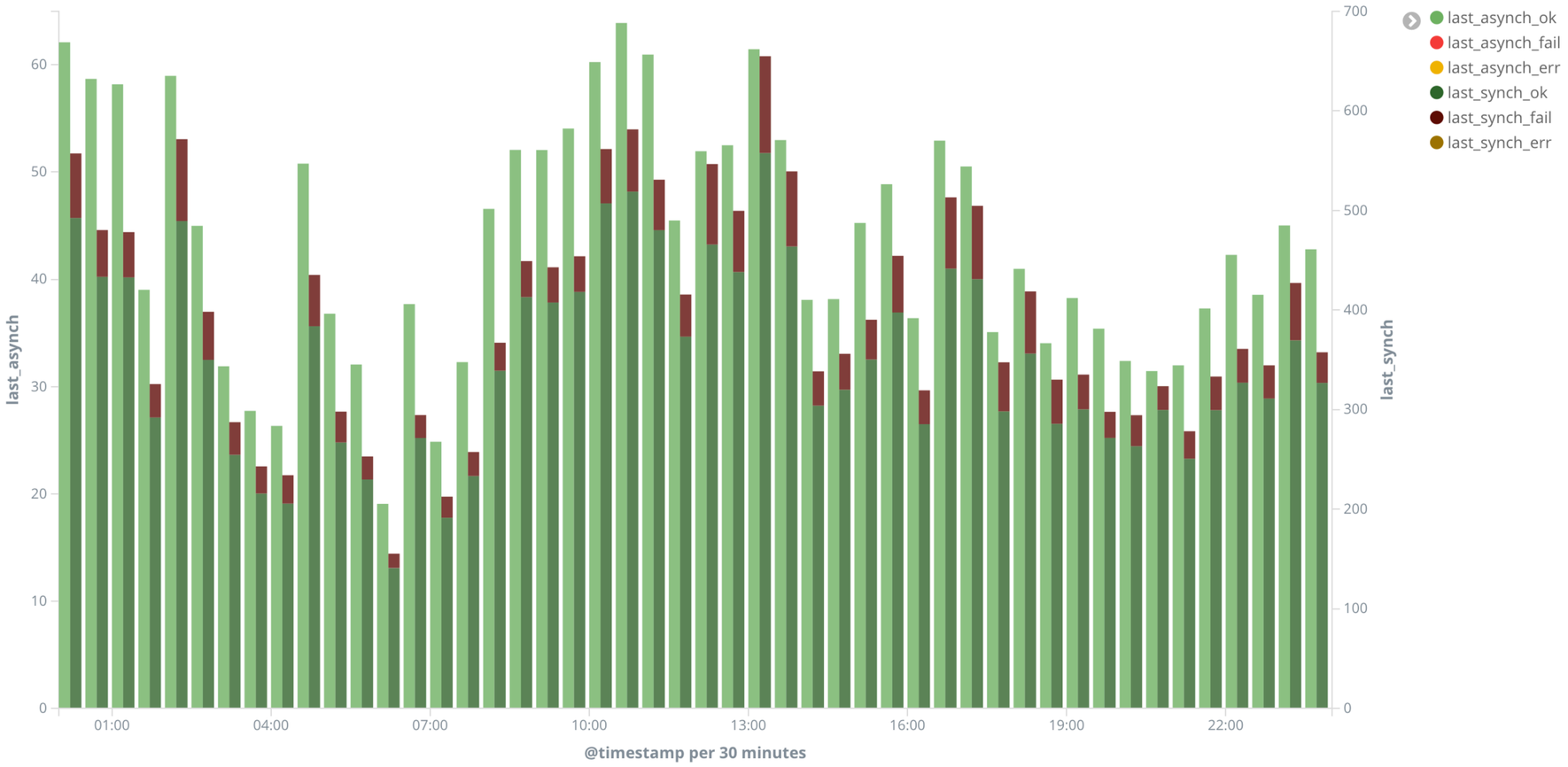
Duration (in milliseconds) of the last Prepare_to_Get and Prepare_to_Put in the *heartbeat* log.



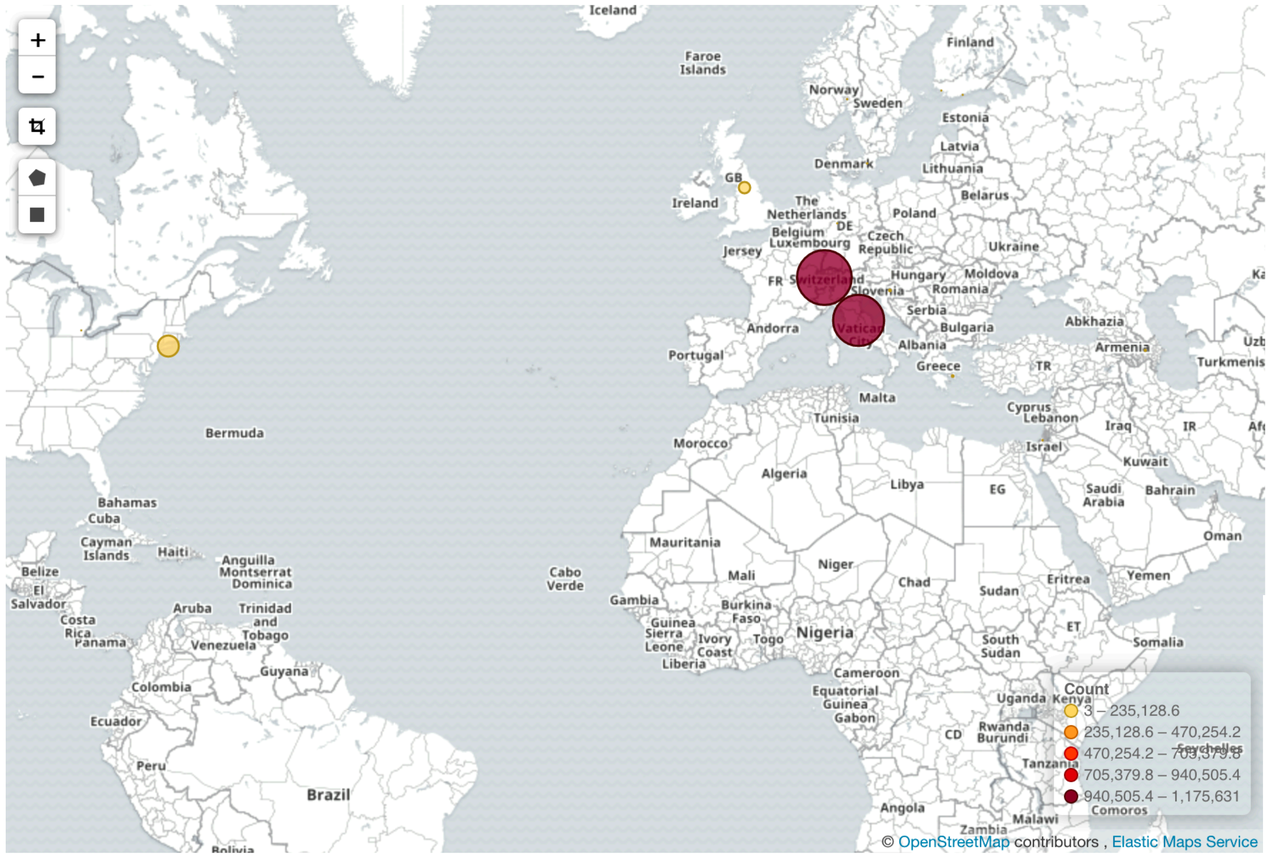
Duration (in milliseconds) of the last Prepare_to_Get and Prepare_to_Put in the *heartbeat* log.

Difference between the total last ptp (ptg) and the one which were successful, in the *heartbeat* log. (A value different than zero implies some failed request).

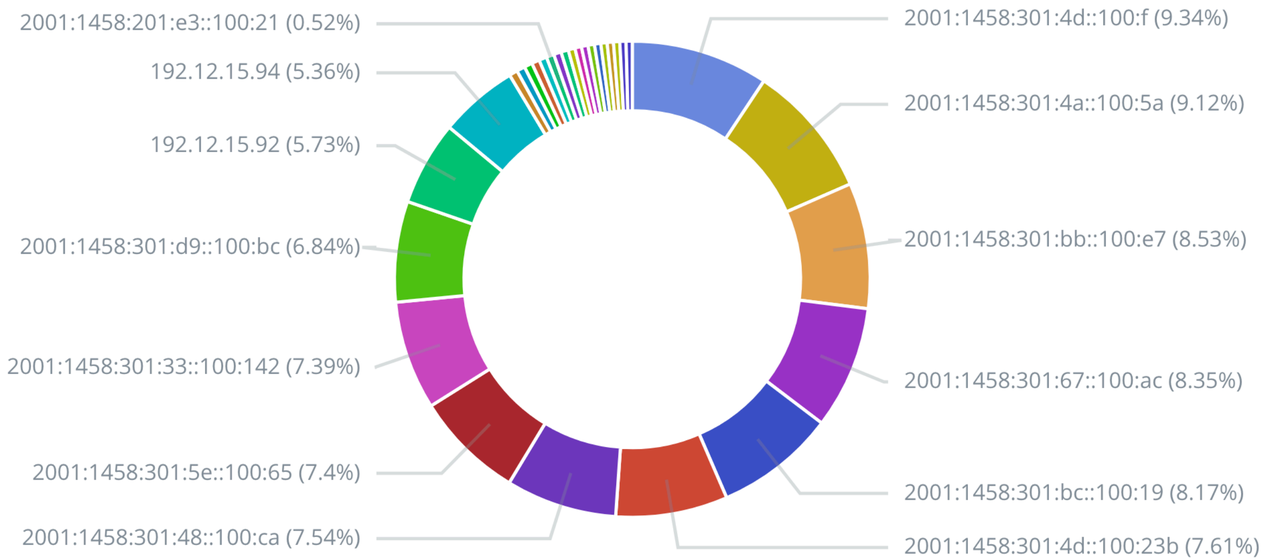




Synchronous and asynchronous counts in the StoRM *monitoring* log, divided into two separate columns.

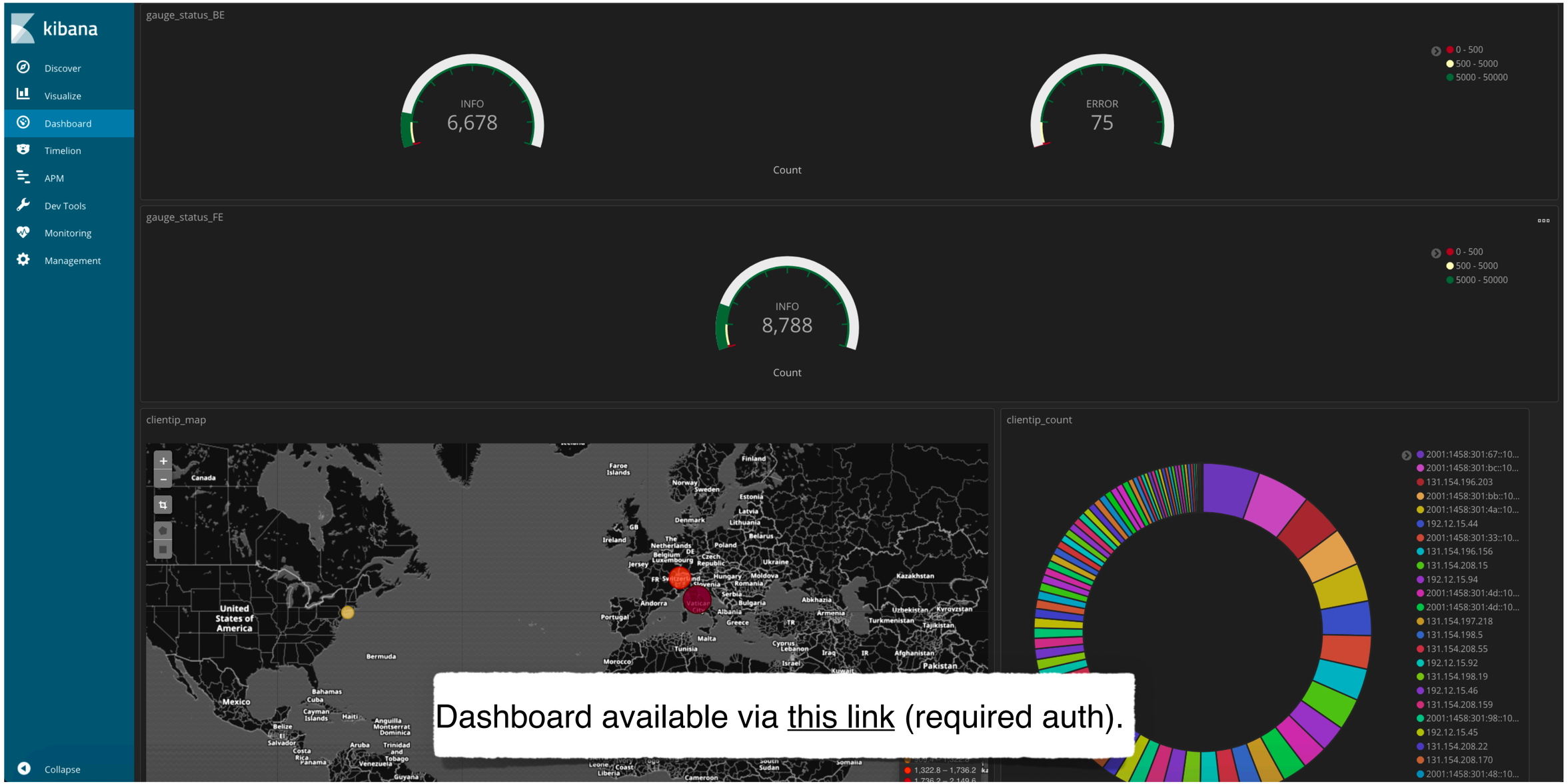


Map of client IP addresses location and frequency of the top 30 (not more for visualisation purposes).



Prototype Dashboard using information from logs:

Prototype Dashboard using information from logs:



Next Steps

- **Short-term** goals:
  - Parse the remaining logs and verify that the de-structuralized information matches the actual status of services running at CNAF.
  - Improve the dashboard, showing useful information regarding the status and possible analytics on real time log stream.

Next Steps

- **Short-term** goals:
  - Parse the remaining logs and verify that the de-structuralized information matches the actual status of services running at CNAF.
  - Improve the dashboard, showing useful information regarding the status and possible analytics on real time log stream.
- **Long-term** goals:
  - Creation of a Machine Learning model that will use such information coming from the ELK Stack, classifying possible critical errors from logs and proactively act before it is too late to intervene.

Conclusions

It was a great experience here at CNAF!

Assessing the most efforts and pain:

Log parsing is quite hard and really time consuming.

I hope this work contributed to allow future students not to start again from the beginning.

Currently, I am a **Ph.D. student in Physics** and I will be moving to something else as major topic soon (probably in the fields of DL and FPGA).

I am potentially interested in the pursuance of the collaboration with CNAF: discussions as of how to set up a *constant interaction* with CNAF developers and operators for a better log understanding and to improve log quality may be a good starting point.

Thanks!