# Log managing at PIC

A. Bruno Rodríguez Rodríguez

Port d'informació científica
Campus UAB, Bellaterra
Barcelona

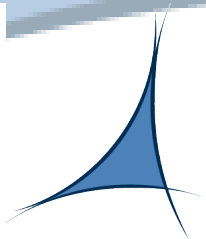
December 3, 2013

# What will I talk about...

1. Who am I and why I came
   - A very little introduction
   - The "log" self-created problem
   - Is really worth it?
2. What did we use...
   - To ship the logs
   - To receive the logs
   - To index (and query) the logs
   - To show everything in a comprehensible way
3. How did we apply everything in the real world
   - Requirements? Requirements!
4. Questions? Demos? Etc.

# About our institute and myself

- I am working for Port d'informació científica. We are physically located at UAB, a bit less than 30 Km. far away from here.
- We are a Tier-1 for LHC. Yes, Higgs Boson and all of that. We are currently storing data and running jobs for Atlas, CMS and LCHb experiments, as well as some other "smaller" projects.
- We are keeping about 600 servers and about 12 Petabyte of data, everything connected with a 10 Gbps link.
- I would like to thank l'Institut de física d'altes energies (IFAE) which pays my rent and CIEMAT, which pays the rent of some of my colleagues, too
- I am in charge of part of the puppet servers, the installation system, some computing parts and the stuff I'm about to talk now

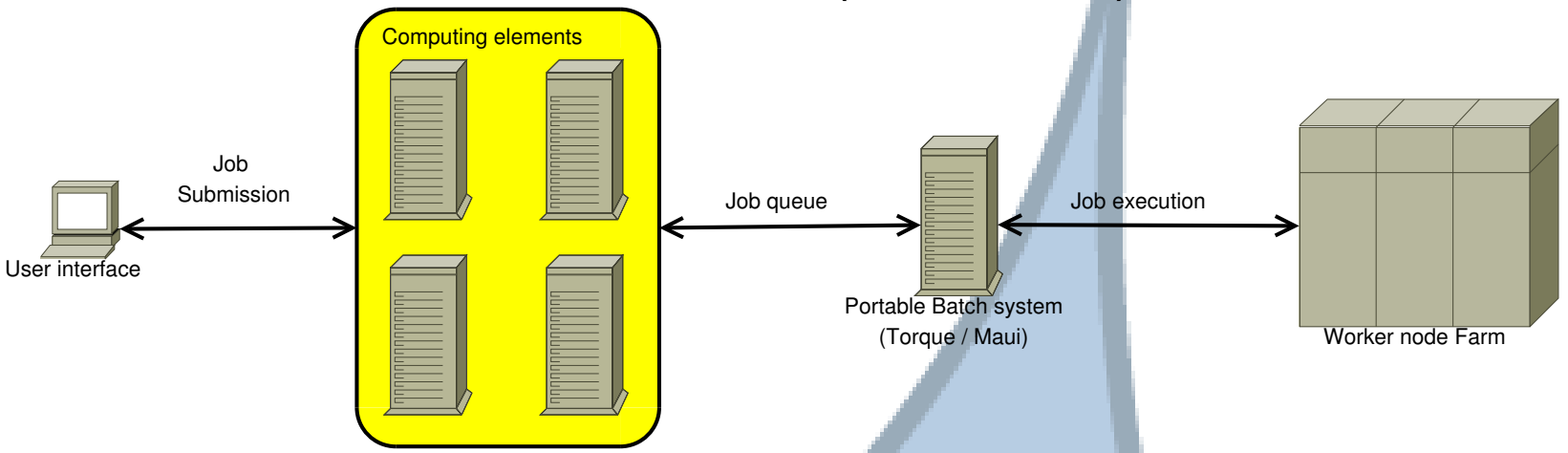


PIC
port d'informació
científica

Centro de Investigaciones
Energéticas, Medioambientales
y Tecnológicas

# Overview

A small part of our infraestructure (as instance):




- Important logs in CE's and PBS
- Also important know which WN is executing the job

# Problems

Everybody here loves CLI, but...

- Different Job IDs on CEs and PBS
- Grepping in some logs in different machines
- Users and permissions in every machine, or
- ...configure a central syslog, and grep the same
- Logrotates, zcats for compressed logs, etc.

```
ssh computing-element
grep iGSZKfQywRaseB-dlV-0w /some/long/path/ce.log
XXX XXX XXX XXX CREAM900912193 XXX XXX XX
exit

ssh pbs
grep CREAM900912193 /using/tab/here/pbs.log
XXX XXX XX XXX 989080.pbs04.pic.es XXX XX XX
```

# Centralized syslog server tree...

```
user@logserver $ ls -l /logs
drwxr-xr-x syslog syslog 12288 Nov 26 12:41 20101101
drwxr-xr-x syslog syslog 12288 Nov 26 12:41 20131102
...
drwxr-xr-x syslog syslog 12288 Nov 29 19:03 20131129
drwxr-xr-x syslog syslog 12288 Nov 30 09:00 20131130
```

```
user@logserver $ ls -l /logs/20131130
drwxr-xr-x syslog syslog 59 Nov 30 00:13 server01
drwxr-xr-x syslog syslog 59 Nov 30 00:13 server02
...
drwxr-xr-x syslog syslog 59 Nov 30 00:13 serverXX
```

# Old problems, old solutions

Of course, we already had a centralized syslog server, and we still keep it. But...

# Old problems, old solutions

Of course, we already had a centralized syslog server, and we still keep it. But...

- Configuring it to log certain files can become horrible and
- We still have to use find and grep/awk/etc.

Most people here must be thinking I'm lazy...

# Old problems, old solutions

Of course, we already had a centralized syslog server, and we still keep it. But...

- Configuring it to log certain files can become horrible and
- We still have to use find and grep/awk/etc.

Most people here must be thinking I'm lazy...

And you're right!

But imagine those "advanced" user wanting to know what happens with their jobs... Aren't they lazy?

# Old problems, old solutions

Of course, we already had a centralized syslog server, and we still keep it. But...

- Configuring it to log certain files can become horrible and
- We still have to use find and grep/awk/etc.

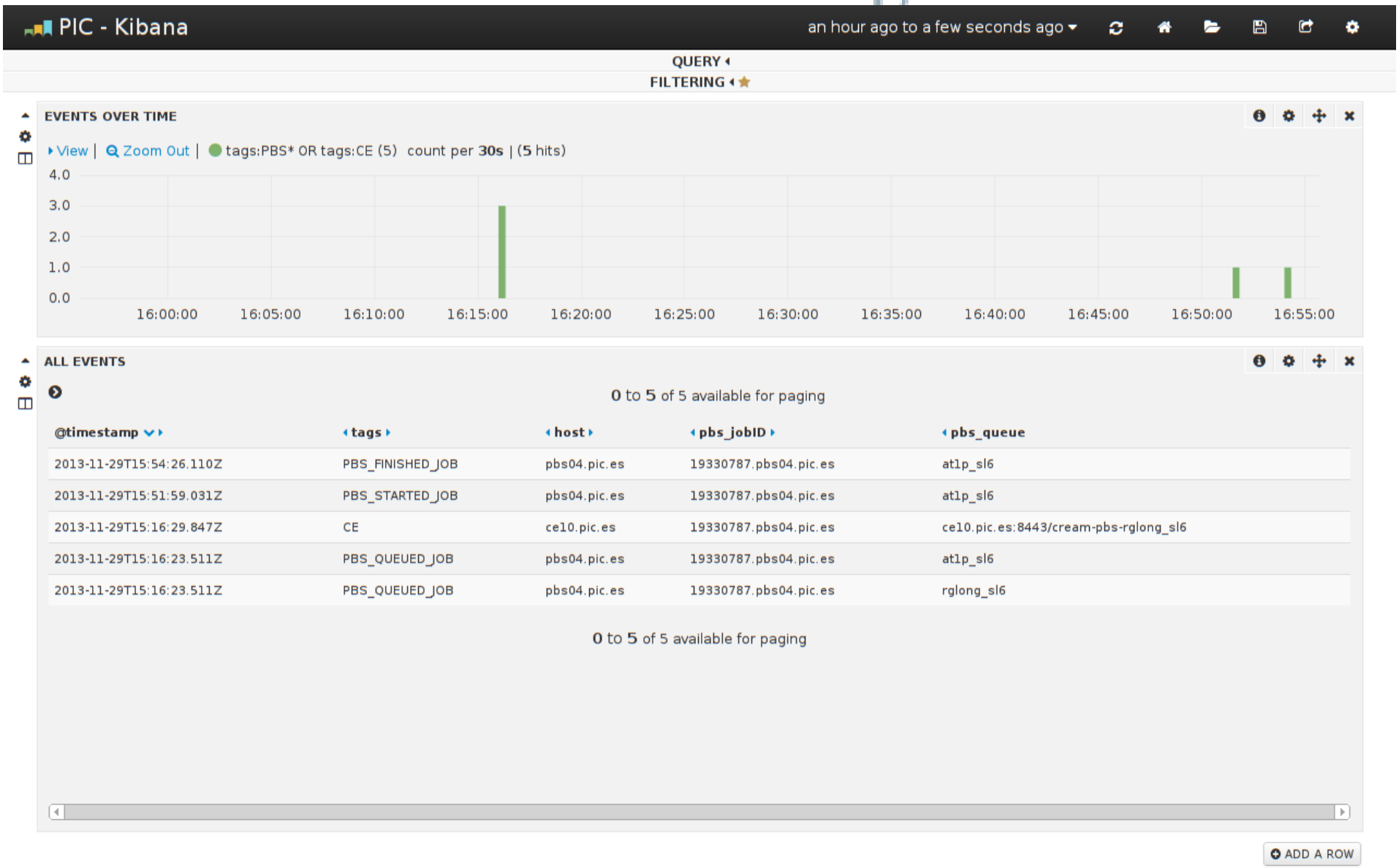
Most people here must be thinking I'm lazy...

And you're right!

But imagine those "advanced" user wanting to know what happens with their jobs... Aren't they lazy?

An never forget: "advanced" is the opposite of "retarded", so probably they know the minimum effort law (and they like to apply it).

# So the log search now looks like...



# Everybody *insert verb here* syslog

Probably most people here knows syslog (RFC 3164, RFC 5424)

- Well-known port to ship and receive logs: 514 UDP. . . not so easy to crypt communication.
- Priority levels

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Debug | Info | Notice | Warn | Error | Critical | Alert | Panic |

- Check syslog daemon memory usage, it can be relevant in servers with small (less than 256MB) RAM.

# Everybody *insert verb here* syslog

Probably most people here knows syslog (RFC 3164, RFC 5424)

- Well-known port to ship and receive logs: 514 UDP. . . not so easy to crypt communication.
- Priority levels

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Debug | Info | Notice | Warn | Error | Critical | Alert | Panic |

- Check syslog daemon memory usage, it can be relevant in servers with small (less than 256MB) RAM.

## Fast and easy configuration

Supposing you have a logstashserver listening on port UDP 10514:

```
cat >> /etc/rsyslog.d/logstash.conf << EOF
*.err,daemon.none @logstashserver:10514
EOF
```

And HUP or restart rsyslog. This would ship only system messages with priority less or equal than error.

# Lumberjack (and he's OK)

When it comes to ship just some logfiles or an stdout (pipe), Lumberjack became a very useful tool:

- Small and "clean": a single executable, less than 2MB shared RAM usage.
- You are able to add some custom fields.
- SSL crypted communications by default.

# Lumberjack (and he's OK)

When it comes to ship just some logfiles or an stdout (pipe), Lumberjack became a very useful tool:

- Small and "clean": a single executable, less than 2MB shared RAM usage.
- You are able to add some custom fields.
- SSL crypted communications by default.

**Never forget logrotate!**

If a log being accessed by lumberjack is rotated, the file pointer will change, so you'll need to restart lumberjack in order to access the new log file!

# Logstash: A log with a moustache!!

Logstash is a "new generation" log collector:

- Embedded in a JRuby container, runs wherever a JRE is installed (which I suppose is good)

# Logstash: A log with a moustache!!

Logstash is a "new generation" log collector:

- Embedded in a JRuby container, runs wherever a JRE is installed (which I suppose is good)
- Has three types of plugins:
  - Input: Listens for a socket, checks a file...
  - Filter: Parses, mutates, greps...
  - Output: Inserts into database, sends a mail...

# Logstash: A log with a moustache!!

Logstash is a "new generation" log collector:

- Embedded in a JRuby container, runs wherever a JRE is installed (which I suppose is good)
- Has three types of plugins:
  - Input: Listens for a socket, checks a file...
  - Filter: Parses, mutates, greps...
  - Output: Inserts into database, sends a mail...
- And a kinda weird logo image

# logstash.conf file example

## Input

```
input {
  plugin_name {
    type => "mytype"
    option => "value"
    ...
  }
}
```

## output

```
output{
  plugin_name{
    option => "value"
    ...
  }
}
```

## Filter

```
filter {
  if [type]=="mytype" {
    plugin_name {
      option => "value"
      ...
    }
  }
}
```

- The conditionals were added on logstash 1.2!
- Adding tags is a good point to know which plugins are applied on every message

# Some logstash input plugin config examples

## Syslog input on logstash.conf

```
udp {
    port => 10514
    type => "linux-syslog"
}
```

# Some logstash input plugin config examples

## Syslog input on logstash.conf

```
udp {
    port => 10514
    type => "linux-syslog"
}
```

After this all of the logs received by port UDP 10514 will be of type "linux-syslog". We can filter them in the section filter of the config file with a conditional:

```
if [type] == "linux-syslog"
{
    ...
}
```

# Some logstash input plugin config examples

## Lumberjack input on logstash.conf

```
lumberjack{
 port => 11111
 ssl_certificate => "/etc/logstash/keys/logstash.pub"
 ssl_key => "/etc/logstash/keys/logstash.key"
 type => "Computing-PBS"
}
```

# Some logstash filter plugin config examples

## kv (key-value) on logstash.conf

```
kv {
 field_split => ";"
}
```

# Some logstash filter plugin config examples

## kv (key-value) on logstash.conf

```
kv {
 field_split => ";"
}
```

would make of a line like

`key1=v1;key2=v2`

# Some logstash filter plugin config examples

## kv (key-value) on logstash.conf

```
kv {
 field_split => ";"
}
```

would make of a line like

```
key1=v1;key2=v2
```

a structure like

```
{ ...
    "message" => "key1=v1;key2=v2",
    "key1" => "v1",
    "key2" => "v2",
... }
```

# Some logstash filter plugin config examples

## grok on logstash.conf

```
grok {
 match => [ "message", "^%{DATA:field\}␣%{POSINT:value\}" ]
}
```

# Some logstash filter plugin config examples

## grok on logstash.conf

```
grok {
 match => [ "message", "^%{DATA:field\}␣%{POSINT:value\}" ]
}
```

this regular expresion over

`MemUsage 35`

# Some logstash filter plugin config examples

## grok on logstash.conf

```
grok {
 match => [ "message", "^%{DATA:field\}␣%{POSINT:value\}" ]
}
```

this regular expresion over

`MemUsage 35`

would produce

```
{ ...
    "message" => "MemUsage 35",
    "field" => "MemUsage",
    "value" => "35",
... }
```

# Elasticsearch (ES to make it shorter)

Elasticsearch is an indexing engine for JSON documents (think of them as JavaScript arrays)

- Self discovering clusters (via multicast).
- Configuring shards (stripes) and replicas (mirrors) of data chunks is as difficult as changing a yaml file.
- REST API on port 9200 to make queries, and it supports Lucene syntax, i.e. you can query data via curl
- It is also a Java piece of software.

# Elasticsearch (ES to make it shorter)

Elasticsearch is an indexing engine for JSON documents (think of them as JavaScript arrays)

- Self discovering clusters (via multicast).
- Configuring shards (stripes) and replicas (mirrors) of data chunks is as difficult as changing a yaml file.
- REST API on port 9200 to make queries, and it supports Lucene syntax, i.e. you can query data via curl
- It is also a Java piece of software.

**But it's got a BIG problem:**

It's not crypted. Not without using a plugin to accomplish it.

# Elasticsearch (ES to make it shorter)

Elasticsearch is an indexing engine for JSON documents (think of them as JavaScript arrays)

- Self discovering clusters (via multicast).
- Configuring shards (stripes) and replicas (mirrors) of data chunks is as difficult as changing a yaml file.
- REST API on port 9200 to make queries, and it supports Lucene syntax, i.e. you can query data via curl
- It is also a Java piece of software.

**But it's got a BIG problem:**

It's not crypted. Not without using a plugin to accomplish it.
Anyway, for internal accesses is great. And you won't set the log querying system in a public network, will you?

# Some Elasticsearch-Lucene examples

Lucene syntax is quite powerful for querying information:

## Get the jobs at the PBS

```
curl 'http://es_server:9200/_all/_search?pretty&q=@tags:PBS_*_JOB'
```

# Some Elasticsearch-Lucene examples

Lucene syntax is quite powerful for querying information:

## Get the jobs at the PBS

```
curl 'http://es_server:9200/_all/_search?pretty&q=@tags:PBS_*_JOB'
```

That would return something like:

```
{
      "_index" : "logstash-2013.10.11",
       ...
      "_score" : 1.0, "_source" : {
         "@source":"lumberjack://pbs.pic.es/var/log/20131011",
         "@tags":["PBS_STARTED_JOB"],
         "@fields":{},
         "@timestamp":"2013-10-11T13:38:36.650Z", ...
    } ]
  }
```

Lucene provides wildcards, fuzzy searches, etc. As the search field in kibana interface uses it, it's quite interesting to learn a bit of it.

# Kibana (it's also a japanese district)

Kibana is the interface we saw some slides ago. We are using Kibana 3, which is:

- Totally HTML5 based. No need for any middle-layer server with PHP, Ruby or similars.
- Accesses directly to Elasticsearch. As JSON is a Javascript array those can be directly read.

# Kibana (it's also a japanese district)

Kibana is the interface we saw some slides ago. We are using Kibana 3, which is:

- Totally HTML5 based. No need for any middle-layer server with PHP, Ruby or similars.
- Accesses directly to Elasticsearch. As JSON is a Javascript array those can be directly read.

## Dashboards

Dashboards are customizable, and they can be saved in Elasticsearch or exported. If you save them in Elasticsearch, take care of the kibana-int index.

# Kibana (it's also a japanese district)

Kibana is the interface we saw some slides ago. We are using Kibana 3, which is:

- Totally HTML5 based. No need for any middle-layer server with PHP, Ruby or similars.
- Accesses directly to Elasticsearch. As JSON is a Javascript array those can be directly read.

## Dashboards

Dashboards are customizable, and they can be saved in Elasticsearch or exported. If you save them in Elasticsearch, take care of the kibana-int index.

## Accessing Kibana

It's not necessary but... Better don't forget to set a user/password or something similar for accessing to Kibana web.

# Infraestructure used right now

Right now everything we can keep everything running on:

- 3 virtual machines hosted on a Redhat EV, each one has
- 2GB RAM
- 30GB Hard disk drive (each one)
- Normal memory usage about 50% and normal CPU usage about 10%
- Most of the installation/configuration was quite “puppetizable”

# Infraestructure used right now

Right now everything we can keep everything running on:

- 3 virtual machines hosted on a Redhat EV, each one has
- 2GB RAM
- 30GB Hard disk drive (each one)
- Normal memory usage about 50% and normal CPU usage about 10%
- Most of the installation/configuration was quite “puppetizable”

**Every day’s syslog for all of our servers (about 600) would need about 1.5GB**

# Infraestructure used right now

Right now everything we can keep everything running on:

- 3 virtual machines hosted on a Redhat EV, each one has
- 2GB RAM
- 30GB Hard disk drive (each one)
- Normal memory usage about 50% and normal CPU usage about 10%
- Most of the installation/configuration was quite "puppetizable"

**Every day's syslog for all of our servers (about 600) would need about 1.5GB**

So... We are not logging every single syslog event. Filtering by syslog priority is important!

# Questions start with “Quest”

**Thanks for your time!**