# Anàlisi de fitxers de Log en sistemes amb un gran volum d'informació i contingut crític i sensible

Facultat d'informàtica de Barcelona

Universitat Politècnica de Catalunya (UPC)

Grau en Enginyeria Informàtica

Tecnologies de la Informació

Autor: **Jordi Farran Centelles**

Ponent: **Xavier Martorell Bofill**

Directors: **Iván Jesús Correa Negrín**

**Emilio Pastor Moreno**

Juny 2016

# Agraïments

Primer de tot vull agrair als responsables del projecte per part de l'empresa everis per oferir-me la possibilitat de dur a terme aquest projecte com un treball final de grau. Concretament al Iván Jesús Correa per oferir-me aquest projecte i també a l'Emilio Pastor pel seu constant suport en les reunions, la seva orientació i la confiança d'acompanyar-me durant la trajectòria d'aquest.

Per altra part també agrair al ponent per part de la Facultat d'Informàtica de Barcelona, concretament en Xavier Martorell, per la seva exigència durant la realització d'aquest, els seus consells i sobretot per acceptar fer-se càrrec d'aquest projecte.

També voldria agrair a en Joan Subirats, tutor de GEP, per tots els seus consells per dur a terme una bona memòria i una bona defensa durant la presentació del projecte.

També voldria agrair a totes aquelles persones amb que he pogut intercanviar opinions i dubtes i que han fet que aquest projecte fos més fàcil i més entretingut de dur-lo a terme.

Finalment agrair a tota la meva família per fer més fàcil aquest camí amb tot el seu recolzament.

# Resum

Aquest projecte és un Treball Final de Grau (TFG) de modalitat B, això significa que està desenvolupat en una empresa.

El principal objectiu d'aquest projecte és desenvolupar una plataforma per poder consultar traces de Logs en temps real d'una forma senzilla i simple, ja que actualment els treballadors de la empresa dediquen més temps del necessari en dur a terme aquestes cerques. El principal problema és que quan volem consultar els Logs d'una aplicació s'ha d'accedir via *ssh* al servidor on s'ha executat, una vegada estem connectats hem de saber en quin fitxer s'ha guardat i després trobar la traça de dades dins d'aquest.

Per dur a terme aquest objectiu utilitzarem les eines ElasticSearch, Logstash i Kibana, les quals en permetran agafar totes les dades de les aplicacions, centralitzar-les i finalment fer la cerca, utilitzant tot tipus de filtres.

# Resumen

Este proyecto es un Trabajo Final de Grado (TFG) de modalidad B, esto significa que está desarrollado en una empresa.

El principal objetivo de este proyecto es desarrollar una plataforma para poder consultar trazas de Logs en tiempo real de una forma sencilla y simple, ya que actualmente los trabajadores de la empresa dedican más tiempo del necesario a llevar a cabo estas búsquedas. El principal problema es que cuando queremos consultar los Logs de una aplicación se tiene que acceder vía *ssh* al servidor dónde se ha ejecutado, una vez estamos conectados tenemos que saber en qué fichero se ha guardado y luego encontrar la traza de datos dentro de este.

Para llevar a cabo este objetivo utilizaremos las herramientas ElasticSearch, Logstash y Kibana, las cuales nos permitirán coger todos los datos de las aplicaciones, centralizarlos y finalmente hacer la búsqueda utilizando todo tipo de filtros.

# Summary

This project is a Degree Final Project modality B, this means that it is developed in a company.

The main purpose of this project is to develop a platform to consult trace Logs in real time in an easy and simple way, so nowadays these company's employees spend too much time doing these searches. The main problem is when we want to consult the Logs from an application, that we must access through *ssh* to the server where it has run, once we are connected we need to know in which file is saved and then find the data.

To bring out this goal we will use the ElasticSearch, Logstash and Kibana tools, which allow us to get all application data, centralize it and finally do the search using all kinds of filters.

# Índex de continguts

1. Introducció ..... 7
   1.1. Context
   1.2. Actors implicats
2. Formulació del problema ..... 10
   2.1. Problema
   2.2. Objectiu
3. Estat de l'Art ..... 11
   3.1. Eines actuals
   3.2. Comparacions i Elecció
   3.3. Eines addicionals
4. Abast ..... 19
   4.1. Eines
   4.2. Requeriments
   4.3. Obstacles
5. Metodologia i Rigor ..... 21
   5.1. Metodologia àgil
   5.2. Eines de seguiment
   5.3. Eines de control
6. Planificació Temporal ..... 22
   6.1. Calendari
   6.2. Fases del Projecte
   6.3. Diagrama de Gantt
7. Gestió Econòmica ..... 28
   7.1. Costos directes per tasca
   7.2. Costos indirectes
   7.3. Contingència
   7.4. Imprevistos
   7.5. Pressupost
8. Control de gestió ..... 31
9. Sostenibilitat ..... 32
   9.1. Matriu de Sostenibilitat
   9.2. Dimensió Econòmica
   9.3. Dimensió Social

9.4. Dimensió Ambiental
10. Disseny ..... 34
11. Implementació ..... 35
11.1. Vagrant
11.2. Ansible
11.3. Docker
11.4. Node.js
11.5. Logstash
11.6. ElasticSearch
11.7. Kibana
12. Proves ..... 68
13. Conclusions ..... 69
14. Referències ..... 70
15. Annex ..... 72

## Índex de Figures

Figura 1. Splunk 1 ..... 11
Figura 2. Splunk 2 ..... 11
Figura 3. Kibana ..... 12
Figura 4. Nagios 1 ..... 13
Figura 5. Nagios 2 ..... 13
Figura 6. Fases del projecte ..... 22
Figura 7. Esquema Disseny ..... 34
Figura 8. Configuració Vagrant ..... 35
Figura 9. Estructura de directoris d'Ansible ..... 38
Figura 10. Format del playbook ..... 39
Figura 11. Execució d'Ansible ..... 43
Figura 12. Docker ps -a ..... 47
Figura 13. Estructura Logstash ..... 50
Figura 14. Informació ElasticSearch ..... 56
Figura 15. Pàgina d'inici Kibana ..... 60
Figura 16. Menú Kibana ..... 61
Figura 17. Propietats d'un índex del Kibana ..... 61
Figura 18. Logs/Temps ..... 62
Figura 19. Creació d'una gràfica ..... 62
Figura 20. Propietats d'una gràfica ..... 63
Figura 21. Gràfica de sectors ..... 64
Figura 22. Dashboard Final ..... 65
Figura 23. Dashboard Final 2 ..... 66
Figura 24. Estadistiques Kibana ..... 67
Figura 25. Gràfica temps per cerca ..... 67

## Índex de Taules

Taula 1. Costos directes del projecte ..... 28
Taula 2. Costos indirectes del projecte ..... 29
Taula 3. Costos de contingència del projecte ..... 29
Taula 4. Cost dels imprevistos del projecte ..... 30
Taula 5. Pressupost del projecte ..... 30
Taula 6. Matriu sostenibilitat ..... 32

# 1. Introducció

## 1.1. Context

L'ús d'aparells electrònics ha tingut un gran impacte en la nostra societat ens les darreres dècades, aquest canvi ha estat tant gran, que actualment hi ha milers de productes tecnològics en desenvolupament.

Aquests productes poden ser molt diferents, però a la vegada hi ha una part essencial, que sense aquesta, el producte no tindria tota la intel·ligència i utilitats desitjades. Estem parlant de la fase de programació del producte, per tal que compleixi la seva tasca tal i com nosaltres l'hem pensat.

Aquesta fase és pot dur a terme de moltes maneres diferents, existeixen diferents plataformes, diferents llenguatges de programació, diferents eines, etc. Però totes es basen en la mateixa idea, poder desenvolupar el codi pel producte final de la millor manera possible.

És per això que en grans projectes on hi treballen desenes de persones i on és van fent actualitzacions de codi molt sovint, hi hagi diferents tipus de proves per saber que tots els canvis efectuats, garanteixen el correcte funcionament de la versió anterior i a més a més poder portar un control exhaustiu del flux que segueix el codi.

Per poder controlar aquestes accions, s'acostuma a utilitzar uns tests de proves automàtiques. Quant aquests tests fallen, i per tant, l'aplicació o servei que estem desenvolupant no es comporta com nosaltres desitjaríem, hem d'esbrinar el que està succeint, i és aquí on entra amb una gran força la part dels Logs[1], també anomenats registres, ja que gràcies a aquests, podem observar quin flux segueix el codi que estem desenvolupant.

Una altra gran utilitat dels Logs és poder tenir informació sobre qui, què, quant, on i perquè succeeix un esdeveniment en el nostre servei.

Per tant quan alguna execució ha anat malament o bé es vol obtenir informació, els Logs hi juguen una part molt important, ja que és com un llibre on és guarda tot el que succeeix.

[1] Son traces d'informació que contenen informació sobre esdeveniments que han succeït al nostre sistema.

És per això que el meu TFG (Treball Final de Grau) estarà dedicat a l'anàlisi d'aquests fitxers i al desenvolupament d'una plataforma per poder oferir una visió d'aquests quant en tenim una gran quantitat.

## 1.2. Actors implicats

En el context d'aquest projecte hi ha tota un sèrie de persones o entitats implicades directament o bé indirectament.

A continuació s'exposen aquests actors.

### 1.2.1. Desenvolupador

Aquesta persona serà la més activa en el desenvolupament del projecte, ja que és l'encarregada de dur-lo tot a terme i de complir tots els terminis establerts.

S'encarregarà tant de la part pràctica, com de la gestió del projecte, el desenvolupament de la memòria i de la preparació de la defensa.

Tota aquesta feina però, estarà consultada i prèviament definida, tant amb el director del projecte com el ponent.

### 1.2.2. Director, Ponent i suport

S'encarregaran de guiar, opinar i donar suport al desenvolupador en totes les seves tasques, per assegurar el correcte desenvolupament del projecte, i en cas d'algun imprevist, intentar aconsellar-lo de la millor manera possible. Aquestes persones són:

- Directors:
  - Iván Jesús Correa Negrín
  - Emilio Pastor Moreno
- Ponent: Xavier Martorell Bofill
- GEP: Joan Subirats Soler

### 1.2.3. Usuaris

Els usuaris d'aquest projecte seran totes les persones que actualment treballen a l'empresa Everis i que pertanyen al projecte de l'entitat bancària per a qui treballem i òbviament a la pròpia entitat bancària.

### 1.2.4. Beneficiaris

Com ja he comentat al punt anterior, els usuaris seran de l'empresa Everis o bé de l'entitat bancària, per tant, els beneficiaris seran ells mateixos, ja que amb la solució proposada, no s'haurà d'invertir tant de temps en mirar els Logs, perquè tindran una accessibilitat més fàcil.

# 2. Formulació del problema

## 2.1. Problema

Actualment en l'empresa on estic fent pràctiques, estem desenvolupant aplicacions i serveis, per una entitat bancària.

Per garantir el correcte funcionament i tenir una breu idea de com és comporta tot el que estem desenvolupant, fem un gran ús dels Logs.

Com el projecte és bastant gran, i te una certa envergadura, per a que les proves és facin en un cert temps raonable, es disposa de diferents servidors de proves on es paral·lelitzen els tests o bé on s'executen les aplicacions ja finalitzades.

Aleshores quan és vol fer una cerca d'un error o una prova en concret, s'ha d'esbrinar en quina màquina ha estat executada, obtenir el fitxer de Logs corresponent, i aleshores, fer una cerca al document de les traces que a nosaltres ens interessen i volem obtenir.

Tot això comporta una gran pèrdua de temps al treballador cada cop que ha de fer la comprovació, i per tant, suposa una petita pèrdua de productivitat.

## 2.2. Objectiu

Es proposa arribar a una solució que mostri d'una forma més visual, per exemple, una gràfica execucions/temps, per a que l'usuari pugui seleccionar una franja de temps i vegi les traces durant aquell interval, a més a més, que pugui filtrar les cerques, i així, fer que se li mostri exactament la traça d'informació que ell està buscant. Això implica centralitzar tots els Logs dels diferents servidors en una màquina i després fer la cerca adient.

# 3. Estat de l'Art

Actualment per a poder realitzar aquest projecte ja hi ha diferents eines al mercat que t'ajuden a poder-ho dur a terme.

## 3.1. Eines actuals

### 3.1.1. Splunk[1]

Una eina molt utilitzada actualment és *Splunk*, la qual et permet fer una cerca sobre el contingut que vols analitzar i una vegada analitzat, t'ho mostra en gràfiques i dades. Es caracteritza per tenir una gran velocitat i per la seva facilitat d'integració en qualsevol entorn.


*Figura 1. Splunk 1*

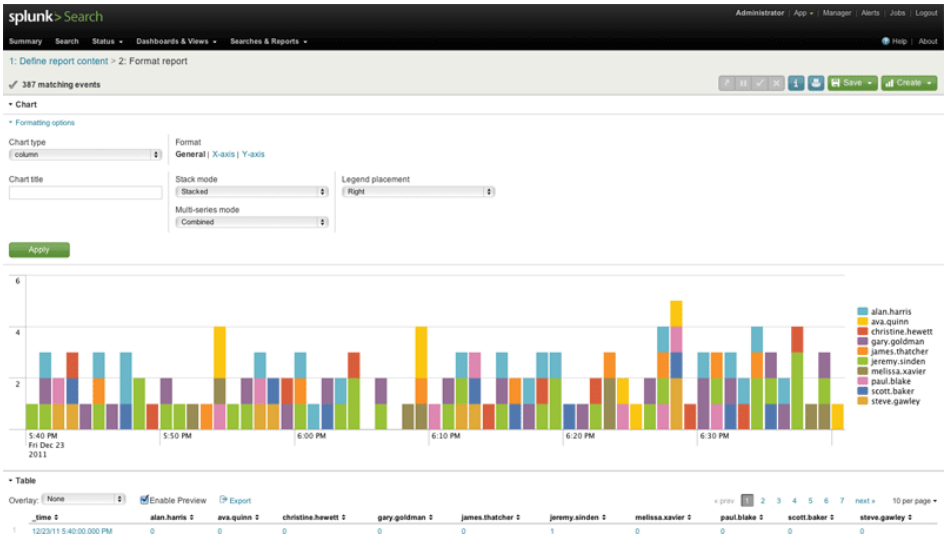


*Figura 2. Splunk 2*

### 3.1.2. ELK[2] (ElasticSearch – Logstash – Kibana)

ElasticSearch, Logstash i Kibana són projectes Open Source que ajuden a l'usuari a obtenir les dades de qualsevol font de dades, amb qualsevol format i fer una cerca i un anàlisi de les mateixes i poder visualitzar-les en temps real.

És per això, que ELK ha tingut una gran rebuda per part de la comunitat. És una eina amb un potencial molt gran, que pot utilitzar-se en molts àmbits diferents, tant per poder veure estadístiques de les dades o fins i tot tenir un control detallat de tot el que està succeint en cada moment.

La companyia que està darrera d'aquests productes s'anomena Elastic i va ser fundada l'any 2012, fins a l'actualitat ha tingut un gran creixement, fet que li ha permès expandir-se per les ciutats més importants del món. Els productes de llicència gratuïta que ofereix Elastic son ElasticSearch, Logstash i Kibana, però a banda d'aquests, ofereix altres eines que els hi donen suport, i que faciliten el treball al usuari. Aquestes altres eines ja compten amb una llicència de pagament.

A continuació mostrarem quin es l'objectiu de cada part.

- **Logstash**: S'encarrega d'obtenir les dades de les diferents màquines on ha estat instal·lat i enviar-les cap a ElasticSearch.
- **ElasticSearch**: S'encarrega de rebre les dades enviades per Logstash, guardar-les a la seva base de dades no relacional i aleshores permetre realitzar cerques sobre aquestes.
- **Kibana**: S'encarrega d'executar les consultes a ElasticSearch i rebre les dades ja filtrades i mostrar-les d'una forma senzilla i entenedora de cara al usuari.

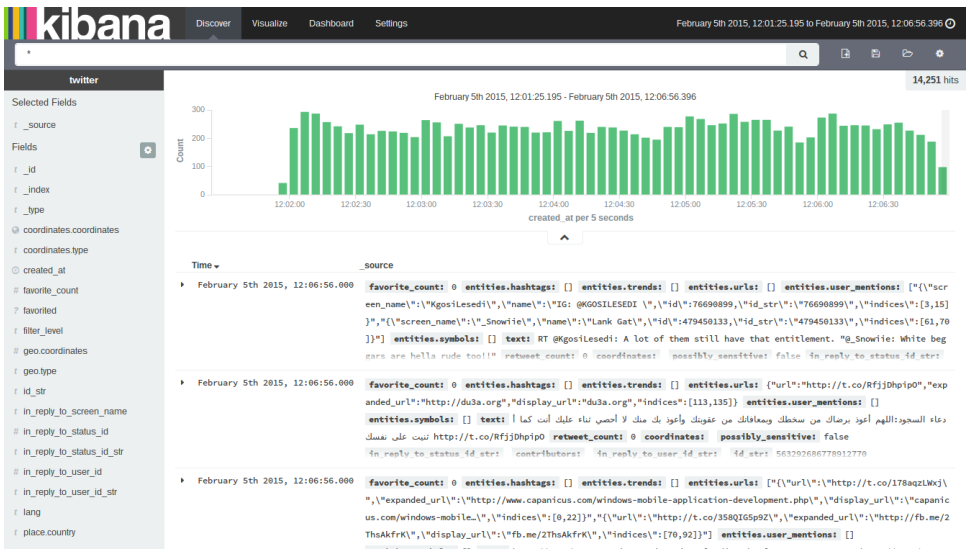
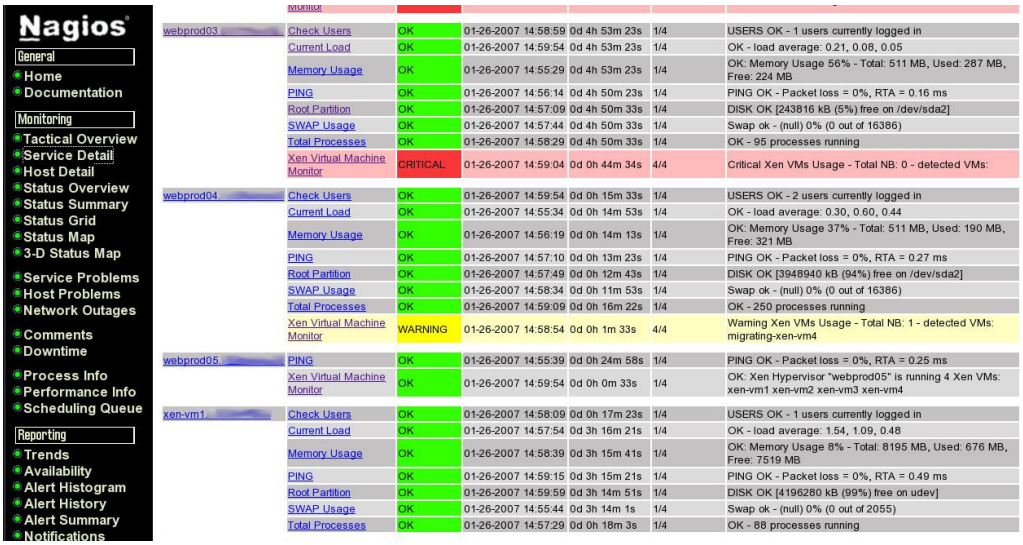



*Figura 3. Kibana*

### 3.1.3. Nagios[3]

*Nagios* et permet monitoritzar el sistema, obtenint els fitxers i fent una cerca amb els paràmetres que el desenvolupador estableixi, també dóna la oportunitat de poder fer saltar alertes al usuari depenent de quin sigui el resultat. Aquest últim aspecte és el que li ha donat bastanta popularitat ens els darrers anys, ja que és una eina molt potent, que combinada amb altres, se'n pot treure un gran rendiment.




| | | | | | | |
|---|---|---|---|---|---|---|
| webprod03 | Check Users | OK | 01-26-2007 14:58:59 | 0d 4h 53m 23s | 1/4 | USERS OK - 1 users currently logged in |
| | Current Load | OK | 01-26-2007 14:59:54 | 0d 4h 53m 23s | 1/4 | OK - load average: 0.21, 0.08, 0.05 |
| | Memory Usage | OK | 01-26-2007 14:55:29 | 0d 4h 53m 23s | 1/4 | OK: Memory Usage 56% - Total: 511 MB, Used: 287 MB, Free: 224 MB |
| | PING | OK | 01-26-2007 14:56:14 | 0d 4h 50m 23s | 1/4 | PING OK - Packet loss = 0%, RTA = 0.16 ms |
| | Root Partition | OK | 01-26-2007 14:57:09 | 0d 4h 50m 33s | 1/4 | DISK OK [243816 kB (5%) free on /dev/sda2] |
| | SWAP Usage | OK | 01-26-2007 14:57:44 | 0d 4h 50m 33s | 1/4 | Swap ok - (null) 0% (0 out of 16386) |
| | Total Processes | OK | 01-26-2007 14:58:29 | 0d 4h 50m 33s | 1/4 | OK - 95 processes running |
| | Xen Virtual Machine Monitor | CRITICAL | 01-26-2007 14:59:04 | 0d 0h 44m 34s | 4/4 | Critical Xen VMs Usage - Total NB: 0 - detected VMs: |
| webprod04 | Check Users | OK | 01-26-2007 14:59:54 | 0d 0h 15m 33s | 1/4 | USERS OK - 2 users currently logged in |
| | Current Load | OK | 01-26-2007 14:55:34 | 0d 0h 14m 53s | 1/4 | OK - load average: 0.30, 0.60, 0.44 |
| | Memory Usage | OK | 01-26-2007 14:56:19 | 0d 0h 14m 13s | 1/4 | OK: Memory Usage 37% - Total: 511 MB, Used: 190 MB, Free: 321 MB |
| | PING | OK | 01-26-2007 14:57:10 | 0d 0h 13m 23s | 1/4 | PING OK - Packet loss = 0%, RTA = 0.27 ms |
| | Root Partition | OK | 01-26-2007 14:57:49 | 0d 0h 12m 43s | 1/4 | DISK OK [3948940 kB (94%) free on /dev/sda2] |
| | SWAP Usage | OK | 01-26-2007 14:58:34 | 0d 0h 11m 53s | 1/4 | Swap ok - (null) 0% (0 out of 16386) |
| | Total Processes | OK | 01-26-2007 14:59:09 | 0d 0h 16m 22s | 1/4 | OK - 250 processes running |
| | Xen Virtual Machine Monitor | WARNING | 01-26-2007 14:58:54 | 0d 0h 1m 33s | 4/4 | Warning Xen VMs Usage - Total NB: 1 - detected VMs: migrating-xen-vm4 |
| webprod05 | PING | OK | 01-26-2007 14:55:39 | 0d 0h 24m 58s | 1/4 | PING OK - Packet loss = 0%, RTA = 0.25 ms |
| | Xen Virtual Machine Monitor | OK | 01-26-2007 14:59:54 | 0d 0h 0m 33s | 1/4 | OK: Xen Hypervisor "webprod05" is running 4 Xen VMs: xen-vm1 xen-vm2 xen-vm3 xen-vm4 |
| xen-vm1 | Check Users | OK | 01-26-2007 14:58:09 | 0d 0h 17m 23s | 1/4 | USERS OK - 1 users currently logged in |
| | Current Load | OK | 01-26-2007 14:57:54 | 0d 3h 16m 21s | 1/4 | OK - load average: 1.54, 1.09, 0.48 |
| | Memory Usage | OK | 01-26-2007 14:58:39 | 0d 3h 15m 41s | 1/4 | OK: Memory Usage 8% - Total: 8195 MB, Used: 676 MB, Free: 7519 MB |
| | PING | OK | 01-26-2007 14:59:15 | 0d 3h 15m 21s | 1/4 | PING OK - Packet loss = 0%, RTA = 0.49 ms |
| | Root Partition | OK | 01-26-2007 14:59:59 | 0d 3h 14m 51s | 1/4 | DISK OK [4196280 kB (99%) free on udev] |
| | SWAP Usage | OK | 01-26-2007 14:55:44 | 0d 3h 14m 1s | 1/4 | Swap ok - (null) 0% (0 out of 2055) |
| | Total Processes | OK | 01-26-2007 14:57:29 | 0d 0h 18m 3s | 1/4 | OK - 88 processes running |

*Figura 4. Nagios 1*

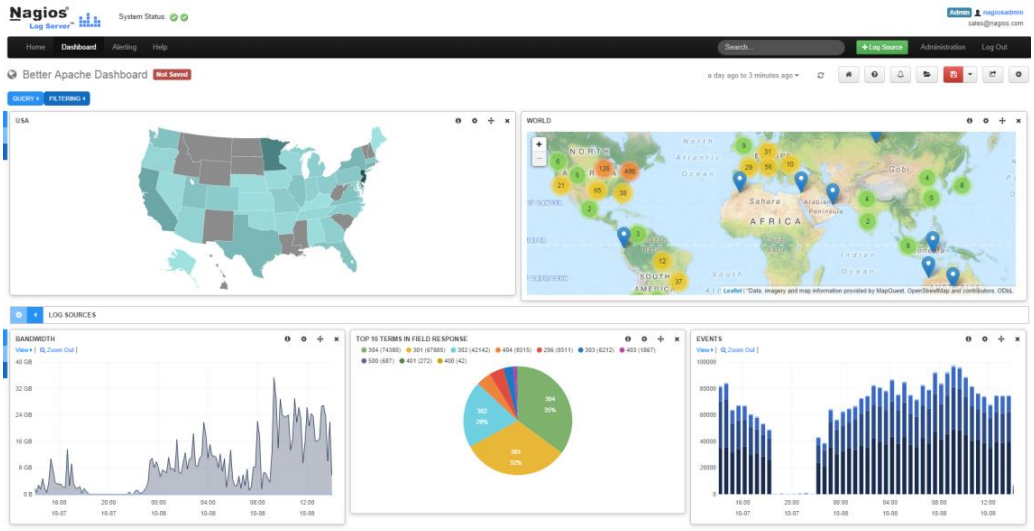



*Figura 5. Nagios 2*

## 3.2. Comparacions i Elecció

Tot i que *Splunk* és una de les millors eines que hi ha actualment al mercat, ja sigui per la seva facilitat o bé per la seva eficiència, el principal problema d'aquesta eina és obtenir la llicencia, ja que en general son bastant cares, i com nosaltres volem solucionar un problema que pel nostre projecte no és crític, aquesta inversió no seria la millor solució possible.

Per acabar de fer la comparació econòmica de les altres dos eines[4], *ELK* és completament gratuït i en canvi *Nagios* s'ha de pagar una llicència, bastant més barata que *Splunk*, per tant podria ser una opció assumible.

Com ja hem comentat abans *Nagios* pot augmentar el seu potencial si li combinem diferents eines, una d'aquestes podria ser *ElasticSearch*, per tant això ens pot indicar quins són els punts dèbils de *Nagios*, que seria en la part de fer la cerca.

Finalment ens decantarem per utilitzar *ELK* ja que creiem que li podem treure un rendiment molt més elevat ja que al dividir-se en tres parts recalcables, ens serà molt més manejable utilitzar-lo en diverses màquines i per tant serà escalable. A més a més podrem configurar més a baix nivell totes les opcions de cerca i tenir un control i coneixement més elevat en tot el procés d'anàlisi de Logs.

## 3.3. Eines addicionals

Addicionalment, per dur a terme aquest projecte utilitzarem tres eines per poder desenvolupar-lo i poder desplegar-lo de la forma més ràpida i eficient possible.

Aquestes eines seran Vagrant[5], Ansible[6], Docker[7] i Node.js[8].

### 3.3.1. Vagrant

Vagrant és va començar a desenvolupar al Gener del 2010 per Mitchell Hashimoto, durant els 3 primers anys, va ser un projecte personal portat a terme per el mateix Hashimoto, on hi treballava en el seu temps lliure. No va ser fins al Novembre del 2012 quan Hashimoto va crear l'empresa HashiCorp i s'hi va dedicar a temps complert, juntament amb la incorporació de nous membres. Aquesta eina és Open Source i ha estat desenvolupada utilitzant el llenguatge Ruby.

Vagrant és una eina que et permet construir entorns complets de desenvolupament, en altres paraules, et crea una màquina virtual amb les característiques desitjades, d'una forma fàcil de configurar, reproduir i de transportar en l'entorn que necessitis.

El potencial d'aquesta eina és que configurant un simple document, pots definir moltes màquines virtuals i gestionar-les, a més a més les pots configurar amb el software desitjat que vols que tingui instal·lat la primera vegada que hi accedeixes i així ja complir tots els teus requeriments.

Per tant per poder utilitzar Vagrant només necessitem instal·lar el binari en la màquina que utilitzarem per desenvolupar el nostre projecte i addicionalment un gestor de màquines virtuals com podria ser Oracle VM Virtual Box.

Cal destacar que aquesta eina només és necessària al moment de dur a terme el desenvolupament del projecte, ja que una vegada estigui implementat ELK, aquest serà instal·lat a les màquines de la empresa que ja compten amb el seu propi sistema operatiu.

### 3.3.2. Ansible

La primer release de Ansible va sortir al mercat el Febrer del 2012, aquesta eina ha estat desenvolupada per l'empresa Red Hat i és una eina d'automatització dels serveis que podria dur a terme un administrador de sistemes.

Tal com s'ha comentat aquesta eina ens permet automatitzar diverses tasques d'un sistema operatiu llençant únicament una única comanda, aquesta eina és molt senzilla de mantenir, degut a com està definida la seva estructura.

Ansible és molt útil en entorns on és tenen moltes màquines i és volen configurar exactament totes igual, ja que només cal tenir l'eina instal·lada en la màquina en qüestió i copiar el directori principal, on està ubicat tota la estructura de fitxers i requeriments que necessita l'usuari. Una vegada és compleixen aquests dos requisits només cal executar la següent comanda.

```
$ ansible-playbook /path/playbook.yml –c servidor
```

La qual és detallarà explícitament en la implementació.

L'ús d'aquesta eina ens aportarà una instal·lació senzilla i ràpida en el desplegament d'ELK a les màquines de la empresa.

### 3.3.3. Docker

La primera versió de Docker va ser llençada al mercat el Març del 2013, per tant és un software relativament nou que ha anat adquirint una gran popularitat i actualment ofereix uns grans avantatges al moment d'utilitzar-lo, concretament pel seu fàcil ús i totes les seves característiques que ens aporta.

Docker és un projecte Open Source que automatitza el desplegament d'aplicacions ubicades dins d'una espècie de contenidors de software. En altres paraules un contenidor conté tot el necessari per a que una aplicació software funcioni (codi, llibreries, dependències, etc.), l'únic que ha de fer l'usuari és executar-lo. A més a més al estar desenvolupat d'aquesta forma et garanteix que sempre s'executarà el mateix software.

Qualsevol usuari pot crear els seus propis contenidors i afegir el que ell desitgi. Per a la realització d'aquest projecte utilitzarem les imatges dels contenidors de la pàgina oficial de Docker, ja que dóna suport a Logstash, ElasticSearch i Kibana i a més a més és van actualitzant a les noves versions.

L'ús d'aquesta eina ens permetrà transportar les imatges d'ELK d'una forma senzilla i sense haver de preocupar-nos per possibles dependències que pugin tenir. A més a més d'instal·lar sempre la mateixa versió.

### 3.3.4. Node.js

Node.js és un software de desenvolupament Open Source i multi plataforma que bàsicament serveix per desenvolupar aplicacions web. La primera release d'aquest software va sortir al mercat el Maig del 2009 i va ser desenvolupat per Ryan Dhal.

Tal com és mostrarà en apartats futurs, al realitzar el projecte en una empresa i treballar per un client, aquest no vol que és mostri cap tipus de dades sensibles que a ell li puguin afectar, per tant per poder mostrar aquest projecte en l'àmbit acadèmic s'ha tingut de crear un servei que llenci Logs ficticis però que tinguin un cert valor informatiu.

Finalment s'ha decidit utilitzar Node.js per dur-lo a terme ja que és un llenguatge que últimament s'ha posat molt de moda i que té un gran potencial.

# 4. Abast

Una vegada que ja sabem quin problema volem solucionar i ja hem estudiat i valorat quines eines hi ha al mercat per poder fer possible el projecte, hem de saber quins seran els requeriments del projecte per poder completar-lo amb èxit. També haurem de tenir en compte quins possibles obstacles ens podem trobar durant la seva realització.

## 4.1. Eines

Finalment ens decantem per utilitzar *ELK*, ja que com hem dit en la comparativa, ens permetrà una major flexibilitat a la hora de desenvolupar tot el projecte i a la vegada tenir una configuració personalitzada i òptima.

## 4.2. Requeriments

- La solució final ha de ser més senzilla d'utilitzar de la que hi ha ara actualment.
- El temps d'utilitzar aquesta solució no pot ser superior al temps que es tarda actualment per fer les cerques desitjades, de fet, hauria de ser molt més baix.
- El resultat final ha de ser el més intuïtiu possible, per a que tots els usuaris en puguin fer un bon ús.

## 4.3. Obstacles

Durant la realització d'aquest projecte poden sorgir diferents problemes o contratemps, si es donés el cas, s'intentaria solucionar-los amb la major rapidesa i solvència possible.

### 4.3.1. Cancel·lació del projecte

Com actualment estem treballant i desenvolupant aquest projecte de cara a un client en concret, si el client decidís que prefereix donar preferència a un altre projecte o bé no el vol prioritzar, s'hauria de complir, i per tant deixar-lo de banda.

### 4.3.2. Problemes amb els Servidors

Aquest projecte està pensat en executar-lo sobre les màquines de producció de l'entitat bancària, per tant, si per algun motiu els servidors deixessin de funcionar aquesta aplicació també ho faria i no es podria provar el seu correcte funcionament.

### 4.3.3. Restricció Temporal

Aquest projecte és durà a terme com un TFG, per tant el temps de que es disposa és bastant limitat, i qualsevol imprevist o mala planificació podria ocasionar que el seu desenvolupament no fos l'adequat.

### 4.3.4. Posposar la distribució al entorn real

Al moment de distribuir aquest projecte al entorn real podria ser que per diverses circumstancies és decidís no distribuir-lo encara i esperar un cert temps, ja que hi poden haver altres prioritats, en aquest cas s'intentaria simular el més possible a la realitat per comprovar el seu correcte funcionament.

# 5. Metodologia i Rigor

## 5.1. Metodologia àgil

Per a la correcta realització d'aquest projecte utilitzarem la metodologia àgil *Scrum* ja que és un model de referència iteratiu i incremental i això ens permetrà poder anar desenvolupant algunes tasques en paral·lel, sense tenir la necessitat de tenir acabada una part per poder avançar amb la següent.

Un altre punt a favor és que durant el transcurs del projecte el resultat final pot variar una mica del prèviament definit amb el client i amb aquesta metodologia això no comporta cap problema.

## 5.2. Eines de seguiment

Per al correcte seguiment de les tasques que estan acabades i de les que hem de realitzar utilitzarem la plataforma *JIRA*[2].

Aquesta eina te un gran potencial, ja que podem preveure el temps estimat que estarem desenvolupant una tasca, i a mesura que l'anem desenvolupant podem anar imputant les hores i així dur un seguiment exhaustiu de si ens sobren hores o bé estem anant massa lents i per tant necessitem reduir temps d'algun altre lloc per poder entregar el projecte a la data acordada.

## 5.3. Eines de control

Com que també haurem de desenvolupar una part de codi, farem servir un gestió de versions com podria ser *GitHub* o un *Subversion*, per així assegurar que no perdem cap part important de codi i si mai volem fer un *rollback* per recuperar informació ens serà molt útil.

[2] Es una plataforma per portar un control intensiu de les hores dedicades a cada tasca.

# 6. Planificació Temporal

## 6.1. Calendari

Aquest projecte té una durada estimada d'uns 4 mesos i escaig, amb inici al 22 de febrer i amb la finalització al 26 de juny.

La dedicació estimada serà d'unes 30 hores setmanals durant 18 setmanes, per tant aquest projecte tindrà una durada total de 540 hores.

## 6.2. Fases del Projecte

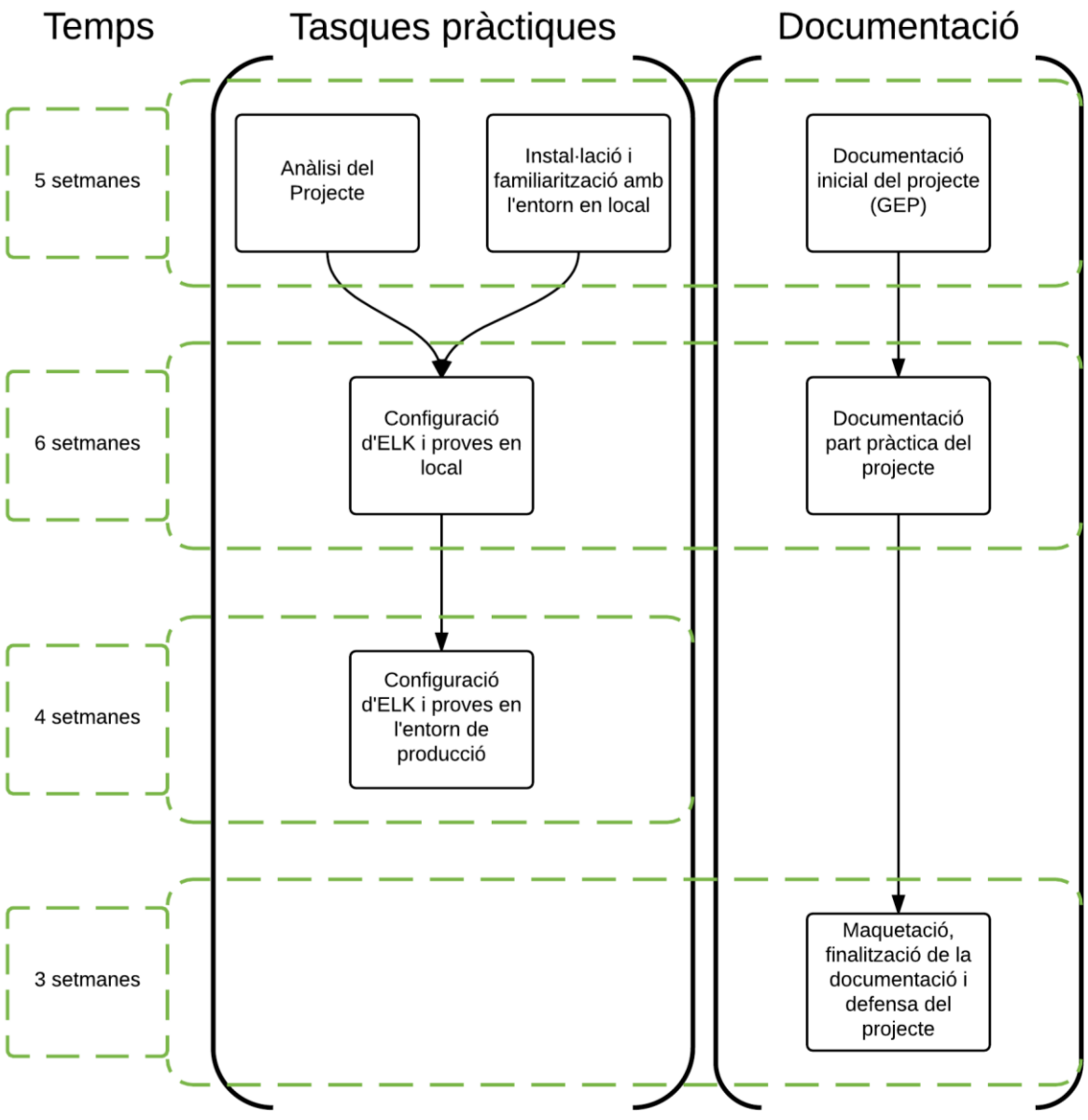



*Figura 6. Fases del projecte*

## 6.3. Diagrama de Gantt

**El diagrama de Gantt és troba en l'annex.**

Aquest diagrama de gantt mostra la divisió de tasques durant el projecte. Com que es un TFG, aquest, estarà desenvolupat completament per jo mateix, amb l'ajuda i consell dels directors i del ponent.

Aquestes seran les tasques prèviament definides en que es dividirà el projecte. La seva durada és pot veure en el mateix Gantt.

### 6.3.1. Primer Lliurable GEP

Aquesta fase pertany al curset introductori de GEP, en la qual començarem a definir l'abast, l'estat de l'art i la contextualització del nostre projecte.

En aquesta tasca no tenim cap risc potencial i els recursos que bàsicament intervindran serà un ordinador, un editor de textos i connexió a internet.

### 6.3.2. Segon Lliurable GEP

Aquesta fase pertany al curset introductori de GEP, en la qual es defineix la planificació temporal per poder dur el projecte amb suficient visió per no tenir problemes de temps, i així ja saber on podrem tenir els principals problemes.

En aquesta tasca no tenim cap risc potencial i els recursos que bàsicament intervindran serà un ordinador, un editor de textos i connexió a internet.

### 6.3.3. Tercer Lliurable GEP

Aquesta fase pertany al curset introductori de GEP, en la qual es tracta la gestió econòmica i la sostenibilitat del projecte, per saber quines despeses tindrem i totes les seves conseqüències.

En aquesta tasca no tenim cap risc potencial i els recursos que bàsicament intervindran serà un ordinador, un editor de textos i connexió a internet.

### 6.3.4. Quart Lliurable GEP

Aquesta fase pertany al curset introductori de GEP, i tracta en realitzar una petita presentació gravada en vídeo de tot el que s'ha desenvolupat fins al moment.

Com a prerequisits tindrem haver realitzat les tres primeres tasques de GEP.

En aquesta tasca no tenim cap risc potencial i els recursos que bàsicament intervindran serà un ordinador, una càmera de vídeo i connexió a internet.

### 6.3.5. Cinquè Lliurable GEP

Aquesta fase pertany al curset introductori de GEP, i tracta de revisar totes les competències especificades al moment d'inscripció del projecte i veure si les complirem tal com nosaltres havíem pensat.

Com a prerequisits tindrem haver realitzat el quart lliurable de GEP.

En aquesta tasca no tenim cap risc potencial i els recursos que bàsicament intervindran serà un ordinador i connexió a internet.

### 6.3.6. Sisè Lliurable GEP

Aquesta fase pertany al curset introductori de GEP, i tracta d'ajuntar tot el que s'ha escrit i transformar-ho en la base del que serà el nostre punt de partida de la memòria del TFG.

Com a prerequisits tindrem haver realitzat el quart lliurable de GEP.

En aquesta tasca no tenim cap risc potencial i els recursos que bàsicament intervindran serà un ordinador, un editor de textos i connexió a internet.

### 6.3.7. Cerca d'informació

Aquesta fase pertany a la cerca d'informació, amb la finalitat d'obtenir tota la informació necessària que necessitem a priori per poder opinar del projecte i intentar buscar la informació més rellevant per a que quan estiguem desenvolupant el projecte sapiguem que hem de fer.

Òbviament durant tot el projecte estarem cercant informació però la fase més exhaustiva serà aquesta per poder conèixer tot aquest món.

En aquesta tasca no tenim cap risc potencial i els recursos que bàsicament intervindran serà un ordinador i connexió a internet.

### 6.3.8. Instal·lació i familiarització amb l'entorn

Aquesta fase pertany al començament del desenvolupament, ja que abans de poder ficar-nos a treballar i desenvolupar el nostre projecte, hem de tenir l'entorn configurat i a més a més que ens sigui còmode poder-hi treballar perquè ja sabem on és troba tot.

Aquesta tasca en principi no ha de suposar cap risc, ja que en la pàgina oficial d'ELK hi ha diferents documents que donen suport per entendreu tot i els recursos que bàsicament intervindran serà un ordinador i connexió a internet.

### 6.3.9. Configuració en local del Logstash

Aquesta fase pertany a la part de desenvolupament del projecte i tracta de configurar el client Logstash en local per a que es comuniqui amb ElasticSearch i que sàpiga quins fitxers ha d'agafar ja que s'ha de fer una tria.

Aquesta tasca té un risc molt petit, ja que en la pàgina web oficial d'ELK hi ha tota la informació corresponent, si fos necessari dedicar més temps en aquesta tasca, no suposaria un gran problema ja que tot i que la configuració d'ElasticSearch ho te com a prerequisit és podrien obtenir manualment el que realitza Logstash.

De recursos bàsicament intervindran un ordinador i connexió a internet.

### 6.3.10. Configuració en local d' ElasticSearch

Aquesta fase pertany a la part de desenvolupament del projecte i tracta de configurar ElasticSearch en local per a que rebi els fitxers que enviem des de el client Logstash prèviament instal·lat, i una vegada ja els té, fa la cerca amb els paràmetres definits per a que ho tregui en un altre fitxer i ho vegi el Kibana.

El perill d'aquesta tasca és molt semblant al del Logstash, tot i que està marcat com a prerequisit per començar a desenvolupar la part de Kibana si tinguéssim algun problema podríem començar a desenvolupar Kibana sense grans problemes.

Els recursos necessaris seran un ordinador i connexió a internet.

### 6.3.11. Configuració en local de Kibana

Aquesta fase pertany a la part del desenvolupament de la part gràfica del nostre projecte que és la que s'encarregarà de mostrar les dades d'una forma simple i fàcil d'entendre per a l'usuari, segurament aquesta tasca serà la més llarga ja que hi ha moltes formes de desenvolupar-la i és poden afegir molts filtres personalitzats.

Aquesta tasca té un gran punt crític i és que si aquesta no és realitza correctament i no funciona com s'espera no podrem avançar en la instal·lació en les maquines que nosaltres desitgem. Per això també és la tasca de desenvolupament on s'hi dedica més temps.

Els recursos necessaris seran un ordinador i connexió a internet

### 6.3.12. Comprovacions en local

Aquesta fase ja pertany a la comprovació del correcte funcionament del nostre projecte en local, on haurem de veure si tenim errors o *bugs* per poder solucionar-los abans de passar a la propera tasca.

Com a prerequisit indispensable tenim haver acabat tot el desenvolupament d'ELK.

En aquesta tasca no tenim cap risc potencial i els recursos que bàsicament intervindran serà un ordinador i connexió a internet.

### 6.3.13. Instal·lació i configuració d'ELK en les màquines reals

En aquesta fase s'instal·laran tots els clients en les màquines desitjades i també és passarà a fer la configuració pertinent que serà molt semblant a la que ja prèviament hem fet en local, però amb petits canvis en les comunicacions de Logstash i ElasticSearch.

Com a prerequisits ens trobem que abans de fer la instal·lació a les màquines el projecte ha d'estar completament funcional i sense errors/*bugs*.

Aquesta tasca no és preveu cap risc però és deixen bastants dies ja que sempre hi ha algun impediment a la hora de realitzar la instal·lació en llocs productius, els recursos que bàsicament intervindran seran les màquines on hem d'instal·lar el projecte i connexió a internet.

### 6.3.14. Comprovacions en les màquines de producció

Tot i que ja hem comprovat que en local funcionava correctament hem de tornat a testejar que funciona correctament a les màquines de producció i no hi ha errors o bugs.

Com a prerequisits ens trobem que tenim de tenir l'entorn instal·lat i configurat correctament en aquestes màquines.

En aquesta tasca no tenim cap risc potencial, els recursos que bàsicament intervindran seran les màquines on hem d'instal·lar el projecte i connexió a internet.

### 6.3.15. Crear un servei que llenci *Logs* amb Node.js

Aquesta part és una tasca extra que hauré de fer per poder ensenyar com funciona el meu projecte en la defensa del mateix, ja que per termes legals no és pot ensenyar codi de l'entitat bancària ni els Logs que creen les aplicacions.

Per tant hauré de fer un servei que anirà creant Logs amb una certa lògica per poder ensenyar el funcionament del projecte, els recursos que intervindran seran un ordinador i connexió a internet.

### 6.3.16. Temps extra per possibles contratemps

Tal com indica, aquesta part servirà per si tenim algun contratemps i endarrerim el projecte, poder tenir un marge de temps per intentar-ho solucionar.

Si tot va bé aquest temps no ens farà falta emprar-lo.

### 6.3.17. Documentació projecte

Aquesta fase tracta sobre la documentació del projecte final, i per tant ja és pot anar desenvolupant a mesura que es va fent el projecte.

En aquesta tasca no tenim cap risc potencial i els recursos que bàsicament intervindran serà un ordinador, un editor de textos i connexió a internet.

### 6.3.18. Presentació per la defensa del TFG

Aquesta fase ja serà la final del projecte i contindrà la creació de la presentació per fer la exposició final.

Com a prerequisits d'aquesta tasca tenim haver acabat completament tot el projecte de manera satisfactòria.

En aquesta tasca no tenim cap risc potencial i els recursos que bàsicament intervindran serà un ordinador, un editor de presentacions i connexió a internet.

# 7. Gestió Econòmica

En aquest apartat farem un anàlisi dels recursos necessaris per a la realització d'aquest projecte. També farem una estimació del cost que aquest suposaria.

## 7.1. Costos directes per tasca

Els costos directes engloben els recursos humans implicats en cada activitat. Per poder decidir aquests costos ens guiarem a través del diagrama de Gantt explicat anteriorment i suposant que les tasques de gestió i documentació ho fa el cap de projecte (45 €/h), les tasques d'anàlisis i disseny les durà a terme l'analista(35 €/h) i les tasques d'implementació les durà a terme el programador(30 €/h).

| ***Activitat*** | **Hores** | **Preu de mercat** | **Cost de mercat** |
|---|---|---|---|
| *Documentació inicial (GEP)* | 78h | 45€ | 3.510 € |
| *Recerca Informació i Anàlisi ELK* | 78h | 35€ | 2.730 € |
| *Instal·lació ELK i Familiarització amb l'entorn* | 30h | 30€ | 900 € |
| *Configuració en local Logstash* | 48h | 30€ | 1.440 € |
| *Configuració en local ElasticSearch* | 36h | 30€ | 1.080 € |
| *Configuració en local Kibana* | 60h | 30€ | 1.800 € |
| *Comprovacions en local* | 30h | 30€ | 900 € |
| *Instal·lació i Configuració producció* | 30h | 30€ | 900 € |
| *Comprovacions en producció* | 18h | 30€ | 540 € |
| *Anàlisis Disseny servei amb Node.js* | 18h | 35€ | 630 € |
| *Creació servei amb Node.js* | 18h | 30€ | 540 € |
| *Documentació projecte* | 36h | 45€ | 1.620 € |
| *Preparació Defensa + Presentació* | 30h | 45€ | 1.350 € |
| *Temps extra* | 30h | | |
| *Total* | **540h** | | **17.940€** |

*Taula 1. Costos directes del projecte*

## 7.2. Costos indirectes

Els costos indirectes engloben la resta de recursos necessaris per a que el desenvolupador pugui a dur a terme el projecte, en el nostre cas serà:

- Connexió a Internet
  - Calculem que la tarifa costa uns 45€ mensuals, durant 5 mesos i que l'utilitzarem 4h al dia, per tant un 16,66%.
- Amortització Hardware
  - El ordinador te un valor de 750€, aquest serà útil durant 4 anys (48 mesos), si el fem servir durant 5 mesos la dedicació serà del 9.6%
- Llum Consumida
  - Consumim 1,8kWh/dia i suposem que cada més té 22 dies hàbils de treball.
- Llicència JIRA
  - La llicència del JIRA ens costa 7€ més durant 5 mesos que dura el projecte.
- Impressió memòria
  - Cada pàgina impresa te un cost de 5 cèntims, suposem que caldrà imprimir unes 600 pàgines en total.

| Producte | Unitats | Preu Unitari | Dedicació | Cost Estimat |
|---|---|---|---|---|
| *Connexió Internet* | *5 mesos* | *45€/mes* | *16,66%* | *37,35€* |
| *Amortització Hardware* | *5 mesos* | *750€* | *9,6%* | *72€* |
| *Llum consumida* | *5 mesos* | *0,15€/kWh* | *1,8kWh/dia* | *29.7€* |
| *Llicència JIRA* | *5 mesos* | *7€/mes* | *100%* | *35€* |
| *Impressió memòria* | *600 pàg.* | *0,05€/pàg.* | *100%* | *30€* |
| *Total* | | | | **204€** |

*Taula 2. Costos indirectes del projecte*

## 7.3. Contingència

Reservarem una part del pressupost per la partida de contingència, concretament reservarem un 15% del cost total, per tenir marge per possibles problemes.

| Producte | Percentatge | Preu | Cost |
|---|---|---|---|
| Costos directes | 15% | 17.940€ | 2.691€ |
| Costos indirectes | 15% | 204€ | 30,6€ |
| *Total* | | | **2.722€** |

*Taula 3. Costos de contingència del projecte*

## 7.4. Imprevistos

Per poder actuar contra imprevistos tenim un marge d'una setmana per poder actuar si tenim algun retràs en alguna tasca que hem descrit al Gantt prèviament i que en els costos directes em deixat sense contar.

| Imprevistos | Probabilitat | Unitats | Preu | Cost |
|---|---|---|---|---|
| Retard 7 dies | 20% | 30h | 35€ | 210€ |
| Avaries | 5% | 1 | 750€ | 37,5€ |
| *Total* | | | | **248€** |

*Taula 4. Cost dels imprevistos del projecte*

## 7.5. Pressupost

El pressupost mostra tots els costos que hem previst que poden anar sorgint a mesura que s'avança en el projecte, per tant el total seria:

| Concepte | Cost |
|---|---|
| Costos directes | 17.940€ |
| Costos indirectes | 204€ |
| Contingència | 2.722€ |
| Imprevistos | 248€ |
| *Total* | **21.114€** |
| *Total + IVA(21%)* | **25.548€** |

*Taula 5. Pressupost del projecte*

# 8. Control de gestió

Per fer un control de gestió de les hores proposades anteriorment s'utilitzarà el JIRA, com ja hem mencionat en l'apartat d'Eines de control. Gràcies aquest programa podrem anar imputant les hores a mesura que les anem realitzant i finalment quedarà quantes hores reals hem dedicat a cada tasca per així quan acabem poder veure on s'ha tingut de dedicar més temps del previst o bé on hem retallat temps.

Una vegada acabat el projecte podem veure que s'han complert bastant bé la previsió de hores. Teníem previstes que el projecte tingués una durada de 540 hores i finalment s'han complert unes 510 hores[3] per dur-lo a terme.

Per tant la desviació total del projecte és:

(hores estimades – hores reals) * cost per hora

(540 - 510) * 30€ = **900€**

Per tant això significa que tenim una desviació positiva de 900€ en aquest projecte.

Si tinguéssim de destacar alguna tasca que finalment ha resultat tenir un desviament molt diferent del que teníem previst seria la tasca d'ElasticSearch, ja que al principi sense saber res vàrem assignar 36 hores de les quals només s'han complert 20 hores, bàsicament perquè la configuració bàsica d'ElasticSearch a nosaltres ja ens servia per la realització d'aquest projecte i s'ha tingut de modificar poca cosa, per altra banda, no s'havia previst que per a la instal·lació d'ELK utilitzaríem les eines d'Ansible i Docker i per tant aquesta part ha tingut una desviació negativa que s'ha igualat amb la desviació positiva d'ElasticSearch.

[3] Aquestes hores son una aproximació ja que al principi no és portava el control adient i no va ser fins una mica després de GEP que és va començar a portar-lo.

# 9. Sostenibilitat

## 9.1. Matriu de Sostenibilitat

| | PPP | Vida útil | Riscos |
|---|---|---|---|
| Ambiental | Consum del disseny | Petjada ecològica | Riscos ambientals |
| | **8** | **15** | **0** |
| Econòmic | Factura | Pla de viabilitat | Riscs econòmics |
| | **5** | **15** | **-5** |
| Social | Impacte personal | Impacte Social | Riscos socials |
| | **5** | **20** | **-10** |
| Rang de sostenibilitat | **20** | **50** | **-15** |
| | **55** | | |

*Taula 6. Matriu sostenibilitat*

## 9.2. Dimensió Econòmica

Tal com ja hem vist en l'apartat anterior, en aquest projecte s'ha fet una estimació dels costs del projecte tant per la part de recursos humans com per la part del material emprat. També s'han tingut en compte diferents problemes que puguin anar sorgint com per exemple la reparació del ordinador.

Aquest projecte podria ser bastant competitiu ja que el seu preu final es el mencionat anteriorment i no disposa de software addicional de pagament, a més a més, el temps d'aquest projecte és molt ajustat ja que és tracta d'un TFG, per tant rarament si podria dedicar molt menys temps.

Com hem pogut observar en el diagrama de Gantt, s'ha dedicat més temps a certes tasques, depenent de la seva importància, per exemple en la part del desenvolupament del Kibana, ja que és la part que veurà l'usuari final i per tant ha de ser el mes entenedora, senzilla i intuïtiva possible, i que mostri el resultat desitjat.

Aquest projecte s'està desenvolupant en l'empresa Everis, jo m'encarrego de la part de la configuració i comunicació d'ELK, tot i això el projecte avarca més propostes i és mes gran, però amb el temps disposat per dur a terme el TFG no és possible d'aplicar-les i desenvolupar-les.

## 9.3. Dimensió Social

Aquet projecte serà útil de cara als treballadors de la empresa perquè tal com s'ha comentat en l'apartat d'actors implicats, implicarà no haver de gastar tant de temps en la cerca de la traça de Log desitjada, i així, agilitzar aquest procés. Per tant amb la realització d'aquest projecte no hi haurà cap col·lectiu perjudicat, ja que s'implementarà un nou servei de cara a tots els seus usuaris.

## 9.4. Dimensió Ambiental

Bàsicament l'impacte ambiental d'aquest projecte no serà molt gran, ja que es desenvoluparà un servei que correrà a les diferents màquines que ja hi ha actualment i per tant no s'hauran de comprar de noves.

Si aquest projecte no fos dut a terme com a TFG l'impacte ambiental seria gairebé el mateix l'única diferencia seria que no caldria fer alguna impressió de paper puntual per algun membre del jurat sinó que és podria penjar en una guia dins l'empresa.

Com ja hem calculat anteriorment la energia requerida per dur a terme el projecte seria d'uns 198kWh (durant el temps de desenvolupar-lo).

Els recursos que necessitarem per dur-lo a terme són: un ordinador, un editor de text i connexió a internet, cal destacar que tot el software utilitzat és gratuït.

# 10. Disseny

Tal com s'ha comentat anteriorment per executar ELK, utilitzarem les eines d'Ansible i Docker per fer-ho d'una forma més senzilla i més automatitzada, per tant un possible esquema d'aquesta solució seria:

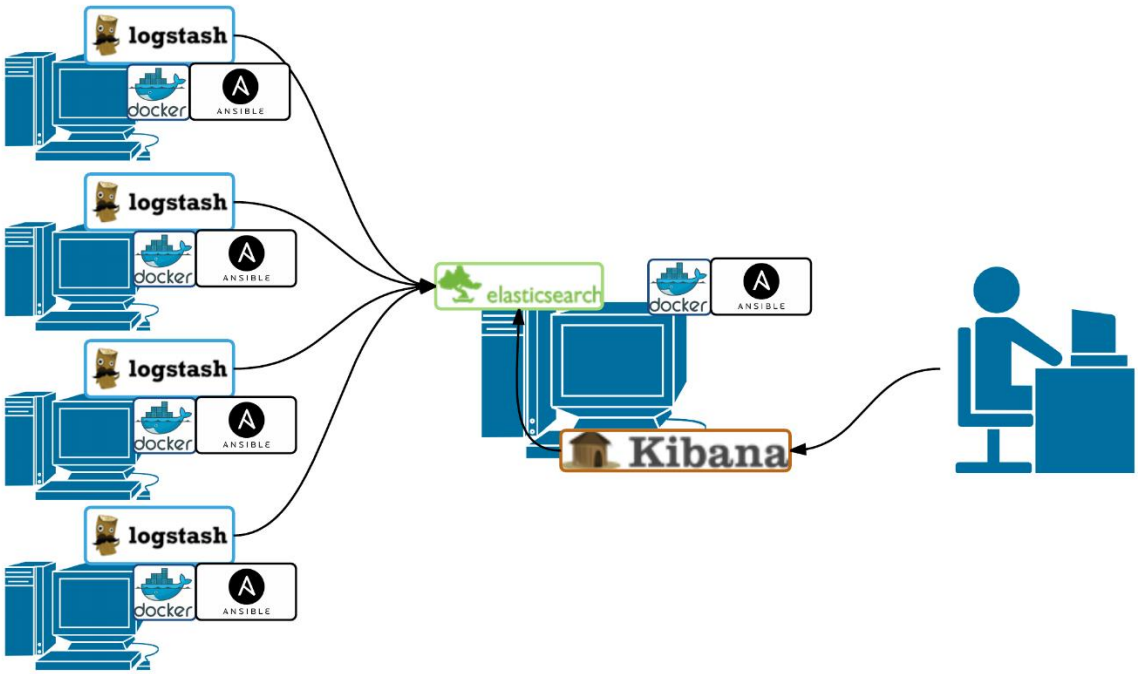



*Figura 7. Esquema Disseny*

La idea bàsica és tenir ElasticSearch i Kibana en un servidor nostre en el que hi puguem interactuar fàcilment. Aleshores només necessitaríem instal·lar els clients Logstash en els servidors del client, d'aquesta forma és fàcilment escalable quan és vulgui afegir algun altre servidor. Cal destacar que tot el que instal·larem als servidors del client, serà el màxim transparent possible, ja que un dels objectius és que aquesta solució consumeixi els menors recursos possibles.

També cal remarcar que gràcies a Ansible i Docker la instal·lació és farà de la manera més fàcil i neta possible.

Addicionalment, per la defensa del projecte, com que no és pot mostrar continguts de l'empresa, utilitzarem un servei Node.js per poder tenir unes dades fictícies i així mostrar la part pràctica sense cap problema.

# 11. Implementació

Una vegada que ja sabem quin és el nostre objectiu, l'abast, requeriments del projecte i disseny, ja podem començar a implementar-lo. A continuació és començarà detallant les eines que duran a terme el desplegament eficient d'ELK amb les parts de la seva configuració i finalment és detallarà Logstash, ElasticSearch i Kibana.

## 11.1. Vagrant

Una vegada tenim Vagrant i el gestor de màquines virtuals instal·lats correctament, el procediment és tant simple com crear un fitxer anomenat "VagrantFile", en el que definirem les propietats principals de la màquina virtual que desitgem crear. Aquest fitxer pot estar en qualsevol directori.

En el nostre projecte una possible configuració seria aquest fitxer, el qual, a continuació s'explicarà:

```
1   VAGRANTFILE_API_VERSION = "2"
2
3   Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
4     config.vm.box = "ubuntu/trusty64"
5     config.vm.synced_folder ".", "/vagrant"
6     config.vm.define "logstash" do |node|
7       node.vm.hostname = "logstash"
8       node.vm.network "private_network", ip: "10.100.199.200"
9       node.vm.network "forwarded_port", guest: 25826, host: 25826
10      node.vm.provision :shell, path: "./vagrant_init/ansible_logstash.sh"
11      node.vm.provision :shell, inline: 'ansible-playbook
    /vagrant/ansible/logstash.yml -c local -v'
12      node.vm.provider "virtualbox" do |v|
13        v.memory = 2048
14      end
15    end
16    config.vm.define "elki" do |node|
17      node.vm.hostname = "elki"
18      node.vm.network "private_network", ip: "10.100.199.201"
19      node.vm.network "forwarded_port", guest: 5601, host: 5601
20      node.vm.network "forwarded_port", guest: 9200, host: 9200
21      node.vm.provision :shell, path: "./vagrant_init/ansible_elki.sh"
22      node.vm.provision :shell, inline: 'ansible-playbook
    /vagrant/ansible/elki.yml -c local -v'
23      node.vm.provider "virtualbox" do |v|
24        v.memory = 2048
25      end
26    end
27  end
```

*Figura 8. Configuració Vagrant*

- **Línia 1**: S'està definint la versió amb la que escriurem el VagrantFile (actualment accepta el valor 1 o 2).
- **Línia 4**: És defineix quina imatge de les possibles[9] utilitzarem en la creació de les següents màquines virtuals, en el nostre cas utilitzarem la versió d'Ubuntu 14.04 LTS de 64 bits.
- **Línia 5**: Es defineix quin o quins directoris estaran sincronitzats amb la màquina virtual.
- **Línia 6-15**: És defineix la configuració d'una màquina virtual.
    - **Línia 7**: És defineix el nom de la màquina virtual.
    - **Línia 8**: És defineix el tipus de xarxa i la seva IP.
    - **Línia 9**: És defineix un *forwarded port*[4] per poder-nos comunicar amb la màquina virtual a través dels ports de la màquina host.
    - **Línia 10-11**: Executa amb la Shell[5] la comanda descrita, més endavant s'explicaran detalladament aquestes comandes.
    - **Línia 12-14**: Definim quanta memòria RAM podrà utilitzar la màquina virtual en qüestió.
- **Línia 16-26**: Creació d'una altra màquina virtual similar.

Per finalitzar, en el nostre cas, haurem de crear uns altres fitxers, el quals seran, anisble_logstash.sh i ansible_elki.sh, i que contindran totes les comandes que necessitarem per instal·lar el software desitjat que volem tenir dins de la màquina virtual. En el cas concret d'aquests fitxers bàsicament s'està instal·lant l'eina Ansible (que explicarem en el següent apartat) i descarregant la imatge de Logstash utilitzant Docker (que també explicarem més endavant).

Una vegada amb aquests fitxers creats i configurats, si des de la ruta on els tenim ubicats, escrivim la comanda:

```
$ vagrant status
```

Obtindrem la informació del estat de les màquines:

```
Current machine states:

logstash          not created(virtualbox)
elki              not created(virtualbox)
```

[4] Permet unir els ports de la màquina virtual amb els ports de la màquina física.
[5] Altrament anomenat, consola de comandes.

Ara mateix com encara no les hem creat, ens dóna aquesta informació, per poder crear/arrancar totes les màquines virtuals definides al fitxer haurem d'escriure:

```
$ vagrant up
```

O bé, per només crear/arrancar una sola màquina:

```
$ vagrant up nom_màquina
```

Una vegada les nostres màquines virtuals estiguin engegades per poder aturar una màquina amb seguretat utilitzarem la següent comanda:

```
$ vagrant halt nom_màquina
```

Si per qualsevol imprevist o canvi en els nostres objectius hem de destruir aquesta màquina, perquè ja no ens fa falta haurem d'utilitzar la següent comanda:

```
$ vagrant destroy nom_màquina
```

S'ha de tenir bastant cura al moment d'utilitzar aquesta comanda, ja que ens eliminarà totes les dades de la màquina virtual del nostre sistema.

Òbviament, Vagrant disposa de moltes més propietats i configuracions per crear màquines virtuals, però que no explicarem, ja que estan totes documentades en la documentació oficial[10] de la seva pàgina web i no s'ha tingut la necessitat d'utilitzar-les.

## 11.2. Ansible

L'estructura d'Ansible és bastant simple, cosa que ajuda a entendre en tot moment el que s'està fent. A continuació, és mostrarà l'estructura de la nostra configuració i s'explicarà pas per pas:

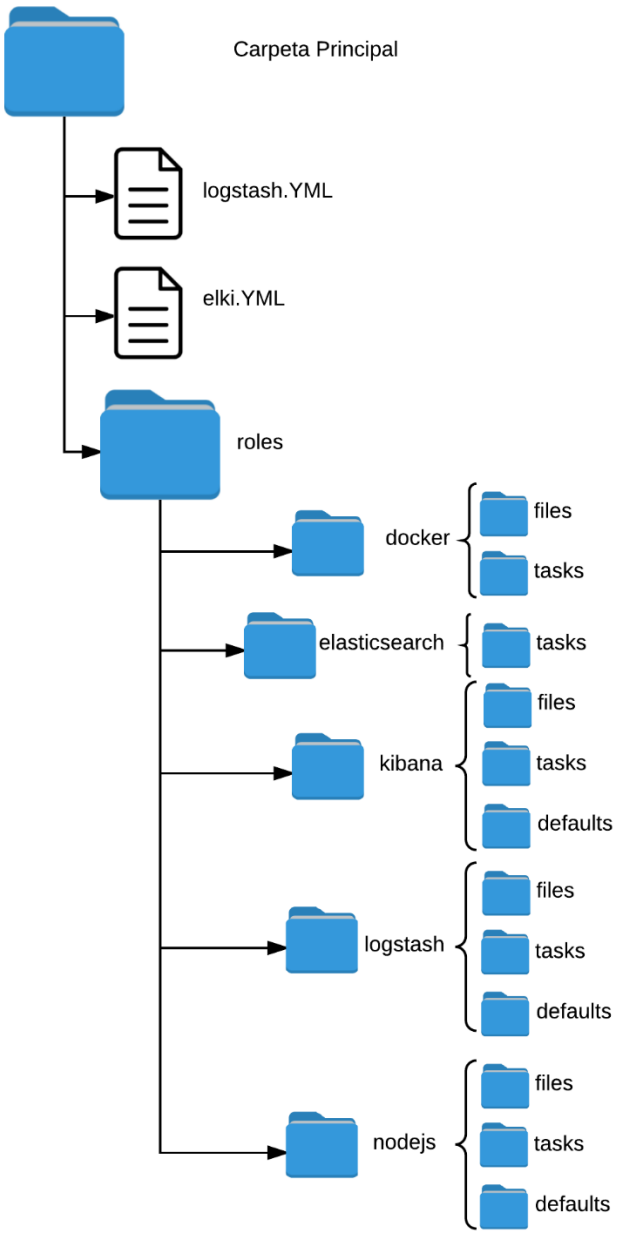



*Figura 9. Estructura de directoris d'Ansible*

Primerament és tracta de tenir en una ruta (directori principal des de on s'executarà el playbook d'Ansible) almenys un fitxer amb extensió YML, en el qual hi haurà definits els rols que executarà el playbook en qüestió. També és poden tenir diferents playbooks, on tinguin definits tots els rols o només una selecció d'ells. Cal destacar que els rols s'executen per ordre d'aparició al playbook. El format bàsic d'aquest fitxer seria el següent:

```
1  - hosts: host
2    sudo: yes|no
3    roles:
4      - role 1
5      - role 2
6      -   .
7      -   .
8      - role N
```

*Figura 10. Format del playbook*

- **Línia 1**: És defineix en quin host ho vols executar, pot ser una IP coneguda o bé en local (*localhost*).
- **Línia 2**: Permís amb que executarà els rols definits.
- **Línia 4-8**: És defineixen els diferents rols.

Un rol pot ser qualsevol intervenció que un administració de sistemes podria fer en el sistema en qüestió, des de fer una instal·lació d'un software determinat o bé una actualització fins a la realització d'un Backup.

Tal com podem observar en la Figura 9, dins del directori roles hi ha els directoris que corresponen a cada rol en general, aquesta estructura bàsicament és per diferenciar les tasques genèriques i fer-ho més fàcil de mantenir i gestionar (Best Practices). Dins d'un rol hi ha una certa estructura de directoris per facilitar el seu manteniment i simplificar el desenvolupament:

- **tasks**: En aquest directori es defineix en un fitxer amb extensió YML, les tasques que durà a terme el rol en qüestió. (Més endavant s'explicarà detalladament aquest fitxer).
- **handlers**: Aquest directori conté en un fitxer amb extensió YML, tots els tipus de *handlers* definits per l'usuari.
- **files**: Aquest directori, conté tots els fitxers extra que utilitzaran les tasques del rol, per exemple arxius de configuració.
- **vars**: Aquest directori conté un fitxer amb extensió YML de totes les variables definides per l'usuari de cada tasca.
- **defaults**: Aquest directori conté un fitxer amb extensió YML, on hi ha definides les variables que s'executaran per defecte en l'execució de la tasca, sempre i quan, no s'hagi redefinit la variable en el directori "vars".
- **meta**: Aquest directori conté en un fitxer amb extensió YML, les dependències que pot arribar a tenir una tasca en concret.

En el nostre cas, tal com és mostra en la Figura 9, podem observar que nomes tenim 3 directoris dels que s'han esmentat anteriorment, això és degut a que no és necessari utilitzar totes aquells directoris i per tant per simplificar no és necessari posar-los, l'únic que és imprescindible és el directori *tasks*.

A continuació és mostraran exemples del format del fitxer ubicat en *tasks*, tal com s'ha comentat anteriorment és on hi ha totes les tasques definides que aquest rol executarà.

### 11.2.1. Exemples

Afegir un repositori:

```
1  - name: Add Docker repository and update apt cache
2    apt_repository:
3      repo: deb https://get.docker.com/ubuntu docker main
4      update_cache: yes
5      state: present
6    tags: [docker]
```

- **Línia 1**: És defineix el nom de la tasca i el valor que és mostrarà quan s'executi un playbook que contingui aquest rol.
- **Línia 2-5**: Aquesta estructura correspon en afegir un repositori al sistema operatiu d'ubuntu, concretament en aquest cas afegim el repositori de Docker.
- **Línia 6**: Defineix quin rol s'està executant actualment i també és mostra en l'execució del playbook.

Instal·lar una aplicació i guardar l'estat:

```
1  - name: Install nodejs
2    apt: name=nodejs state=present
3    register: nodejs_installed
4    tags: [nodejs]
```

- **Línia 2**: S'instal·la el software desitjat, en aquest cas Node.js.
- **Línia 3**: Aquest paràmetre és pot ficar en qualsevol tasca d'Ansible i serveix per guardar quin ha estat l'estat d'executar aquesta tasca (més endavant s'explicarà quins poden ser aquests estats).

Instal·lar una aplicació amb un condicional:

```
1  - name: Install npm
2    when: nodejs_installed|changed
3    apt: name=npm state=present
4    register: npm_installed
5    tags: [nodejs]
```

- **Línia 2**: La opció *when* serveix per realitzar aquesta tasca només si es compleix la condició, en aquest cas, només instal·larem el software *npm* si node.js ha estat instal·lat. Sabrem que ha estat instal·lat perquè la tasca d'abans ha provocat un canvi al sistema si no estava instal·lat.

Ús de la Shell:

```
1  - name: Install moment-timezone
2    shell: npm install moment-timezone
3    when: npm_installed|changed
4    args:
5      chdir: /data/nodejs
6    tags: [nodejs]
```

- **Línia 2**: S'executa a la Shell la comanda desitjada.
- **Línia 5**: Serveix per passar qualsevol paràmetre, en aquest cas, estem obligant que és baixi aquest plugin en la ruta */data/nodejs*.

Còpia d'un arxiu:

```
1  - name: Config file is present
2    copy:
3      src: logstash.conf
4      dest: "{{ conf_dir }}/logstash.conf"
5    tags: [logstash]
```

- **Línia 2**: Executa la comanda de copiar un fitxer**.**
- **Línia 3**: S'especifica el fitxer *source* que volem copiar.
- **Línia 4**: S'especifica el destí del fitxer a copiar. Podem observar com aquí fem ús de la variable *{{ conf_dir }}*, tal com hem comentat abans, aquesta variable està definida en el directori *defaults* d'aquest rol.

A més a més d'aquestes funcionalitats mostrades, Ansible en té moltes de possibles, les quals no mostrarem, ja que estan totes documentades en la seva documentació oficial a la seva pàgina web[11].

Tot i això farem èmfasis a una última tasca d'Ansible específica, que és la que a nosaltres ens aporta un gran valor en aquest projecte:

Posada en marxa d'un Docker:

```
1   - name: Container is running
2     docker:
3       image: logstash
4       name: logstash
5       volumes:
6         - "/data/logstash/config:/conf"
7       expose:
8         - 25826
9       ports:
10        - 25826:25826
11        - 25826:25826/udp
12      command: "logstash -f /conf/meu.conf"
13    tags: [logstash]
```

- **Línia 2-12**: Ens permet executar imatges de Docker amb la configuració desitjada, totes les propietats de Docker s'explicaran en el seu apartat.

Amb aquesta propietat, ens permet integrar i executar Dockers amb molta més facilitat i a més a més automatitzar-los i controlar-los.

### 11.2.2. Execució d'Ansible

Una vegada tenim l'estructura de directoris completada amb els seus fitxers de configuració definits, és el moment d'executar el playbook d'Ansible amb la comanda següent:

```
$ ansible-playbook /path/playbook.yml –c servidor
```

La comanda que mostrem a continuació, executa el playbook anomenat *elki*, el qual conté tot el necessari per a que ElasticSearch i Kibana s'executi correctament.

```
$ ansible-playbook /vagrant/ansible/elki.yml –c local
```

Com nosaltres volem que s'executi en local al moment de seleccionar el servidor hem de seleccionar local, opció -c. Addicionalment podem utilitzar la opció –v per mostrar més informació al moment d'executar, mode *debug*. Una vegada executada aquesta comanda veuríem la següent traça d'informació.

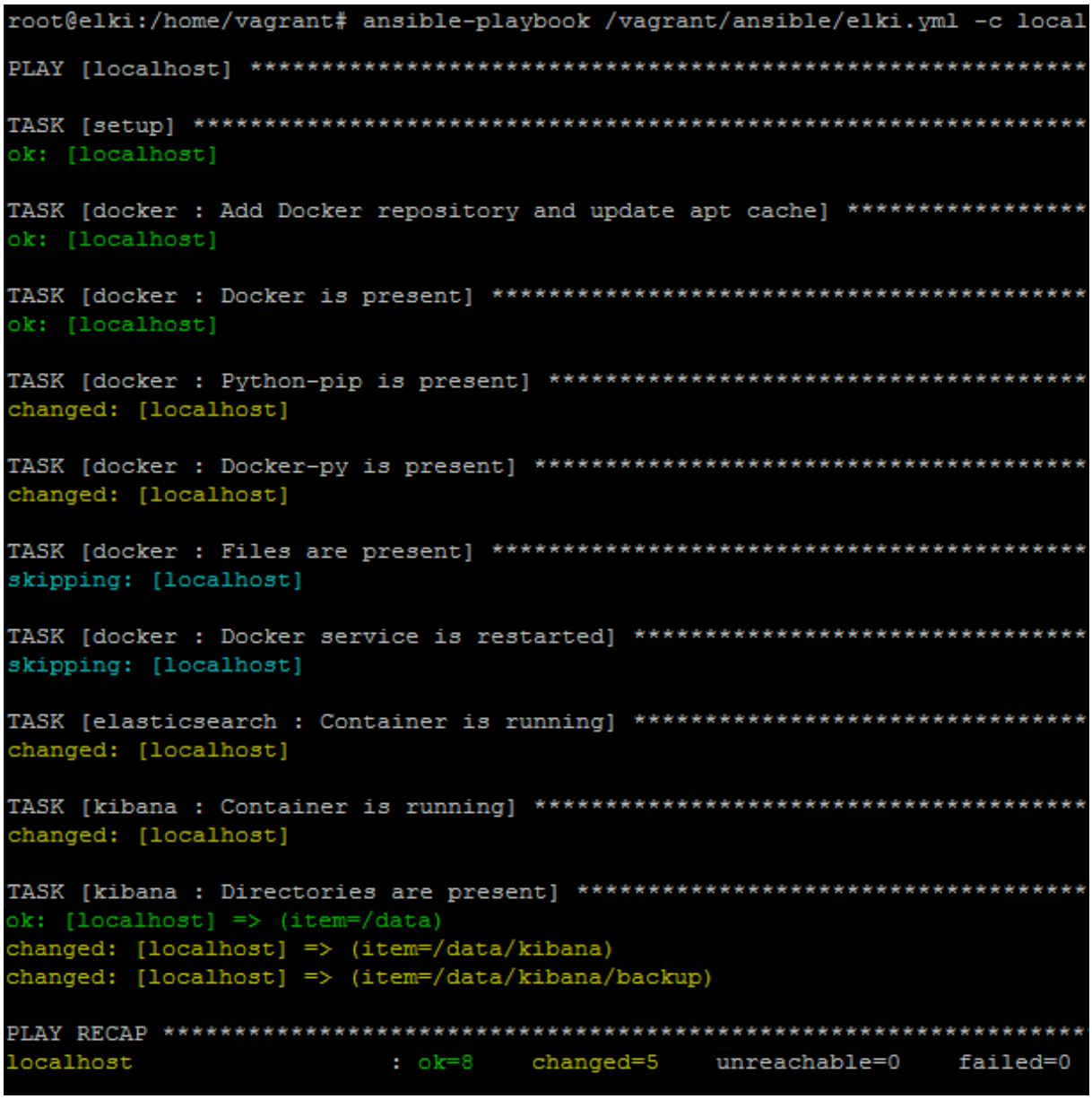



*Figura 11. Execució d'Ansible*

En aquesta imatge podem observar com s'ha executat la comanda i quina informació ens dóna dels rols executats. Quan surt la línia que comença per *TASK* significa que és una tasca definida en el directori *task* de cada rol, per identificar el rol, el podem veure a continuació de la paraula *TASK*, amb el *tag* que nosaltres li hem escrit al fitxer de configuració i finalment el nom que hem definit a cada tasca.

Una vegada intenti executar un tasca aquesta pot acabar en 4 estats diferents:

- **ok**: Significa que s'ha executat, i que no ha provocat cap canvi en el sistema, pot ser degut a que una aplicació que és vol instal·lar ja ho estigui.
- **changed**: Significa que s'ha executat correctament, però que en el sistema s'ha produït un canvi, això passa el primer cop que és fa una instal·lació o bé quan es posa en marxa una imatge d'un Docker per primer cop.
- **unreachable**: Significa que no pot realitzar la tasca desitjada perquè no troba alguna funcionalitat.
- **failed**: Significa que la tasca que ha intentat realitzar ha fallat.

Una vegada finalitzat el playbook, mostra un recompte de tots els resultats, si tot ha anat bé només en tindria d'aparèixer en *ok* i *changed*. Cal destacar que en el moment que el resultat d'una tasca no és cap d'aquests dos, s'avorten les següents tasques.

## 11.3. Docker

El primer que hem de fer una vegada tenim instal·lada l'eina Docker, és baixar-nos les imatges d'ELK amb la següent comanda (en el nostre cas, ho tenim automatitzat):

```
$ docker pull logstash
```

```
$ docker pull elasticsearch
```

```
$ docker pull kibana
```

Per no tenir problemes en un futur escollirem que sempre es baixi la mateixa versió de cada producte, en el nostre projecte utilitzarem les següents versions:

- Logstash:2.3.2
- ElasticSearch: 2.3.3
- Kibana: 4.5.1

Una vegada tinguem tot això ja podrem crear la tasca per a que Ansible executi els contenidors.

A continuació és mostrarà les tasques de cada contenidor que s'executen en l'Ansible i s'explicaran a baix nivell, ja que tota la informació a ELK és mostrarà en els següents apartats.

Aquí podem veure la tasca de Logstash:

```
1   - name: Container is running
2     docker:
3       image: logstash
4       name: logstash
5       volumes:
6         - "/data/logstash/config:/conf"
7       expose:
8         - 25826
9       ports:
10        - 25826:25826
11        - 25826:25826/udp
12      command: "logstash -f /conf/logstash.conf"
13    tags: [logstash]
```

- **Línia 3**: Defineix quin Docker vol que s'executi.
- **Línia 4**: Defineix quin serà el nom del Docker, per poder operar amb ell.

- **Línia 5 i 6**: Defineix quina ruta voldrà compartir amb el Docker. Això significa que la ruta de la nostra màquina amb els seus fitxers és trobaran al Docker en el directori /conf.
- **Línia 7 i 8**: Especifica en quin port estarà escoltant el contenidors.
- **Línia 9 - 11**: Defineix amb quin port podràs accedir en el contenidor, també és pot especificar el protocol.
- **Línia 12**: Executa addicionalment la comanda especificada.

Aquí podem veure la tasca d'ElasticSearch, on tots els seus paràmetres han estat explicats en la tasca Logstash.

```
1  - name: Container is running
2    docker:
3      image: elasticsearch
4      name: elasticsearch
5      ports:
6        - "9200:9200"
7      volumes:
8        - /data/elasticsearch:/usr/share/elasticsearch/data
9    tags: [elasticsearch]
```

Aquí podem veure la configuració del Kibana:

```
1  - name: Container is running
2    docker:
3      image: kibana
4      name: kibana
5      ports:
6        - 5601:5601
7      links:
8        - elasticsearch:elasticsearch
9    register: kibana_result
10   tags: [kibana]
```

- **Línia 7 i 8**: Com que ElasticSearch i Kibana estan en la mateixa màquina, es pot definir que els Dockers és comuniquin entre ells utilitzant un link.

Una vegada introduïda la configuració en les tasques d'Ansible i d'haver executat el playbook, per comprovar que les imatges dels contenidors estan executant-se i a més a més observar certa informació, podem executar la comanda següent:

```
$ docker ps -a
```

Aquesta comanda ens retornarà la següent informació:

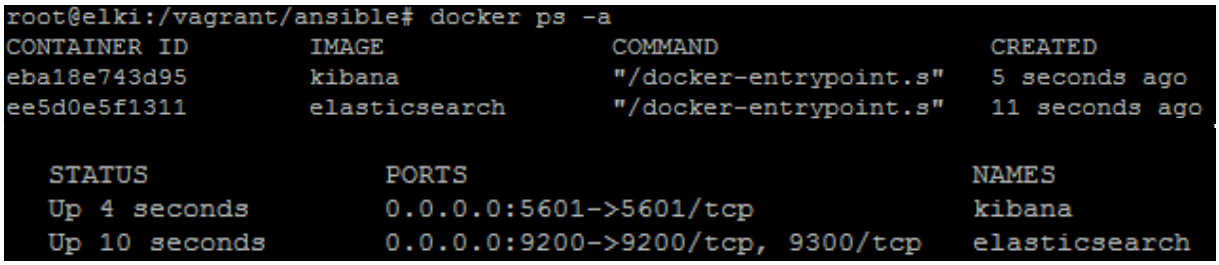



*Figura 12. Docker ps -a*

A continuació s'explicarà què significa cada paràmetre:

- **Container ID**: Conté l'ID únic de cada contenidor.
- **Image**: Mostra el nom per defecte del contenidor.
- **Command**: Ens mostra la comanda que ha utilitzat del contenidor.
- **Created**: Mostra el temps que fa que ha estat creada la imatge del contenidor.
- **Status**: Mostra l'estat del contenidor, per saber si està executant-se o bé s'ha caigut.
- **Ports**: Mostra quins ports s'han definit.
- **Names**: Mostra els noms que nosaltres em definit al moment d'executar el Docker.

Una vegada tenim les imatges de Docker corrent correctament si volguéssim fer algun canvi en algun Docker, podríem accedir-hi dins utilitzant la següent comanda:

```
$ docker exec -it NAME /bin/bash
```

Les funcionalitats de Docker, són molt més extenses, ja que com hem comentat abans, és pot crear els teus propis contenidors introduint el software que l'usuari desitgi, si és volen consultar totes les seves funcionalitats complertes i totes les seves propietats, és poden trobar en la documentació oficial de Docker en la seva pàgina web[12], ja que nosaltres ens hem centrat en el moment de posar-los en marxa.

## 11.4. Node.js

La idea bàsica és crear un servei que ens insereixi Logs en un fitxer i a més a més que tinguin un cert sentit per després poder-los interpretar i poder extraure tot el potencial d'ELK.

Per mostrar que ELK accepta de tot tipus de traces de dades el que farem serà crear 3 serveis diferents.

El primer llençarà Logs amb el format:

Data i Hora | Tipus de Log | Servidor | User | Entorn | Missatge

El segon llençarà Logs amb el format:

Data i Hora | Tipus de Log | Servidor | Entorn | Permisos | Missatge

El tercer llençarà Logs amb el format d'un erro de Java (*StackTrace*)

Els valors de cada variables podran ser els següents:

- Data i Hora: Data i Hora de quan és crea el Log.
- Tipus de Log: Info, Error i Debug.
- Servidor: Alpha, Bravo, Charlie, Delta, Echo, Foxtrot i Golf.
- User: 0539, 8022, 7394, 9346, 7204, 1993, 6666, 9438, 0930 i 6307.
- Entorn: Desenvolupament, Test, PreProducció i Producció.
- Permisos: Alt, Mitja i Baix
- Missatge: Missatges variats depenen dels paràmetres anteriors seleccionats.

A continuació és mostrarà una part del codi desenvolupat.

```
1  var fs = require("fs");
2  var sleep = require('/data/nodejs/node_modules/sleep');
3  while(1){
4        var tipo = get_tipus();
5        var server = get_servidor();
6        var usuari = get_usuari();
7        var entorn = get_entorn();
8
9        var operacio = get_operacio(tipo);
10
11       fs.appendFileSync('/data/logstash/config/logs/tfg_logs1.txt',
12 data() + ' [' + tipo + '] [' + server +'] [' + usuari +'] [' +
13 entorn +'] - ' + operacio + '\n');
14       sleep.usleep(100000 * random(7,25));
15       debug_consola();
16 }
```

No mostrarem el que fan les funcions ja que l'únic que fan és agafar un valor aleatori amb unes certes probabilitats, per tant assumirem que cada funció retorna el resultat que correspon camp.

Els serveis bàsicament el que faran serà esperar-se un cert temps i llençar una traça d'informació aleatòria però, que depenen el tipus de valors a les variables assignades tregui un missatge adient a aquelles.

Cada servei node.js guardarà els resultats en documents diferents i Logstash serà l'encarregat d'obtenir les dades.

Cal destacar que els Logs s'han desenvolupat de la forma que s'assemblin més a la realitat però utilitzant dades fictícies.

## 11.5. Logstash

Logstash és el client que s'instal·la a la màquina desitjada per a que pugui transmetre cap a ElasticSearch tota la informació de les dades que nosaltres desitgem.

Partint de la idea de que utilitzarem els mètodes anteriorment explicats per inicialitzar Logstash, l'únic que haurem de definir és el fitxer de configuració que llençarà Ansible. El fitxer en qüestió s'anomena logstash.conf, i en el nostre cas està ubicat en la ruta */data/logstash/config/logstash.conf*.

Aquest fitxer conté tota la informació relativa que utilitzarà per poder obtenir les dades, filtrar-les, *parsejar*-les i finalment enviar-les cap a ElasticSearch.

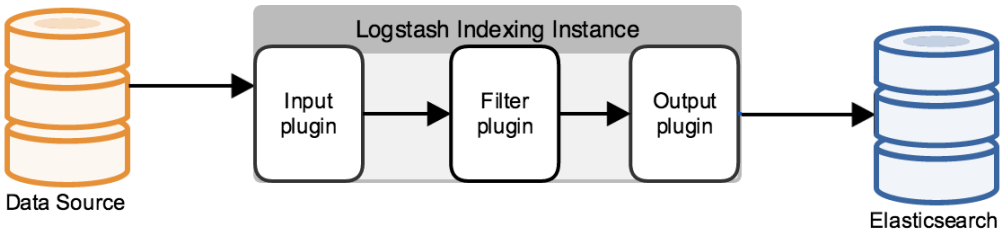



*Figura 13. Estructura Logstash*

El fitxer de configuració Logstash conté tres seccions diferenciables:

```
input {
  ...
}
filter {
  ...
}
output {
  ...
}
```

Cada secció pot contenir un o més *plugins*, un *plugin* ve definit per un nom genèric i un bloc de propietats que són els que el configuren. A continuació explicarem quina és la funció de cada part.

- **Input**: En aquesta secció és defineix la ubicació d'on el client Logstash estarà escoltant i quina informació obtindrà.
- **Filter**: En aquesta secció és defineixen els patrons de les dades, ja que quan és reben només son considerades com un simple *String*.
- **Output**: En aquesta secció és defineix en quina ubicació enviarà les dades rebudes.

Una vegada tenim clar que defineix cada secció, entrarem amb més de detall al que és pot fer en cada una i mostrarem la nostra configuració.

### 11.5.1. Input

Hi ha una gran quantitat de *plugins* que és poden utilitzar per recuperar les dades que l'usuari desitja, aquí només s'anomenaran una petita quantitat d'aquests i s'entrarà amb més detall al que s'ha utilitzat per dur a terme el projecte.

- Exec: Aquest *plugin* captura el resultat d'una comanda llençada des d'una *Shell*.
- File: Aquest *plugin* ens permet obtenir les dades dels fitxers de text.
- Http: Aquest *plugin* ens permet obtenir dades d'esdeveniments via *http* o *https*.
- Log4j: Aquest *plugin* permet rebre esdeveniments sobre *sockets* TCP que utilitzen log4j.
- Pipe: Aquest *plugin* permet rebre dades de esdeveniments que estiguin executant-se utilitzant una *pipe*.
- Sqlite: Aquest *plugin* ens permet obtenir les dades d'esdeveniments d'una base de dades *SQLite*.
- Udp: Aquest plugins ens permet obtenir dades d'esdeveniments via *udp*.

En el nostre cas, com hem d'obtenir dades de serveis i aplicacions, aquestes és troben en fitxers de text, per tant haurem d'utilitzar el *plugin file*.

A continuació és mostra la nostra part de configuració del Input:

```
1  input {
2     file {
3      path  => "/conf/logs/tfg_logs*.txt"
4
5      start_position => "end"
6
7      codec => multiline {
8        pattern => "^\s|^\t"
9        what => "previous"
10     }
11   }
12 }
```

En la nostra configuració podem observar com estem dient que ens agafi la informació dels fitxers que estiguin en la ruta */conf/logs/* i que el nom del fitxer sigui tfg_logsXXX.txt on XXX significa qualsevol nom de qualsevol llargada.

També veiem la propietat *start_position* que està definida en *end* això significa que cada cop que s'actualitzi el fitxer només agafarà la nova informació escrita, sinó agafaria totes les dades del fitxer un altre cop.

La propietat còdec conjuntament amb *multiline* ens permet ajuntar diferents línies de dades en un sol esdeveniment ja que Logstash separa els esdeveniments per línia, aquest *plugin* l'utilitzem bàsicament per el tractament dels errors, ja que quan algun servei o aplicació en llença un, aquest no està especificat en una sola línia, sinó que estan descrits en diverses (*Stacktrace*).

Aleshores el que definim concretament en aquesta propietat és que si la línia de dades que estem obtenint comença amb un espai o bé una tabulació (format del *Stacktrace* de Java) significa que pertany al conjunt de dades anteriors.

Aquesta és la nostra configuració d'input, però tal com hem comentat abans hi ha molts altres tipus d'obtenir dades i de configuracions que no explicarem ja que estan documentades a la documentació oficial de Logstash en l'apartat del *plugin* d'Input[13].

### 11.5.2. Filter

Hi ha una gran quantitat de *plugins* que és poden utilitzar per tractar les dades que rebem del *input*, a continuació és nombraran alguns dels *plugins* més interessants per la selecció de les dades i s'entrarà en detall amb els que hem utilitzat.

- Aggregate: Ens permet agregar diferents esdeveniments com si fos una única tasca.
- Checksum: Ens permet crear un *checksum* dels esdeveniments per poder comprovar que no han hagut modificacions.
- Drop: Ens permet eliminar esdeveniments que provinguin del *input*.
- Grok: Ens permet analitzar gramaticalment el text rebut des del *input* i a més a més donar-li una estructura (és un dels plugins essencials i amb mes potencial).
- Json: Ens permet analitzar gramaticalment un text en format JSON.
- Mutate: Ens permet canviar el nom de diversos paràmetres.
- Multiline: Ens permet ajuntar diverses línies de text rebudes en un sol esdeveniment.
- Uuid: Ens afegeix un ID únic per a cada esdeveniment.
- Xml: Ens permet analitzar gramaticalment un text en format XML.

En el nostre cas, com estem rebent línies de text, bàsicament utilitzarem el *plugin grok*.

A continuació es mostra la nostra part de configuració del *Filter*:

```
filter {
  grok {
    match => {
      "message" => [
        "(?<TimeStamp>[a-zA-Z0-9:, /]{23})
         (?<Tipus>[a-zA-Z:'\[\]]{0,10})
         (?<Servidor>[a-zA-Z0-9:'\[\]]{0,15})      Comprovació 1
         (?<User>[0-9\[\]]{6})
         (?<Entorn>[a-zA-Z0-9:'\[\]]{0,20})" ,

        "(?<TimeStamp>[a-zA-Z0-9:, /]{23})
         (?<Tipus>[a-zA-Z:'\[\]]{0,10})
         (?<Servidor>[a-zA-Z0-9:'\[\]]{0,15})      Comprovació 2
         (?<Entorn>[a-zA-Z0-9:'\[\]]{0,20})
         (?<Permisos>[a-zA-Z0-9:'\[\]]{0,15})",

        "(?<StackTrace>(.|\r|\n|\t)*)"             Comprovació 3
      ]
    }
      remove_field => [ "@version"]
  }
  if "multiline" in [tags] {
    mutate {
      add_field => [ "Tipus", "[Error]" ]
    }
  }
}
```

En la nostra configuració podem observar com utilitzem el *plugin grok* esmentat anteriorment, bàsicament estem comprovant quin patró segueix la traça que ens arriba, en el cas de la comprovació 1, estem mirant que la traça comenci amb el següent format:

Data i Hora | Tipus de Log | Servidor | User | Entorn

En la segona comprovació comprovem que tingui el següent format:

Data i Hora | Tipus de Log | Servidor | Entorn | Permisos

I en la última comprovació acceptem qualsevol informació.

Si ens fixem la forma de definir quins caràcters estan permesos la sintaxi és de expressions regulars de Oniguruma[6,[14]].

[6] Oniguruma és un llenguatge que defineix expressions regulars de tal forma que és pot indicar quins caràcters estan permesos i quina llargada ha de tenir la paraula en concret per a que és compleixi la condició.

Cal destacar que les comprovacions han d'anar de més restrictives a menys, ja que la primera que és compleixi serà la seleccionada, per tant cal tenir molta cura al moment de decidir les restriccions i l'ordre.

Una vegada s'hagi complert una condició s'assignaran les paraules a la variable designada per més tard fer les cerques amb ElasticSearch.

A més a més el *plugin grok* ens permet eliminar els camps que nosaltres no volem mantenir, en el nostre cas, la versió no ens interessa i per tant la podem eliminar.

Si fem memòria, recordarem que en l'input s'ha definit un còdec amb el *plugin multiline*. Aquest *plugin* quan acaba de generar un esdeveniment deixa en un camp la paraula *multiline*, per tant quan trobem que en aquest camp hi hagi aquesta condició sabrem que és tracta d'un error i per tant per donar més valor al error li afegirem que és de tipus error.

En el nostre cas només ens ha sigut necessari utilitzar aquestes propietats però podem observar la essència de que si sabem com estan definits els Logs podem treure molt de valor en ells i estructurar-los tots.

Tal com ja hem comentat abans hi ha molts altres *plugins* que és poden trobar definits en la documentació oficial de Logstash en l'apartat del plugin *Filter*[15].

### 11.5.3. Output

Hi ha una gran quantitat de *plugins* que és poden utilitzar per comunicar-nos on hem d'enviar les dades, a continuació nombrarem alguns plugins interessants per l'enviament de les dades i entrarem en més detall en el que em utilitzat nosaltres:

- Csv: Escriu les dades en un document amb format *csv*.
- Cloudwatch: Permet enviar les dades cap a AWS[7].
- ElasticSearch: Permet enviar les dades a ElasticSearch.
- File: Guarda les dades en un fitxer de text.
- Nagios: Permet enviar les dades a Nagios.
- TCP: Envia els esdeveniments via un *socket* TCP.
- UDP: Envia els esdeveniments via UDP.

[7] Amazon Web Services.

A continuació és mostra la nostra part de configuració del Output:

```
output {
  elasticsearch {
    hosts => "10.100.199.201:9200"
  }

}
```

En la nostra configuració podem observar com només en fa falta indicar que les dades s'enviaran a ElasticSearch i especificar la IP. És poden definir diferents IP's i també si pot afegir un Proxy en cas de que sigui necessari. Com la demostració és fa en local només cal assignar la IP que s'ha definit a la màquina virtual que conté ElasticSearch.

També és pot definir amb quin índex de cerca volem que és guardi a ElasticSearch, per defecte és *Logstash-any.mes.hora*. En el nostre cas creiem que és útil guardar-ho així ja que amb una simple comanda de *cron*[8], podem fer que cada dia s'eliminin els Logs que fa més d'una setmana que estan emmagatzemats, i així no haver-nos de preocupar per l'espai que poden ocupar tots els Logs.

Tal com s'ha comentat abans hi ha molts altres tipus de *plugins* per el *Output* i que és poden trobar definits en la documentació oficial de Logstash en l'apartat del plugin *Output*[16].

[8] Cron és un procés en *background* (*deamon*) que executa les tasques desitjades per l'usuari cada cert període de temps.

## 11.6. ElasticSearch

Una vegada ja tenim ElasticSearch executant-se amb la configuració correcta, Logstash ja hi pot enviar les dades.

Una vegada comencem a enviar dades, si accedim a la URL:

*host_name:9200/_cat/indices?v*

veurem la següent informació:

```
health status index                pri
yellow open   .kibana                1
yellow open   logstash-2016.06.13    5

docs.count docs.deleted store.size pri.store.size
        24            0     47.9kb          47.9kb
        70            0      225kb           225kb
```

*Figura 14. Informació ElasticSearch*

- **Health**: Aquesta propietat ens dóna informació de com tenim el node d'ElasticSearch. Si està en *green* significa que tot funciona correctament i que tenim mes d'un node, si està en *yellow* significa que funciona correctament però que només tenim un node i si esta en *red* significa que no funciona.
- **Index**: Fa referència amb l'índex que s'ha guardat a ElasticSearch.
- **Pri**: Indica la prioritat d'aparició de cada índex.
- **Docs.count**: Indica quants esdeveniments s'han registrat amb aquest índex.
- **Docs.deleted**: Indica quants esdeveniments s'han esborrat amb aquest índex.
- **Store.size**: Indica la quantitat de memòria que ocupa aquest índex emmagatzemat al disc.
- **Pri.store.size**: Indica la quantitat de memòria que ocupa aquest índex emmagatzemat al disc sense contar les repliques si n'hi ha (en altres nodes).

ElasticSearch també et permet tenir diferents clústers i nodes per l'obteniment de les dades, i d'aquesta forma si alguna instància es queda aturada pots continuar rebent les dades a les altres. Tenir diferents clústers significa tenir un balancejador de carrega per a detectar on enviar les dades que es tenien de transmetre al node que actualment estaria caigut, per tant necessitem una eina addicional com podria ser l'ús de Nginx. Aquesta part no l'hem dut a terme ja que no

estava plantejada al inici del projecte, i requeria dedicar-hi més temps per poder-la completar amb èxit.

Per altra banda una vegada tenim les dades emmagatzemades a ElasticSearch, les necessitem recuperar utilitzant certes crides, tal com veurem mes endavant Kibana ens proporciona una estructura gràfica i fàcil d'utilitzar per abstraure al desenvolupador de fer totes les cerques manuals.

Però tot i això és mostraran alguns exemples de com fer cerques a ElasticSearch ja que per un futur podria ser útil saber-ho, ja que Kibana et permet personalitzar les cerques i al final et pots crear les teves cerques personalitzades ja que les cerques en ElasticSearch són com una API Rest.

### 11.6.1. Cerques

Podem "crear" tot tipus de cerques:

**Exemple 1**

```
curl -XPOST 'localhost:9200/logstash*/_search?pretty' -d '
{
  "query": { "match": { "Tipus": "[Info]" } }
}'
```

Accedeix a la ruta on és troba ElasticSearch indicant el índex que és vol consultar i realitza la cerca de totes les dades que tinguin un camp que es digui "Tipus" amb valor "[Info]".

**Exemple 2**

```
curl -XPOST 'localhost:9200/logstash*/_search?pretty' -d '
{
  "query": {
    "bool": {
      "must": [
        { "match": { "Tipus": "[Info]" } },
        { "match": { "Entorn": "[Produccio]" } }
      ]
    }
  }
}'
```

En aquest cas estem cercant tots els Logs que continguin un camp que es digui "Tipus" amb valor "[Info]" i un altre que es digui "Entorn" amb valor "[Producció]".

És pot canviar el *must* per *should*, i seria com fer una *OR* o bé *must_not* i retornaria totes les que no continguin aquests valors.

També és poden combinar aquests:

```
curl -XPOST 'localhost:9200/logstash*/_search?pretty' -d '
{
  "query": {
    "bool": {
      "must": [
        { "match": { "Tipus": "[Info]" } }
       ],
      "must_not": [
        { "match": { "Entorn": "[Produccio]" } }
       ]
    }
  }
}'
```

També podem aplicar filtres, per definir intervals:

**Exemple 3**

```
curl -XPOST 'localhost:9200/bank/_search?pretty' -d '
{
  "query": {
    "bool": {
      "must": { "match_all": {} },
      "filter": {
        "range": {
          "balance": {
            "gte": 20000,
            "lte": 30000
          }
        }
      }
    }
  }
}'
```

Aquesta consulta la realitzem sobre una base de dades fictícia d'un banc ja que en el nostre cas no tenim cap variable declarada com un número.

El que faria aquest filtre seria retornar tots els usuaris que tinguin un compte entre 20.000 i 30.000€.

### Exemple 4

També podem definir agregacions, que seria com fer el típic *group_by* en una base de dades.

```
curl -XPOST localhost:9200/logstash*/_search?pretty' -d '
{
  "size": 0,
  "aggs": {
    "group_by_Servidors": {
      "terms": {
        "field": "Servidors"
      }
    }
  }
}'
```

Aquesta consulta et retornaria el numero de Logs que coincideixen amb cada servidor.

Una vegada és saben les cerques més bàsiques és poden generar de més complexes per treure exactament la dada que és vulgui.

En aquest apartat no veurem més exemples ja que poden trobar-se explicats a la documentació de la pàgina oficial de ElasticSearch[17].

## 11.7. Kibana

Una vegada ja tenim ElasticSearch executant-se correctament i Logstash enviant les dades cap ElasticSearch, ja podem utilitzar l'eina del Kibana, per accedir-hi hem d'entrar a la ruta on hem instal·lat la imatge, en el nostre cas com és en una màquina virtual i em fet *port forwarding*, hi podem accedir tant per *localhost:5601* o bé utilitzant la IP de la màquina virtual amb el port 5601. Kibana per defecte ataca al port 5601.

Una vegada estiguem a dins ens apareixerà una pantalla d'aquest estil.

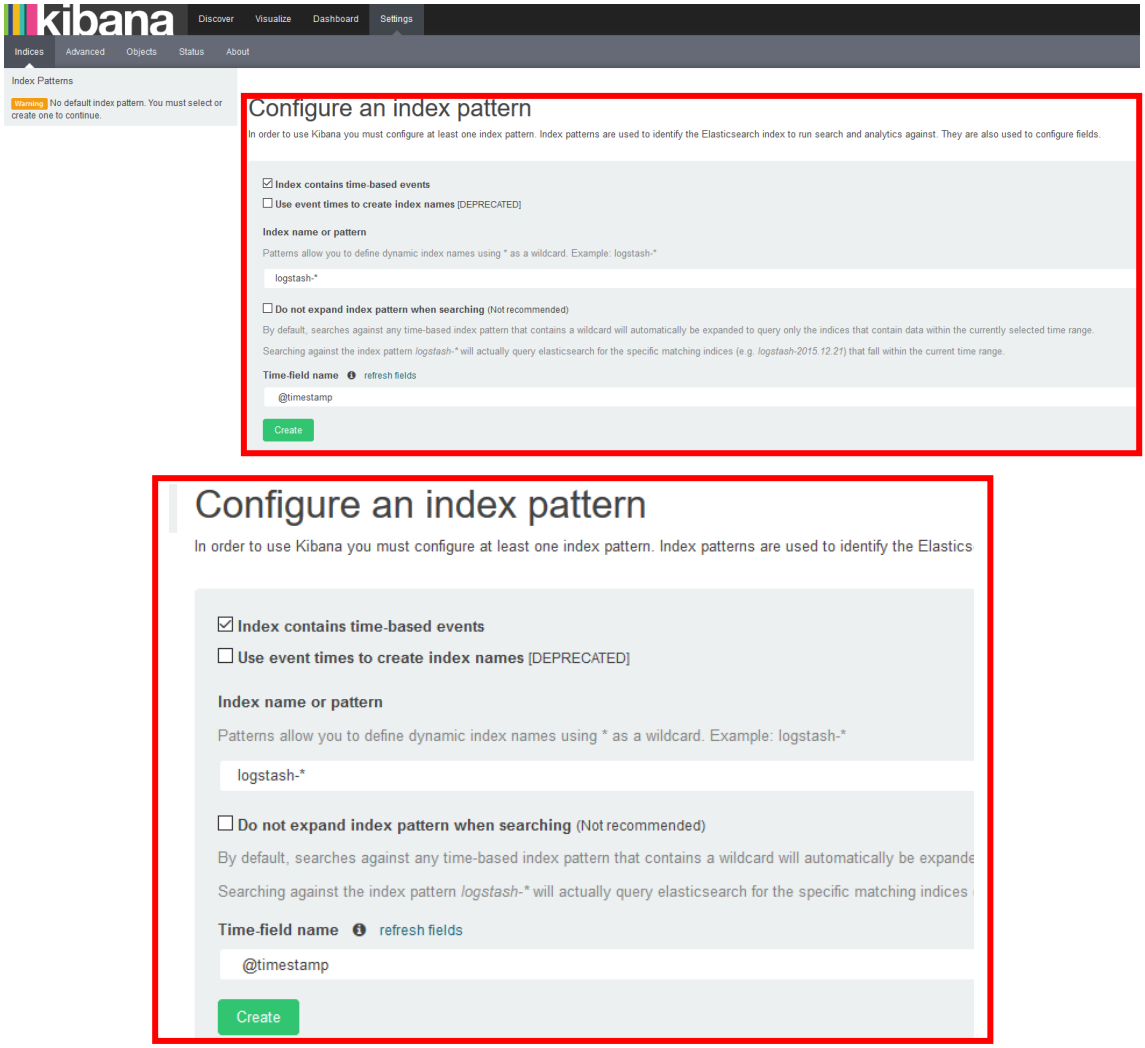



*Figura 15. Pàgina d'inici Kibana*

Primer de tot hem de seleccionar quin índex volem buscar, en el nostre cas seleccionarem *Logstash-**, que significa que cercarem tots els índexs que continguin la paraula Logstash seguidament d'altres caràcters. També haurem de seleccionar una variable de temps, en el nostre cas utilitzarem la que ve per defecte i que mostra el moment en que s'ha afegit aquest esdeveniment a ElasticSearch.

Una vegada tinguem seleccionat un índex apareixerà a la part superior esquerre i ens mostrarà quines variables conté aquest índex.

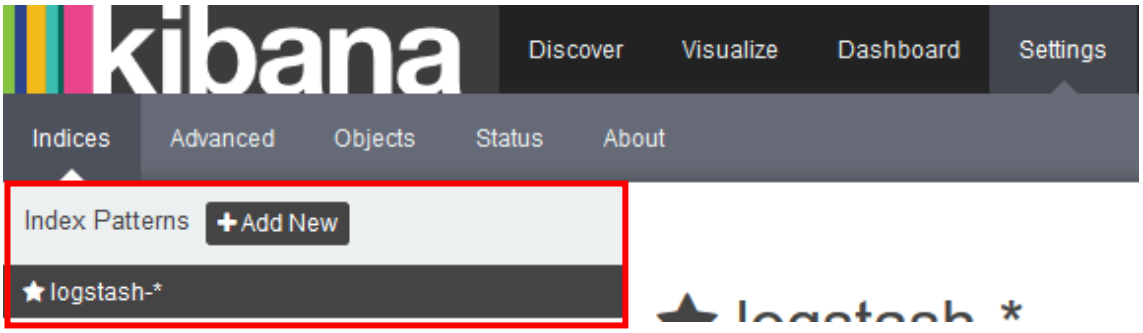



*Figura 16. Menú Kibana*

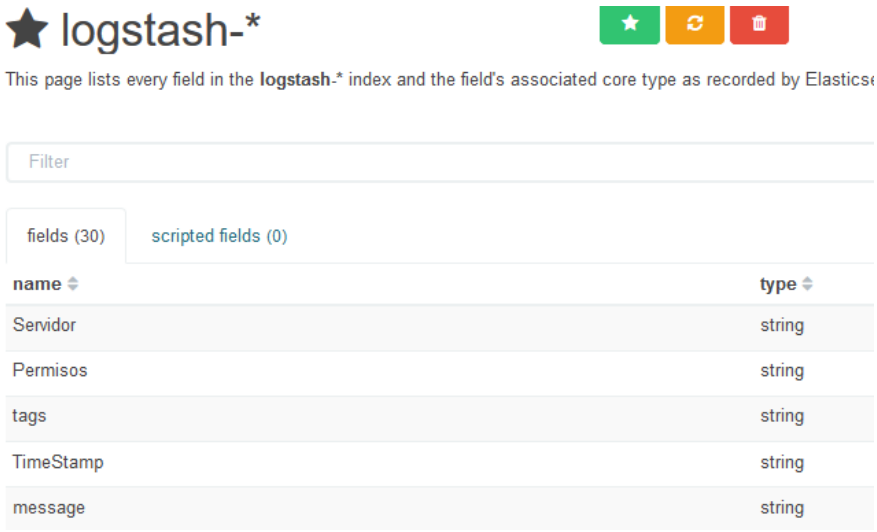



*Figura 17. Propietats d'un índex del Kibana*

Una vegada tinguem seleccionat un index ens apareixeran tres recuadres de diferents colors.

- Si seleccionem el recuadre verd, aquest índex serà el principal i sempre que s'obri apareixerà aquest.
- Si seleccionem el recuadre taronja recargarem les variables ja que és pot donar el cas que n'apareixin més si s'afegeixen dades noves.
- Si seleccionem el recuadre vermell, eliminarem el índex.

Una vegada s'ha definit com a mínim un índex, ja podem anar a veure totes les dades que tenim emmagatzemades fent clic a la pestanya *Discover*, la pantalla que és mostrarà serà d'aquest estil.


*Figura 18. Logs/Temps*

On podem observar en la part superior una gràfica de temps que ens mostra en quin moment les dades han estat guardades a ElasticSearch i a la part inferior podem veure tots els Logs que apareixen en aquesta franja de temps.

Una vegada observem que tenim les dades desitjades, per poder configurar una gràfica hem d'anar a l'apartat *Visualize* i veurem el següent.

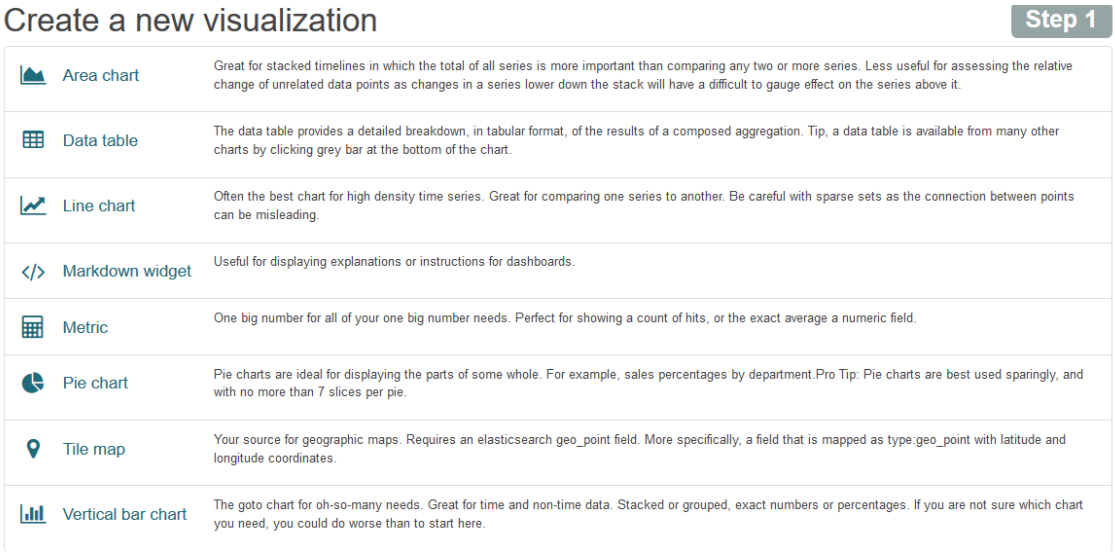



*Figura 19. Creació d'una gràfica*

En aquesta part hem de seleccionar quin tipus de gràfica volem crear i una vegada seleccionada ens apareixerà quines variables són les que voldrem mostrar. Podem escollir entre 8 opcions possibles:

- Area Chart: Ens crea una gràfica d'area, basada en les dades que seleccionem.
- Data Table: Ens crea una taula basada en les dades que seleccionem.
- Line Chart: Ens crea una gràfica de línies, basada en les dades que seleccionem.
- Markdown Widget: Ens permet introduir qualsevol text.
- Metric: Ens permet incloure números de les dades que seleccionem.
- Pie Chart: Ens crea una gràfica de sectors basada en les dades que seleccionem.
- Tile Map: Ens crea un gràfic situant les dades que seleccionem sobre el mapamundi.
- Vertical Bar Chart: Ens crea una gràfica de barres, basada en les dades que seleccionem.

Per posar un exemple seleccionarem la gràfica *Pie Chart*, una vegada la seleccionem observarem la següent informació:

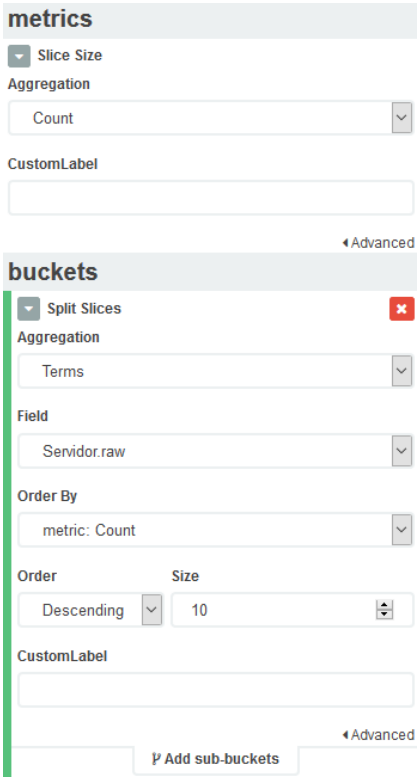



*Figura 20. Propietats d'una gràfica*

En aquest cas la mètrica principal serà la agregació de cada servidor, d'aquesta forma a la gràfica de sectors tindrem un sector per cada servidor i una major area al que tingui més logs. Finalment la gràfica quedaria així:

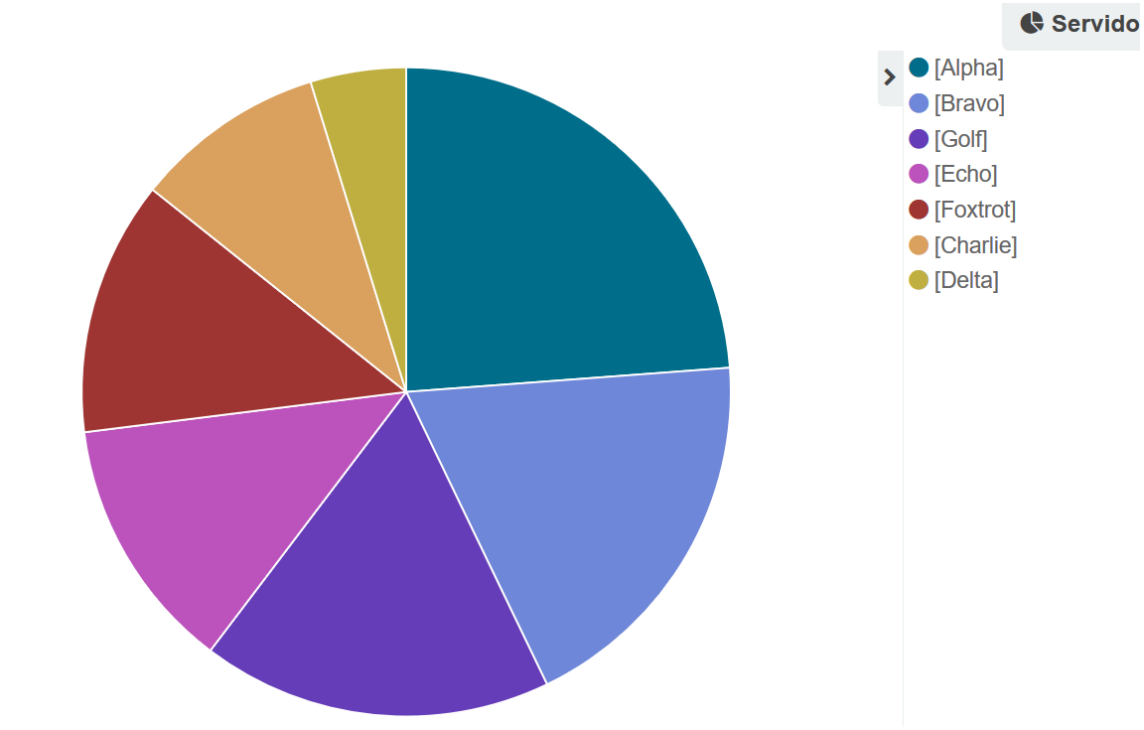



*Figura 21. Gràfica de sectors*

Com podem veure, qualsevol usuari podria crear una visualització sense tenir ni idea de Logstash i ElasticSearch, ja que tot és molt intuitiu i no obliga al usuari a tenir altres coneixaments.

Una vegada tenim diverses gràfiques les podem incluir en el *Dashboard* de Kibana.

Per poder-ho fer, només hem d'anar a la pestanya *Dashboard* i afegir les gràfiques que tenim creades.

Una vegada s'han afegit totes les gràfiques creades, en el nostre cas, un exemple pràctic, quedaria així.

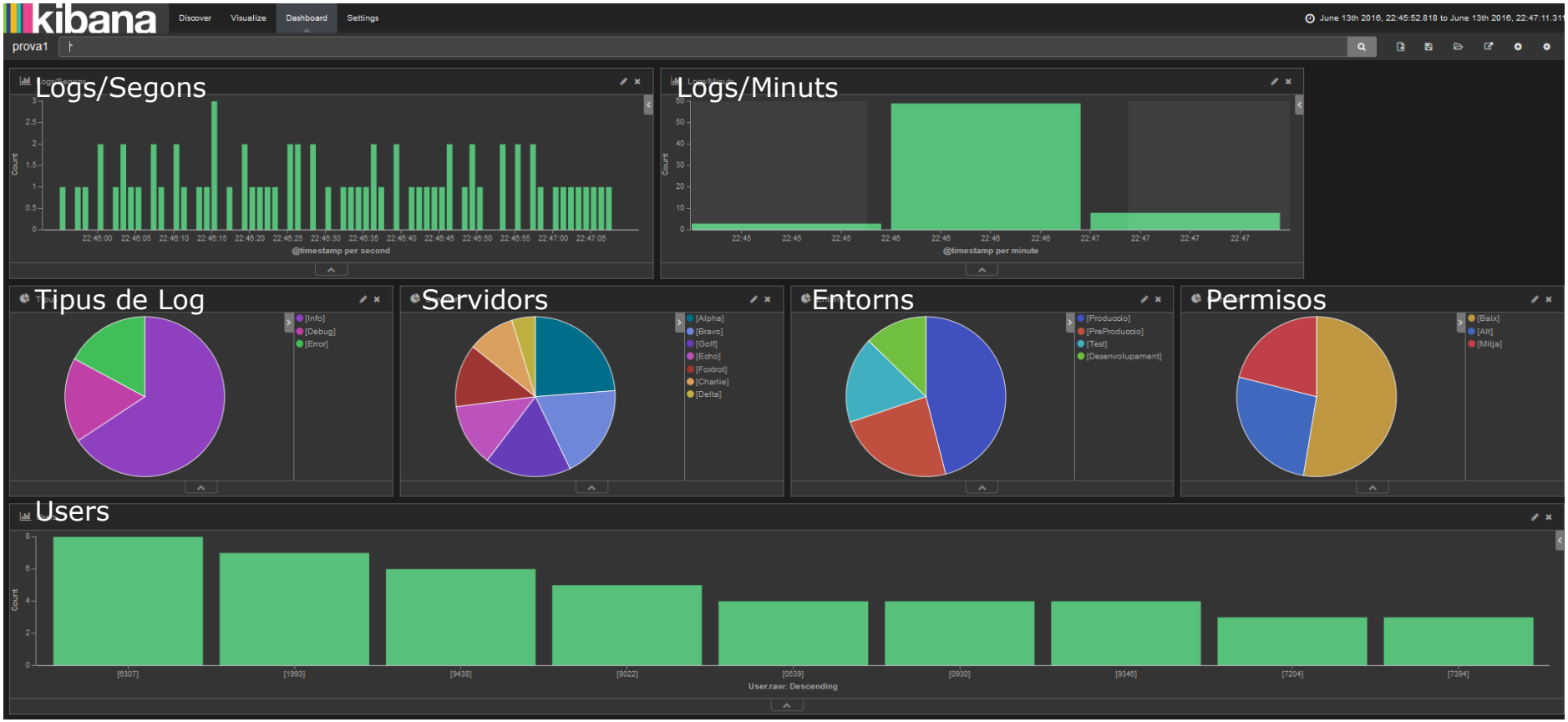



*Figura 22. Dashboard Final*

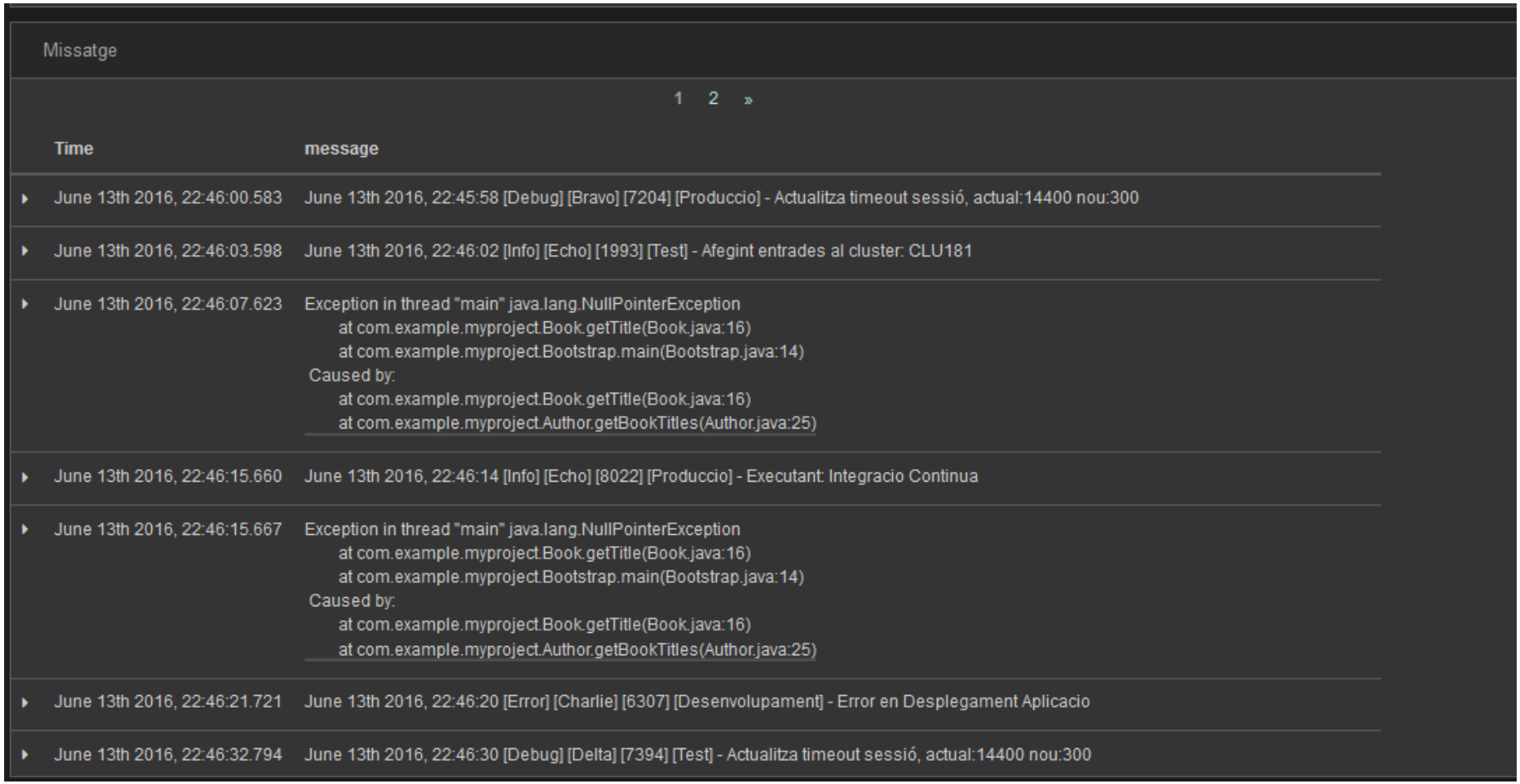



*Figura 23. Dashboard Final 2*

Una vegada estem amb el *Dashboard* creat el podem guardar, per quan hi tornem accedir ja tenir-lo tal com desitgem.

Totes les gràfiques i *Dashboards* és poden importar i exportar.

A més a més al tenir totes les gràfiques en un *Dashboard* si seleccionem un paràmetre, fent clic en una gràfica, per exemple seleccionem l'entorn de Test automàticament s'actualitzen les dades amb només els Logs que contenen aquesta paraula.

En les gràfiques de temps és pot seleccionar una zona en concret i només et mostra els Logs que han succeït en aquella franja de temps.

Addicionalment Kibana ens permet observar estadístiques sobre quant de temps triga en fer les seves cerques. Aquestes dades les podem observar a la pestanya Status.

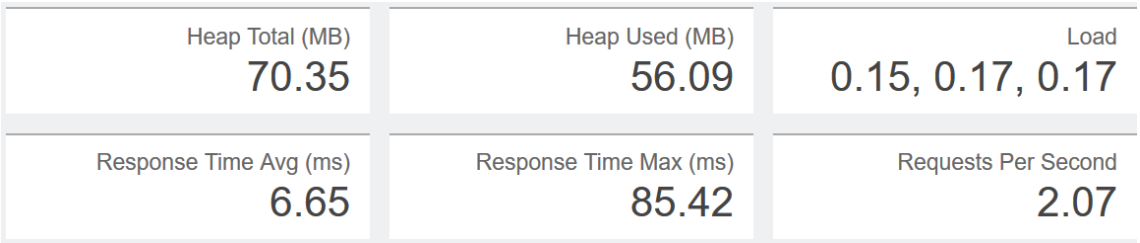



*Figura 24. Estadistiques Kibana*

Podem observar com és mostra la capacitat total, la carrega assignada, la mitja de temps en completar les cerques, el temps màxim i el nombre de sol·licituds per segon.

El temps mig de resposta acostuma a ser molt baix inclús per moltes dades a continuació mostrem una gràfica:

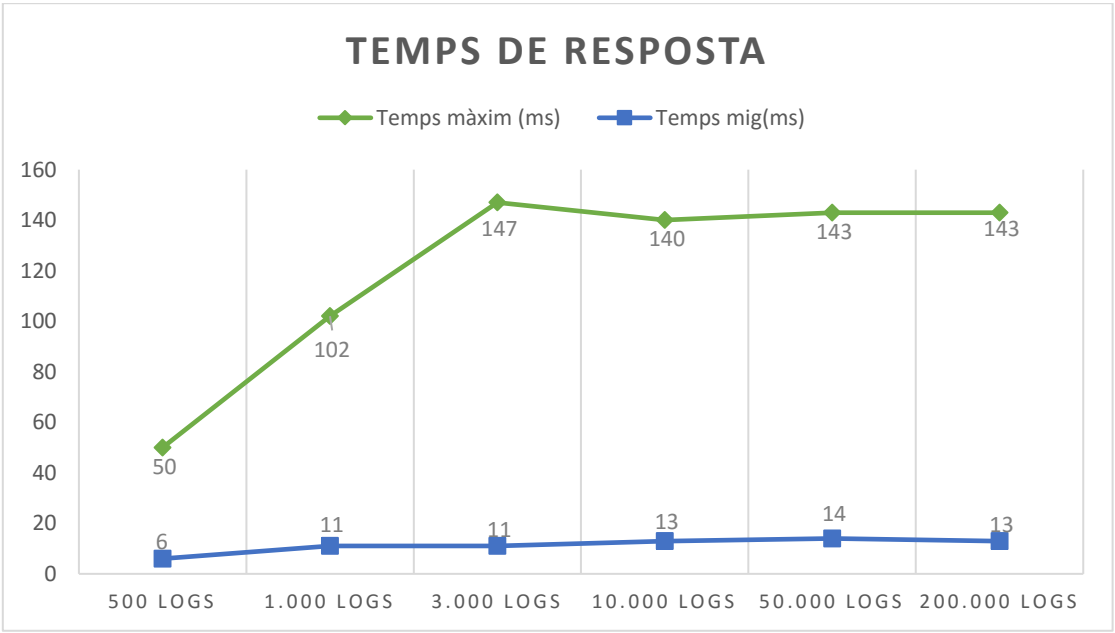



*Figura 25. Gràfica temps per cerca*

# 12. Proves

Per comprovar que tot el conjunt d'eines d'ELK funciona correctament, s'ha accedit a Kibana i s'han realitzat diferents proves de mostreig de les dades totals i també utilitzant filtres per comprovar que les dades que és mostren compleixen estrictament els filtres establerts.

Aquestes proves és duran a terme el dia de la defensa del projecte, ja que per contemplar tots els casos i mostrar-los en el treball escrit seria necessari l'ús de moltes imatges.

Les proves és podrien dividir en dos apartats, les que tracten el desplegament d'ELK, i així observar que sempre s'executa correctament, i les que ens permetran obtenir la traça de Log desitjada, fent ús dels filtres.

Concretament és realitzaran proves de desplegament tant d'un o més clients Logstash en diferents màquines virtuals i de ElasticSearch i Kibana en una altra màquina virtual.

Seguidament és llençaran els serveis Node.js per crear Logs i que els clients Logstash rebin i enviïn les dades cap a la màquina virtual on és troba ElasticSearch. Aleshores accedirem al Kibana i realitzarem diferents cerques dinàmiques dels Logs que haguem rebut, fins trobar la traça que compleixi amb els nostres paràmetres.

# 13. Conclusions

Una vegada realitzat aquest projecte podem observar com aconseguim que les aplicacions que desitgem és pugin instal·lar d'una forma ràpida, senzilla i escalable, gràcies a les eines d'Ansible i Docker. Això ens permetrà instal·lar aquestes aplicacions en uns servidors que no en som els propietaris sense fer gaire soroll.

A més a més, els usuaris que necessitin consultar l'aplicació del Kibana no és condició necessària que sàpiguen com funciona Logstash i ElasticSearch ja que accedint en una direcció web ja poden veure tots els Logs dels servidors i a més a més aplicar filtres. També si ho desitgem poden crear gràfiques que siguin importants pel seu dia a dia i sense molestar als altres usuaris ja que com s'ha comentat és poden tenir diferents *Dashboards* personalitzats en el mateix Kibana.

El principal gran avantatge d'utilitzar aquestes eines és que podrem observar tots els Logs en temps real i de tots els servidors, sense haver de perdre més temps del necessari accedint via SSH als servidors del client, aquest concretament és el principal objectiu del projecte, facilitar la vida als treballadors per poder obtenir les dades de la forma més senzilla i ràpida possible.

També s'ha pogut observar que inclús tenint un nombre molt gran de Logs en la base de dades d'ElasticSearch, el temps mig de cerca és menyspreable.

Per tant, es dóna per finalitzat aquest projecte havent assumit els seus principals objectius, marcats al inici del mateix.

# 14. Referències


[1] Pàgina Oficial de Splunk [en línia]
http://www.splunk.com/en_us/homepage.html [Consulta: 24 de Febrer de 2016]

[2] Pàgina Oficial d'ELK [en línia]
https://www.elastic.co/ [Consulta: 24 de Febrer de 2016]

[3] Pàgina Oficial Nagios [en línia]
https://www.nagios.org/ [Consulta: 24 de Febrer de 2016]

[4] Noticies Nagios [en línia]
https://labs.nagios.com/2014/10/19/nagios-log-server-vs-elasticsearch-logstash-kibana/ [Consulta: 29 de Febrer de 2016]

[5] Pàgina Oficial Vagrant [en línia]
https://www.vagrantup.com/ [Consulta: 15 de Març de 2016]

[6] Pàgina Oficial d'Ansible [en línia]
https://www.ansible.com/ [Consulta: 15 de Març de 2016]

[7] Pàgina Oficial Docker [en línia]
https://www.docker.com/ [Consulta: 15 de Març de 2016]

[8] Pàgina Oficial Node.js [en línia]
https://nodejs.org// [Consulta: 1 d'Abril de 2016]

[9] Imatges Vagrant [en línia]
https://atlas.hashicorp.com/boxes/search/ [Consulta: 15 de Març de 2016]

[10] Documentació Vagrant [en línia]
https://www.vagrantup.com/docs/ [Consulta: 15 de Març de 2016]

[11] Documentació Ansible [en línia]
http://docs.ansible.com/ [Consulta: 15 de Març de 2016]

[12] Documentació Docker [en línia]
https://docs.docker.com/ [Consulta: 15 de Març de 2016]

[13] Documentació Plugin Input Logstash [en línia]
https://www.elastic.co/guide/en/logstash/2.3/input-plugins.html
[Consulta: 6 d'Abril de 2016]

[14] Oniguruma [en línia]
https://manual.macromates.com/en/regular_expressions/
[Consulta: 8 d'Abril de 2016]

[15] Documentació Plugin Filter Logstash [en línia]
https://www.elastic.co/guide/en/logstash/2.3/filter-plugins.html
[Consulta: 8 d'Abril de 2016]

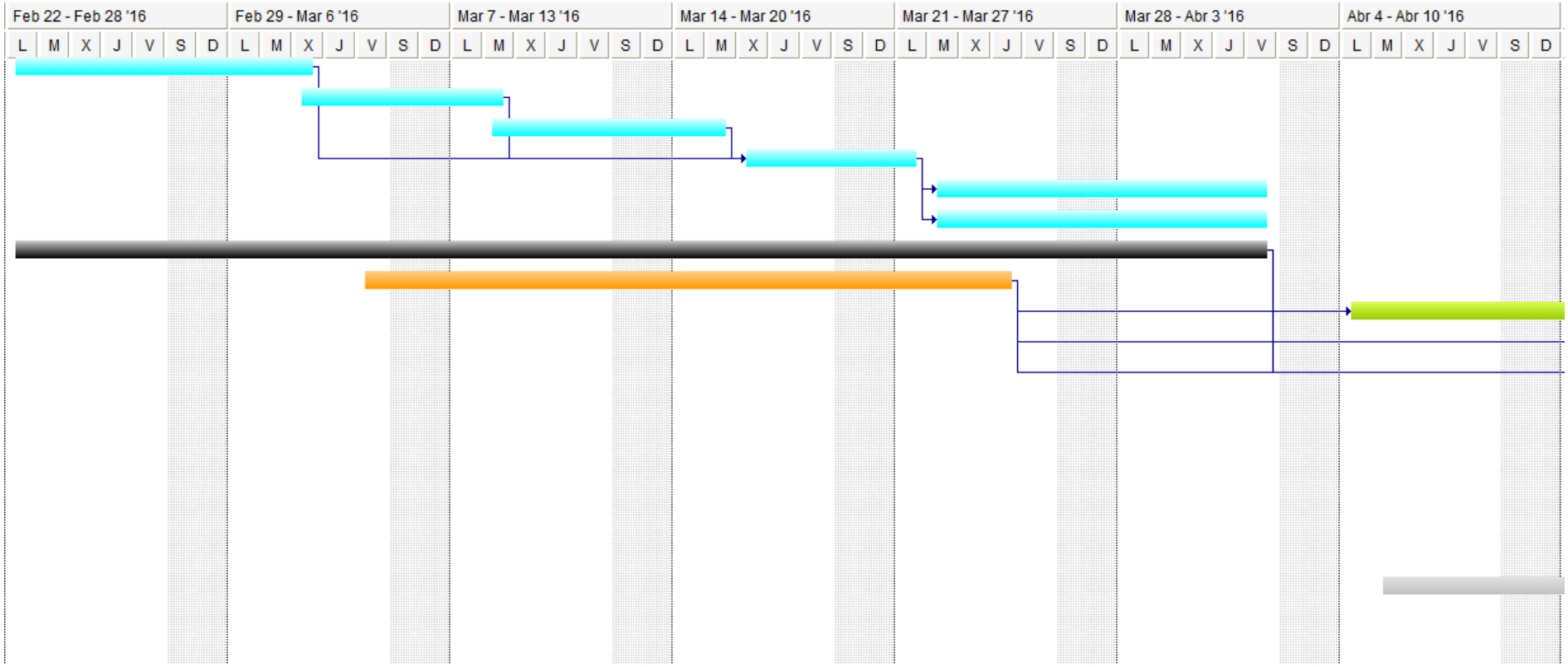
[16] Documentació Plugin Output Logstash [en línia]
https://www.elastic.co/guide/en/logstash/2.3/output-plugins.html
[Consulta: 10 d'Abril de 2016]

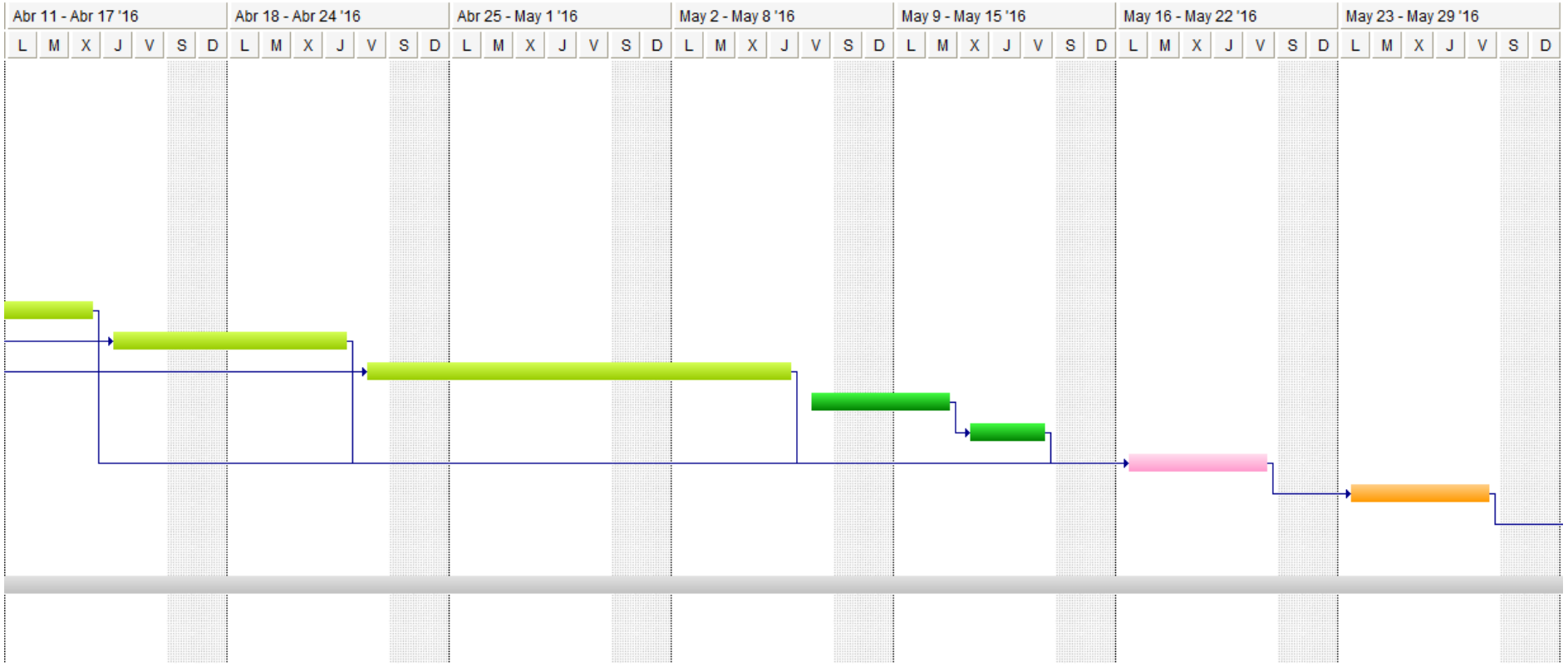
[17] Documentació Cerques ElasticSearch [en línia]
https://www.elastic.co/guide/en/elasticsearch/reference/current/_executing_searches.html
[Consulta: 15 d'Abril de 2016]

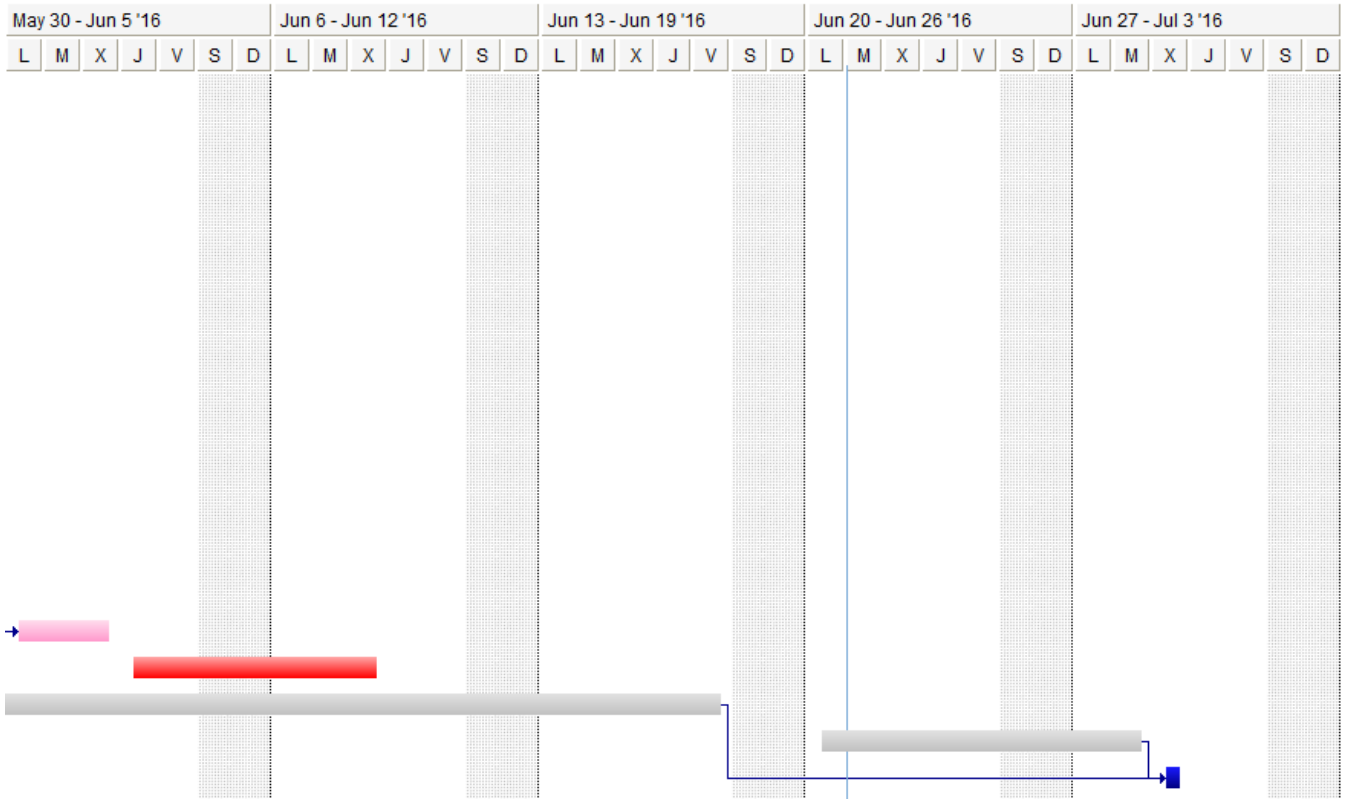
[18] Documentació Kibana [en línia]
https://www.elastic.co/guide/en/kibana/current/visualize.html
[Consulta: 26 d'Abril de 2016]

# 15. Annex

| | | Nombre | Duración | Inicio | Fin | Predecesoras |
|---|---|---|---|---|---|---|
| 1 | | Primer Entregable GEP | 8d | 22/02/2016 | 02/03/2016 | |
| 2 | | Segon Entregable GEP | 5d | 02/03/2016 | 08/03/2016 | |
| 3 | | Tercer Entregable GEP | 6d | 08/03/2016 | 15/03/2016 | |
| 4 | | Quart Entregable GEP | 4d | 16/03/2016 | 21/03/2016 | 1,2,3 |
| 5 | | Cinquè Entregable GEP | 9d | 22/03/2016 | 01/04/2016 | 4 |
| 6 | | Sisè Lliurable GEP | 9d | 22/03/2016 | 01/04/2016 | 4 |
| 7 | | Cerca d'informació | 30d | 22/02/2016 | 01/04/2016 | |
| 8 | | Instal·lació i Familiarització amb l'entorn | 15d | 04/03/2016 | 24/03/2016 | |
| 9 | | Configuració en local Logstash | 8d | 04/04/2016 | 13/04/2016 | 7,8 |
| 10 | | Configuració en local ElasticSearch | 6d | 14/04/2016 | 21/04/2016 | 7,8 |
| 11 | | Configuració en local Kibana | 10d | 22/04/2016 | 05/05/2016 | 7,8 |
| 12 | | Analisi i Disseny servei Node.js | 3d | 06/05/2016 | 10/05/2016 | |
| 13 | | Creació servei que llenci Logs amb Node.js | 3d | 11/05/2016 | 13/05/2016 | 12 |
| 14 | | Comprovacions en local | 5d | 16/05/2016 | 20/05/2016 | 9,10,11,13 |
| 15 | | Instal·lació i configuració d'ELK en les màquines apropiades | 5d | 23/05/2016 | 27/05/2016 | 14 |
| 16 | | Comprovacions que tot està configurat correctament i funciona a le | 3d | 30/05/2016 | 01/06/2016 | 15 |
| 17 | | Temps extra per possibles contratemps | 5d | 02/06/2016 | 08/06/2016 | |
| 18 | | Documentar el projecte | 54d | 05/04/2016 | 17/06/2016 | |
| 19 | | Creació de la presentació per la defensa | 7d | 20/06/2016 | 28/06/2016 | |
| 20 | | Presentació | 1d | 29/06/2016 | 29/06/2016 | 18,19 |


Feb 22 - Feb 28 '16
Feb 29 - Mar 6 '16
Mar 7 - Mar 13 '16
Mar 14 - Mar 20 '16
Mar 21 - Mar 27 '16
Mar 28 - Abr 3 '16
Abr 4 - Abr 10 '16
L M X J V S D


Abr 11 - Abr 17 '16
Abr 18 - Abr 24 '16
Abr 25 - May 1 '16
May 2 - May 8 '16
May 9 - May 15 '16
May 16 - May 22 '16
May 23 - May 29 '16
L M X J V S D


May 30 - Jun 5 '16
Jun 6 - Jun 12 '16
Jun 13 - Jun 19 '16
Jun 20 - Jun 26 '16
Jun 27 - Jul 3 '16
L M X J V S D