

Hector Mercado Valls

**EMMAGATZEMATGE I ANÀLISI AUTOMÀTIC DE DADES MÈDIQUES
RELLEVANTS DE DOCUMENTS O FITXERS DE LOG**

TREBALL DE FI DE GRAU

**dirigit per la Dra. Aïda Valls Mateu,
el Dr. Antonio Moreno Ribas
i el Sr. Marc González Boronat**

Grau d'Enginyeria Informàtica



UNIVERSITAT ROVIRA I VIRGILI

Tarragona

2018

Índex

1.	Introducció	5
1.1	Motivació	6
1.2	Objectius	6
1.3	Estructura del Document.....	7
2.	Especificacions del Projecte.....	8
2.1	Requisits Genèrics	8
2.2	Requisits Específics	8
3.	Disseny	10
3.1	Quines Dades Volem Analitzar	10
3.1.1	Big Data.....	10
3.2	Elecció del Tipus de Base de Dades	11
3.3	Transició del Model Relacional a NoSQL.....	12
3.4	Motors de Cerca.....	13
3.5	Redundància i Rendiment.....	15
3.6	Anàlisi Automàtic de Dades	15
3.7	Normativa GDPR Europea	16
3.8	Arquitectura del Sistema.....	17
4.	Introducció a Elastic Stack.....	18
4.1	Introducció a Elasticsearch	18
4.1.1	Documents.....	18
4.1.2	Índexs	19
4.1.3	Tipus.....	19
4.1.4	Shards.....	19
4.1.5	Node	20
4.1.6	API REST.....	21
4.1.7	Exploració i Descobriments de Nodes.....	21
4.1.8	Funcionament Distribuit.....	21
4.1.9	Split Brain	22
4.1.10	Templates	23
4.1.11	Inverted Index.....	24
4.1.12	Analitzadors.....	25
4.1.13	Rellevància en les Cerques.....	26
4.1.14	Similitud entre Termes	28
4.1.15	Precision vs Recall	28
4.2	Introducció a Logstash.....	30

4.2.1	Pipelines	30
4.2.2	Resiliència de Dades	30
4.3	Introducció a Kibana.....	31
4.3.1	Extracció d'Informació de Documents	31
4.3.2	Extracció d'Informació d'Elastic Stack	36
5.	Desenvolupament.....	37
5.1	Entorn de desenvolupament.....	37
5.1.1	VirtualBox.....	37
5.1.2	Configuració d'Elasticsearch	41
5.1.3	Configuració de Logstash.....	42
5.1.4	Configuració de Kibana	43
5.1.5	Seguretat.....	44
5.1.6	Redirecció de Ports i DDNS.....	46
5.1.7	Reverse Proxy SSL.....	47
5.2	Tractament de Fitxers de Log	51
5.2.1	Definició dels Templates.....	54
5.2.2	Definició dels Pipelines.....	55
5.2.3	Execució del Pipeline	57
5.2.4	Construcció de Visualitzacions	59
5.3	Tractament i Anàlisi d'Incidències	66
5.3.1	Tractament de les Descripcions Textuals.....	68
5.3.2	Filtratge per Idioma Predominant	70
5.3.3	Definició del Template per Incidències	73
5.3.4	Indexació d'Incidències.....	75
5.3.5	Classificador.....	77
5.3.6	Score.....	84
5.3.7	Desnormalització d'Accions	85
6.	Resultats i Avaluació.....	86
6.1	Entorn de Desenvolupament.....	86
6.1.1	Monitoratge Elasticsearch.....	86
6.1.2	Monitoratge Logstash.....	88
6.1.3	Monitoratge i Seguretat Kibana	89
6.1.4	Monitoratge Fallides/Split Brain.....	90
6.2	Tractament de Fitxers de Log	92
6.3	Tractament i Anàlisi d'Incidències	96
6.3.1	Qualitat de l'Algorisme amb Dades Controlades.....	96

6.3.2	Qualitat de l'Algorisme amb Dades no Controlades	104
7.	Conclusions i Treball Futur.....	109
7.1	Entorn de Desenvolupament.....	109
7.2	Tractament de Fitxers de Log	109
7.3	Tractament i Anàlisi d'Incidències	109
7.4	Valoració Personal del TFG.....	110
8.	Recursos Utilitzats.....	111
	Acrònims	112
	Referències.....	114

1. Introducció

Actualment els sistemes informàtics emmagatzemen gran quantitat de dades de molt diferent naturalesa, de senyals obtingudes de sensors, missatges en *tweets*, correus electrònics, logs de servidors web, imatges, vídeos, etc. Així doncs, creix la necessitat d'implementar processos per transformar aquestes dades en informació útil per a la presa de decisions, o si més no, per enriquir i complementar la informació que s'obté a través d'altres dades més estructurades on els processos de transformació i anàlisi ja fa molts anys que s'estan utilitzant.

La paraula *Big Data* és ambigua i sovint maltractada. És difícil definir en quin moment es creua la línia vermella entre *Data* i *Big Data*. Sovint s'intenta explicar dient "quan utilitzes tècniques d'aprenentatge automàtic", o "quan els conjunts de dades superen la capacitat d'emmagatzemament d'una base de dades típica". Una definició prou acceptada és que *Big Data* són el conjunt de dades necessàries per transformar-les en informació de valor afegit per poder prendre decisions amb un impacte positiu dins d'un objectiu o projecte establert [1][2].

El tractament de quantitats massives de dades requereix un disseny especial dels sistemes informàtics, que en permetin la manipulació a una velocitat raonable. A més a més, però, cal tenir en compte que moltes d'aquestes dades no són estructurades i per tant cal determinar com s'han d'analitzar i interpretar, perquè quin tipus d'anàlisi o interpretació es pot fer d'una gravació de veu, o d'un text escrit? Generalment s'accepta que les dades no estructurades representen el 80% de la informació disponible en el marc actual, per tant, es dedueix que solament estem traient valor d'aproximadament un 20% del total d'informació existent, i això vol dir que hi ha latent un gran potencial de dades esperant que s'extregui valor d'elles.

L'anàlisi del text, conegut també pels termes *Text Mining* o *Natural Language Processing* (NLP) és la ciència que aporta un conjunt d'eines per poder donar valor al text [3]. Aquestes tècniques permeten convertir aquestes dades no estructurades en informació estructurada, extreure informació rellevant dels textos (per exemple, identificar el sentiment dels comunicadors o el motiu per què està passant un fet), o fins i tot classificar de forma automàtica el text (o fragments), per exemple per conèixer els titulars d'una conversa o una notícia, entre d'altres.

Per poder realitzar aquest tipus d'anàlisi, es necessita una tecnologia que porti la capacitat d'absorció, emmagatzematge i gestió de grans volums de dades en format text, així com de proveir eines d'anàlisi integrades. La plataforma de codi obert Elastic Stack (conegut també com ELK) és una de les possibles solucions i és la que s'ha escollit en aquest projecte. Elastic Stack conté diferents eines com Elasticsearch, Logstash i Kibana.

Elasticsearch és un motor de cerca que incorpora una base de dades NoSQL distribuïda orientada a documents, que permet indexar¹ un gran volum de dades per poder realitzar consultes sobre elles a posteriori. Logstash és una eina per recollir, transformar i carregar qualsevol tipus d'esdeveniment i Kibana és des d'on es podrà visualitzar, realitzar l'anàlisi i extreure valor de les dades.

¹ Utilitzem la paraula "indexar" per indicar que un document és inserit a la base de dades d'Elasticsearch. També utilitzem la mateixa paraula per indicar el procés de modificació de l'índex invertit a partir d'un text.

En aquest treball fi de grau estudiarem i aplicarem aquestes eines de management de dades textuais per resoldre un problema real que es planteja a continuació.

1.1 Motivació

El servei GINSA AIE (Gestió Integral Sanitària i Assistencial AIE), amb seu a Reus, dóna suport a un conglomerat d'unes 10 empreses de serveis sanitaris, essent la més gran l'Hospital Universitari Sant Joan de Reus. Dins l'entorn sanitari, gestionat per aquesta empresa, es disposen de registres i informes de pacients i incidències redactats en text pla que, si es poguessin analitzar, segur que podrien afegir valor a les dades mèdiques quantitatives que actualment existeixen en els registres electrònics dels pacients. El director del servei d'organització i sistemes ens va contactar per desenvolupar una solució informàtica que es pugui incorporar a la seva unitat.

Concretament, durant el 2017 es van rebre al voltant de 18.000 incidències/demandes a l'àrea de TIC de GINSA AIE. El 80% d'aquestes incidències són de l'Hospital Sant Joan de Reus.

La recollida i emmagatzemament d'aquestes incidències permetrà analitzar, entre d'altres:

- Mètriques de càrregues (RAM, CPU) i tràfic de xarxa de servidors a fi de trobar comportaments anòmals.
- Temps de resposta de les aplicacions i llurs components.
- Utilització de les aplicacions (en un altre tipus de negoci, es podria implementar un sistema de recomanacions).
- Tipus d'incidències més freqüents.
- Durada de la resolució de les incidències per tipus.

Amb una plataforma que emmagatzemi aquests documents, es podria recollir, centralitzar i automatitzar tot l'anàlisi, monitoratge i implementació d'alertes.

Personalment, aquest repte representa una bona oportunitat per formar-me en tecnologies *Big Data* per resoldre problemes molt comuns en situacions que es volen explotar dades no estructurades per extreure informació.

1.2 Objectius

L'objectiu del projecte consisteix a emmagatzemar i analitzar dades generades pel sistema d'informació hospitalari, sobre les quals aplicarem coneixement del domini que ens permetrà extreure informació útil per al gestor de l'hospital. En definitiva, es tracta de construir un sistema de suport a la decisió senzill (DSS) que permetrà al gestor analitzar de forma més ràpida i senzilla la gran quantitat de dades que diàriament arriben als centres de gestió informàtica.

Aquest objectiu es pot descompondre en els següents subobjectius:

1. Aquest projecte té com a primer objectiu implementar una base de dades NoSQL que permeti administrar i consultar documents informàtics de l'hospital. La base de dades ha de permetre la indexació de grans volums de dades de forma contínua.

2. Com a segon objectiu es vol desenvolupar un sistema que analitzi dades estructurades i sigui capaç de donar sentit a una gran quantitat d'esdeveniments provinents de fitxers de log que recullen l'activitat del personal de l'hospital.
3. El tercer objectiu consisteix a incorporar tècniques de NLP per categoritzar de forma automàtica els missatges sobre incidències que envien els usuaris al centre informàtic de l'hospital.

Els avantatges principals d'aquest sistema de suport a la decisió són, d'una banda, poder visualitzar el comportament del sistema informàtic de l'hospital de forma àgil, identificant els punts febles i així poder introduir millores allà on sigui adient, i d'altra banda, estalviar temps al personal del servei d'informàtic eliminant l'etiquetatge manual que es fa actualment.

1.3 Estructura del Document

El present document està estructurat en 7 capítols. Després de la introducció, al capítol 2 es formalitzen les especificacions del sistema a desenvolupar. Aquestes especificacions s'han obtingut després de diverses entrevistes amb el director d'organització i sistemes de GINSA AIE.

Al capítol 3 es parla dels diferents tipus de bases de dades, centrant-nos en el cas de dades massives poc estructurades, com els documents de text. Es presenten alguns entorns ben coneguts actualment i es justifica l'ús d'Elastic Stack.

Al capítol 4 s'expliquen tots els conceptes necessaris per familiaritzar-se amb Elastic Stack com a mínim des del punt de vista teòric per així entendre la part pràctica dels capítols posteriors.

El capítol 5 proposa una implementació del sistema DSS per satisfer els objectius i requisits especificats.

Al capítol 6 es pot veure els resultats aconseguits quan el sistema es posa en marxa i realitza totes les inicialitzacions pertinents.

El capítol 7 recull les conclusions extretes dels capítols 5 i 6.

2. Especificacions del Projecte

A partir d'entrevistes amb el director del servei informàtic de l'Hospital Universitari Sant Joan de Reus es van formular els següents requeriments del sistema que es vol desenvolupar.

2.1 Requisits Genèrics

- R1: Construir un entorn de desenvolupament per simular un entorn de producció.
- R2: Indexar dades estructurades com ara logs de QlikView i dades semiestructurades com incidències.
- R3: Automatitzar el procés d'indexació a la base de dades.
- R4: Establir redundància de dades evitant el fenomen *split brain*.
- R5: Proporcionar un sistema segur d'emmagatzemament, accés i monitoratge de les dades.
- R6: Facilitar l'escalabilitat del sistema.

2.2 Requisits Específics

Dins els requisits genèrics anteriors podem determinar-ne uns requisits més específics, que s'indiquen a continuació:

R1:

- L'entorn de desenvolupament s'ha de poder instal·lar als ordinadors de l'hospital, però durant el projecte es limitarà només a un sol equip personal.

R2: Els tipus concrets de fitxers que s'han de poder tractar són:

- **Estructurats:** Fitxers de text (logs) recollits pel software QlikView codificats en UTF-8-BOM. QlikView és una plataforma de *business discovery* que ofereix un *Business Intelligence* d'autoservei per tot tipus d'usuaris d'una empresa o organització [4]. QlikView està dissenyat per analitzar dades estructurades i fer de suport per a la presa de decisions. Amb QlikView es pot recollir i centralitzar tots els logs per poder analitzar-los, monitorar-los i implementar alertes. La recollida d'aquests fitxers de log són rellevants per obtenir informació sobre:
 - o Càrregues (RAM, CPU) de servidors i monitorar tràfic de xarxa a fi de trobar comportaments anòmals.
 - o Temps de resposta de les aplicacions i llurs components.
 - o Utilització de les aplicacions.

Un exemple d'un esdeveniment en un dels logs:

```
2017-07-12 03:25:48 2017-08-01 17:04:55 4 100 Notice License:
License leased to user GRUPSGS\19989
```

Diem que aquestes dades són estructurades perquè contenen una capçalera que defineix uns camps fixes que tenen unes dades amb un format i un rang de valors conegut i acotat de possibilitats de forma sistemàtica [5].

- **Semiestructurats:** Fitxers d'incidències i accions del sistema d'informació hospitalari en format JSON codificats en UTF-8. Exemple d'un esdeveniment:

```
...{
  "ID_INCIDENT":86687,
  "ENTRY_CHANNEL": "#",
  "ASSIGNED_TO":10,
  "CREATION_DATE":"2016-05-30",
  "CLOSE_DATE":"2016-06-02",
  "CENTRE_INT":"HOSPITAL SANT JOAN DE REUS",
  "CENTRE_PET_DEP":null,
  "TIPUS":"Incidència",
  "CLASSE":"HW_PRN",
  "TITLE":"no funciona la impressora de reanimación",
  "DESCRIPTION":"no sale en la lista."
},...
```

Diem que aquestes dades són semiestructurades perquè no tenen un format fixe, però tenen etiquetes (claus JSON) que permeten separar les dades [5]. Entre les dades que podem separar, hi ha dades estructurades com dates o identificadors, i dades no estructurades de text natural en els camps TITLE i DESCRIPTION.

Utilitzarem les dades estructurades per demostrar les capacitats visuals de Kibana, mentre que les semiestructurades ens serviran per aplicar tècniques de tractament de llenguatge natural per classificar documents d'Elasticsearch.

R3:

- El sistema ha de poder llegir els formats dels fitxers anteriors i indexar-los automàticament dins del Elasticsearch.
- Cal determinar les etiquetes d'interès a priori perquè es puguin indexar segons els conceptes importants per al gestor.

R4:

- Es configurarà un clúster dins de l'ordinador personal del projectista.

R5:

- S'activaran les directives de seguretat d'Elastic Stack necessàries.

R6:

- L'escalabilitat s'aconseguirà mitjançant un disseny adequat de la base de dades.
- Els procediments implementats (indexació i anàlisi) seran també escalables.

3. Disseny

En aquesta secció es justifica l'ús d'Elastic Stack suggerit pel director d'organització i sistemes a GINSA AIE per solucionar els problemes que ell mateix va proposar i que es recullen en les especificacions del projecte.

3.1 Quines Dades Volem Analitzar

Es volen guardar dades en forma de documents curts semiestructurats que tenen una part estructurada (dates, identificadors) i poden tenir una part no estructurada (text lliure en català). A continuació analitzem més en detall les necessitats i determinem el disseny del sistema.

3.1.1 Big Data

En un sistema d'informació com el d'un hospital, tant els registres que genera un metge per ordinador, com els logs dels programes que integren tot el sistema informàtic, es guarden de forma persistent en algun tipus de format i codificació, i en diferents llocs.

Durant el 2017 es van rebre sobre les 18.000 incidències/demandes a l'àrea de TIC. El 80% d'aquestes incidències són de l'Hospital Sant Joan de Reus.

Per això necessitem un sistema que aportï flexibilitat i rapidesa per absorbir totes les dades que puguin ser rellevants per després fer consultes i anàlisis sobre aquesta enorme quantitat de dades. Per resoldre aquest problema necessitem un sistema amb les següents característiques:

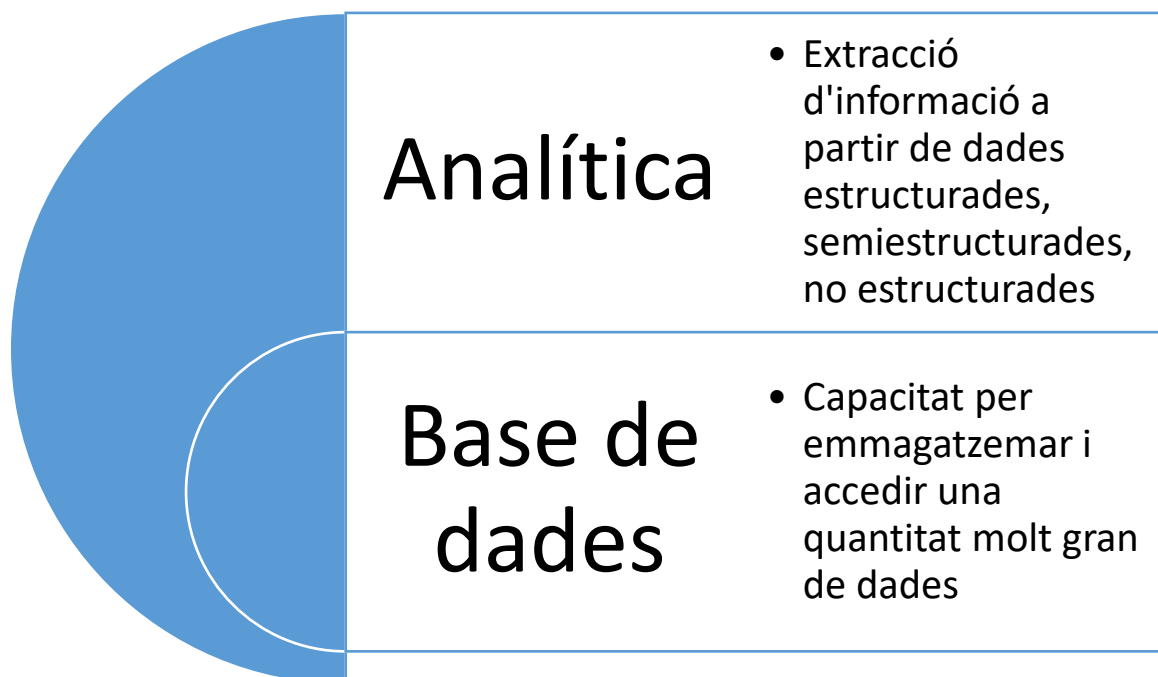


Figura 1. Esquema de les funcionalitats principals del sistema

3.2 Elecció del Tipus de Base de Dades

Per decidir quin tipus de base de dades s'ajusta millors als nostres objectius, tindrem en compte els següents criteris:

- Quantitat de dades:

Per tractar *Big Data* necessitem una base de dades NoSQL. Aquest tipus de bases de dades han sigut dissenyades per manipular grans volums de dades de forma molt ràpida i no segueixen el model entitat-relació típic de les bases de dades tradicionals.

Les bases de dades relacionals, encara que tinguin unes característiques molt potents per la consulta de dades mitjançant el llenguatge SQL, el seu funcionament s'alenteix considerablement quan augmenta el volum de les dades a manipular. Les bases de dades tradicionals basen el seu funcionament en taules, *joins*, i transaccions ACID, mentre que les bases de dades NoSQL no requereixen una estructura de dades en forma de taules i relacions entre elles, i no imposen un esquema prefixat [5].

Les característiques principals de les bases de dades NoSQL són:

- o Emmagatzematge de gran quantitat de dades.
- o Manipulació de gran volum de dades a gran velocitat.
- o Alta escalabilitat horitzontal sense pèrdua de rendiment.
- o Manca d'un esquema predeterminat.

- Tipus de dades:

Les bases de dades NoSQL són més flexibles, ja que permeten informació en formats alternatius a les taules com ara:

- o Clau-valor.
- o Mapeig de columnes.
- o Documents.
- o Grafs.

E-mails, incidències, informes, logs... Són entitats que contenen dades que es poden distingir amb etiquetes. Però cada instància d'aquestes entitats pot tenir o bé etiquetes diferents o bé diferent nombre d'etiquetes. Aquest tipus de dades semiestructurades són ideals per les bases de dades NoSQL orientades a documents.

- Finalitat d'ús d'aquestes dades:

Les bases de dades NoSQL són idònies per aplicacions que requereixin lectura/escriptura de grans volums de dades. Aquestes bases de dades no garanteixen les característiques ACID de la mateixa forma que ho fan les relacionals. Però com estem parlant d'un sistema DSS, podem utilitzar un sistema BASE sense posar en perill cap component del sistema d'informació hospitalari.

Així doncs el millor model per a l'emmagatzemament de les dades mèdiques d'aquest projecte és una base de dades NoSQL orientada a documents. Aquesta base de dades serà el nucli de la figura 1.

3.3 Transició del Model Relacional a NoSQL

Els fitxers de les incidències i accions que tractarem més endavant segueixen un esquema relacional. Per fer compatible aquestes dades amb la base de dades escollida, s'ha d'aplicar un procés de desnormalització.

Les BD relacionals SQL són molt eficients pel que fa a la consistència i l'espai d'emmagatzematge que es requereix. Les taules d'aquestes BD eviten repetir dades i asseguren la seva correctesa, ja que cal un disseny previ de relacions i tipus abans de permetre insercions.

El problema d'aquestes BD es presenta quan es volen mantenir una quantitat extraordinària de dades, i a sobre aquestes dades tenen una estructura variable. Quan hi ha moltes dades els recorreguts per *joins* comporten una latència que en alguns sistemes es consideren inacceptables. Pel que fa a l'estructura i tipus de dades, s'obligaria a la BD a desapropiar alguns camps o redissenyar-la, cosa que comporta un cost molt elevat de temps i esforç.

El model NoSQL prioritza l'accés a les dades sobre la seva modificació o eliminació. En el nostre sistema NoSQL té sentit perquè accedirem a les dades moltes més vegades que les modifiquem. Però per aconseguir aquest efecte cal desnormalitzar les dades en algunes situacions. A continuació mostrem un exemple de desnormalització:

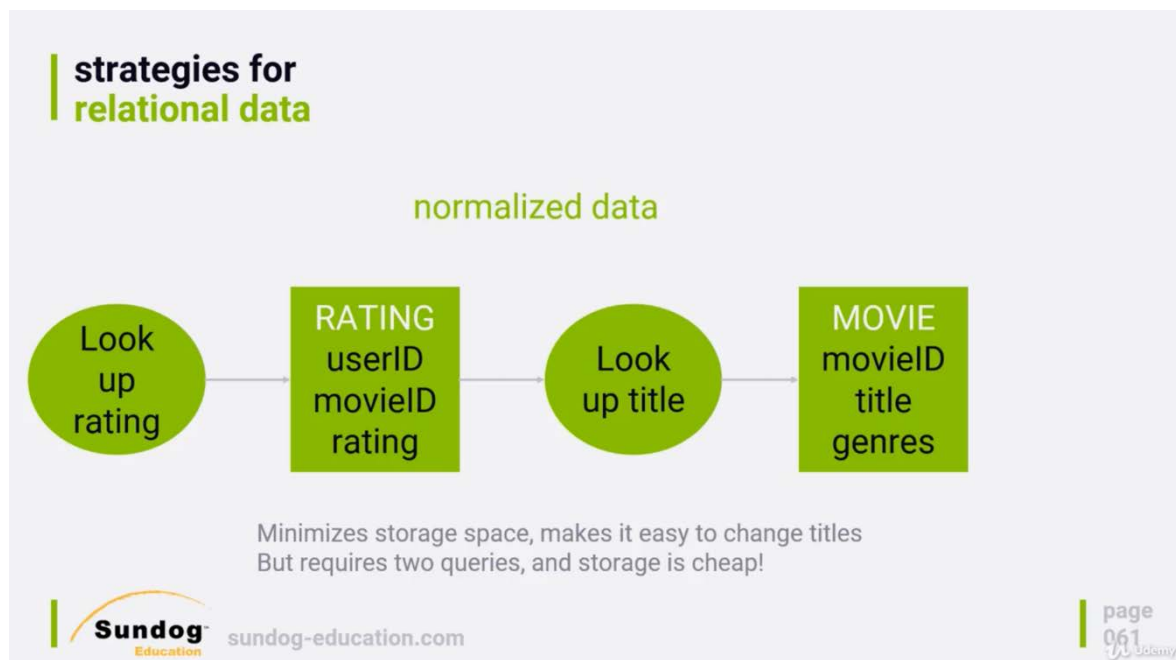


Figura 2. Dades normalitzades [6]

En el cas que volem veure quines pel·lícules tenen una valoració més gran que un valor X, amb dades normalitzades es necessiten dos passos per aconseguir la resposta. Mentre que en el cas següent, ho aconseguirem només en un pas:

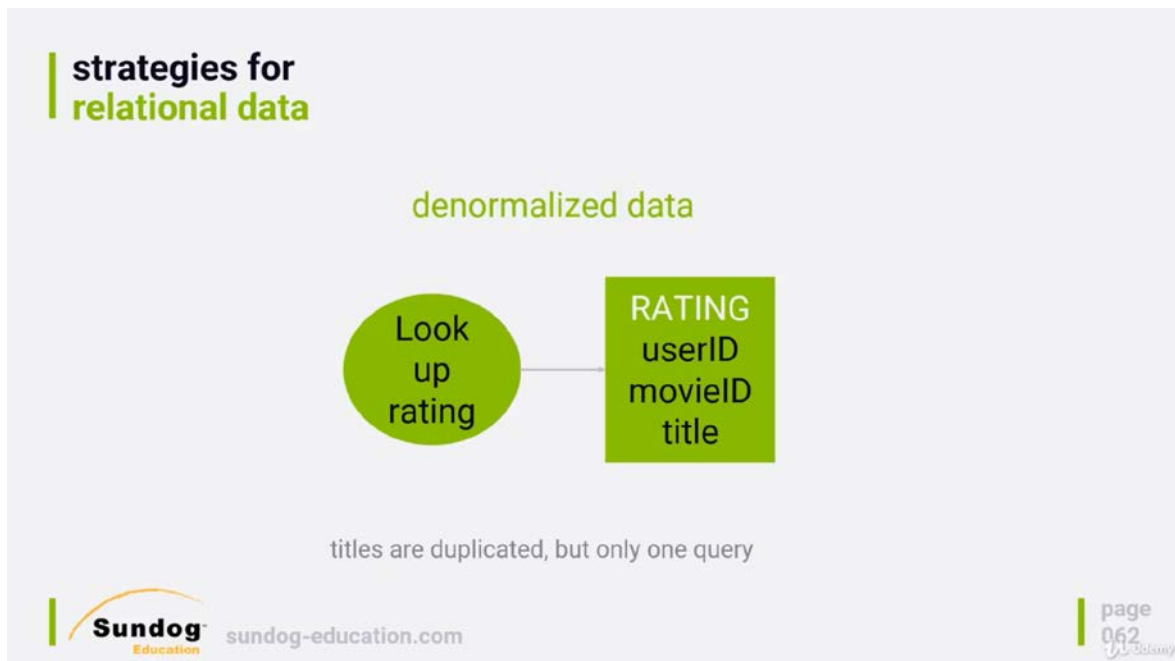


Figura 3. Dades desnormalitzades

És clar que en aquest cas sacrificuem memòria per processament duplicant dades, però tenint en compte que tenir un espai gran d'emmagatzematge pot ser bastant assequible, aquest canvi val la pena. A més si l'accés es pot fer en paral·lel, el rendiment encara és millor.

3.4 Motors de Cerca

Fins aquest punt només s'ha definit com serà la base de dades de la figura 1. Però una vegada s'han indexat els documents, la informació s'ha d'extreure per mitjà de consultes/cerques/*queries* específiques per arribar a la visualització del resultat d'aquestes consultes.

Per poder oferir aquestes funcionalitats, Elasticsearch implementa la base de dades escollida com a suport, una BD NoSQL orientada a documents. Els motors de cerca més populars del mercat són:

17 systems in ranking, June 2018

Rank			DBMS	Database Model	Score		
Jun 2018	May 2018	Jun 2017			Jun 2018	May 2018	Jun 2017
1.	1.	1.	Elasticsearch 	Search engine	131.04	+0.60	+19.48
2.	2.	 3.	Splunk	Search engine	65.78	+0.68	+8.26
3.	3.	 2.	Solr	Search engine	62.06	+0.55	-1.55
4.	4.	4.	MarkLogic	Multi-model 	10.97	+0.58	-1.16
5.	5.	5.	Sphinx	Search engine	6.38	+0.06	-0.39
6.	6.	6.	Microsoft Azure Search	Search engine	4.19	-0.05	+1.14
7.	7.	 8.	Algolia	Search engine	3.22	+0.06	+0.84
8.	8.	 7.	Google Search Appliance	Search engine	2.70	+0.02	-0.34
9.	9.	9.	Amazon CloudSearch	Search engine	2.13	-0.15	-0.22
10.	 11.	 11.	Xapian	Search engine	0.45	-0.01	-0.14
11.	 10.	 10.	CrateDB 	Multi-model 	0.35	-0.13	-0.51
12.	12.	12.	SearchBlox	Search engine	0.18	-0.07	-0.12
13.	13.	 16.	DBSight	Search engine	0.00	±0.00	-0.07
13.	13.	 14.	Exorbyte	Search engine	0.00	±0.00	-0.18
13.	13.	13.	Indica	Search engine	0.00	±0.00	-0.24
13.	13.		Manticore Search	Search engine	0.00	±0.00	
13.	13.	 17.	searchxml	Multi-model 	0.00	±0.00	±0.00

Figura 4. Ranking dels motors de cerca [7]

Els gestors del sistema estan interessats a utilitzar software *open-source* per motius de seguretat. Aquesta és una de les raons perquè utilitzen Debian als servidors. A nivell *open-source* i amb funcionalitats similars a Elasticsearch, podem trobar Solr. Sense entrar en més detalls, tots dos són similars i estan basats en Apache Lucene però Elasticsearch disposa d'una popularitat i comunitat major, fent que com a mínim la seva documentació i solucions dels errors siguin millors. A més la utilitat d'Elasticsearch es multiplica amb l'ús integrat amb altres tecnologies d'Elastic Stack:

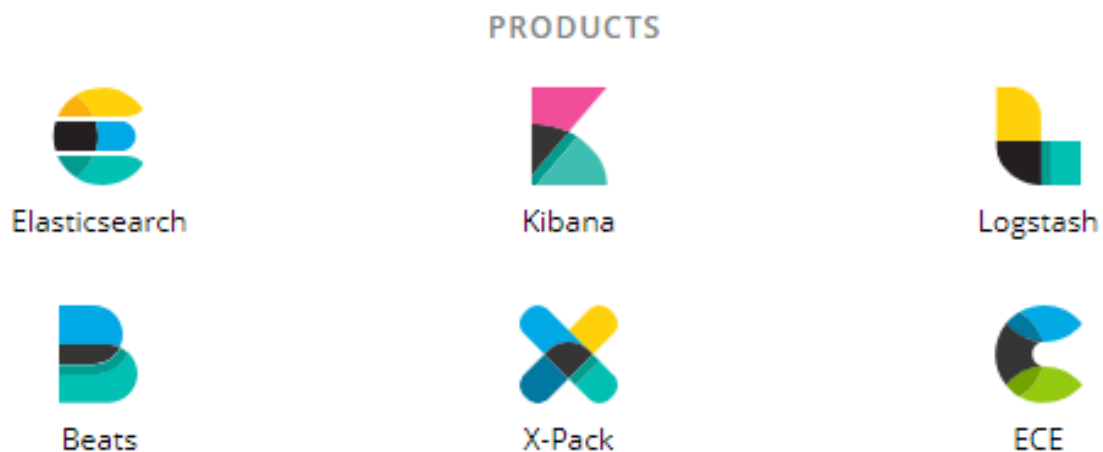


Figura 5. Elastic Stack [8]

Arribant a oferir un ampli ventall de solucions:

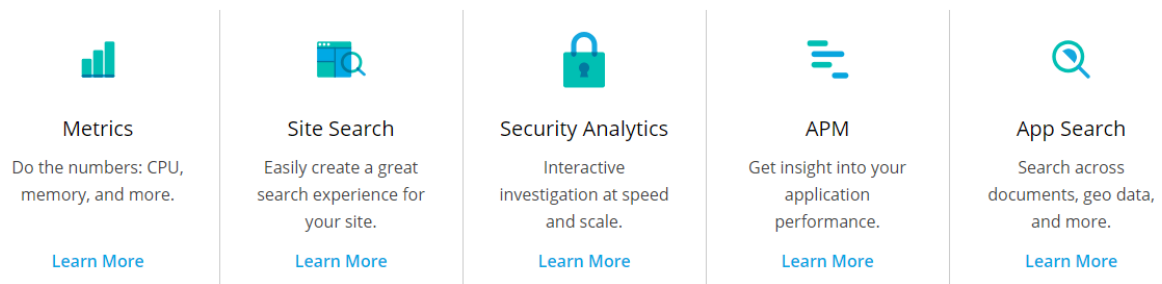


Figura 6. Solucions d'Elastic Stack [8]

El motiu principal llavors per utilitzar Elasticsearch és per les seves capacitats:

- Com a motor de cerca.
- Analítiques, agregacions i integració amb Kibana.
- Recol·lecció de logs, mètriques i integració amb Logstash.
- Emmagatzematge de gran quantitat de documents.
- Plena flexibilitat en la distribució i escalabilitat d'Elastic Stack.
- Seguretat, *open-source*, bona documentació.

3.5 Redundància i Rendiment

Duplicant les dades per diferents nodes d'un clúster obtenim redundància i més paral·lelisme en la lectura de les dades. No obstant hi ha un cost major per crear, modificar i eliminar elements, ja que els nodes s'han de sincronitzar per mantenir una consistència global encara que sigui eventual.

El paral·lelisme no només es troba en la lectura sinó que també es pot trobar en el tractament de les dades quan es llegeixen. Per exemple, en una cerca d'elements, el processament que cal fer es pot repartir entre els diferents nodes que contenen les dades i així millorar el rendiment del sistema.

Com més nodes, més redundància i rendiment obtenim. Tal com veurem després, per motius de *split brain*, el mínim de nodes necessaris perquè el clúster sigui tolerable a particions són 3. Aquest fet sumat amb una limitació de recursos econòmics per simular un clúster físic i una limitació de recursos computacionals per simular un clúster virtual, ens porta a dissenyar un clúster de 3 nodes en l'entorn de desenvolupament.

Elasticsearch facilita molt la configuració i escalabilitat del clúster, informant en tot moment del rendiment i estat del sistema. Amb una configuració per defecte *out-of-the-box* només hem de canviar unes quantes línies a *elasticsearch.yml* perquè el clúster comenci a funcionar.

3.6 Anàlisi Automàtic de Dades

La Intel·ligència Artificial (IA) és una disciplina que ens proporciona diverses tècniques per fer una anàlisi de dades i extreure coneixement que pot ser de gran utilitat per a l'usuari del sistema [3]. Dins de la IA hi ha diferents branques d'estudi. En el cas de tractament de dades textuais es parla de "Processament del Llenguatge Natural" (NLP).

Com que en aquest projecte es vol analitzar el text lliure dels documents mèdics es necessiten eines de NLP. Elasticsearch implementa per defecte moltes d'aquestes eines, entre les quals hi ha un analitzador predefinit de català que ens serà de gran utilitat en aquest projecte. Aquest analitzador s'executarà automàticament cada vegada que indexem o consultem documents amb text en català.

3.7 Normativa GDPR Europea

La GDPR és una regulació de les lleis europees sobre protecció i privacitat de dades, que afecta a tots els individus de la unió europea i de l'espai econòmic europeu [9]. Aquesta regulació que va entrar en vigor el 25 de maig de 2018, té com a objectiu cedir el control als ciutadans sobre les seves dades personals i simplificar l'entorn regulador en la unió europea.

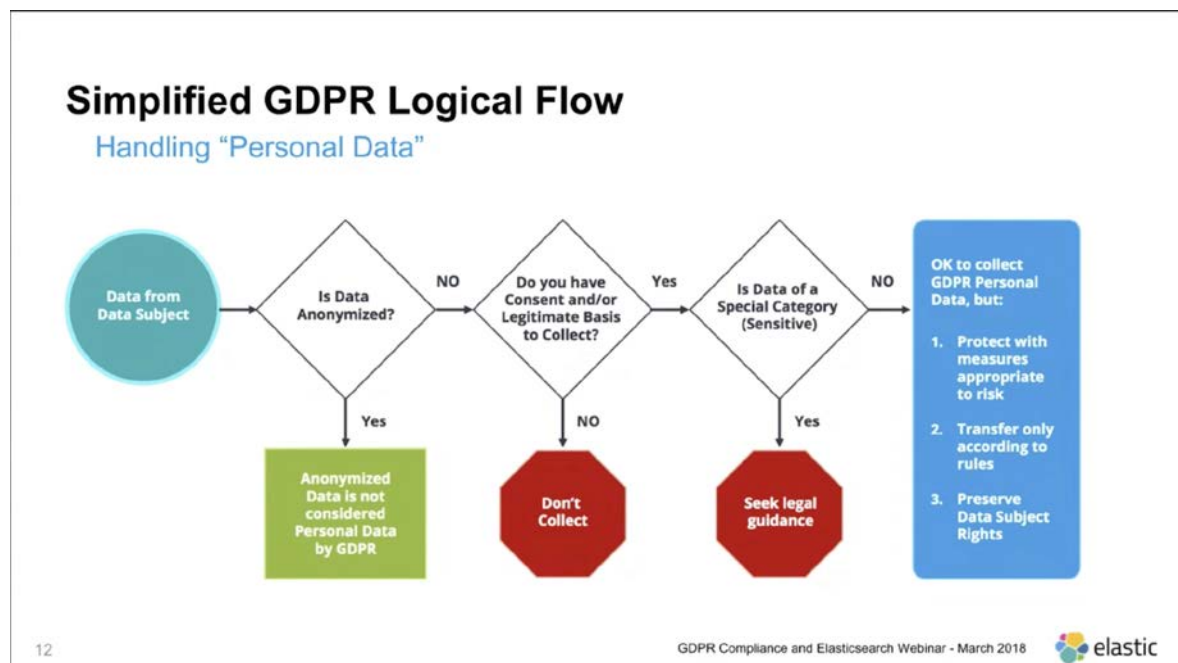


Figura 7. Normativa GDPR [10]

Elastic Stack està dissenyat per poder superar els requisits de la normativa europea sobre dades de protecció personal. Per exemple, si s'anonimitzen les dades personals dels usuaris, aquestes dades deixen de ser considerades com personals per GDPR i per tant ja no caldria aplicar les mateixes mesures estrictes de protecció. L'anonimització és vàlida si és irreversible. Logstash permet transformar dades personals en camps *fingerprint* identificadors mitjançant operacions de *hashing*.

3.8 Arquitectura del Sistema

Amb el sistema DSS proporcionem una informació útil als administradors perquè puguin resoldre els problemes de forma més eficient o fins i tot puguin prevenir-los [11].

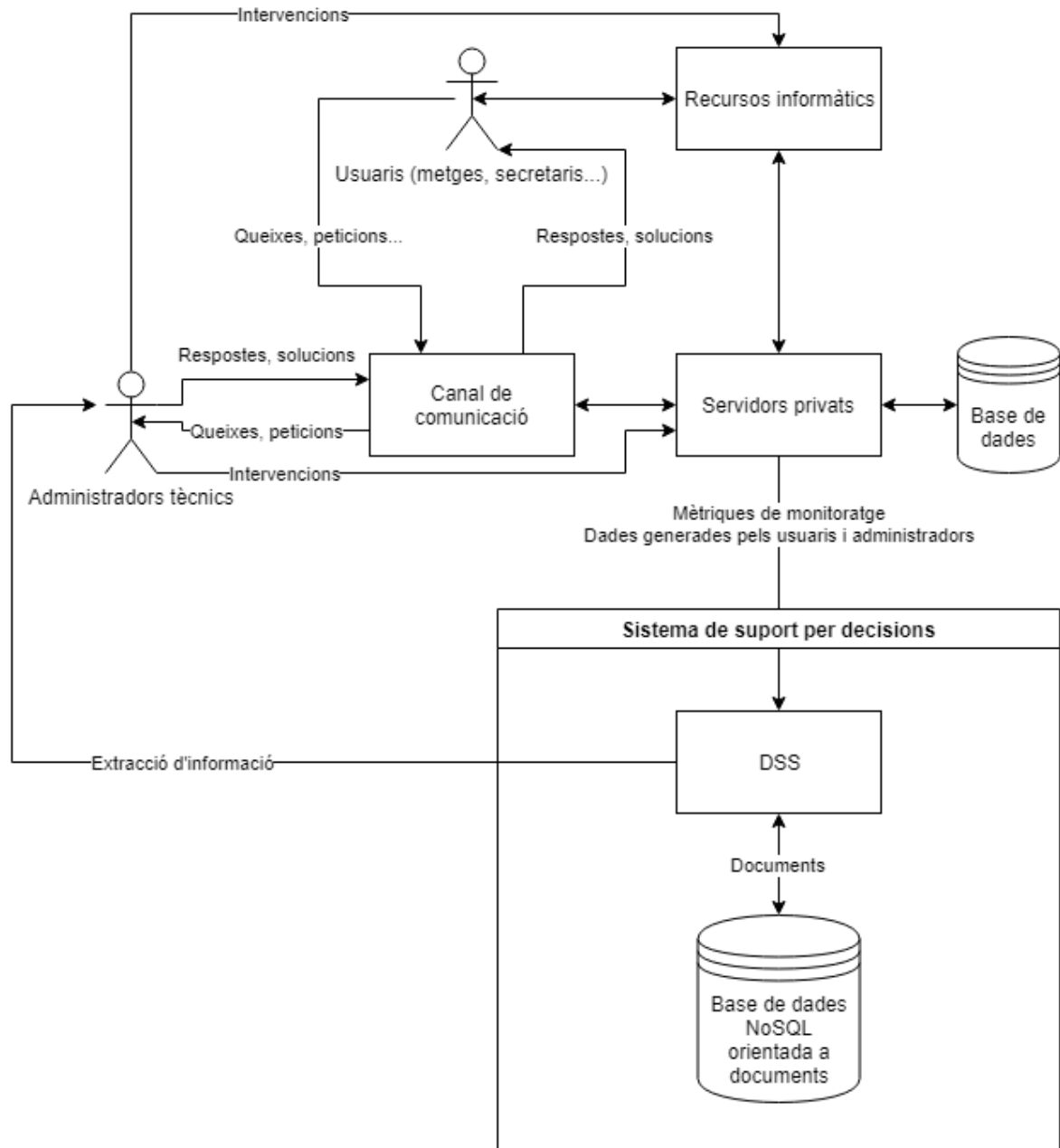


Figura 8. Integració DSS

Cal ressaltar que cap element crític del sistema d'informació resultarà negativament afectat per la integració del nou DSS, ja que la base de dades NoSQL s'utilitza només per l'anàlisi de dades i extracció d'informació.

4. Introducció a Elastic Stack

Elastic Stack és una col·lecció de software *open-source* basat en els programes Elasticsearch, Logstash i Kibana. Aquesta col·lecció està dissenyada per extreure el màxim profit quan s'utilitzen a la vegada de forma integrada, però Elasticsearch i Logstash també es poden utilitzar de manera independent amb altre tipus de software.

4.1 Introducció a Elasticsearch

Elasticsearch és un motor de cerca i anàlisi de dades altament escalable, amb una BD NoSQL orientada a documents com a suport. Permet emmagatzemar, cercar i analitzar grans quantitats de dades de forma instantània. Es tracta d'una tecnologia reconeguda mundialment i utilitzada en organitzacions tan grans com Wikipedia, The Guardian, Stack Overflow, Github...

La unitat bàsica d'informació que es guarda en aquest sistema és el document. Elasticsearch és una plataforma de cerca quasi a temps real. Això significa que hi ha una petita latència (normalment d'un segon) entre el moment d'indexació d'un document fins que aquest es troba disponible per a la cerca.

4.1.1 Documents

Uns dels motius pels quals l'OOP és tan popular, és perquè els objectes ens ajuden a representar i manipular entitats del món real amb estructures de dades complexes.

Tradicionalment les dades s'han guardat persistentment en files i columnes en bases de dades relacionals, que són incompatibles amb la flexibilitat i expressivitat dels objectes.

Però hi ha una forma de guardar els objectes com a tal, sense perdre flexibilitat. Un objecte depèn del llenguatge de programació i és una estructura de dades resident a memòria. Per enviar, guardar i recuperar l'objecte en una BD, cal representar-ho amb el format estàndard NoSQL, anomenat JSON. Aquest format de fitxer de text és fàcil de llegir i permet la compatibilitat de dades entre diferents programes, millorant la interoperabilitat. Quan un objecte és serialitzat a JSON, el resultat és un document JSON o simplement un document.

JSON està format per parelles clau-valor. La clau és el nom d'algun camp, que dona sentit semàntic al valor. Aquest valor pot ser un *string*, un número, un booleà, un *array* de valors, o una altra parella clau-valor. Com per exemple:

```
{
  "name": "John Smith",
  "age": 42,
  "confirmed": true,
  "join_date": "2014-06-01",
  "home": {
    "lat": 51.5,
    "lon": 0.1
  },
  "accounts": [
    {
      "type": "facebook",
      "id": "johnsmith"
    },
    {
      "type": "twitter",
```

```

    "id": "johnsmith"
  }
]
}

```

Codi 1. Exemple JSON [12]

Quan s’indexa un document a la BD, se li afegeix metadades per poder administrar-ho correctament. Els documents d’ES són immutables, per tant, si es vol alterar el contingut d’un document, es reindexa el document sencer.

4.1.2 Índexs

Un índex és una ubicació lògica en Elasticsearch que utilitzem per referir-nos a una carpeta o container lògic de documents indexats a la BD. El que succeeix físicament és que els documents pertanyen a *shards* primaris. Cada *shard* primari pot tenir fins a un màxim teòric de `Integer.MAX_VALUE` – 128 documents. Per tant si necessitem més documents que aquesta quantitat, assignarem més d’un *shard* primari a un determinat índex.

4.1.3 Tipus

És un element obsolet a partir de la versió ES 6.X, però que encara s’utilitza parcialment per compatibilitat. Antigament un índex es podia subdividir en tipus que realment eren els containers lògics dels documents. Però ara un índex només pot tenir el tipus “doc”, així que pràcticament podem dir que l’índex és el contenidor lògic dels documents.

4.1.4 Shards

Apache Lucene és una llibreria *open-source* de cerca d’alt rendiment escrita en Java. És una tecnologia adient per qualsevol aplicació que necessiti cerca *full-text*, especialment per a un entorn multiplataforma.

Els *shards* són instàncies d’Apache Lucene, executant-se en una instància d’Elasticsearch. Normalment només hi ha una instància ES per cada node físic del clúster. En ES s’utilitza indistintament la paraula “instància d’ES” i “node (lògic)”.

Els *shards* es divideixen en dues categories, primaris i secundaris. Quan es crea un índex, s’especifiquen el nombre de *shards* primaris i secundaris. Els primaris serveixen per distribuir els documents amb *hashing*. Per exemple, si tenim 5 *shards* primaris i volem indexar un document a l’índex format per aquests *shards*, s’assignarà un *shard* a aquest document mitjançant *hashing*, i el document acabarà indexat a un dels 5 *shards* amb una probabilitat uniforme. Així que els *shards* són els containers físics dels documents.

$$\text{shard} = \text{hash}(\text{routing}) \% \text{number_of_primary_shards} \quad (1)$$

El *routing* per defecte és el valor del camp “_id” que figura en les metadades del document.

Els *shards* secundaris serveixen per redundància i són rèpliques dels *shards* primaris repartits pels nodes del clúster, de manera que cada node tingui un nombre similar de *shards* del mateix índex. Les operacions d’escriptura com ara la creació i l’eliminació d’un document, s’efectuen primer als *shards* primaris abans de replicar els canvis als *shards* secundaris.

A diferència dels secundaris, la quantitat de *shards* primaris no es poden modificar una vegada creat l'índex. Si canviéssim el nombre de *shards* primaris, no trobaríem el *shard* propietari d'un document indexat abans del canvi perquè el resultat de la fórmula (1) seria diferent.

Elasticsearch és horitzontalment escalable perquè es poden crear *shards* secundaris a mesura que afegim més nodes al clúster sense pràcticament limitació.

4.1.5 Node

Existeixen 5 tipus de nodes dins d'un clúster:

- **Master:** És responsable d'executar accions que afecten a tot el clúster com ara crear o esborrar un índex, seguiment dels nodes del clúster, decidir quins *shards* s'assignen als nodes...

Per defecte tots els nodes són candidats a ser el *master*, però els nodes candidats es posen d'acord per decidir-ne només un.

- **Data-node:** Serveixen per emmagatzemar persistentment els documents dels *shards*. Aquests nodes són responsables de les operacions CRUD, cerca i agregacions. Aquests nodes són els que més sobrecarrega I/O, memòria i CPU tenen.
- **Ingest-node:** Processen i modifiquen els documents just abans d'indexar-los a la BD.
- **Coordinating-only-node:** Encamina el tràfic de peticions, ja que tenen informació sobre l'estat del clúster.

Es comporten com balancejadors de càrrega per peticions de lectura de documents on tant els *shards* primaris com els secundaris poden respondre. Escullen una còpia de *shard* diferent per cada petició per així repartir la càrrega

Per exemple, una petició de cerca que necessita consultar tots els documents d'un determinat índex s'executa en dues fases. En la fase *scatter*, el node coordinador trasllada la petició als data-nodes propietaris dels *shards* que mantenen l'índex corresponent. Aquests data-nodes cerquen en paral·lel i retornen els resultats al node coordinador. En la fase *gather*, el node coordinador replega i junta els resultats en un únic document que retorna al client com a resposta.

Per peticions d'escriptura de documents s'adrecen directament al data-node que té el *shard* primari corresponent

Tots els tipus de nodes són coordinadors implícitament, poden rebre peticions HTTP i redirigir el tràfic segons la petició. Si hi ha més d'un node coordinador disponible, es recomana repartir les peticions en comptes de comunicar-se només amb un.

- **Machine-learning node:** Executen tasques de *machine learning* per detectar anomalies relacionades amb valors o freqüències, excepcions estadístiques, comportaments inusuals d'un membre en una població...

Per defecte un node té els 4 primers rols. En un entorn de producció caldria tenir en compte aquests rols per repartir la feina i evitar saturacions de càrrega en alguns nodes.

4.1.6 API REST

Els usuaris poden comunicar-se amb ES mitjançant RESTful API al port 9200. Els mètodes HTTP s'associen amb les operacions CRUD seguint l'esquema següent:

Create i Update => PUT, POST

Read => GET, HEAD

Delete => DELETE

La URL s'utilitza segons el format que especifiquen les APIs de totes les funcionalitats disponibles d'ES per al client. Aquestes APIs a vegades requereixen un cos HTTP en format JSON per especificar com s'utilitza l'operació.

Un resum de les funcionalitats que ES ofereix, són les següents:

- Document APIs: Administració de documents.
- Search APIs: Obtenir informació sobre la BD i el seu contingut.
- Indices APIs: Administració i configuració dels índexs.
- cat APIs: Mostra informació sobre el sistema de forma molt clara.
- Cluster APIs: Mostra informació i estadístiques sobre el clúster i el seu estat.
- Query DSL: Defineix les especificacions de cerca en el cos HTTP en format JSON.
- X-Pack APIs: Per configurar i accedir a característiques de X-Pack.

Els usuaris també tenen l'opció de fer ús de comunicació segura HTTPS al mateix port si es configura Elasticsearch perquè faci ús del mòdul de seguretat proporcionat per X-Pack.

4.1.7 Exploració i Descobriment de Nodes

Elasticsearch implementa un protocol propi anomenat Zen per l'exploració i descobriment dels nodes del clúster. Perquè un node es pugui unir al clúster només cal que contacti amb un node que ja hi formi part.

4.1.8 Funcionament Distribuït

Elasticsearch intenta fer transparent la complexitat dels sistemes distribuïts. Alguns exemples de les tasques de les quals s'encarrega ES automàticament són:

- Particionar els documents en diferents contenidors (*shards*), que poden ser guardats en un únic node o en múltiples nodes.
- Repartir els *shards* de manera equitativa entre els nodes del clúster per balancejar la indexació i la cerca.
- Repartir còpies de redundància dels *shards* primaris per disminuir el risc de pèrdua de dades en cas de fallida de hardware.
- Reenviar peticions des de qualsevol node cap als nodes que tenen les dades desitjades.

- Integrar nous nodes al clúster o redistribuir els *shards* per recuperar-se de la pèrdua dels nodes.

4.1.9 Split Brain

Quan hi ha un error de xarxa, per exemple un encaminador avariament que desconnecta el clúster i el separa en dos o més grups, pot aparèixer el fenomen *split brain* si no es té en compte la consistència del sistema.

Si el clúster es descompon en subgrups independents, arribaríem a un estat inconsistent amb dades malmeses. Per evitar això podem configurar els nodes perquè només formin un clúster i operin amb normalitat si la majoria de nodes es troben disponibles, aconseguint així només un únic clúster en cada instant mantenint consistència eventual.

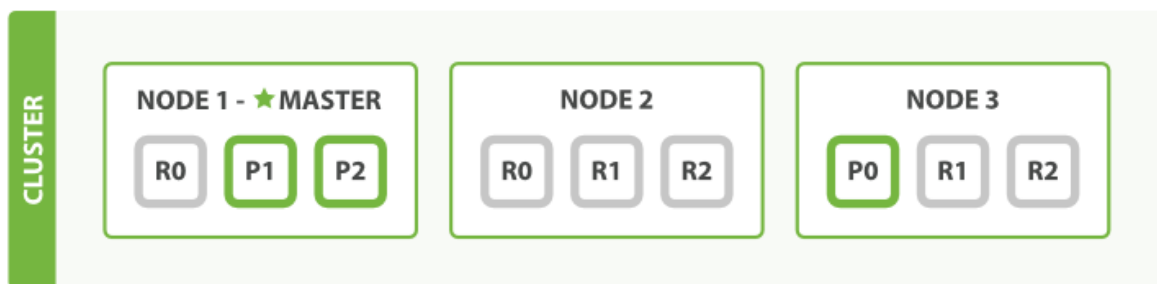


Figura 9. Abans de la fallida [13]

Elasticsearch pot elevar la categoria dels *shards* secundaris a *shards* primaris si els primaris no es troben disponibles, mantenint així la disponibilitat d'indexació.



Figura 10. Després de la fallida del node 1 [14]

En les figures 9 i 10 es demostra aquest comportament amb els *shards* primaris (P0, P1, P2) i els secundaris (R0, R1, R2).

4.1.10 *Templates*

Elasticsearch proporciona un model *schema-free* on no cal definir cap estructura de document abans de fer la indexació. Aquest és un dels avantatges de les BD orientades a documents. No obstant això, Elasticsearch ha d'associar algun tipus als camps indexats i per tant els ha d'endevinar en cas que no s'hi defineixin prèviament.

Els tipus de camps poden ser els següents:

Core datatypes:

- *String*
 - o *text and keyword*
- *Numeric datatypes*
 - o *long, integer, short, byte, double, float, half_float, scaled_float*
- *Date datatype*
 - o *Date*
- *Boolean datatype*
 - o *Boolean*
- *Binary datatype*
 - o *Binary*
- *Range datatypes*
 - o *integer_range, float_range, long_range, double_range, date_range*

Complex datatype

- *Array datatype*
 - o *Array support does not require a dedicated type*
- *Object datatype*
 - o *object for single JSON objects*
- *Nested datatype*
 - o *nested for arrays of JSON objects*

Geo datatypes

- *Geo-point datatype*
 - o *geo_point for lat/lon points*
- *Geo-Shape datatype*
 - o *geo_shape for complex shapes like polígons*

Specialised datatypes

- *IP datatype*
 - o *ip for IPv4 and IPv6 addresses*
- *Completion datatype*
 - o *completion to provide auto-complete suggestions*
- *Token count datatype*
 - o *token_count to count the number of tokens in a string*
- *mapper-murmur3*

- *murmur3 to compute hashes of values at index-time and store them in the index*
- *Percolator type*
 - *Accepts queries from the query-dsl*
- *join datatype*
 - *Defines parent/child relation for documents within the same index*

Multi-fields

A vegades és útil indexar el mateix camp de formes diferents per diferents finalitats. Un camp de tipus *string* es podria definir a la vegada com un camp de text per cerca *full-text* i com un camp *keyword* per poder fer ordenacions i agregacions en Kibana. La majoria de tipus suporten *multi-fields* mitjançant el paràmetre *fields*.

Amb tantes possibilitats de tipus, és probable que Elasticsearch no apliqui el tipus que volem que endevini. És per això que Elasticsearch permet la definició i tipus de camps a través dels anomenats *templates* abans de fer la creació d'un índex per la posterior indexació de documents.

A banda de predefinir el mapeig dels camps, els *templates* també serveixen per establir la configuració de l'índex, com la quantitat de *shards* primaris i secundaris que es dedicaran a l'índex quan aquest es creï.

La dinàmica per utilitzar *templates* és la següent:

1. Tenim una font d'entitats/esdeveniments similars, amb camps anomenats amb les mateixes etiquetes, que volem indexar a la BD.
2. Definim aquests camps amb les mateixes etiquetes en un *template*, especificant el seu tipus i les seves propietats. Definim el nom de l'índex pel qual aquest *template* s'aplicarà i configurem les opcions d'aquest l'índex si cal.
3. Enviem el *template* a Elasticsearch mitjançant API REST.
4. Creem l'índex pel qual es va destinar el *template*.
5. Indexem els documents del punt 1.

4.1.11 Inverted Index

Elasticsearch utilitza una estructura anomenada *inverted index* per permetre cerques *full-text* molt ràpides. Un índex invertit consisteix en una llista ordenada sense repeticions de totes les paraules que apareixen en qualsevol document, i per cada paraula, una llista dels documents en la que aquesta apareix:

Per exemple, si tenim dos documents amb un camp amb la mateixa etiqueta X que contenen respectivament el text:

1. "The quick brown fox jumped over the lazy dog"
2. "Quick brown foxes leap over lazy dogs in summer"

Per crear l'índex invertit, primer trunquem el contingut dels camps en paraules (també anomenat termes o *tokens*) i després les afegim a la llista de forma única. L'acció

d'afegir *tokens* a l'índex invertit i marcar en quins documents es troba aquest *token* l'anomenem **indexació**:

Term	Doc_1	Doc_2
Quick		X
The	X	
brown	X	X
dog	X	
dogs		X
fox	X	
foxes		X
in		X
jumped	X	
lazy	X	X
leap		X
over	X	X
quick	X	
summer		X
the	X	

Figura 11. Índex invertit del camp X després de la indexació dels dos documents [15]

Tal que per la cerca “quick brown”, Elasticsearch ens retornaria els dos documents perquè “quick” apareix al primer document mentre que “brown” apareix als dos documents:

Term	Doc_1	Doc_2
brown	X	X
quick	X	
Total	2	1

Figura 12. Resultats de la cerca

Perquè puguem trobar “quick” o “brown” a l'índex invertit a partir de la cerca “quick brown”, s'ha d'aplicar el mateix procés de normalització en la *query*, que en aquest cas només consisteix a trencar el text en *tokens*.

4.1.12 Analitzadors

Hi ha casos que és necessari aplicar un procés de normalització més complex. Per exemple:

- Si tractem “Quick” i “quick” com paraules diferents, hi haurà casos que cercant “Quick” no rebrem cap resultat perquè es va indexar “quick” en comptes de “Quick”.
- Si tractem “fox” i “foxes” com paraules diferents, passa el mateix que abans.
- Ídem per paraules sinònimes com “jumped” i “leap”.

Per resoldre aquest tipus de problemes lingüístics existeixen els analitzadors que processen de la mateixa manera tant els camps a indexar com les *queries* sobre aquests camps, aplicant transformacions més enllà que truncar el text per espais en blanc.

Un analitzador predefinit o personalitzat és un paquet que conté tres nivells aplicats en ordre:

- **Filtre de caràcters:**

Rep el text original com un *stream* de caràcters que pot transformar afegint, eliminant o canviant aquests caràcters. Per exemple un filtre de caràcter podria ser utilitzar per eliminar elements HTML com “” del *stream*. Un analitzador pot tenir zero o més filtres de caràcters.

- **Tokenizer:**

Rep el *stream* de caràcters filtrat, el trenca en *tokens* (normalment paraules) i retorna un *stream* de *tokens*. Per exemple, el *whitespace tokenizer* trunca el text en *tokens* partint el text pels espais en blanc. Per exemple, convertiria el *stream* de caràcters "Quick brown fox!" en els termes [Quick, brown, fox!]. Un analitzador només pot tenir un *tokenizer*.

- **Filtre de tokens:**

Afegeix, elimina o canvia *tokens* del *stream* de *tokens*. Per exemple, *lowercase token filter* converteix tots els *tokens* a minúscules. Un analitzador té zero o més filtres de tokens.

Després de l'últim nivell, els *tokens* resultants es guarden en l'índex invertit del camp corresponent si estem indexant camps de documents, o es cerquen en aquest índex si estem fent una *query*.

Per finalitzar, cal entendre molt bé la diferència entre el tipus text i *keyword* per als *strings* indexats en Elasticsearch. Si definim un camp com tipus text, Elasticsearch aplicarà automàticament un analitzador *standard* perquè es puguin fer cerques *full-text* a posteriori sobre aquest camp. Aquesta funcionalitat és útil per text lliure. Però si es tracta de *strings* que només tenen sentit com un conjunt compacte de paraules i símbols especials, per exemple un identificador o una categoria de document, s'han d'indexar en la seva totalitat sense cap analitzador. Per això existeix el tipus *keyword*. Si un *string* pot ser útil amb les dues funcionalitats, el camp es pot declarar a la vegada com text i *keyword* utilitzant el tipus *multi-field*.

4.1.13 Rellevància en les Cerques

Elasticsearch permet dues modalitats de cerques:

- **Filter context:** Retorna els documents que tenen termes determinats.
- **Query context (full-text):** A banda de retornar els documents que tenen determinats termes, calcula una puntuació per cada document retornat, basada en la similitud del contingut del document amb la cerca.

En el cas de **query context**, Elasticsearch ens retornarà un conjunt de documents en ordre decreixent de rellevància segons la consulta que li fem.

La rellevància o puntuació de cada document es representa per un número positiu decimal associat al camp “_score” en el JSON retornat per Elasticsearch. Com més alt sigui el valor d’aquest camp, més rellevant serà el document.

Quan fem una consulta en *query context* no ens interessa només que els documents retornats continguin paraules de la consulta, sinó que volem saber amb quina exactitud s’assemblen a la consulta realitzada. Llavors per comparar la similitud de dos documents només necessitem comparar el seu camp “_score” per determinar quin document és més rellevant segons la consulta.

Per calcular el camp “_score” i decidir quant s’apropa un document respecte al text de la *query*, s’utilitzen algorismes de similitud semàntica. Podem personalitzar el nostre algorisme de similitud en Elasticsearch, o podem seleccionar un dels que Elasticsearch ja té implementats:

- Okapi BM25 based on TF/IDF model
- Classic based on TF/IDF model
- DFR
- DFI
- IB
- LM Dirichlet
- LM Jelinek Mercer

L’estudi de cada algorisme queda fora dels objectius d’aquesta memòria perquè no afectarà la implementació proposada en el capítol 5. Només explicarem l’algorisme Okapi BM25 perquè és l’algorisme que Elasticsearch utilitza per defecte i és un algorisme típic en *information retrieval* utilitzat pels cercadors de text per determinar la rellevància dels documents. Aquest algorisme està basat en el model TF/IDF [16]. Donada una *query* Q (exemple “sabates vermelles”) amb n keywords² $q_0, q_1 \dots q_i$ (q_0 = sabates, q_1 = vermelles), calculem el camp “_score” pel document D amb els següents paràmetres:

$$_score(D, Q) = \sum_{i=1}^n IDF(q_i) \cdot \frac{TF(q_i, D) \cdot (k_1 + 1)}{TF(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{avgdl}\right)} \quad (2)$$

Per defecte, Elasticsearch té els paràmetres $k_1 = 1.2$ i $b = 0.75$; on $|D|$ és el nombre de paraules que té el document D i *avgdl* és la mitjana del nombre de paraules que hi ha en els N documents. Com més gran sigui TF, més rellevant serà el document, ja que té un efecte multiplicador al numerador:

$$TF(q_i, D) = \sqrt{\text{vegades que el terme } q_i \text{ apareix al document } D} \quad (3)$$

Quan en més documents aparegui q_i , menor serà IDF i per tant menor serà la rellevància que aportarà aquest terme, contrarestant així un possible TF alt:

² N i n són paràmetres diferents

$$IDF(q_i) = \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} \quad (4)$$

On N és el nombre total de documents que estem cercant i $n(q_i)$ és el nombre de documents que contenen el terme q_i

Si estem cercant tots els documents d'un índex, Elasticsearch primer filtra de forma booleana tots els documents que tinguin els termes “sabates” OR “vermelles” abans d'aplicar l'algorisme per així començar l'algorisme amb només els documents que puguin ser rellevants i executar-ho més eficientment. Per tant el paràmetre N serà el nombre de documents resultants d'aplicar el filtre booleà. Podem modificar els paràmetres d'una cerca basada en TF/IDF perquè en comptes de OR s'utilitzi AND.

A la secció 5.3 veurem com Elasticsearch ignora aquest algorisme en el nostre projecte per classificar els documents en categories.

4.1.14 *Similitud entre Termes*

Un *match* es produeix quan un terme de l'índex invertit coincideix amb un terme d'una *query* després de ser processat per l'analitzador (si s'analitza). Però en el món real poden haver errors ortogràfics tant en els índexs invertits com en la *query*. Per permetre que dos termes, un correcte i un altre amb errors es tractin com la mateixa paraula introduïm la distància de Damerau-Levenstein (o *edit distance*) que mesura la mínima quantitat de transformacions d'un caràcter que calen aplicar a una paraula errònia perquè sigui correcta.

Existeixen quatre tipus de transformacions:

- Substitució: pebre → negre
- Inserció: millor → millora
- Esborrament: taula → aula
- Transposició: cantar → acntar

Damerau va observar que el 80% de les errates lingüístiques produïdes tenen una distància 1. De fet, considerar distàncies majors pot tenir un impacte negatiu perquè dues paraules diferents i correctes podrien considerar-se iguals aplicant més d'una transformació.

4.1.15 *Precision vs Recall*

La qualitat dels algorismes de cerca intel·ligent de documents solen avaluar-se tenint en compte dues mesures:

- **Precision:** Retornar el mínim possible de documents irrellevants.
- **Recall:** Retornar el màxim nombre de documents rellevants.

Per calcular aquestes mesures es necessita tenir un conjunt de documents prèviament etiquetats segons la informació que s'hi busca. Aleshores es poden extreure els següents indicadors:

	Retornats?	No retornats?
Documents classificats	Rellevants?	Vertader positiu (TP)
	Irrellevants?	Fals positiu (FP)
		Fals negatiu (FN)
		Vertader negatiu (TN)

Taula 1. Resultats de la cerca [17]

Quan major sigui el nombre de vertaders positius i negatius, millor serà la cerca. Anàlogament, quan major siguin el nombre de falsos positius i negatius, pitjor serà la cerca.

Per mesurar la qualitat de la cerca es tenen en compte les següents mètriques:

$$Precision = \frac{TP}{TP + FP} \quad (5)$$

$$Recall = \frac{TP}{TP + FN} = \text{True Positive Rate (TPR)} \quad (6)$$

$$\text{False Positive Rate (FPR)} = \frac{FP}{FP + TN} \quad (7)$$

$$F1 \text{ score} = \frac{2 * (Recall * Precision)}{(Recall + Precision)} \quad (8)$$

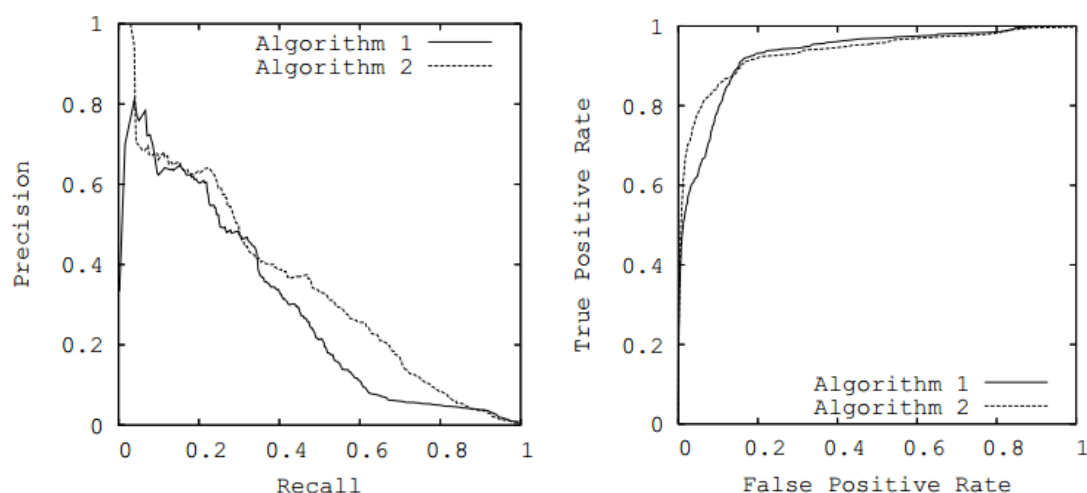


Figura 13. Gràfiques a partir de les mètriques [18]

4.2 Introducció a Logstash

Logstash és un motor de recol·lecció de dades *open-source* amb capacitat de tractament de dades en temps real. Logstash pot unificar tota mena de dades de diferents fonts i normalitzar-les per enviar-les a diferents destinacions, com ara Elasticsearch. Per aconseguir aquesta tasca, Logstash compta amb una gran quantitat de plugins que faciliten el tractament de dades.

4.2.1 Pipelines

Logstash executa el tractament de dades en tres fases consecutives que formen un *pipeline*:

- **Inputs:** Generen *events*
- **Filters:** Modifiquen els *events*
- **Outputs:** Envien els *events*

Inputs i outputs suporten codificació i descodificació de dades perquè així no s'hagin d'utilitzar diferents filtres segons el format de les dades.

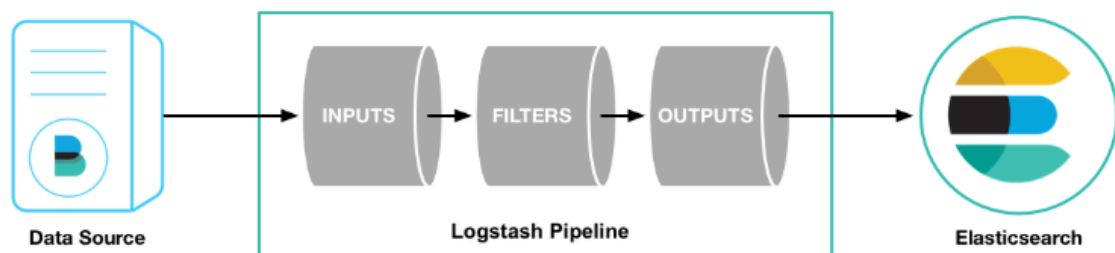


Figura 14. Fases del *pipeline* [19]

Logstash junt amb el framework Beats proporcionen una solució d'indexació de documents escalable i resilient amb les següents característiques:

- Escalabilitat horitzontal, alta disponibilitat, balanceig de càrrega.
- Durabilitat dels *events* amb garantia de com a mínim un enviament satisfactori.
- Transport punt a punt segur amb autenticació i encriptació.

4.2.2 Resiliència de Dades

Mentre les dades són processades pel *pipeline*, hi poden haver algunes situacions que evitin que Logstash pugui enviar els *events* als diferents destinataris. Per exemple, dades que continguin tipus inesperats de dades o un tall de llum podrien provocar que els *events* es perdin. Logstash proporciona dues cues persistents d'*events* per així evitar perdre dades processades a mig camí que no s'hagin pogut entregar al destinatari:

- **Persistent queues:** Guarda tots els *events* en un buffer persistent entre la fase d'input i filtre. Elimina la necessitat d'utilitzar un mecanisme extern de buffer de missatges com ara Redis o Apache Kafka. Garanteix que els *events* aturats en el *pipeline* per aturades de Logstash siguin enviats, ja que es guardaran en el buffer fins que s'enviïn correctament com a mínim una vegada.

- **Dead letter queues:** Quan Logstash troba un *event* que no es pot processar per un error de mapeig o qualsevol altre problema, per defecte Logstash deixa que l'*event* es perdi. Amb *dead letter queues* activat Logstash guarda els *events* insatisfactoris en una cua que consumirà un *pipeline* diferent i específic per tornar a intentar el processament i solucionar-ho.

4.3 Introducció a Kibana

Kibana és una plataforma d'anàlisi i visualització *open-source* dissenyada per funcionar amb Elasticsearch que s'utilitza per cercar, veure i interactuar amb les dades guardades als índexs d'ES, facilitant així l'anàlisi i la comprensió de grans volums de dades en temps real. Com altres components d'Elastic Stack, Kibana també compta amb bastants plugins, però només ens centrarem en els més rellevants per aquest projecte.

4.3.1 Extracció d'Informació de Documents

Kibana és un servidor que fa d'intermediari entre l'usuari i Elasticsearch, fent que la manipulació de dades per via API REST sigui transparent mitjançant un entorn gràfic. Kibana està dissenyat per permetre una gran flexibilitat a l'hora d'extreure la informació que sigui necessària, però els passos generals per arribar a obtenir un conjunt de visualitzacions desitjades són els següents:

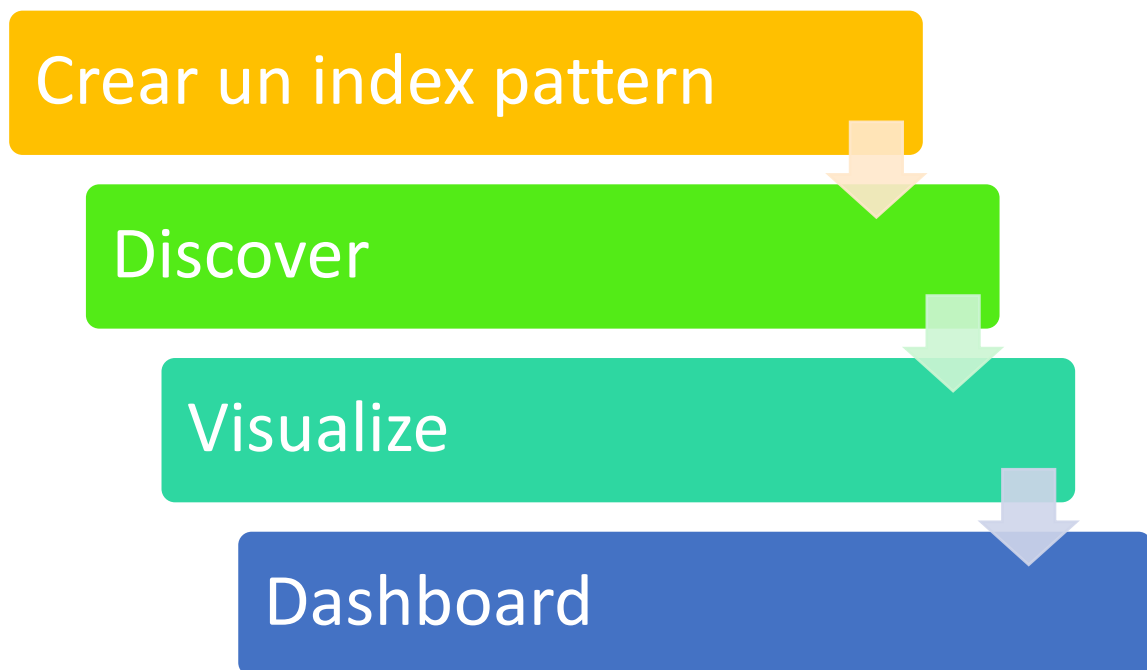


Figura 15. Fases d'exploració a Kibana

- **Crear un index pattern:** Per fer cerques de documents abans s'ha d'especificar quin o quins índexs es volen tenir en compte. Per això, podem escriure el nom de l'índex per restringir la cerca a tal índex, o podem fer ús del *wildcard* "*" per restringir la cerca a un conjunt d'índexs.

Figura 16. Creació del *index pattern*

- **Discover:** En aquesta fase es poden veure tots els documents indexats que pertanyin als índexs trobats pel patró d'índex creat anteriorment.

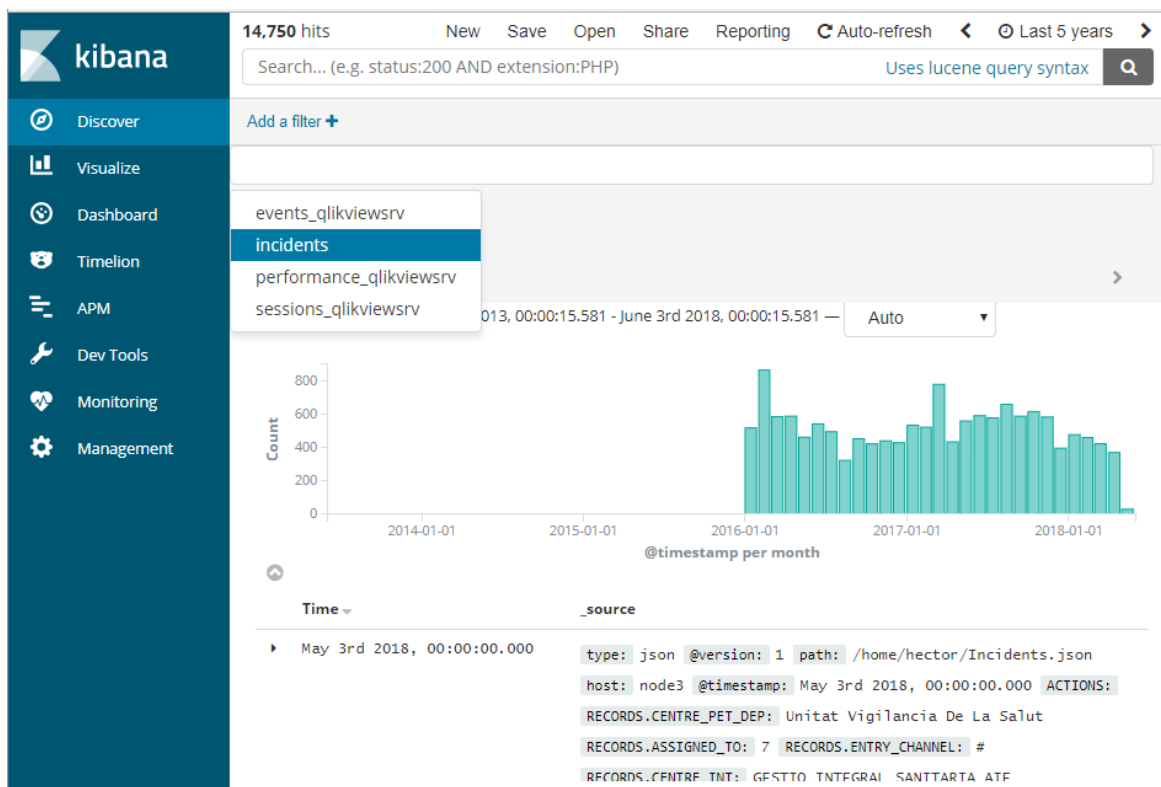


Figura 17. Fase *Discover*

A partir d'aquí es pot utilitzar filtres cronològics o per camps, veure els continguts dels documents, utilitzar *lucene query syntax*... Però la funcionalitat més potent es troba en la personalització de la cerca amb Query DSL:

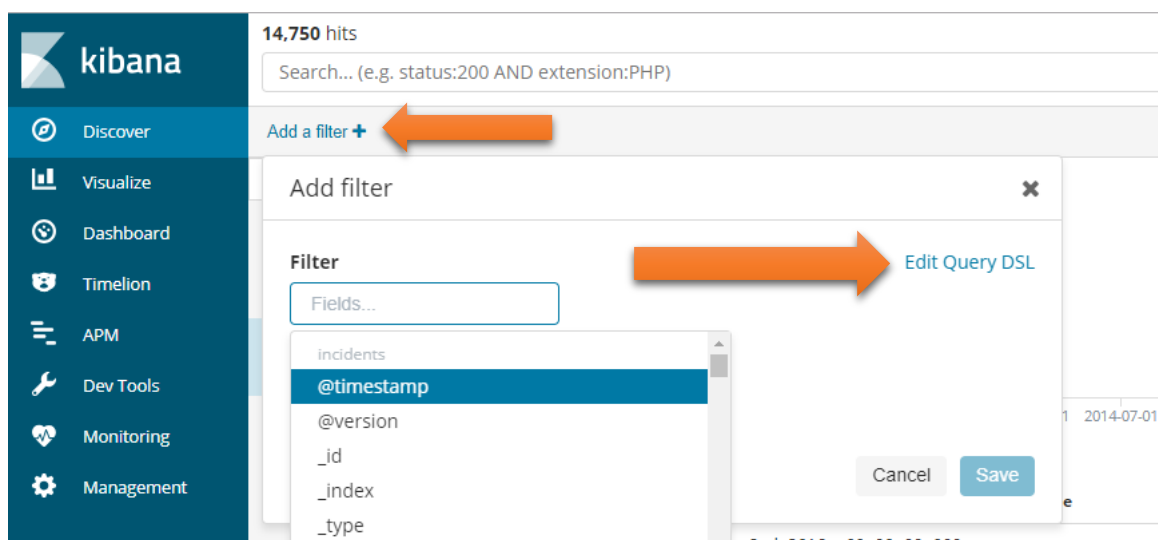


Figura 18. Cerca avançada en Kibana

- **Visualize:** Una vegada filtrats i cercats els documents per recuperar les dades que convinguin, cal guardar la cerca per visualitzar-la en aquesta fase. En la pestanya *Visualize* es pot seleccionar el tipus de visualització més convenient per extreure la informació desitjada.

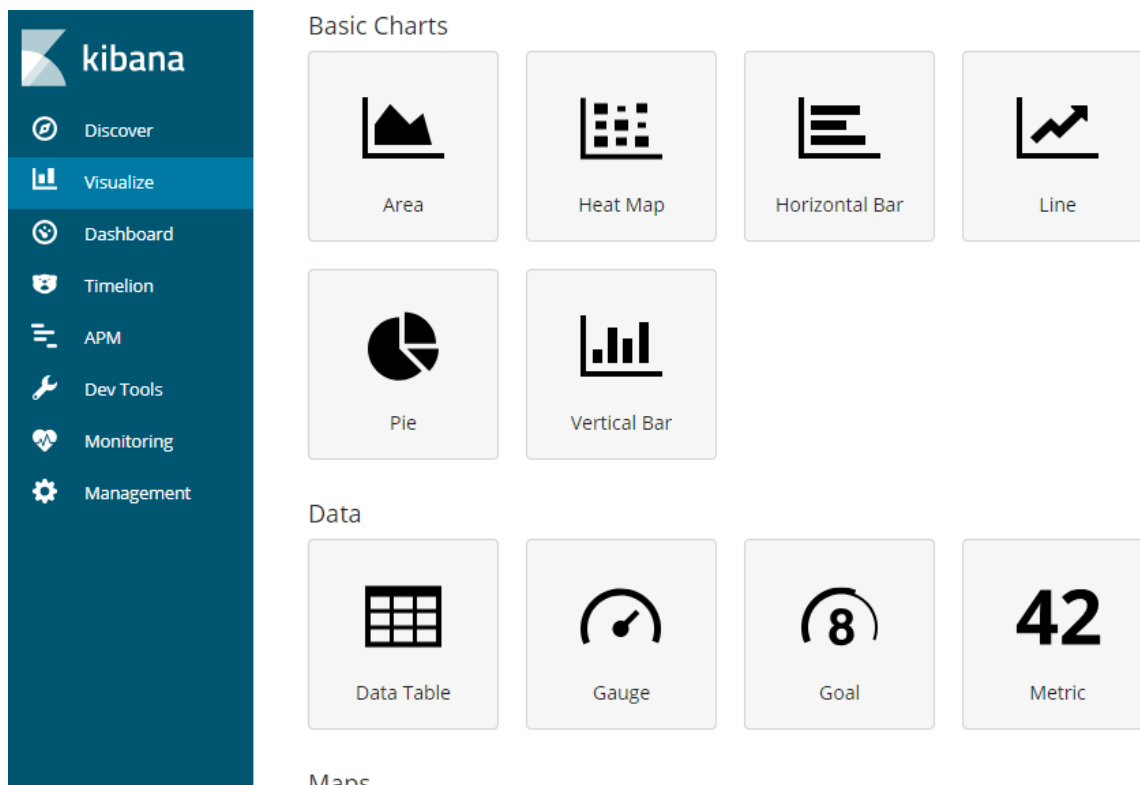


Figura 19. Tipus de visualitzacions

Després se selecciona la cerca guardada anteriorment:

Or, From a Saved Search

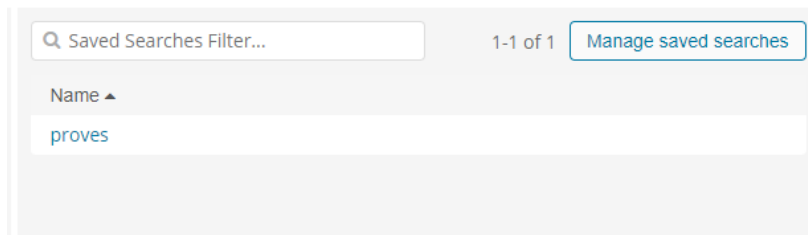


Figura 20. Selecció de cerques

I es configuren els paràmetres que convinguin per personalitzar la visualització:

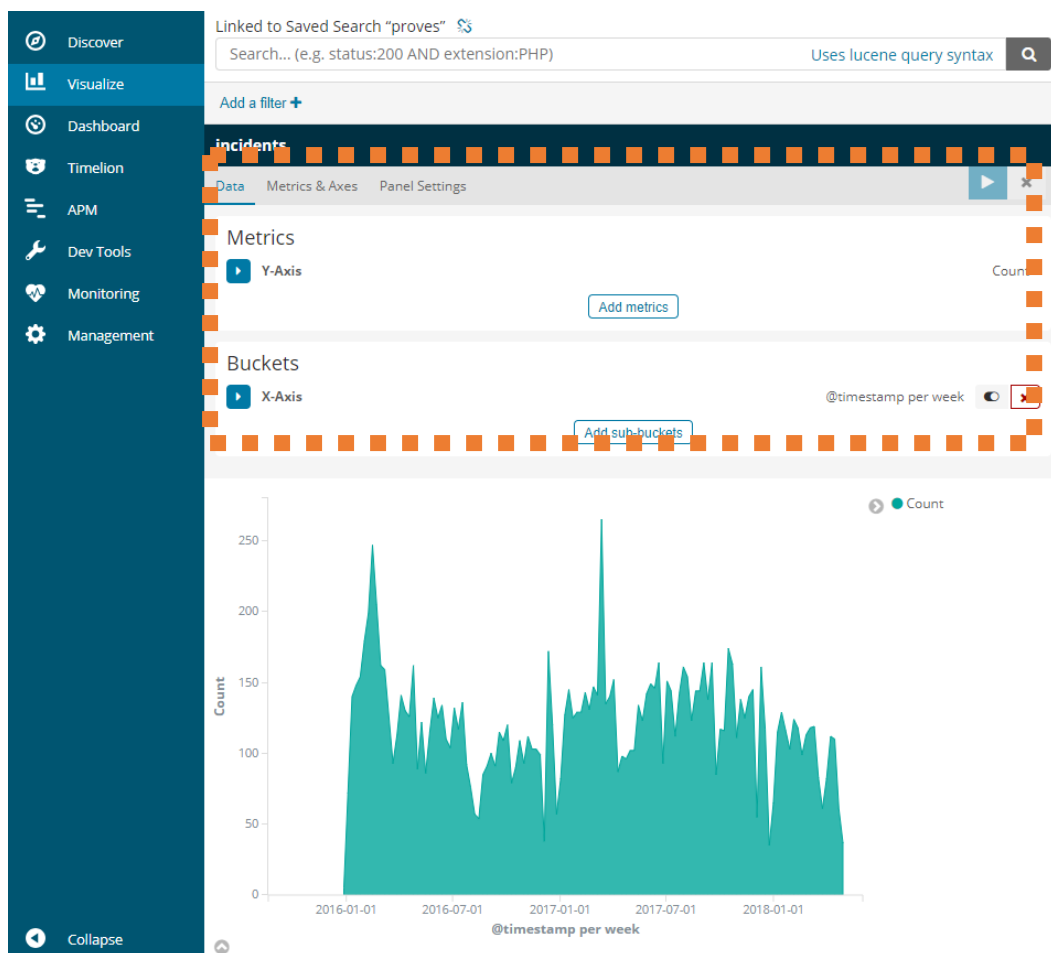


Figura 21. Personalització de visualitzacions

Algunes funcionalitats de Kibana estan restringides a certs tipus de camps. Per exemple, la visualització d'operacions matemàtiques com mitjanes, només poden ser aplicades a camps numèrics, mentre que el recompte de *strings* iguals només pot ser aplicat a camps del tipus *keyword*. Per això cal tenir present com volem visualitzar la informació quan definim els *templates*. Una vegada l'índex s'hagi creat, la modificació del *template* no tindrà cap efecte.

- **Dashboard:** Probablement és necessari més d'un tipus de visualització per extreure la informació desitjada. Per això, Kibana permet agrupar les visualitzacions guardades per mostrar-les en un panell *dashboard*.

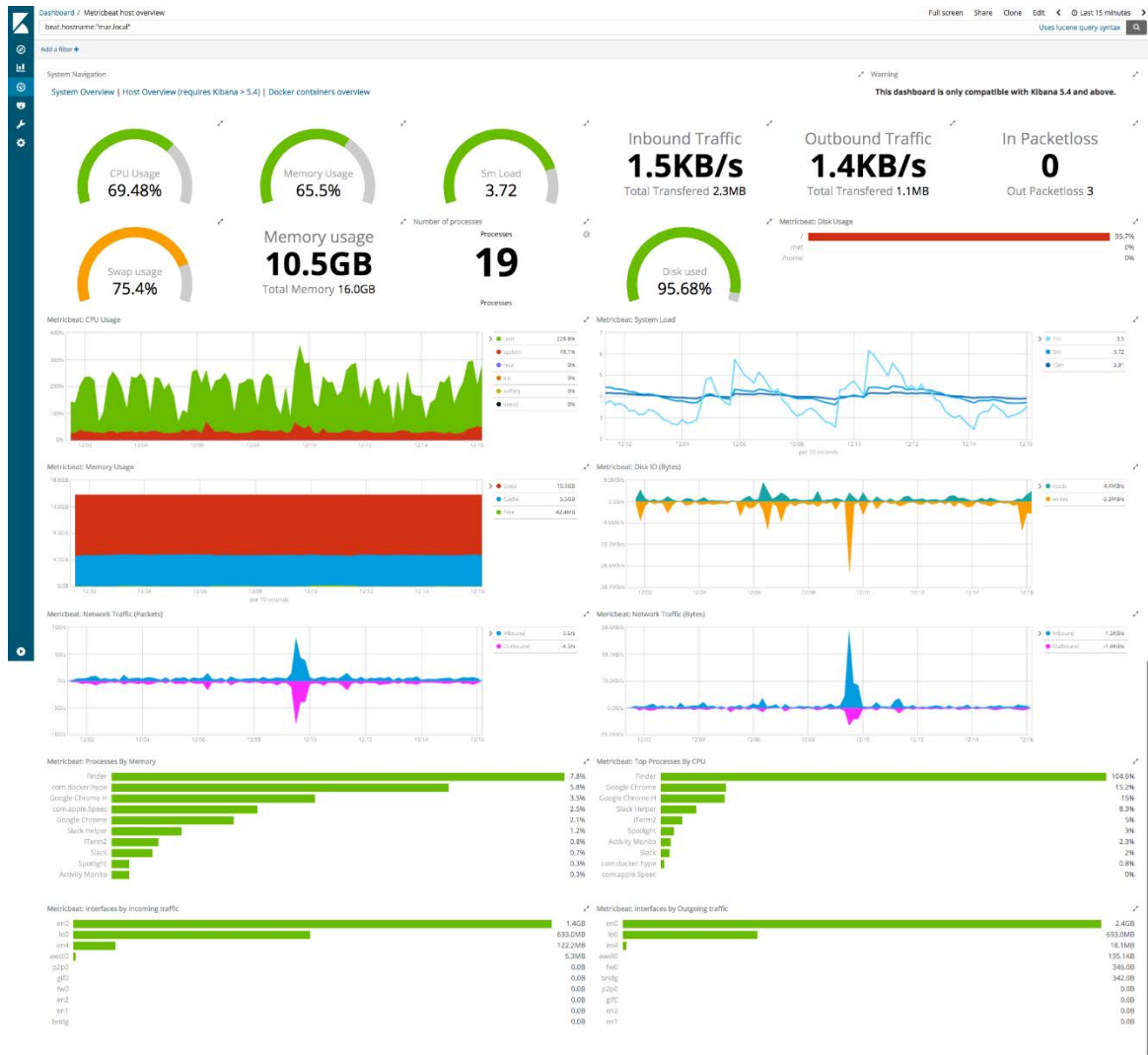


Figura 22. Exemple d'un *dashboard* [20]

4.3.2 Extracció d'Informació d'Elastic Stack

Amb Kibana també podem extreure informació sobre l'estat d'Elastic Stack, recollint dades de monitoratge d'Elasticsearch, Logstash i Kibana per detectar possibles errors o colls d'ampolla:

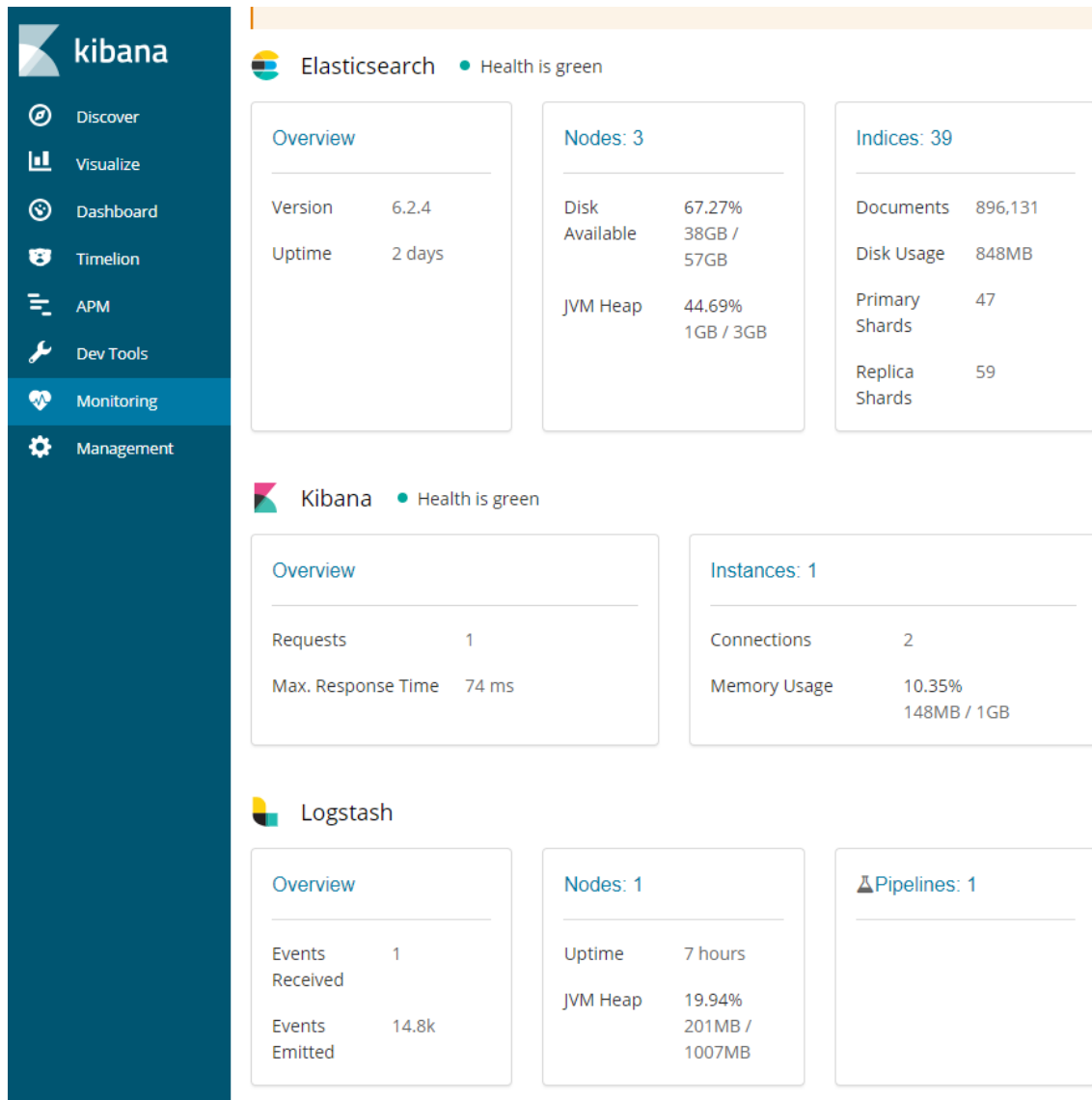


Figura 23. Panell principal de monitoratge

5. Desenvolupament

En aquest capítol es donen els detalls del desenvolupament d'aquest treball fi de grau. S'explica com s'han abordat els diferents objectius a partir dels requisits identificats en el capítol 2.

El capítol s'estructura en tres parts principals: primer s'explica com s'ha preparat l'entorn de desenvolupament, detallant la configuració de la infraestructura i d'Elastic Stack per formar un clúster. En segon lloc es detalla la implementació de la indexació automàtica dels esdeveniments de fitxers de logs a Elasticsearch amb ajuda de Logstash. En tercer lloc es presenten les tècniques utilitzades pel tractament i anàlisi dels registres d'incidències, basades en l'anàlisi de llenguatge natural i incloent un posterior classificador de documents.

5.1 Entorn de desenvolupament

Per demostrar les propietats d'Elasticsearch com a sistema distribuït cal un clúster de com a mínim tres nodes per motius de *split brain*.

Volem un entorn de desenvolupament que simuli de forma suficientment fidedigna a l'entorn de producció, que estaria format per un clúster físic amb diferents servidors Linux. Tanmateix, construir un clúster físic per fer proves requereix més feina i diners del necessari, ja que existeixen programes per simular entorns com ara VirtualBox o Docker.

Mentre que Elastic ofereix suport per Docker, el motiu pel qual utilitzem VirtualBox és perquè no només volem simular el clúster en l'àmbit d'aplicació sinó també detectar possibles problemes en la integració amb el sistema operatiu invitat per així aplicar solucions a l'entorn de producció.

5.1.1 VirtualBox

VirtualBox permet simular diferents màquines de forma aïllada en la màquina amfitrió.

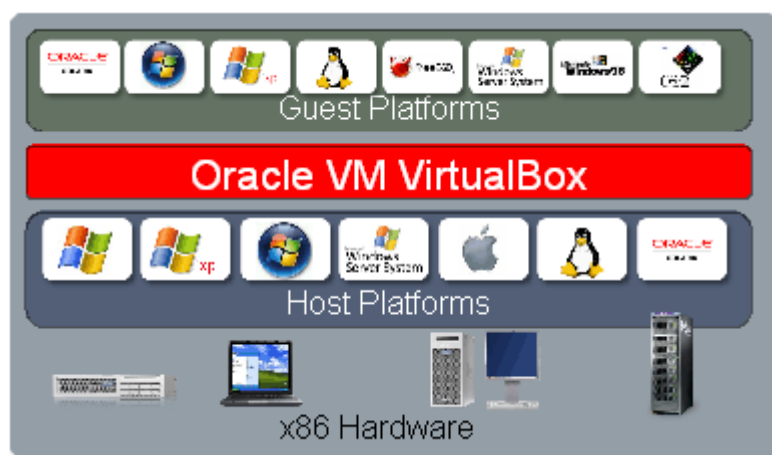


Figura 24. Arquitectura VirtualBox [20]

En aquest projecte el tipus i versió dels diferents components són els següents:

Guest Platforms:

- Linux 4.9.0-6-amd64 #1 SMP Debian 4.9.88-1+deb9u1 (2018-05-07) x86_64

Oracle VM VirtualBox:

- Version 5.2.12 r122591 (Qt5.6.2)

Host Platform:

- Windows 10 Pro 1803 17134.48 x64

Hardware CPU:

- AMD A10-5800K APU 3.80GHz

Hardware RAM:

- 8 GB DDR3

Amb VirtualBox configurem tres màquines virtuals Debian perquè a banda de ser el SO utilitzat en l'entorn de producció, és una de les distribucions més estables i utilitzades en servidors, i podem trobar configuració específica sobre Debian en la documentació d'Elastic. Com que l'ordinador de proves compta amb 8 GB de RAM a repartir entre el SO amfitrió i les màquines virtuals, realitzem una instal·lació mínima de la distribució Debian amb espai suficient de disc per la instal·lació d'Elastic Stack. També configurem un adaptador pont d'accés a xarxa per a cada node i així simular millor un entorn de producció real.

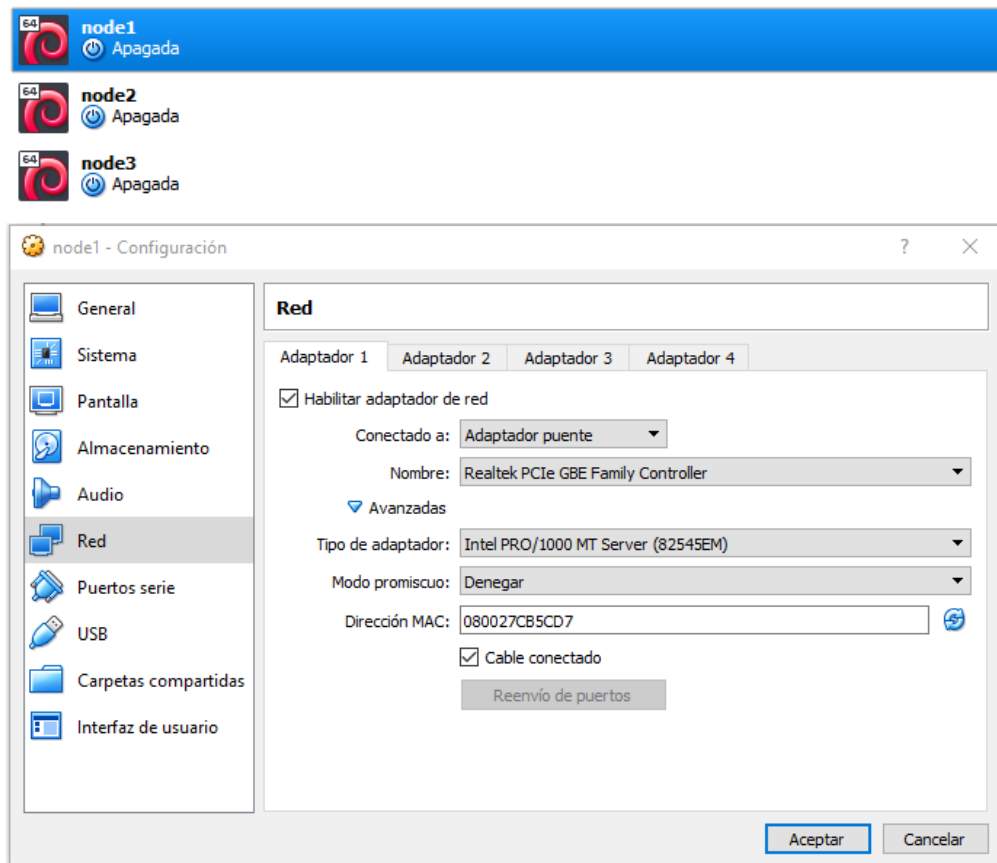


Figura 25. Configuració VirtualBox

Idealment cada node del clúster hauria de tenir una instància Elasticsearch aprofitant al màxim els seus recursos. Però la distribució d'Elastic Stack sobre el nostre clúster queda així:

node1 i node2:

- 1 instància Elasticsearch.
- 1024 MB de RAM reservats a VirtualBox.
- 2 Cores CPU reservats a VirtualBox.
- 20 GB d'espai d'emmagatzematge.

node3:

- 1 instància Elasticsearch.
- 1 instància Logstash.
- 1 instància Kibana.
- 2048 MB de RAM reservats a VirtualBox.
- 2 Cores CPU reservats a VirtualBox.
- 30 GB d'espai d'emmagatzematge.

Pels següents motius:

- En l'entorn de producció es preveu utilitzar Elastic Stack sobre Linux.
- Per cada màquina virtual s'han de reservar més recursos en la màquina amfitrió, així que crearem el mínim de màquines virtuals per formar el clúster de 3 nodes i fer una bona simulació.
- En cas que Logstash o Kibana tinguessin un error greu que provoqués el reinici del node, millor reduir aquesta possibilitat a un únic node.
- Oracle recomana que la RAM i la CPU d'una màquina virtual no superin la meitat del que ofereix la màquina amfitrió. És a dir, si sumem la RAM i CPU del clúster virtual, tenim 4096 MB i 6 cores, que en el cas de la RAM, és la meitat del que ofereix la màquina amfitrió. S'ha de tenir en compte que la màquina amfitrió també necessita capacitat de còmput pel SO i VirtualBox. De forma experimental, 6 cores en total pel clúster virtual suposen un rendiment òptim.

En comptes de muntar un sistema DNS i assignar noms a les màquines virtuals, apuntem la direcció MAC de cadascuna en el servidor DHCP del encaminador domèstic per assignar IPs estàtiques als tres nodes.

DHCP Estáticas

Lista de equipos -- selecciona uno -- Copiar

Identidad	Nombre	IP	Dirección MAC
1	Hector-PC	192.168.1.100	FC : AA : 14 : C0 : 6E : A2
2	node1	192.168.1.200	08 : 00 : 27 : CB : 5C : D7
3	node2	192.168.1.201	08 : 00 : 27 : 67 : D3 : 99
4	node3	192.168.1.202	08 : 00 : 27 : 3E : CC : 98

Figura 26. Configuració DHCP

Tal que l'esquema de xarxa queda així:

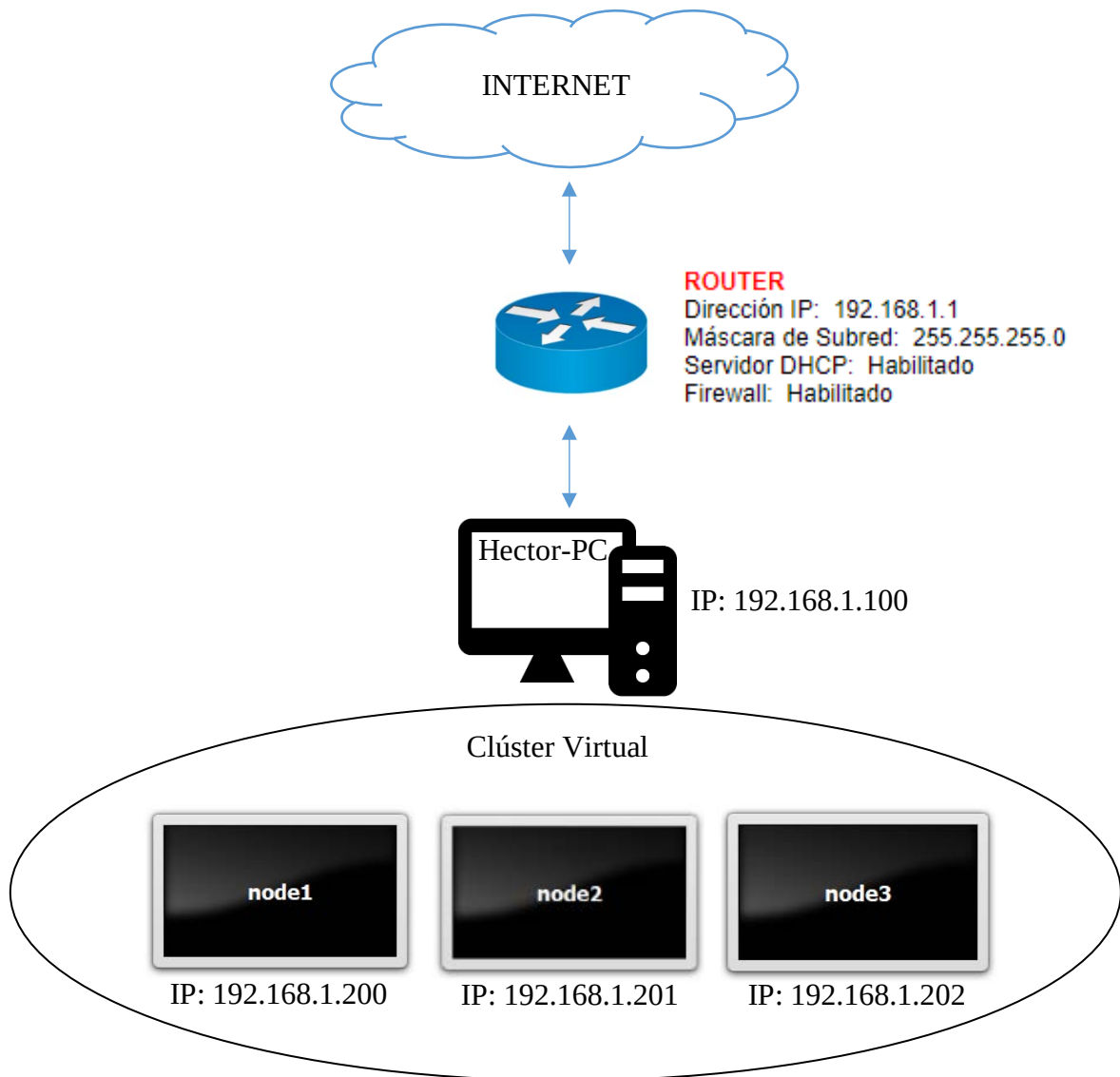


Figura 27. Configuració IP [22] [23]

Un cop muntat el clúster, amb les instruccions que es troben a <https://www.elastic.co/guide/en/elastic-stack/current/installing-elastic-stack.html> realitzem la instal·lació d'Elastic Stack 6.2.4.

5.1.2 Configuració d'Elasticsearch

En la carpeta `/etc/elasticsearch` es troben els següents arxius de configuració:

- **elasticsearch.yml**: Per configurar Elasticsearch.
- **jvm.options**: Per configurar les opcions JVM d'Elasticsearch.
- **log4j2.properties**: Per configurar el *logging* d'Elasticsearch.

El fitxer que s'ha de modificar és **elasticsearch.yml** (format YAML). També es poden modificar alguns paràmetres una vegada el node està actiu, a través de les APIs. Les modificacions que es fan en aquest apartat són suficients per a un entorn de desenvolupament. En un entorn de producció s'hauria de ser més estrictes amb els paràmetres que afecten la integració d'Elasticsearch amb el sistema operatiu per garantir més estabilitat i eficiència. A cada node les configuracions seran diferents per les IPs.

Paràmetres que no es deixen per defecte, **exemple node1**:

`node.name: ${HOSTNAME}`

Per defecte Elasticsearch utilitza els 7 primers caràcters d'un UUID aleatori que s'utilitza com la identificació del node. Node.name li dona un nom significatiu al node utilitzant la variable d'entorn HOSTNAME.

`network.host: 192.168.1.200`

Adreça IP en la que Elasticsearch escolerà per comunicar-se. Depèn del node.

`discovery.zen.ping.unicast.hosts: ["192.168.1.201", "192.168.1.202"]`

Adreces IPs d'un subconjunt de nodes del clúster que sempre estiguin actius per permetre que nous nodes puguin unir-se al clúster. En aquest cas posem les IPs dels node2 i node3.

`discovery.zen.minimum_master_nodes: 2`

Per evitar el fenomen *split brain*, configurem el mínim de nodes candidats a *master* que calen que estiguin disponibles en xarxa perquè Elasticsearch funcioni i formi un clúster. Per solucionar *split brain* només cal configurar la majoria de nodes amb:

$$(\text{master_eligible_nodes} / 2) + 1 \quad (4)$$

Com que tenim 3 nodes candidats a *master*, $\text{int}(3/2) + 1 = 2$

`xpack.security.enabled: false`

Ens assegurem que el mòdul X-Pack de seguretat està desactivat, simplificant la modificació del clúster, índexs o documents sense cap mena d'autenticació amb rols ni contrasenyes. Amb la llicència gratuïta d'1 any no es pot utilitzar el mòdul de seguretat, però per defecte Elasticsearch utilitza la llicència *trial* i activa el mòdul de seguretat si hem instal·lat X-Pack.

Amb un script executat des del node3 podem configurar i iniciar el clúster amb els fitxers de cada node (node1.yml, node2.yml, node3.yml) situats en la mateixa carpeta que el script.

```
#!/bin/bash
cp node3.yml /etc/elasticsearch/elasticsearch.yml
scp node1.yml root@192.168.1.200:/etc/elasticsearch/elasticsearch.yml
scp node2.yml root@192.168.1.201:/etc/elasticsearch/elasticsearch.yml
ssh root@192.168.1.200 'service elasticsearch restart'
ssh root@192.168.1.201 'service elasticsearch restart'
service elasticsearch restart
```

Codi 2. setup.sh

Abans però cal configurar /etc/ssh/sshd_config per poder connectar-nos com usuari *root* per via ssh, modificant o afegint la línia *PermitRootLogin yes*

5.1.3 Configuració de Logstash

En la carpeta */etc/logstash* es troben els següents fitxers de configuració:

- **logstash.yml**: Per configurar Logstash.
- **pipelines.yml**: Per configurar múltiples pipelines en una sola instància de Logstash.
- *jvm.options*: Per configurar les opcions JVM de Logstash.
- *log4j2.properties*: Per configurar el logging de Logstash.
- **conf.d**: Carpeta que conté les definicions de cada pipeline.

pipelines.yml i els fitxers de **conf.d** depenen de les dades que vulguem tractar. Però per configurar Logstash com a instància cal configurar **logstash.yml en el node3**:

Paràmetres que no es deixen per defecte:

```
config.reload.automatic: true
```

Útil per desenvolupar. Amb aquesta línia Logstash detecta canvis en els fitxers de configuració i les carrega automàticament.

```
xpack.monitoring.enabled: true
```

Activa el mòdul de monitoratge per després poder visualitzar l'estat de Logstash en Kibana.

```
xpack.monitoring.elasticsearch.url: ["http://192.168.1.200:9200",  
"http://192.168.1.201:9200", "http://192.168.1.202"]
```

Configura les IPs dels nodes amb Elasticsearch per enviar dades de monitoratge que després seran consultades per Kibana.

5.1.4 Configuració de Kibana

En la carpeta `/etc/kibana` modificarem el fitxer **kibana.yml en el node3**:

Paràmetres que no es deixen per defecte:

`server.host: "192.168.1.202"`

IP en la que Kibana escoltarà peticions HTTP dels usuaris que vulguin entrar i utilitzar Kibana.

`server.name: "kibanaccess.ddns.net"`

Nom de la instància de Kibana per motius d'estètica.

`elasticsearch.url: http://192.168.1.202:9200`

IP d'un node amb Elasticsearch a la que Kibana farà les peticions necessàries per executar les funcions que l'usuari utilitzi dins de Kibana.

`server.ssl.enabled: true`

Activa la capa SSL per utilitzar el protocol HTTPS quan els usuaris utilitzin Kibana.

`server.ssl.certificate: /etc/kibana/server.crt`

Camí del certificat SSL amb clau pública. S'explica perquè es configura en el següent apartat.

`server.ssl.key: /etc/kibana/server.key`

Camí de la clau privada que pertany al certificat anterior. S'explica perquè es configura en el següent apartat.

`xpack.security.enabled: false`

Aquest paràmetre ha d'estar igual que en *elasticsearch.yml* per poder comunicar-se correctament.

5.1.5 Seguretat

La seguretat és de vital importància quan Elasticsearch conté informació sensible sobre pacients i sistemes informàtics sanitaris. El nivell de seguretat que aporta Elastic Stack depèn de la llicència i si tenim instal·lat X-Pack:

	OPEN SOURCE	BASIC	GOLD	PLATINUM	ENTERPRISE
	Free Download	Free License	Request Info	Request Info	Request Info
X-PACK					
^ Security (formerly Shield)			✓	✓	✓
Encrypted communications			✓	✓	✓
Role-based access control			✓	✓	✓
Native, LDAP, Active Directory authentication			✓	✓	✓
SAML authentication				✓	✓
Audit logging			✓	✓	✓
Field- and document-level security				✓	✓
Third-party integrations (custom realms)				✓	✓
Encryption at rest support				✓	✓
^ Monitoring (formerly Maudit)		✓	✓	✓	✓

Figura 28. Seguretat X-Pack [24]

Punts clau de la seguretat en Elastic Stack:

- **Autenticitat:** Si tenim una llicència superior a la BASIC, amb l'accés per rols podem autenticar usuaris amb contrasenyes i agrupar-los en grups amb diferents permisos per controlar el que poden fer amb el clúster. Els usuaris poden ser persones o també poden ser usuaris especials, com els dedicats per Elasticsearch, Logstash i Kibana.

La llicència superior ens permet encriptar les comunicacions entre els nodes del clúster, i entre nodes i usuaris. És a dir, podem tenir nodes Elasticsearch repartits pel món comunicant-se per internet mentre que una persona s'instal·la Kibana al seu ordinador i es connecta al clúster per internet de forma segura. De fet aquest exemple es pot aplicar a Elastic Cloud.

- **Privacitat:** Amb l'accés per rols també es pot restringir el que els usuaris poden veure, com per exemple alguns documents o camps seleccionats.
- **Accessibilitat:** Amb la llicència superior també es poden utilitzar totes les funcions de monitoratge i alerta per mantenir el clúster en bon estat. La llicència BASIC també incorpora bona part del monitoratge per defecte. Elasticsearch activa automàticament el mòdul de monitoratge després d'instal·lar X-Pack.

	OPEN SOURCE	BASIC	GOLD	PLATINUM	ENTERPRISE
	Free Download	Free License	Request Info	Request Info	Request Info
X-PACK					
Security (formerly Shield)			✓	✓	✓
Monitoring (formerly Marvel)		✓	✓	✓	✓
Full stack monitoring		✓	✓	✓	✓
Multi stack monitoring support			✓	✓	✓
Configurable retention policy			✓	✓	✓
Automatic alerts on stack issues			✓	✓	✓
Management		✓	✓	✓	✓
Alerting (via Watcher)			✓	✓	✓
Highly available, scalable alerting			✓	✓	✓
Notifications via email, Slack, Pagerduty, Jira, or webhook			✓	✓	✓

Figura 29. Monitoratge X-Pack

Afortunadament no necessitem cap llicència de pagament per poder desenvolupar les tasques necessàries del projecte. Amb la llicència BASIC és suficient. Per aconseguir aquesta llicència ens registrem en <https://register.elastic.co/> i descarreguem la llicència en format JSON. Després des de la PowerShell de Windows de la màquina amfitrió podem executar el següent per activar la llicència descarregada:

```
gc .\license.json | Invoke-WebRequest -uri
http://192.168.1.200:9200/_xpack/license -Method Put -ContentType
"application/json"

gc .\license.json | Invoke-WebRequest -uri
http://192.168.1.200:9200/_xpack/license?acknowledge=true -Method
Put -ContentType "application/json"
```

Codi 3. Canvi de llicència [25]

De moment podem utilitzar Elastic Stack en la xarxa local. Ara bé, per permetre l'accés als directors del projecte mentre es desenvolupa, cal que Kibana sigui accessible des d'internet. Això comporta quatre problemes:

1. L'encaminador domèstic funciona per NAT i per tant no és possible que un client exterior arribi a establir una nova connexió amb la IP 192.168.1.202 local que és on escolta Kibana.
2. La IP pública de l'encaminador pot canviar, per tant és necessari DDNS.
3. Encriptar les comunicacions entre Kibana i el client que el vulgui utilitzar. La capa de seguretat SSL està disponible en Kibana sense necessitat d'incloure X-Pack i sense importar la llicència.

4. És una conseqüència d'utilitzar la llicència BASIC que no permet l'ús del mòdul de seguretat del X-Pack. És a dir, ni hi ha comunicació encriptada entre Kibana i el clúster, ni podem restringir l'accés per autenticació d'usuaris amb contrasenya. Com que Kibana està instal·lat a un dels nodes del clúster, realment el problema és que qualsevol pot accedir una vegada solucionat el primer problema.

5.1.6 Redirecció de Ports i DDNS

Com tenim previst establir una comunicació encriptada HTTPS entre client i Kibana, haurem de redirigir les connexions entrants noves TCP pel port 443 cap a la IP local del node3, ja que s'ha de permetre l'accés per l'URL `https://ip_pública_del_router`. Amb la redirecció de ports indiquem a l'encaminador que redirigeixi aquestes peticions cap al node3.

Mapeo de Puertos

Es posible configurar el router como un servidor que redirija todas las peticiones que se realicen desde Internet a servicios determinados (FTP, HTTP, etc.) de tu Red Local a ciertos equipos (direcciones IP locales) y puertos. Es decir, dependiendo de qué tipo de servicio se trate (qué puerto TCP/UDP), el router asignará ese servicio a un servidor concreto de la red local identificado por su dirección IP local y puerto para tratarlo. Es posible configurar rangos de puertos, una lista de ellos, o una combinación de ambos.

Por ejemplo:

Rango de puertos: ej. 100-150

Lista de puertos: ej. 25,110,80

Combinación: ej. 25-100,80

No.	LAN IP	Tipo de Protocolo	Puerto en la LAN	Puerto público	Activar	Configurar
1	192.168.1.202	TCP	443	443	✓	 

Figura 30. Redirecció de ports

Kibana escolta el port 5601 per defecte. Però ara suposarem que escolta el 443 així ja tenim feina feta pel següent apartat. En aquest punt hem resolt el primer problema. Ara cal enllaçar l'encaminador amb un nom DNS tipus A. Per això ens podem registrar a <https://www.noip.com/> i crear un *hostname* que en aquest cas anomenem `kibanaccess.ddns.net`

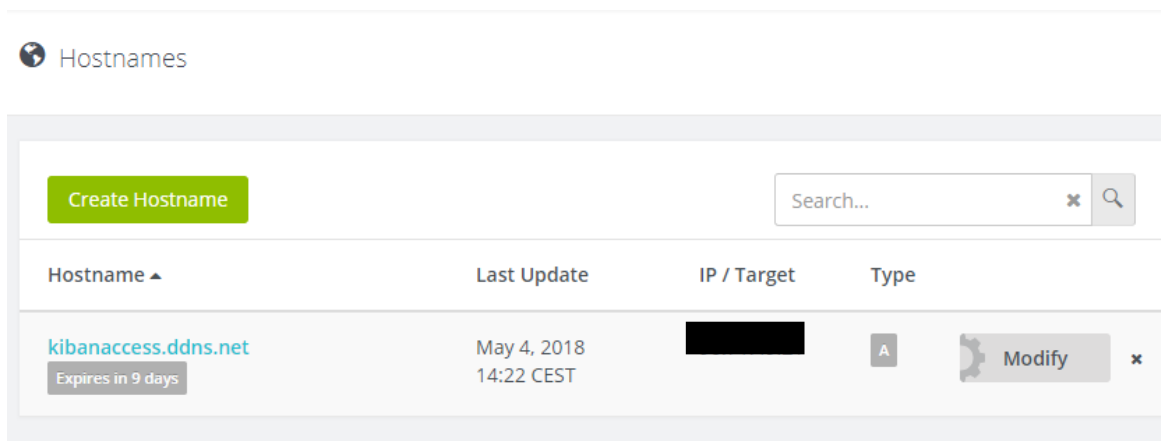


Figura 31. Creació de DNS

I enllacem el *hostname* amb l'encaminador utilitzant les credencials que vam utilitzar en el registre d'usuari a noip.com:

Configuración de un servidor DNS dinámico (DDNS)

Los servidores de direcciones DNS dinámicos son muy útiles cuando tu proveedor te asigna direcciones IP de forma dinámica, ya que permiten asociar un nombre permanente a tu router para que sea accesible desde Internet, sin necesidad de conocer en cada momento cual es la dirección IP asignada. El servidor DDNS actualizará tu dirección IP cada vez que te conectes a Internet y la asociará a ese nombre.

La función DDNS la proporcionan proveedores externos que generalmente necesitan que se haya creado previamente una cuenta con un nombre de usuario y contraseña

DNS dinámico	☒ Estado Actual: Registrado
Proveedor	No-IP ▼
Nombre de Dominio	kibanaccess.ddns.net
Cuenta o Dirección de e-mail	[Redacted]
Contraseña	

Figura 32. Configuració DDNS a l'encaminador

Ara també hem resolt el segon problema.

5.1.7 Reverse Proxy SSL

En un entorn de producció caldria mantenir un certificat SSL signat per una CA reconeguda per mantenir l'autenticitat del servidor Kibana a banda de la integritat i confidencialitat en la connexió per qualsevol usuari que es connecti a Kibana. Però en el nostre cas generarem un certificat autosignat que afegirem als certificats de confiança del nostre navegador. Perquè el certificat sigui vàlid en navegadors com Google Chrome, cal afegir l'extensió SAN.

Exemple de creació del certificat per al servidor Kibana al **node3**:

- Anem a la carpeta de configuració de Kibana:

```
cd /etc/kibana
```

- Creem una nova clau privada per utilitzar RSA:

```
openssl genrsa -out server.key 2048
```

- Creem el certificat corresponent a la clau privada i el *hostname* de Kibana:

```
openssl req\  
-key server.key\  
-x509\  
-nodes\  
-new\  
-out server.crt \  
-subj "/CN=kibanaccess.ddns.net"\  
-reqexts SAN\  
-extensions SAN\  
-config <(cat /usr/lib/ssl/openssl.cnf\  
    <(printf '[SAN]\nsubjectAltName=DNS:kibanaccess.ddns.net'))\  
-sha256\  
-days 3650
```

Codi 4. Generació del certificat SSL [26]

Ara Kibana trobarà la clau pública i privada en el camí que vam indicar en *kibana.yml*, pel que permetrà només el protocol HTTPS, no HTTP.

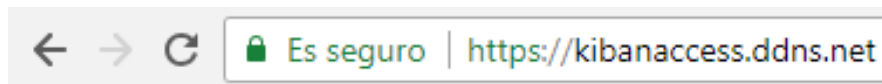


Figura 33. Demostració HTTPS

En aquest punt tenim el tercer problema resolt. Però ara qualsevol usuari que entri a <https://kibanaccess.ddns.net> tindrà accés a l'aplicació sense cap tipus d'autorització. Per evitar-ho, proposem la configuració d'un *reverse proxy*.

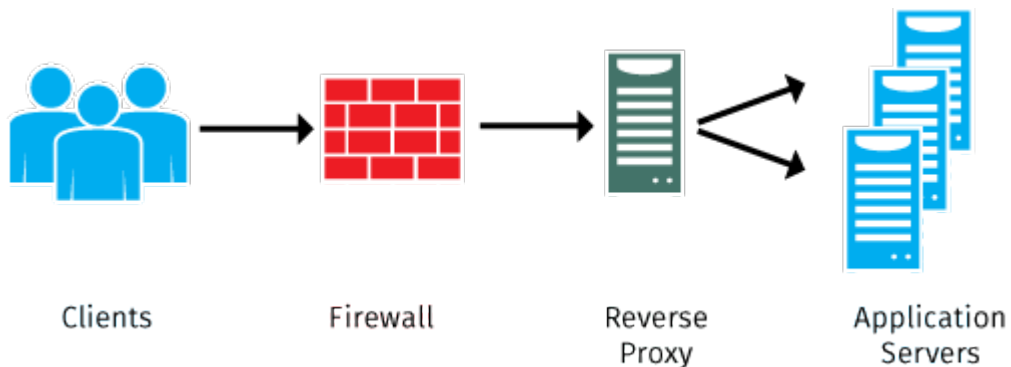


Figura 34. Esquema reverse proxy [27]

Clients	Directors del projecte accedint des d'internet
Firewall	L'encaminador domèstic incorpora un mòdul tallafocs que detecta intrusions i fa inspecció dinàmica de paquets
Reverse Proxy	Se situa entre l'encaminador i Kibana. Permetrà connectar-se a Kibana si els usuaris s'autentiquen correctament amb usuari i contrasenya
Applications Servers	Nodes del clúster on es troba Kibana. En el nostre cas només hi ha 1

Taula 2. Descripció dels agents relacionats amb reverse proxy

Per configurar el *reverse proxy* utilitzarem el servidor web Apache amb els mòduls `proxy_module` i `ssl_module` engegats [28]. Amb la comanda `htpasswd` creem l'usuari "elastic" i guardem el fitxer resultant `htpass` a `/etc/kibana`. Ara cal configurar Apache perquè només deixi accedir a l'usuari "elastic" a Kibana.

Per això editem el fitxer `/etc/apache2/sites-available/000-default.conf`:

```
<VirtualHost *:443>
    ServerAdmin hector.mercado@urv.cat
    ServerName kibanaccess.ddns.net
    SSLEngine on
    SSLCertificateFile /etc/kibana/server.crt
    SSLCertificateKeyFile /etc/kibana/server.key
    <Location />
        AuthType Basic
        AuthName "Restricted Files"
        # (Following line optional)
        AuthBasicProvider file
        AuthUserFile "/etc/kibana/htpass"
        Require user elastic
    </Location>
    SSLProxyEngine on
    SSLProxyVerify none
    SSLProxyCheckPeerCN off
    SSLProxyCheckPeerName off
    SSLProxyCheckPeerExpire off
    ProxyPreserveHost On
    ProxyRequests Off

    ProxyPass / https://192.168.1.202:5601/
    ProxyPassReverse / https://192.168.1.202:5601/

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Codi 5. 000-default.conf

Amb aquesta configuració Apache l'únic que fa és redirigir el tràfic entre Kibana i el director en cas que aquest s'autentiqui correctament. El moment d'introduir les credencials de l'usuari "elastic" també està encriptat per HTTPS amb les mateixes claus que vam generar per Kibana.

En aquest punt deixem de suposar que Kibana escolta pel port 443 per entendre el resultat final:

1. El director del projecte escriu <https://kibanaccess.ddns.net> en el seu navegador.
2. L'encaminador redirigeix la petició al port TCP 443 del node3.
3. Apache, que escolta pel port 443 com a servei, exigeix autenticació al director.
4. El director introdueix les credencials de l'usuari "elastic" i inicia sessió.

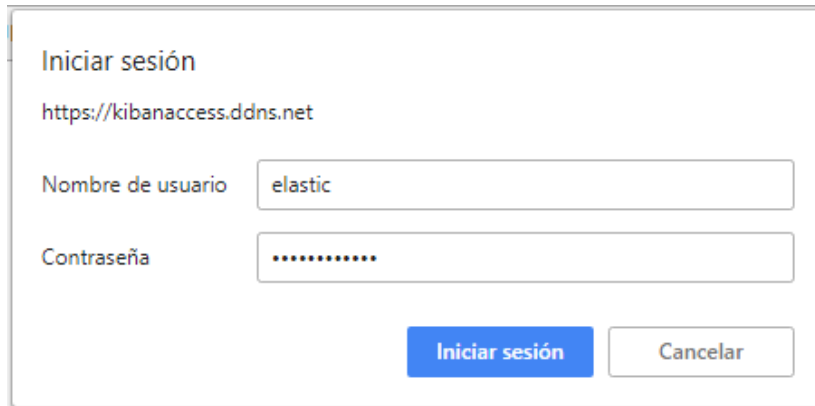
A login form titled "Iniciar sesión" (Log in). It shows the URL "https://kibanaccess.ddns.net". There are two input fields: "Nombre de usuario" (Username) with the value "elastic" and "Contraseña" (Password) with masked characters. At the bottom, there are two buttons: "Iniciar sesión" (Log in) in blue and "Cancelar" (Cancel) in white.

Figura 35. Autenticació del director

5. Apache crea una sessió i redirigeix la comunicació HTTPS del director a la IP i port on escolta Kibana 192.168.1.202:5601 i viceversa.

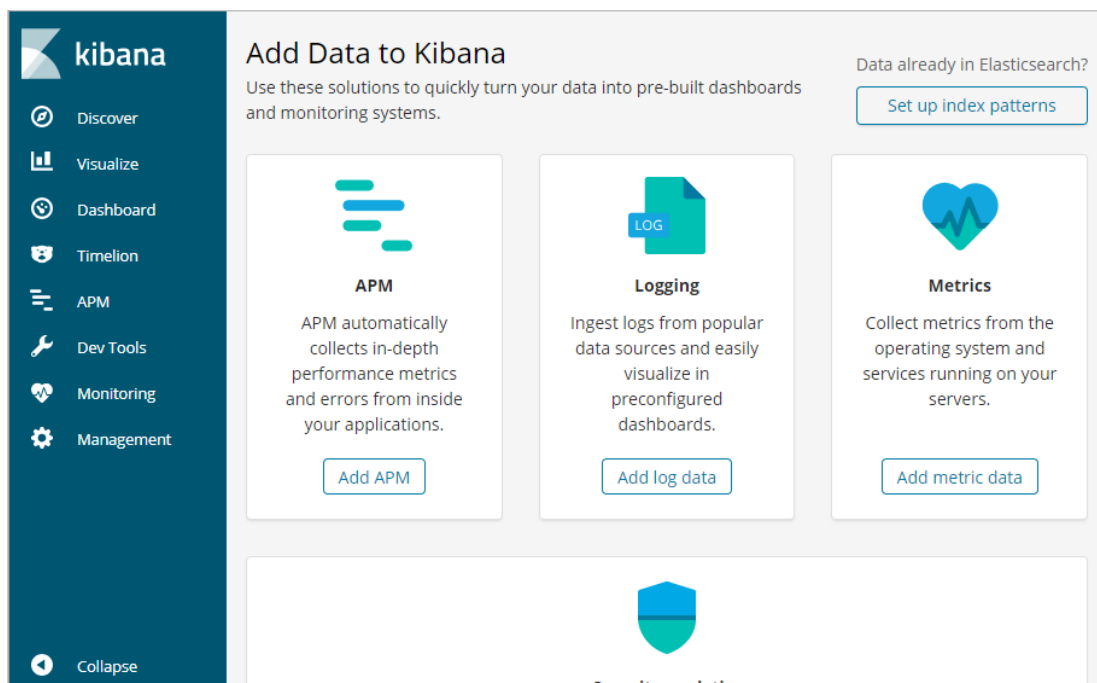


Figura 36. Pàgina principal de Kibana

Ara tenim l'entorn de desenvolupament a punt per començar a afegir dades a Elasticsearch mitjançant Logstash per després extreure informació amb Kibana.

5.2 Tractament de Fitxers de Log

L'objectiu d'aquesta secció consisteix a indexar les dades d'uns logs de QlikView a Elasticsearch de forma automàtica a mesura que QlikView afegeixi noves línies als logs.

Hi ha tres fitxers que ens han demanat emmagatzemar a la base de dades per permetre el seu posterior anàlisi:

Events.log: Similar al log d'esdeveniments de Windows. Enregistra esdeveniments del servidor tipificats com informació, *warnings* o errors.

Performance.log: En una planificació definida, per exemple cada X minuts, s'enregistren estadístiques de rendiment del servidor QlikView, incloent-hi informació de la memòria, CPU utilitzada, usuaris actius, sessions actives...

Sessions.log: Aquest log enregistra informació útil sobre les sessions dels usuaris, com el temps de connexió, tipus de client, documents consultats...

La primera línia d'aquests fitxers descriu els camps que té cada esdeveniment de log. No es tindrà després en compte en el processament del *pipeline* de Logstash perquè no cal que hi estigui als fitxers

Com que les tècniques en el tractament de cada fitxer de log són similars, només posarem d'exemple en aquesta memòria el tractament d'**Events.log**, explicant el seu *template* i *pipeline*, i deixarem de banda *Performance.log* i *Sessions.log*, que es farien de forma anàloga.

L'algoritme bàsic per indexar un esdeveniment d'aquests fitxers:

1. Es llegeix una línia del fitxer
2. Es parteix la línia segons els tabuladors i s'associa cada element segons els camps predefinits en un *template* d'ElasticSearch.
3. Es modifica el *@timestamp* perquè la data del document correspongui a la data de creació del log.

Un cop indexats els fitxers a Elasticsearch, s'han implementat un conjunt de *reports* definits pel gestor de l'hospital. Aquests reports li permetran analitzar de forma ràpida i senzilla el contingut dels logs per extreure informació útil per portar a terme possibles decisions de millora del sistema.

Server Started	Timestamp	SeverityID	EventID	Severity	Message
2015-06-03 09:07:28	2015-06-17 11:24:24	1	300	Error	Shared: Can not open file.
2016-01-22 09:33:48	2016-02-03 15:24:38	1	300	Error	SE_LOG: Server - ScanFile: Error
) occurred during open of C:\QLIKVIEW DEVELOPMENT\DEVELOPINGAPPS\AMBULATORIES\AMBULATORIES_19012016.QVW.					
2017-07-12 03:25:48	2017-08-01 17:04:55	4	100	Notice	License: License leased to user GRUPSGS\19989

Figura 37. Events.log 2,696 KB

Exe Type	Exe Version	Server Started	Timestamp	EntryType	ActiveDocSessions	DocSessions
RLS64	11.20.12852.0409.10	2015-05-13 10:10:46	2015-05-13 10:10:48	Server starting	0	0

ActiveAnonymousDocSessions	AnonymousDocSessions	ActiveTunneledDocSessions	TunneledDocSessions
0	0	0	0

DocSessionStarts	ActiveDocs	RefDocs	LoadedDocs	DocLoads	DocLoadFails	Calls	Selections	ActiveIpAdrs
0	0	0	0	0	0	0	0	0

IpAdrs	ActiveUsers	Users	CPUload	VMCommitted(MB)	VMAllocated(MB)	VMFree(MB)	VMLargestFreeBlock(MB)
0	0	0	87	201	15141	15141	

UsageCalBalance⁴
0.00

Figura 38. Performance.log 40,335 KB

³ Començament de la línia

⁴ Final de la línia

```

5Exe Type   Exe Version Server Started   Timestamp   Document
RLS64 11.20.12852.0409.10   2015-08-20 16:09:32   2015-09-15 13:27:01   C:\QLIKVIEW\PROJECT\GENERAL\QIA.qvw

Document Timestamp      QlikView User      Exit Reason      Session Start      Session Duration CPU spent (s)
2015-09-14 01:00:30      Session expired after idle time      2015-09-15 12:55:01      00:32:00      0.000

Bytes Received   Bytes Sent   Calls Selections   Authenticated user      Identifying user
49282 715595      169 5      GRUPSGS\rmtomas      GRUPSGS\rmtomas

Client machine identification Serial number      Client Type Client Build
7c3d8e3f-925c-4317-96cb-6f9c3c476f8d      Ajax QvWS 11.20.12852.0 browser.msie 7.0

Version      Secure Protocol   Tunnel Protocol   Server Port Client Address      Client Port Cal Type      Cal Usage
On      4747   169.254.28.112      56819 Named User

Count Session6
5547

```

Figura 39. Sessions.log 3,456 KB

Tal com es pot veure, són fitxers organitzats en files i columnes. És a dir, cada fila està partida en tabuladors ‘t’ que separen les cel·les de cada columna. Per exemple a *Sessions.log* tenim que la primera fila és Exe Type/tExe Version/tServer Started/tTimestamp/t... Si volem consultar el Timestamp de qualsevol sessió⁷, només cal saltar 3 tabuladors per arribar a la cel·la corresponent.

⁵ Començament de la línia

⁶ Final de la línia

⁷ Sinònim de fila, línia o esdeveniment

5.2.1 Definició dels Templates

Abans de crear i executar els *pipelines* hem de crear un *template* en Elasticsearch, per a cada fitxer, dedicat a definir l'índex que Logstash crearà (si no existeix) en la fase d'output:

```
{
  "template": "events_qlikviewsrv",
  "version": 50001,
  "settings": {
    "index.refresh_interval": "5s",
    "number_of_shards": "3",
    "number_of_replicas": "2"
  },
  "mappings": {
    "doc": {
      "properties": {
        "@timestamp": {
          "type": "date"
        },
        "@version": {
          "type": "keyword"
        },
        "server_started": {
          "type": "date",
          "format": "yyyy-MM-dd HH:mm:ss"
        },
        "eventID": {
          "type": "keyword"
        },
        "message": {
          "type": "text",
          "fields": {
            "keyword": {
              "type": "keyword"
            }
          }
        },
        "severity": {
          "type": "keyword"
        },
        "severityID": {
          "type": "keyword"
        }
      }
    }
  }
}
```

Codi 6. events_template.json

On "template": "events_qlikviewsrv" fa referència a l'índex que s'ha d'aplicar el *template* quan aquest índex es creï. Després des d'una consola en Linux podem carregar el *template* en Elasticsearch amb:

```
curl -H 'Content-Type: application/json' -XPUT
http://192.168.1.200:9200/_template/events_template.json?pretty -d
@events_template.json
```

Codi 7. update_templates.sh

Si apliquem la resta de *templates*, ens quedarà el següent esquema:

events_template.json defineix l'índex **events_qlikviewsrv** pel fitxer *Events.log*

performance_template.json defineix l'índex **performance_qlikviewsrv** pel fitxer *Performance.log*

sessions_template.json defineix l'índex **sessions_qlikviewsrv** pel fitxer *Sessions.log*

Amb el paràmetre "number_of_shards": "3" podem donar la possibilitat a Elasticsearch de distribuir els 3 *shards* primaris als 3 nodes i així permetre escriptures paral·leles de documents.

Amb el paràmetre "number_of_replicas": "2" permetem que Elasticsearch continuï operatiu amb normalitat si un dels tres nodes cau, i no perdrem cap dada si dos dels tres nodes cauen, gràcies a la redundància de *shards*.

5.2.2 Definició dels Pipelines

La majoria de línies en els fitxers de log corresponen a un esdeveniment diferent. Però pot passar, com en *Events.log*, que hi hagi dues línies consecutives que pertanyin al mateix esdeveniment. Això suposa un problema perquè Logstash tracta automàticament cada línia separada per '\n' com un esdeveniment diferent. Aquest problema el podem solucionar en el cas de *Events.log* amb:

```
codec => multiline {
  pattern => "^\n"
  what => "previous"
}
```

Indicant a Logstash que cada línia que comenci pel caràcter '\n', pertany a la línia anterior.

Els fitxers de log es troben en */home/hector* i serviran d'input per als *pipelines* de Logstash. Cada fitxer s'ha de tractar de forma diferent perquè té diferents camps per diferents tipus de valors, per això cal un *pipeline* per cadascun dels fitxers. Com que es tracta d'entitats diferents, mètriques de diferents tipus d'esdeveniments, dedicarem un índex diferent en Elasticsearch per l'output de cada pipeline.

Si tinguéssim només un *pipeline*, el podríem configurar⁸ a *logstash.yml*, però com que en tenim múltiples, els hem de configurar al fitxer **pipelines.yml**:

- pipeline.id: events
path.config: "/etc/logstash/conf.d/events.conf"
- pipeline.id: performance
path.config: "/etc/logstash/conf.d/performance.conf"
- pipeline.id: sessions
path.config: "/etc/logstash/conf.d/sessions.conf"
- pipeline.id: incidents
path.config: "/etc/logstash/conf.d/incidents.conf"

⁸ "Configurar" un *pipeline* significa configurar-ho a externament, fent ús de característiques com *persistent queues*, múltiples *threads*... Mentre que "definir" un *pipeline* consisteix a configurar les 3 fases.

Els fitxers de definició dels *pipelines* sempre estaran a **conf.d** i només seran executats per Logstash si els hem inclòs en **pipelines.yml** o **logstash.yml**.

Exemple de definició del *pipeline events*:

```
input {
  file {
    path => "/home/hector/Events.log"
    ignore_older => 0
    start_position => "beginning"
    sincedb_path => "/dev/null"
    codec => multiline {
      pattern => "^\)"
      what => "previous"
    }
  }
}
filter {
  dissect {
    mapping => {
      "message" => "%{server_started}    %{timestamp}"
      "%{severityID}    %{eventID}    %{severity}    %{message}"
    }
  }
  date {
    match => ["timestamp", "yyyy-MM-dd HH:mm:ss"]
    remove_field => ["timestamp"]
  }
}
output {
  elasticsearch {
    hosts => ["192.168.1.200", "192.168.1.201", "192.168.1.202"]
    index => "events_qlikviewsrv"
    manage_template => false
  }
  stdout {
    codec => rubydebug
  }
}
Codi 8. events.conf
```

Tal com vam descriure, un *pipeline* està format per tres fases, input, filter i output. La fase de l'input trenca les dades que llegeix en *events*⁹, que després modificarem en l'etapa de filter per després indexar-los com documents en Elasticsearch en l'etapa d'output. En cada etapa es poden utilitzar uns plugins predefinits, per exemple, en la d'input utilitzem el plugin *file* per llegir *Events.log*:

```
path => "/home/hector/Events.log"
```

⁹ Aclariment: “esdeveniments” són els esdeveniments que interpretem nosaltres al fitxer d'input del *pipeline*. “events” són els esdeveniments que interpreta Logstash i resideixen a memòria com un objecte JSON mentre l'*event* sigui processat. “documents” són els *events* que interpreta Elasticsearch després de les fases filter i output de Logstash.

Logstash manté constància internament de quins fitxers ha començat a processar i quantes línies ha processat d'aquests fitxers. Quan desenvolupem, probablement no encertarem a la primera, hi haurà errors en el *pipeline* o mapejarem incorrectament els documents d'Elasticsearch.

```
ignore_older => 0
start_position => "beginning"
sincedb_path => "/dev/null"
```

Amb aquestes línies aconseguirem que cada vegada que reiniciem el servei Logstash, es tornin a processar els fitxers especificats en el *pipeline* des del principi del fitxer. Per tant si hi ha errors amb Elasticsearch, només cal esborrar l'índex que rep l'output d'aquest *pipeline* i reiniciar el servei Logstash.

5.2.3 Execució del Pipeline

A partir d'aquí farem una simulació de com un esdeveniment d'*Events.log*, és processat en Logstash:

1. Logstash s'inicia com a servei:

Amb la distribució de Linux instal·lada al node3 on es troba Logstash, podem fer:

```
systemctl restart logstash
```

o

```
service logstash restart
```

I per veure l'output que Logstash produeix com a servei i així tenir constància visual del processament, obrim dues noves terminals i executem una de les dues comandes en cadascuna:

```
journalctl -f -u logstash
```

```
tail -f /var/log/logstash/logstash-plain.log
```

Aquestes terminals ens donaran informació en temps real. En el meu cas, utilitzo terminals ssh Putty perquè em semblen més còmodes per treballar i visualitzar el procés.

Un dels problemes més freqüents pels quals els serveis Elasticsearch, Logstash o Kibana no s'inicien correctament, sense mostrar cap tipus d'error a vegades, és per la mala configuració dels permisos d'alguns fitxers. Com a serveis, ens hem d'assegurar que els fitxers que llegeixin com ara certificats SSL o configuració de *pipelines*, tinguin permís de lectura per als usuaris "elasticsearch", "logstash" o "kibana". Aquests usuaris s'afegeixen automàticament en Linux a mesura que instal·lem l'Elastic Stack.

2. Logstash començarà a llegir la primera línia d'Events.log, l'anomenarem línia A. Mostrem metacaracters `'/t'` `'/n'` a mode didàctic¹⁰:

```
2015-06-03 09:07:28/t2015-06-17 11:24:24/t1/t300/tError/tShared: Can not
open file./n
```

La línia que hi ha després de la A no comença per `(`, així que la línia A es transforma en *event* per tractar-la en la fase de filtre. L'*event* en aquest cas no és res més que una cadena en memòria RAM esperant a ser processada per la fase filter i output.

3. En la fase filter primer s'aplica el plugin *dissect*.

```
mapping => {
  "message" => "%{server_started}      %{timestamp}      %{severityID}
               %{eventID}  %{severity}  %{message}"
}
```

Mapejem els camps de cada esdeveniment amb els camps definits en el *template* anterior. Per exemple, el contingut que Logstash es trobi abans del primer `'/t'` ho assignarà al camp *top-level* "server_started" de l'objecte JSON (*event*). El document Elasticsearch que resulti d'indexar aquest *event* tindrà la mateixa estructura, claus i contingut que aquest *event* que Logstash li envia a l'etapa d'output.

Events.log conté un camp "timestamp". Els fitxers processats per Logstash per defecte reben un camp de metainformació @timestamp que serveix d'utilitat a Kibana després a l'hora de visualitzar dades en un període de temps determinat. Per defecte, Logstash assigna a @timestamp el moment que l'esdeveniment es processa. Però el @timestamp en aquest cas hauria de descriure el moment que l'esdeveniment va ser creat en el fitxer de log en comptes de quan l'*event* a Logstash va ser processat. Les següents línies solucionen això:

```
date {
  match => ["timestamp", "yyyy-MM-dd HH:mm:ss"]
  remove_field => ["timestamp"]
}
```

Quan ja tenim l'*event* estructurat amb les claus i valors adients, volem enviar l'*event* a algun destinatari, en el nostre cas Elasticsearch i stdout. Aquest procés té lloc en la fase d'output:

```
output {
  elasticsearch {
    hosts => ["192.168.1.200", "192.168.1.201", "192.168.1.202"]
    index => "events_qlikviewsrv"
    manage_template => false
  }
}
```

¹⁰ Aquesta línia del fitxer és massa llarga per encaixar com a línia en aquesta memòria, per això indiquem el `'\n'`

```

    stdout {
      codec => rubydebug
    }
  }
}

```

Amb el plugin elasticsearch aconseguim indexar els *events* balancejant la càrrega entre els nodes identificats per la seva IP:

```
hosts => ["192.168.1.200", "192.168.1.201", "192.168.1.202"]
```

Per defecte Logstash intenta aplicar un *template* dinàmic que s'ajusti als *events*. Però si volem controlar la configuració dels índexs/documents en Elasticsearch, cal que apliquem *templates* personalitzats, desactivant el dinàmic de Logstash:

```
manage_template => false
```

Per veure l'output dels *events* de Logstash com a servei en temps real, utilitzem la comanda *journalctl*:

```

stdout {
  codec => rubydebug
}

```

5.2.4 Construcció de Visualitzacions

Una vegada tenim documents indexats en Elasticsearch, podem analitzar-los com ens sigui de gust per extreure la informació que convingui amb ajuda de Kibana. A l'apartat 6.2 presentem les visualitzacions realitzades amb Kibana a partir dels fitxers de log.

Com que el procediment per crear aquestes visualitzacions és similar, mostrarem la construcció només d'una visualització a tall d'exemple. En concret demostrarem la construcció de la següent visualització:

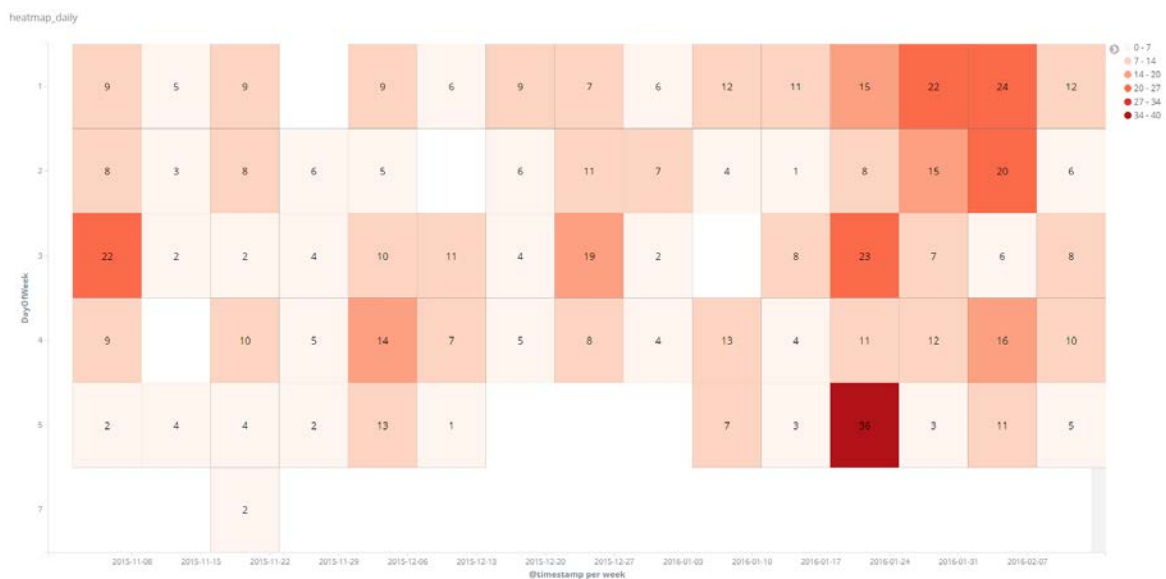


Figura 40. Heatmap de les sessions per dia de la setmana

Aquesta visualització és del tipus *heatmap* i pretén il·lustrar quantes sessions es produeixen segons el dia de la setmana (dilluns, dimarts, dimecres...) setmanalment en una franja de temps seleccionada.

Les dades que necessitem per crear el *heatmap* són totes les sessions produïdes en un període de temps determinat, per així fer un recompte de sessions per dia de la setmana. Per obtenir totes les sessions necessitem configurar l'*index pattern* per seleccionar l'índex on es troben les sessions.

En aquest projecte s'han definit quatre índexs en Elasticsearch:

- events_qlikviewsrv
- sessions_qlikviewsrv
- performance_qlikviewsrv
- incidents

Així que en l'*index pattern* cal referenciar només l'índex de les sessions:

Create index pattern

Kibana uses index patterns to retrieve data from Elasticsearch indices for things like visualizations.

☐ Include system indices

Step 1 of 2: Define index pattern

Index pattern

sessions_qlikviewsrv

You can use a * as a wildcard in your index pattern.
You can't use empty spaces or the characters \, /, ?, ", <, >, |.

> Next step

✓ **Success!** Your index pattern matches **1 index**.

sessions_qlikviewsrv

Figura 41. Creació *index pattern*

Es pot observar que si volguéssim seleccionar tots els documents de logs, només caldria definir un patró amb *wildcard* com “*_qlikviewsrv”.

A continuació seleccionem el camp @timestamp generat per Logstash per així filtrar els documents durant un cert període de temps.

Step 2 of 2: Configure settings

You've defined **sessions_qlikviewsrv** as your index pattern. Now you can specify some settings before we create it.

Time Filter field name Refresh

@timestamp ▼

The Time Filter will use this field to filter your data by time.

You can choose not to have a time field, but you will not be able to narrow down your data by a time range.

> Show advanced options

[< Back](#) [Create index pattern](#)

Figura 42. Selecció de filtre temporal

Creem l'*index pattern* i anem a la pestanya *Discover*. Seleccionem l'*index pattern* creat. Pot ser que ens aparegui el següent missatge:

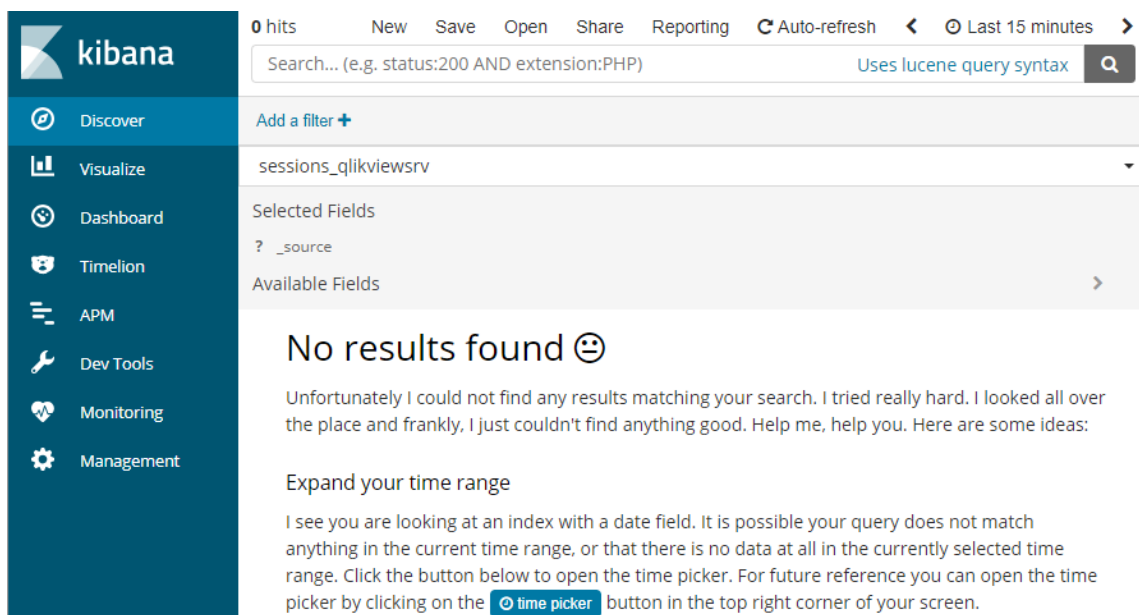


Figura 43. Discover de les sessions amb filtre erroni

Si hem indexat els documents a `sessions_qlikviewsrv`, les sessions poden no aparèixer o bé perquè s'han indexat recentment i Kibana encara no ha actualitzat la vista (en tal cas s'ha d'actualitzar la plana) o bé perquè en la franja de temps escollit no existeix cap document (en tal cas s'ha de modificar la franja).

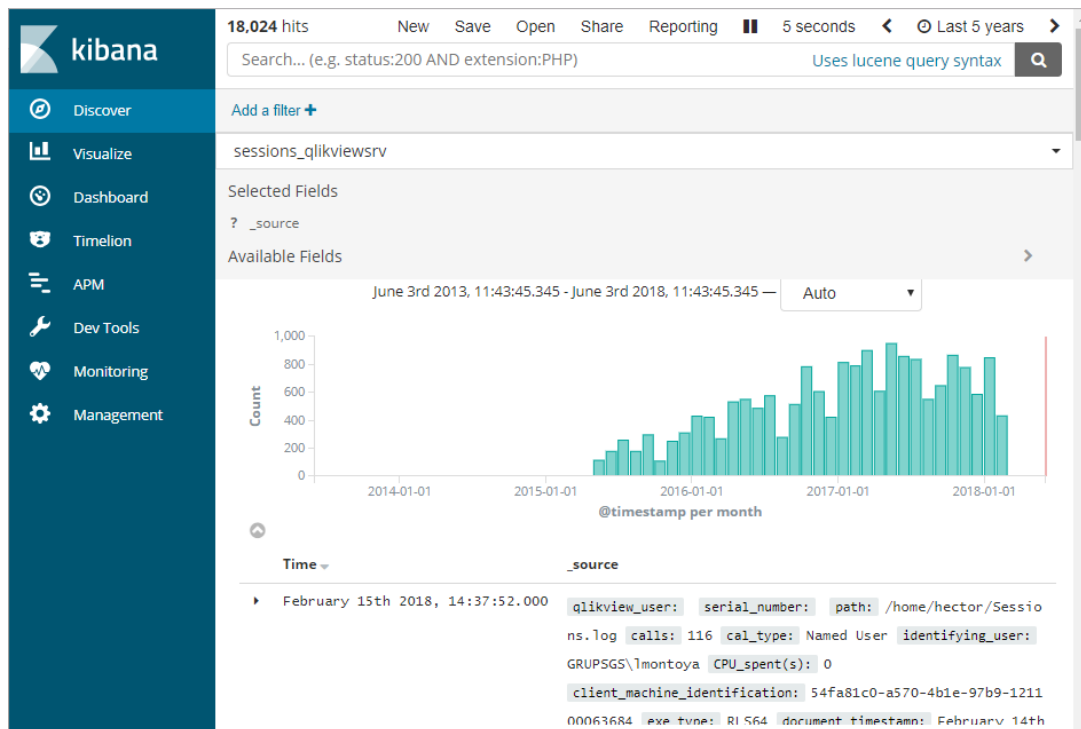


Figura 44. Discover de les sessions

Quan ja surten els documents en pantalla, anem a la pestanya *Visualize* i seleccionem *Heat Map*:

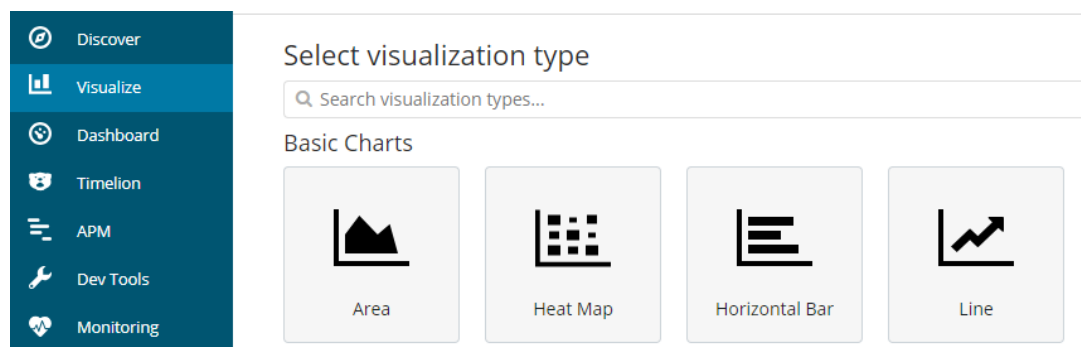


Figura 45. Selecció de gràfica

I seleccionem l'*index pattern* creat anteriorment:

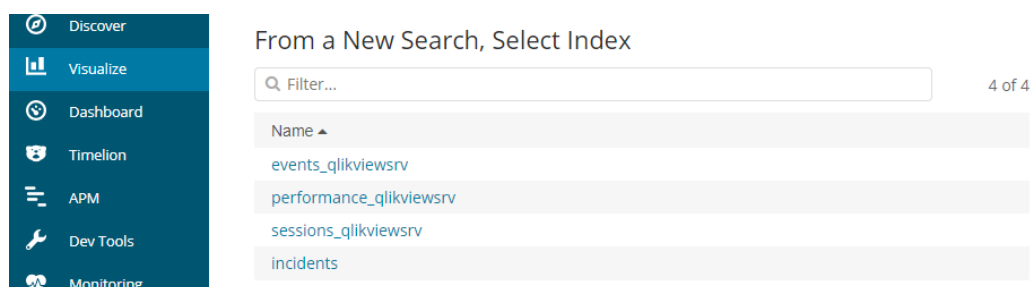


Figura 46. Selecció *index pattern*

Per fer el recompte de documents per dia de la setmana, necessitem la mètrica *count*. Les mètriques pertanyen a un tipus d'agregacions que computa determinades operacions sobre un set de documents:

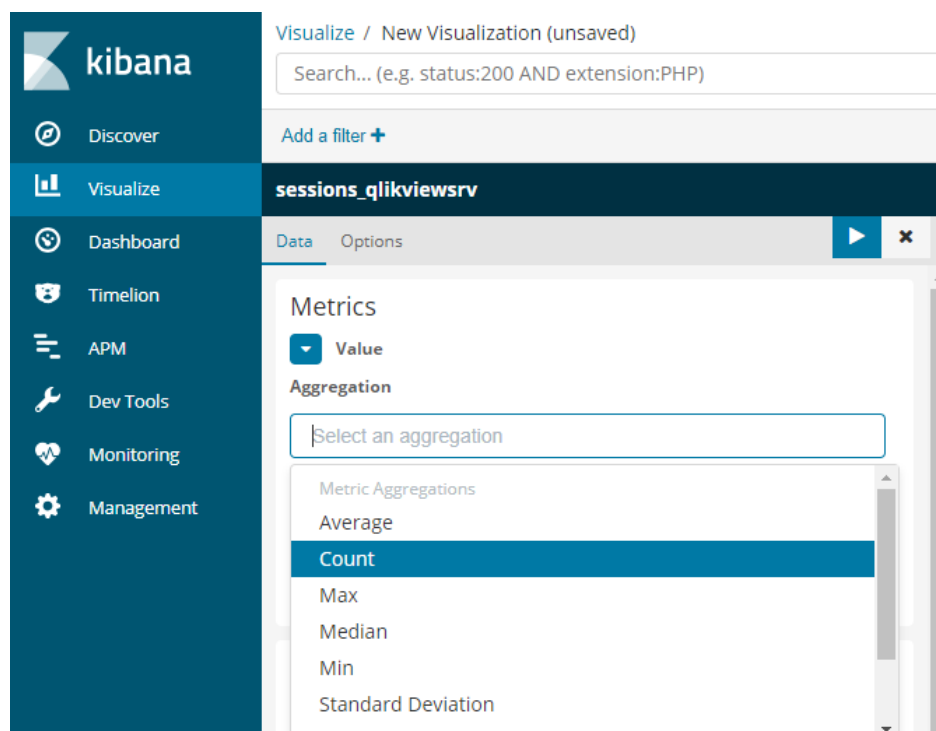


Figura 47. Selecció de mètrica

A continuació configurem els eixos del *heatmap*. Com que volem el recompte per dia de la setmana, en l'eix Y posarem els dies de la setmana, mentre que en l'eix X serà un histograma setmanal:

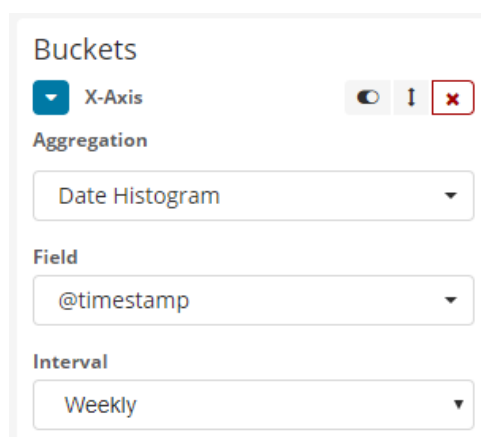


Figura 48. Configuració eix X

Y-Axis [toggle] [info] [close]

Sub Aggregation

Terms

Field: calls

Order By: metric: Count

Order: Descending Size: 5

☐ Group other values in separate bucket [i](#)

☐ Show missing values [i](#)

Custom Label: DayOfWeek

[Advanced](#)

Exclude

Include

JSON Input [i](#)

```
{
  "script": "doc['@timestamp'].date.dayOfWeek"
}
```

Figura 49. Configuració eix Y

Com que Kibana no té una opció predefinida per mostrar els dies de la setmana en l'eix Y, podem forçar-li afegint un script en JSON Input:

```
{
  "script": "doc['@timestamp'].date.dayOfWeek"
}
```

Que retorna un valor numèric segons el dia de la setmana (1: dilluns, 2: dimarts: 3: dimecres....)

Finalment podem personalitzar l'estètica de la visualització:

sessions_qlikviewsrv

Data

Options

Basic Settings

Show Tooltips

☒

Highlight

☐

Legend Position

right

Heatmap Settings

Color Schema

Reds

Reverse Color Schema

☐

Color Scale

linear

Scale to Data Bounds

☐

Percentage Mode

☐

Number of colors

4

Custom Ranges

☐

Show Labels

☒

Rotate

☐

Color

#555

Figura 50. Personalització de heatmap

Resultant en el heatmap inicial:



Figura 51. Heatmap

5.3 Tractament i Anàlisi d'Incidències

El segon tipus de dades a tractar correspon a incidències, i conté camps de text. Així doncs, a més d'indexar els fitxers a Elasticsearch, cal analitzar el text per tal de poder classificar els documents automàticament en unes categories predefinides que permetran fer consultes específiques per conceptes d'interès.

A diferència de l'apartat anterior, ara tenim dos documents relacionats entre si:

Incidents.original.json: Descriuen les incidències, queixes o peticions dels usuaris que interactuen amb el sistema d'informació de l'hospital, com per exemple metges, administradors, empreses externes... També indiquen un codi d'agent responsable per solucionar el problema.

Actions.original.json: Descriuen les mesures preses pels agents responsables de resoldre el problema.

Una **incidència** té 0..N **accions** relacionades per solucionar el problema, per exemple:

```
{
  "ID_INCIDENT": 84163, ...
  "TITLE": "Demandar nova prn laser color",
  "DESCRIPTION": "hp cp2025 de secretaria oncologia s'ha trencat"
},
{
  "ID_INCIDENT": 84163,
  "ID_ACTION": 112119, ...
  "DESCRIPTION_TEXT": "Demandar nova prn laser color per substituir hp
cp2025 de secretaria oncologia"
},
{
  "ID_INCIDENT": 84163,
  "ID_ACTION": 116852, ...
  "DESCRIPTION_TEXT": "HR16052301"
},
{
  "ID_INCIDENT": 84163,
  "ID_ACTION": 116853, ...
  "DESCRIPTION_TEXT": "rebut"
},
```

```

{
  "RECORDS": [
    {
      "ID_INCIDENT":76316,
      "ENTRY_CHANNEL":null,
      "ASSIGNED_TO":10,
      "CREATION_DATE":"2016-01-04",
      "CLOSE_DATE":"2016-01-07",
      "CENTRE_INT":"LABORATORI DE REFERENCIA DE CATALUNYA",
      "CENTRE_PET_DEP":"Laboratori",
      "TIPUS":"Incidència",
      "CLASSE":"SW_Genomi",
      "TITLE":"no funciona el genomi",
      "DESCRIPTION":"stk a tsreus13"
    },
    {
      "ID_INCIDENT":76319,
      "ENTRY_CHANNEL":null,
      "ASSIGNED_TO":10,
      "CREATION_DATE":"2016-01-04",
      "CLOSE_DATE":"2016-01-07",
      "CENTRE_INT":"ABS LA SELVA DEL CAMP - LA SELVA",
      "CENTRE_PET_DEP":"La Selva Del Camp",
      "TIPUS":"Incidència",
      "CLASSE":"SW_Genomi",
      "TITLE":"no pot imprimir la recepta electronica",
      "DESCRIPTION":"error de gestor de java."
    },
    ...
  ]
}

```

Figura 52. Incidents.original.json

```

{
  "RECORDS": [
    {
      "ID_INCIDENT":76328,
      "ID_ACTION":100684,
      "ID_AGENT":7,
      "ACTION_DATE":"2016-01-07",
      "DESCRIPTION_TEXT":"voy revisar equipo y vuelvo poner equipo en
dominio cambiando nombre a Aria24"
    },
    {
      "ID_INCIDENT":76331,
      "ID_ACTION":100470,
      "ID_AGENT":10,
      "ACTION_DATE":"2016-01-04",
      "DESCRIPTION_TEXT":"error de java es treu la de 32bits i es fica la
de 64 bits."
    },
    ...
  ]
}

```

Figura 53. Actions.original.json

Es pot veure com el camp ID_INCIDENT fa de clau primària a *Incidents.original.json* i de clau forana a *Actions.original.json*. No obstant això, el gestor està interessat a obtenir

les incidències junt amb les accions relacionades amb aquestes. És per això que ara és quan té sentit la desnormalització de dades concatenant els camps DESCRIPTION_TEXT de les accions en un nou camp ACTIONS en els documents de les incidències. En l'exemple del codi X, el document per la incidència amb ID_INCIDENT 84163 quedaria així després de la desnormalització:

```
{
  RECODS: [
    "ID_INCIDENT":      , ...
    "TITLE": "Demandar nova prn laser color",
    "DESCRIPTION": "hp cp2025 de secretaria oncologia s'ha trencat"
  ],
  ACTIONS: ": "Demandar nova prn laser color per substituir hp cp2025 de
secretaria oncologia
HR16052301
rebut"
},
```

Per tant, estem descartant part de la informació de les accions ignorant els camps ID_INCIDENT, ID_ACTION, ID_AGENT pels següents motius:

- Evitar duplicació d'incidències per simplificar el projecte.
- L'administrador està més interessat en el text de les accions que en la resta de camps perquè prefereix extreure informació de com solucionar les incidències abans que saber qui i quan va realitzar tal acció. Igualment podem trobar el primer responsable de solucionar la incidència a partir del camp ASSIGNED_TO.

Encara que en aquesta memòria el contingut JSON d'aquests fitxers estiguin ben mostrats amb salts de línia, cal que JSON tingui només un salt de línia al final del fitxer (contràriament a la secció 5.1) perquè es pugui processar correctament en Logstash.

5.3.1 Tractament de les Descripcions Textuals

El motiu principal de la classificació d'incidències és perquè el gestor es queixa que la majoria estan mal classificades pel camp CLASSE.

Com que els analitzadors de text estan relacionats amb el *template* que està relacionat amb l'índex, no podem dur a terme la indexació sense tenir en compte que en la classificació d'aquests documents ens trobem amb tres grans problemes:

1. Mescla d'idiomes, la majoria en català o castellà.
2. Argot informàtic, mèdic, abreviacions, acrònims, etc.
3. Llenguatge informal amb errors ortogràfics...

El primer problema significa que no podem analitzar text en diferents idiomes amb un analitzador d'un idioma específic perquè distorsionaria l'índex invertit, ja que les regles lingüístiques per un idioma no són compatibles amb un altre idioma. S'hauria de separar els camps per idiomes per així aplicar l'analitzador que convingui. En aquest treball ens hem centrat en un sol idioma però la solució que es proposa a continuació és extrapolable al cas de camps amb diferents idiomes. Així doncs, per simplificar, primer filtrarem les incidències i accions segons l'idioma que predomini i després les indexarem.

El segon problema es pot solucionar tal com veurem després, per un expert que conegui a fons el sistema d'informació hospitalari i el seu domini lingüístic.

El tercer problema el solucionarem a partir de consultes amb similitud de termes (*semantic similarity*) i les eines de processament de llenguatge natural d'Elasticsearch.

El procediment per classificar les incidències és el següent:

1. Esbrinar quin és l'idioma predominant en *Incidents.original.json* i *Actions.original.json*
2. Una vegada sabem quin és l'idioma predominant calen dos passos previs a la indexació de les incidències mitjançant Logstash:
 - a. Crear el *template* per incidències amb analitzadors de l'idioma predominant.
 - b. Filtrar els fitxers **.original.json* perquè quedin només esdeveniments en l'idioma predominant.
3. Indexar les incidències amb Logstash.
4. Classificar els documents de les incidències.
5. Desnormalitzar les accions amb les incidències.

En la classificació no tenim en compte les accions perquè:

- a) Hi haurà un espai de temps relativament gran entre la creació de la incidència i la primera acció que es porti a terme per solucionar-la. Per tant si volem classificar les incidències tan ràpid com es produeixin, s'han d'ignorar les accions.
- b) Estem classificant pròpiament les incidències. Potser les accions realitzades poden distorsionar el sentit de la incidència si les tenim en compte a l'hora de fer la classificació.

Per classificar els documents ens interessen els camps en els quals podem trobar text natural.

Incidents*.json:

- RECORDS.TITLE
- RECORDS.DESCRPTION

Posteriorment a la classificació, també ens interessa el següent camp, perquè pot ser d'interès per al gestor per fer una cerca *full-text* d'accions.

Actions*.json:

- RECORDS.DESCRPTION_TEXT

5.3.2 Filtratge per Idioma Predominant

Per esbrinar quin idioma és el que predomina en aquests camps, executarem el següent script:

```
import json
from langdetect import detect, lang_detect_exception

count_ca = 0
count_es = 0
count_mix = 0
count_other = 0

with open('Incidents.original.json', 'r', encoding='utf-8') as f:
    incident_dict = json.load(f)

for doc in incident_dict['RECORDS']:
    title = doc['TITLE']
    des = doc['DESCRIPTION']
    lang1 = ""
    lang2 = ""
    if title:
        try:
            lang1 = detect(str(title))
        except lang_detect_exception.LangDetectException:
            pass
    if des:
        try:
            lang2 = detect(str(des))
        except lang_detect_exception.LangDetectException:
            pass

    if lang1 != "" and lang2 != "":
        if lang1 == "es" and lang2 == "es":
            count_es += 1
        elif lang1 == "ca" and lang2 == "ca":
            count_ca += 1
        elif (lang1 == "ca" and lang2 == "es") or (lang1 == "es" and
lang2 == "ca"):
            count_mix += 1
        else:
            count_other += 1
    elif lang1 == "es" or lang2 == "es":
        count_es += 1
    elif lang1 == "ca" or lang2 == "ca":
        count_ca += 1
    else:
        count_other += 1

print("Incidents")
print("Spanish docs = " + str(count_es))
print("Catalan docs = " + str(count_ca))
print("Mixed docs = " + str(count_mix))
print("Other docs = " + str(count_other))

count_ca = 0
count_es = 0
count_mix = 0
count_other = 0
```

```

with open('Actions.original.json', 'r', encoding='utf-8') as f:
    actions_dict = json.load(f)

for doc in actions_dict['RECORDS']:
    text = str(doc['DESCRIPTION_TEXT'])
    lang = ""
    try:
        lang = detect(text)
        if lang == "es":
            count_es += 1
        elif lang == "ca":
            count_ca += 1
        else:
            count_other += 1
    except lang_detect_exception.LangDetectException:
        count_other += 1

print("Actions")
print("Spanish docs = " + str(count_es))
print("Catalan docs = " + str(count_ca))
print("Mixed docs = " + str(count_mix))
print("Other docs = " + str(count_other))
Codi 9. counts.py

```

Un cop processades les dades originals, hem obtingut que el català és l'idioma predominant:

```

Incidents
Spanish docs = 3490
Catalan docs = 7407
Mixed docs = 1924
Other docs = 11994

Actions
Spanish docs = 11096
Catalan docs = 16499
Mixed docs = 0
Other docs = 12097

```

És a dir, la majoria d'esdeveniments són purament en català. Malgrat això, veiem que hi ha casos en què el sistema els classifica com "Other docs", que indica quants esdeveniments el detector ha descartat que siguin en català o en castellà per falta de confiança o per idioma estranger. "Mixed docs" representa aquells documents que tenen una barreja clara de camps en català i castellà.

Filtrar els esdeveniments per idioma i desnormalitzar les accions comporta una dificultat superior a l'hora de fer-ho amb Logstash, ja que Logstash està pensat per simplement automatitzar el procés d'indexació si ens fixem només en els plugins predefinits. Per tant, sacrifiquem les capacitats d'automatització de Logstash en aquest apartat per centrar-nos en la classificació de documents, utilitzant Logstash solament com a suport per les incidències filtrades per idioma.

Seguint el procediment, ens disposem a fer un pre-procés filtrant les incidències i les accions per crear nous fitxers que continguin només els esdeveniments en català:

```

import json
from langdetect import detect, lang_detect_exception

incident_ca = {"RECORDS": []}
actions_ca = {"RECORDS": []}

with open("Incidents.original.json", 'r', encoding='utf-8') as f:
    incident_dict = json.load(f)

print("n° original incidents: " + str(len(incident_dict['RECORDS'])))

for doc in incident_dict['RECORDS']:
    title = doc['TITLE']
    des = doc['DESCRIPTION']
    lang1 = ""
    lang2 = ""
    if title:
        try:
            lang1 = detect(str(title))
        except lang_detect_exception.LangDetectException:
            pass
    if des:
        try:
            lang2 = detect(str(des))
        except lang_detect_exception.LangDetectException:
            pass

    if lang1 != "" and lang2 != "":
        if lang1 == "ca" and lang2 == "ca":
            incident_ca['RECORDS'].append(doc)
    elif lang1 == "ca" or lang2 == "ca":
        incident_ca['RECORDS'].append(doc)

print("n° filtered incidents: " + str(len(incident_ca['RECORDS'])))

with open("Incidents.json", 'w', encoding='utf-8') as f:
    json.dump(incident_ca, f, ensure_ascii=False)
    f.write('\n')

with open("Actions.original.json", 'r', encoding='utf-8') as f:
    actions_dict = json.load(f)

print("n° original actions: " + str(len(actions_dict['RECORDS'])))

for doc in actions_dict['RECORDS']:
    text = str(doc['DESCRIPTION_TEXT'])
    lang = ""
    try:
        lang = detect(text)
        if lang == "ca":
            actions_ca['RECORDS'].append(doc)
    except lang_detect_exception.LangDetectException:
        actions_ca['RECORDS'].append(doc)

print("n° filtered actions: " + str(len(actions_ca['RECORDS'])))

with open("Actions.json", 'w', encoding='utf-8') as f:
    json.dump(actions_ca, f, ensure_ascii=False)

```

```
f.write('\n')  
Codi 10. filter.py
```

I obtenim el següent:

```
nº original incidents: 24815  
nº filtered incidents: 7442  
nº original actions: 39692  
nº filtered actions: 16769
```

Després d'executar-ho tenim creats els nous fitxers filtrats en català. *Incidents.json* i *Actions.json*. Llavors, de les 24815 incidències originals, queden 7442 incidències purament en català.

5.3.3 Definició del Template per Incidències

L'anàlisi textual dels camps es portarà a terme internament per Elasticsearch a partir de la configuració del *template* dedicat per incidències. En aquest cas no necessitem *template* per les accions, ja que aquestes s'afegiran posteriorment a les incidències amb un script.

```
{  
  "template": "incidents",  
  "version": 50001,  
  "settings": {  
    "index.refresh_interval": "5s",  
    "number_of_shards": "3",  
    "number_of_replicas": "2"  
  },  
  "mappings": {  
    "doc": {  
      "properties": {  
        "@timestamp": {  
          "type": "date"  
        },  
        "@version": {  
          "type": "keyword"  
        },  
        "RECORDS": {  
          "properties": {  
            "ASSIGNED_TO": {  
              "type": "keyword"  
            },  
            "CENTRE_INT": {  
              "type": "text",  
              "analyzer": "catalan",  
              "fields": {  
                "keyword": {  
                  "type": "keyword",  
                  "ignore_above": 256  
                }  
              }  
            },  
            "CENTRE_PET_DEP": {  
              "type": "text",  
              "analyzer": "catalan",  
              "fields": {  
                "keyword": {  
                  "type": "keyword",  
                  "ignore_above": 256  
                }  
              }  
            }  
          }  
        }  
      }  
    }  
  }  
}
```


En aquest *template*, amb la propietat "analyzer":"catalan" estem activant l'analitzador de llenguatge natural predefinit de català que tracta el text de la següent manera:

1. Trencar el text en paraules segons l'algorisme de segmentació de text Unicode Text Segmentation [29]
2. Elimina els apòstrofs. Per exemple, l'escola → escola
3. Converteix les lletres de les paraules a minúscules
4. Elimina les *stopwords* en català. Com per exemple "i" "el" [30]
5. Redueix les paraules a la seva arrel. Per exemple: passar → pas
6. Els *tokens* resultants s'afegeixen a l'índex invertit si s'està indexant, o es comparen amb les que ja n'hi ha si se cerca.

5.3.4 Indexació d'Incidències

Una vegada carregat el *template* procedim a indexar les dades seguint el procediment de l'apartat 5.1, però ara amb una definició de *pipeline* diferent, ja que es tracta d'una entitat diferent:

```
input {
  file {
    path => "/home/hector/Incidents.json"
    ignore_older => 0
    start_position => "beginning"
    sincedb_path => "/dev/null"
    type => "json"
    codec => "json"
  }
}
filter {
  split {
    field => "RECORDS"
  }
  mutate {
    add_field => {"ACTIONS" => " "}
  }
  date {
    match => [ "[RECORDS][CREATION_DATE]", "ISO8601" ]
  }
}
output {
  elasticsearch {
    hosts => ["192.168.1.200", "192.168.1.201", "192.168.1.202"]
    index => "incidents"
    document_id => "%{[RECORDS][ID_INCIDENT]}"
    manage_template => false
  }
  stdout {
    codec => rubydebug
  }
}
}
Codi 12. incidents.conf
```

Els punts que més diferencien aquest *pipeline* són els següents:

```
split {  
  field => "RECORDS"  
}
```

L'etapa de filtratge parteix l'*array* "RECORDS" perquè cada element de l'*array* sigui tractat com un *event*.

No seria necessari, però per simplificar la concatenació de les accions en el camp ACTIONS, creem aquesta clau sense valor amb:

```
mutate {  
  add_field => {"ACTIONS" => " "  
}
```

Per simplificar la desnormalització d'accions assignem el valor d'ID_INCIDENT al camp obligatori "_id" que serveix per identificar cada document en l'índex al qual pertany:

```
document_id => "%{[RECORDS][ID_INCIDENT]}"
```

5.3.5 Classificador

En les incidències s'hi pot observar que els texts són bastant curts. El model TF/IDF afavoreix els documents en els quals una paraula surt moltes vegades en pocs documents i penalitza aquelles paraules que surten en molts documents (perquè són paraules comuns). Però amb els texts tan breus d'incidències, aquest model TF/IDF deixa de ser tan efectiu.

Les categories i subcategories necessàries per a la classificació s'han definit amb l'ajuda del gestor de l'hospital. S'han definit fins a 3 nivells, però es permeten tants com sigui necessaris. En el primer nivell hi ha categories generals, que són les que ja existeixen actualment al camp CLASSE, i que ens permetran validar el sistema. Els conjunts de subcategories han de permetre afinar més el tipus d'incidència, per ajudar al personal tècnic que l'hagi de solucionar:

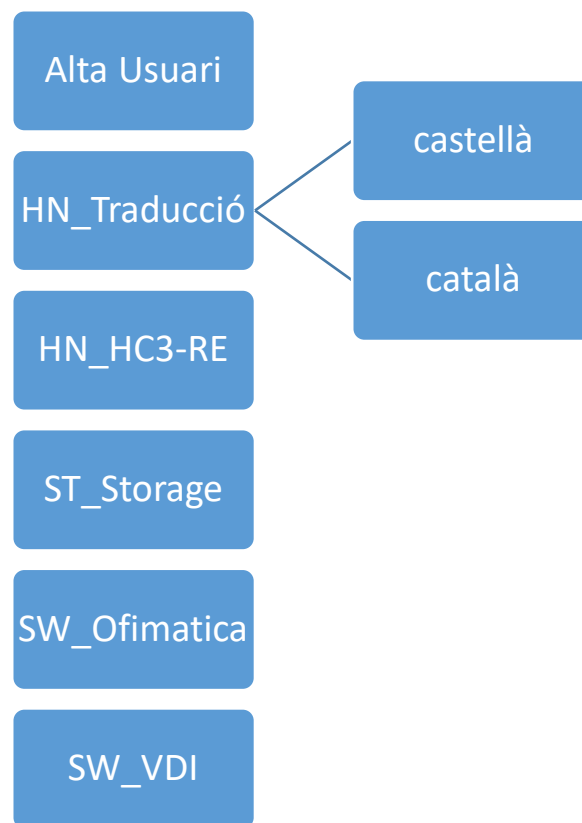


Figura 54. Taxonomia (1)

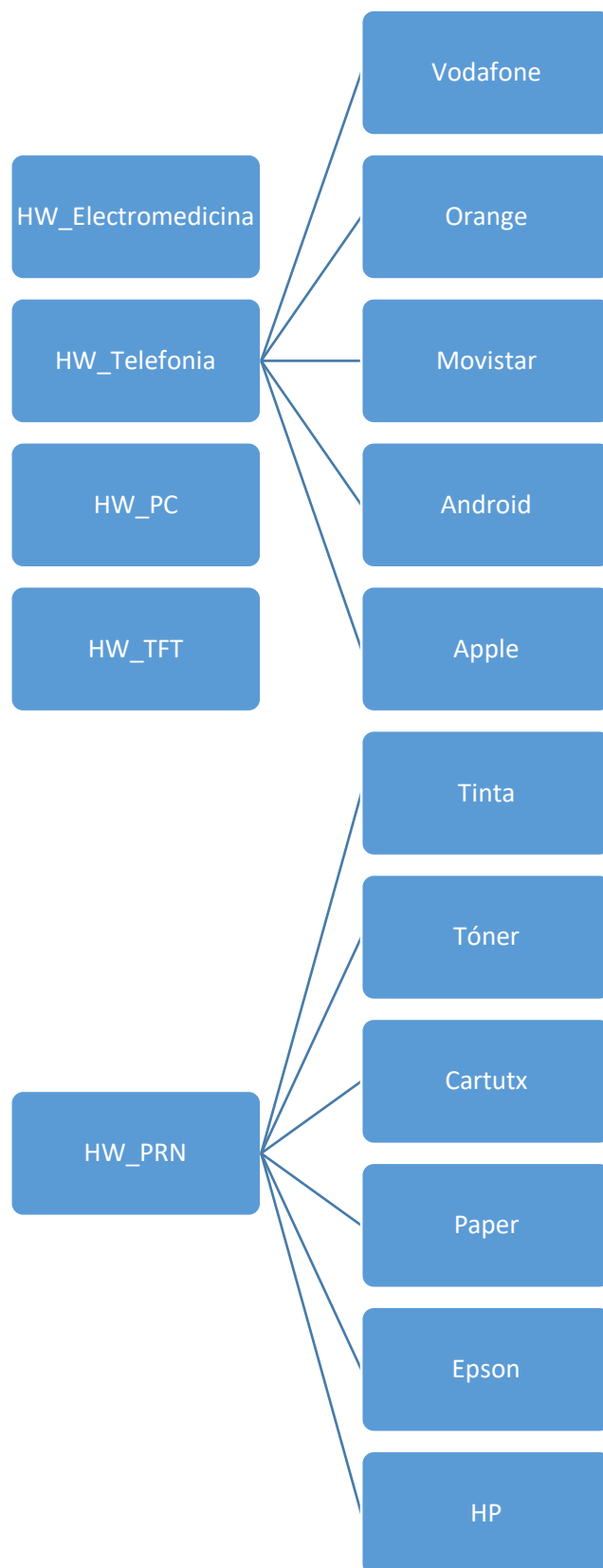


Figura 55. Taxonomia (2)

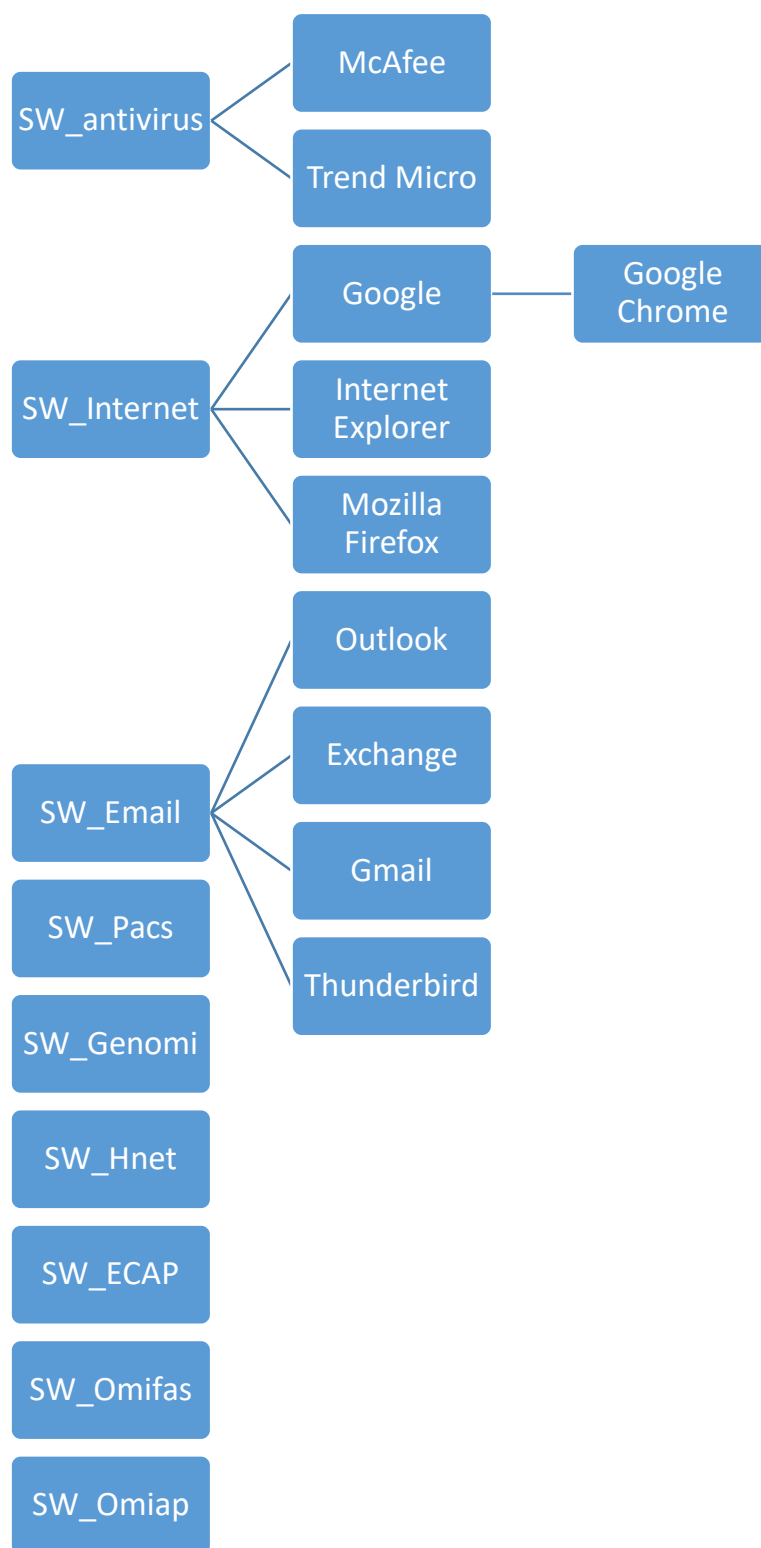


Figura 56. Taxonomia (3)

Aquesta taxonomia es guarda en **tags.json**. Això permet que es puguin modificar, afegir o eliminar els nodes que convinguin en qualsevol moment.

A cada node es defineixen uns *keywords* per poder cercar els documents que pertanyin a aquesta categoria. Aquests *keywords* són paraules claus que només haurien d'aparèixer en els documents de tal categoria. Opcionalment es pot definir recursivament les categories amb l'etiqueta "*tags*".

Un cop s'ha definit la taxonomia de termes i s'han detallat les expressions que ens permetran identificar-los, s'ha dissenyat un algoritme de classificació automàtica dels documents, consistint dels següents passos:

1. Explorar l'arbre de categories definit a *tags.json* recursivament.
2. Per cada node se cerquen les incidències més similars amb una *query* formada pels *keywords* del node i dels seus pares.
3. Els documents resultants es guarden en un diccionari. Un document pot aparèixer diverses vegades com a resultat de diferents cerques, per tant li aplicarem els *tags* que més *_score* li proporcionin.
4. Calcular mètriques per avaluar posteriorment els resultats.
5. Actualitzar la informació de les incidències emmagatzemades a Elasticsearch amb els *tags* extrets per l'algorisme recursiu.

Algorisme recursiu:

```
import copy
import csv
import json

from elasticsearch import Elasticsearch, ElasticsearchException

tags_file = "tags.json"

total = {
    "HW_PRN": 712,
    "SW_Email": 610,
    "HN_HC3-RE": 536,
    "HW_PC": 477,
    "SW_Genomi": 405,
    "SW_antivirus": 291,
    "ST_Storage": 260,
    "SW_Ofimatica": 244,
    "SW_Hnet": 239,
    "SW_ECAP": 219,
    "Alta Usuari": 148,
    "SW_Omifas": 122,
    "SW_Pacs": 121,
    "HW_Telefonia": 120,
    "SW_VDI": 115,
    "SW_Internet": 88,
    "SW_Omiap": 75,
    "HW_TFT": 43,
    "HW_Electromedicina": 38,
    "HN_Traducció": 4
}

all = 0
for v in total.values():
```

```

    all += v

'''
tags_file = "tags.test.json"

total = {
    "Alta Usuari": 20,
    "HN_Traducció": 5,
    "HW_PC": 20,
    "HW_PRN": 20,
    "SW_antivirus": 20,
    "SW_Internet": 20,
    "SW_Email": 20,
    "SW_Pacs": 20
}

all = 145'''

es = Elasticsearch([{'host': '192.168.1.200'}, {'host':
'192.168.1.201'}, {'host': '192.168.1.202'}])
incidents_dict = {}

def generate_constant(dict, lvl):
    filters = ""
    keywords = list(dict.values())[0]
    for keyword in keywords.split():
        filters += ""\n{ "constant_score": {
                                "boost": "" + str(lvl) + "",
                                "filter": { "match": {
"RECORDS.DESCRPTION": \"\"\" + keyword + \"\"\" } }
                                }}, ""
        filters += ""\n{ "constant_score": {
                                "boost": "" + str(lvl * 3) + "",
                                "filter": { "match": {
"RECORDS.TITLE": \"\"\" + keyword + \"\"\" } }
                                }}, ""
    return filters

def generate_body(chain):
    lvl = 1
    filters = ""
    for parent in chain:
        filters += generate_constant(parent, lvl)
        lvl += 1
    if not filters:
        return
    filters = filters[:-1]

    body = ""
    {
        "size" : 2000,
        "query": {
            "bool": {
                "should": [
                    "" + filters + ""
                ]
            }
        }
    }

```

```

    }
    """
    return body

def get_keys(chain):
    l = []
    for item in chain:
        l.append(list(item)[0])
    return l

def add_results(results, chain):
    for doc in results:
        if doc['_id'] in incidents_dict:
            if doc['_score'] > incidents_dict[doc['_id']]['_score']:
                incidents_dict[doc['_id']]['_score'] = doc['_score']
                incidents_dict[doc['_id']]['tags'] = get_keys(chain)
            else:
                incidents_dict[doc['_id']] = doc
                incidents_dict[doc['_id']]['tags'] = get_keys(chain)

def classify(chain):
    body = generate_body(chain)
    if not body:
        return
    results = es.search(index=["incidents"], body=body, scroll="1m")
    add_results(results['hits']['hits'], chain)
    sid = results['_scroll_id']
    scroll_size = results['hits']['total']
    while scroll_size > 0:
        results = es.scroll(scroll_id=sid, scroll='10s')
        add_results(results['hits']['hits'], chain)
        sid = results['_scroll_id']
        scroll_size = len(results['hits']['hits'])

def iterate(cat, chain):
    for tag, keywords in cat.items():
        if tag == "tags":
            for cat in keywords:
                iterate(cat, copy.deepcopy(chain))
        else:
            chain.append({tag: keywords})
            classify(chain)

with open(tags_file, 'r', encoding='utf-8') as f:
    tags_dict = json.load(f)

for cat in tags_dict['tags']:
    iterate(cat, [])

for min_score in range(0, 30):
    tp = {}
    fp = {}
    for inc in incidents_dict.values():
        if inc['_score'] < min_score:
            continue

```

```

        if inc['tags'][0] == str(inc['_source']['RECORDS']['CLASSE']):
            if inc['tags'][0] in tp:
                tp[inc['tags'][0]] += 1
            else:
                tp[inc['tags'][0]] = 1
        elif str(inc['_source']['RECORDS']['CLASSE']) in total:
            if inc['tags'][0] in fp:
                fp[inc['tags'][0]] += 1
            else:
                fp[inc['tags'][0]] = 1

    for k, v in total.items():
        if k not in tp:
            tp[k] = 0
        if k not in fp:
            fp[k] = 0
        if tp[k] == 0 and fp[k] == 0:
            continue
        fn = total[k] - tp[k]
        tn = (all - total[k]) - fp[k]
        tpr = tp[k] / (tp[k] + fn)
        fpr = fp[k] / (fp[k] + tn)
        recall = tp[k] / total[k]
        precision = tp[k] / (tp[k] + fp[k])
        try:
            f1 = (2 * (recall * precision)) / (recall + precision)
        except ZeroDivisionError:
            f1 = 0

        with open('stats/' + k + '.csv', 'a', newline='') as csvfile:
            fieldnames = ['min_score', 'true positive', 'false
positive', 'true negative', 'false negative',
                        'true positive rate', 'false positive
rate', 'recall', 'precision', 'f1score']
            writer = csv.DictWriter(csvfile, fieldnames=fieldnames,
delimiter=';')
            if min_score == 0:
                writer.writeheader()
            writer.writerow(
                {'min_score': min_score, 'true positive': tp[k],
'false positive': fp[k], 'true negative': tn,
'false negative': fn, 'true positive rate': tpr,
'false positive rate': fpr, 'recall': recall,
'precision': precision, 'f1score': f1})

    update = 0
    for id, inc in incidents_dict.items():
        lvl = 1
        for tag in inc['tags']:
            try:
                body = """{
                        "doc" : {
                            "TAG" + str(lvl) + "\" : \"\"\" + tag
+ "\"\"\"\"
                        }
                    }
                """

            es.update(index='incidents', doc_type='doc', id=id,

```

```

        body=body)
    update += 1
except ElasticsearchException as e:
    print(e)

finally:
    lvl += 1

print("Inserted tags: " + str(update))

```

Codi 13. classifier.py

L'algorisme està dissenyat perquè quan s'apliqui un *tag* de nivell N a un document, s'apliquin també els *tags* dels nodes pares.

5.3.6 Score

Per cada cerca amb *keywords* que fem, generem un cos JSON per cercar en Elasticsearch. Aquest cos està definit en Query DSL. Un exemple de cerca a nivell 2, seria el següent:

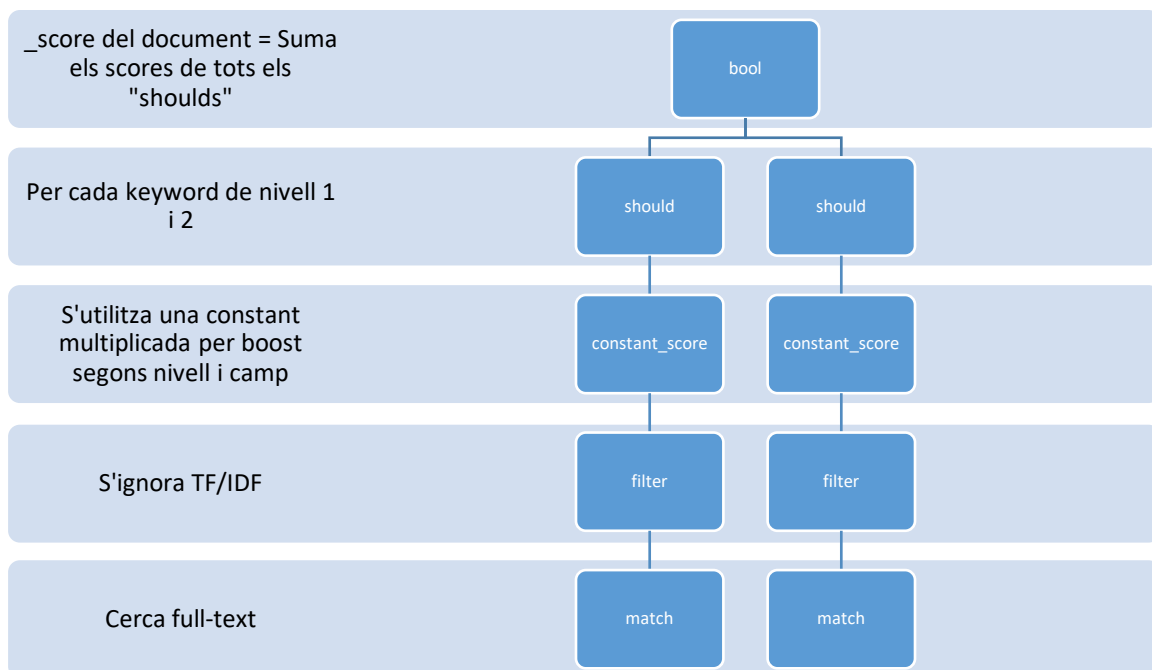


Figura 57. Càlcul de _score final d'un document

La puntuació final és la suma de tots els fills amb camp *should*. Cada fill té constant_score, és a dir, encara que s'utilitzi cerca *full-text* amb *match*, *should* només contribuirà a la suma amb constant_score * boost si el *keyword* del *match* apareix en el document.

Com major sigui el nivell del *keyword* més pes aportarà boost. També es dóna més pes al camp TITLE, ja que és on els usuaris normalment resumeixen la incidència.

Les consultes *match*, per defecte utilitzen un valor 1 de similitud de termes definit prèviament com a "edit distance" (p. 28). També utilitza cerca *full-text* per tractar els *keywords* amb l'analitzador de català.

5.3.7 Desnormalització d'Accions

Després d'indexar i classificar les incidències ajuntem les accions amb l'índex d'incidències en Elasticsearch utilitzant un altre script i Update API. També seria possible classificar les incidències després de desnormalitzar les accions, ja que aquestes no afecten en la classificació:

```
import json
from elasticsearch import Elasticsearch, ElasticsearchException
from unicodedata import normalize

es = Elasticsearch([{'host': '192.168.1.200'}, {'host':
'192.168.1.201'}, {'host': '192.168.1.201'}])

update = 0

with open('Actions.json', 'r', encoding='utf-8') as f:
    actions_dict = json.load(f)
    for doc in actions_dict['RECORDS']:
        try:
            body = """{
                        "script" : {
                            "source": "ctx._source.ACTIONS +=
params.action",
                            "lang": "painless",
                            "params" : {
                                "action" :
\\\"\"\"+repr(normalize("NFKD",
str(doc['DESCRIPTION_TEXT']).replace('\\', '\\')))[1:-1]+\"\"\"\\\"
                                }
                            }
                        }"""

            es.update(index='incidents', doc_type='doc',
id=doc['ID_INCIDENT'],
                    body=body)
            update += 1
        except ElasticsearchException as e:
            print(e)

print("Updated docs " + str(update))
Codi 14. actions.py
```

Resultat:

Updated docs 6436

És a dir, s'han ajuntat 6436 accions que pertanyen a alguna incidència dins de l'índex d'incidències. L'únic que es fa és concatenar el DESCRIPTION_TEXT de les accions en un camp anomenat ACTIONS en les incidències.

6. Resultats i Avaluació

En aquest capítol mostrarem els resultats del sistema de suport a la decisió desenvolupat en aquest treball fi de grau. Distingirem en tres apartats segons les tres tasques en què s'ha dividit la feina: instal·lació i preparació de l'entorn de desenvolupament en un clúster simulat, tractament de fitxers de log, i tractament de fitxers d'incidències.

6.1 Entorn de Desenvolupament

6.1.1 Monitoratge Elasticsearch

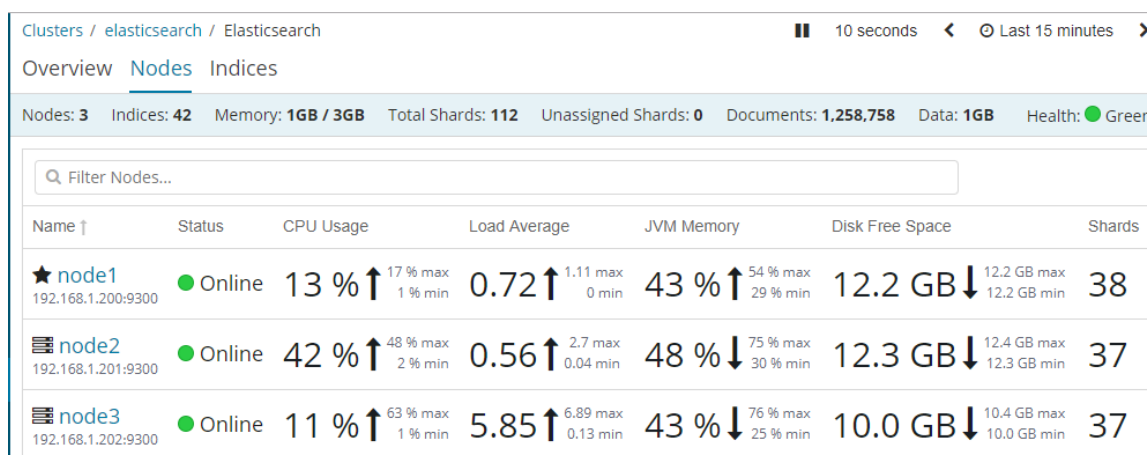


Figura 58. Estat dels nodes ES

Mètriques d'ús de les infraestructures en les quals s'executen les instàncies d'Elasticsearch. En aquesta plana podem identificar el node *master* i veure l'estat dels nodes actius que formen el clúster.

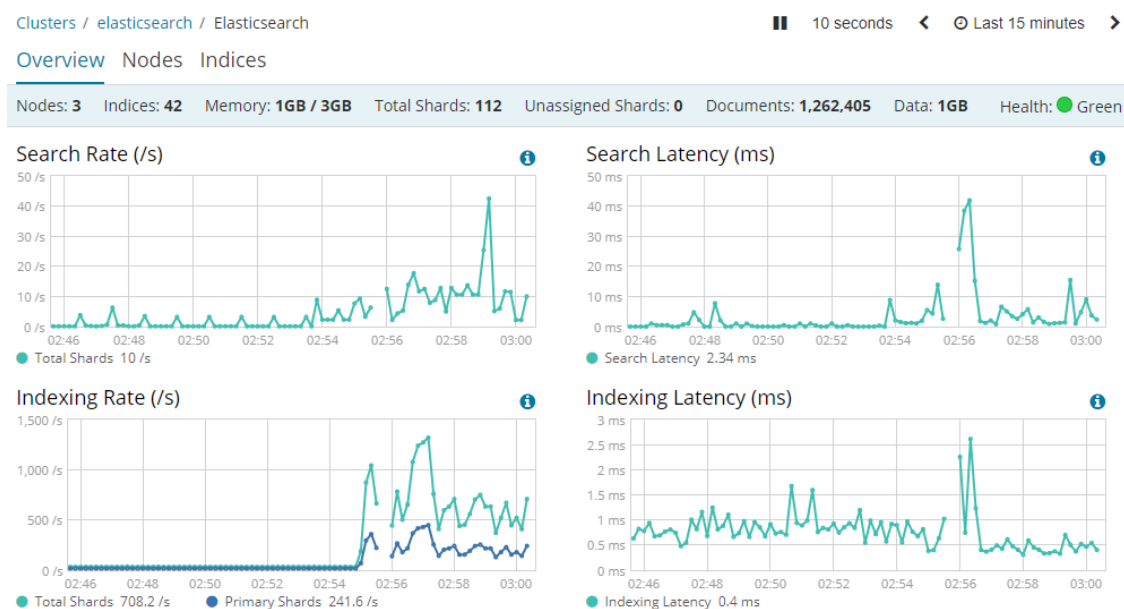


Figura 59. Mètriques de cerca i indexació

Temps de resposta d'Elasticsearch. *Search Rate* pot ser molt útil per aplicacions de cerca intensives com ara Wikipedia.

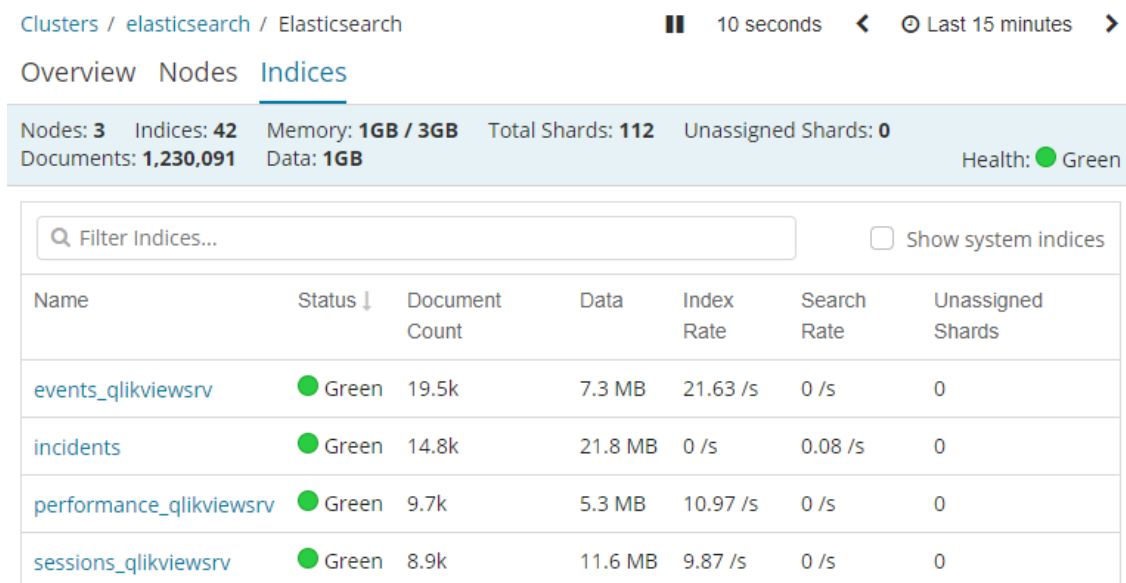


Figura 60. Estadístiques dels índexs

Útil per veure resumit l'estat dels índexs. Els estats poden ser:

- **Green:** Tots els *shards* estan repartits. El clúster és 100% operacional.
- **Yellow:** Alguns *shards* secundaris no estan actius. Risc d'indisponibilitat alt.
- **Red:** Com a mínim ha desaparegut un *shard* primari amb totes les seves rèpliques.

Shard Legend

Primary Replica Relocating Initializing Unassigned Primary Unassigned Replica

Nodes

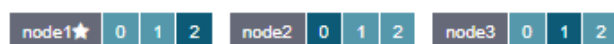


Figura 61. Repartició de *shards*

Distribució dels *shards* primaris i secundaris en el clúster. Es pot observar com Elasticsearch intenta paral·lelitzar l'escriptura, repartint equitativament els *shards* primaris.

6.1.2 Monitoratge Logstash

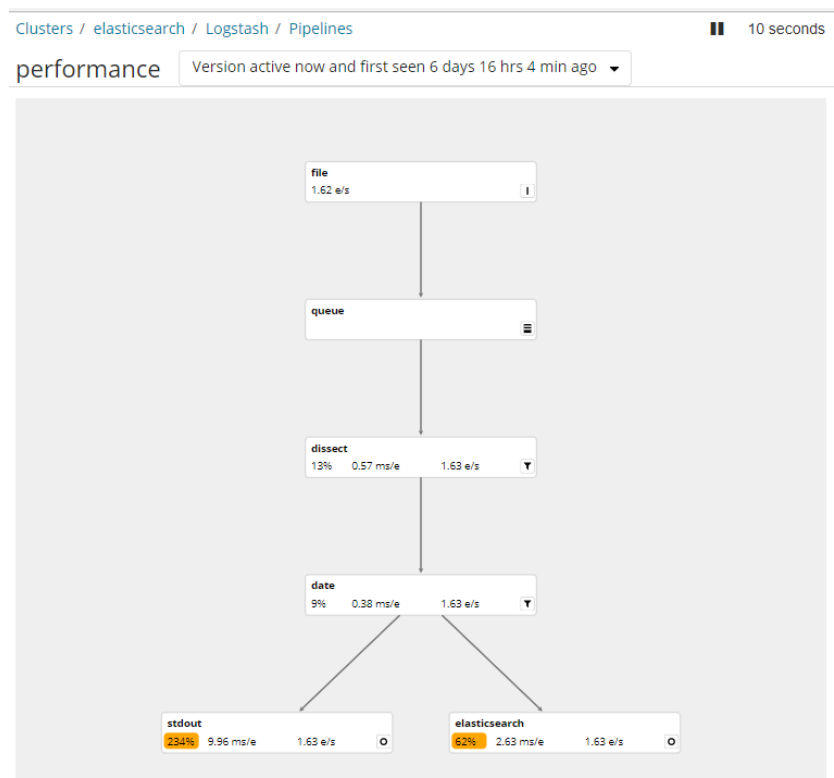


Figura 62. Monitoratge d'un pipeline

La informació més rellevant que es pot extreure d'aquestes figures és la relació events enviats/unitat de temps quan Logstash està processant *pipelines*.

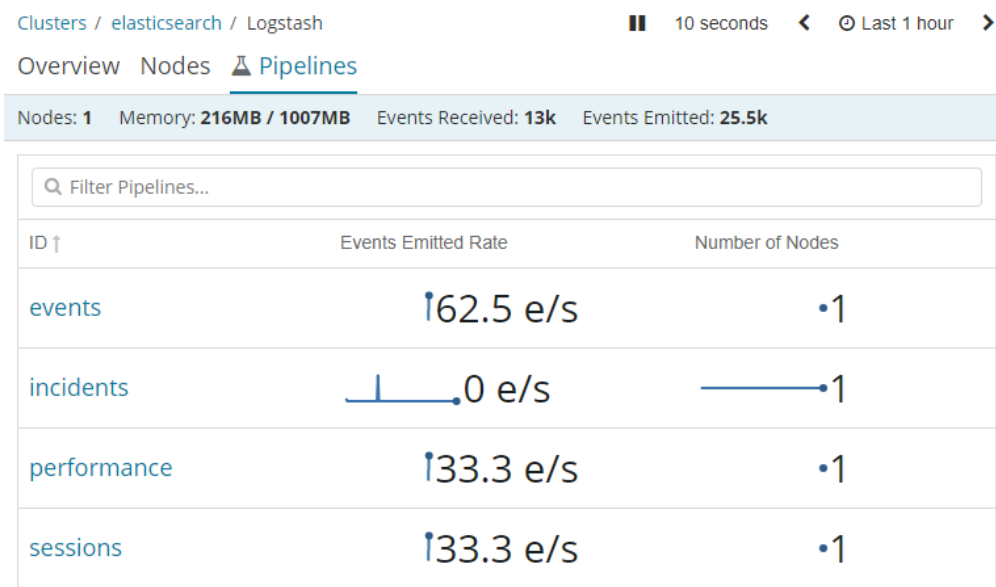


Figura 63. Mètriques dels processament dels pipelines

6.1.3 Monitoratge i Seguretat Kibana

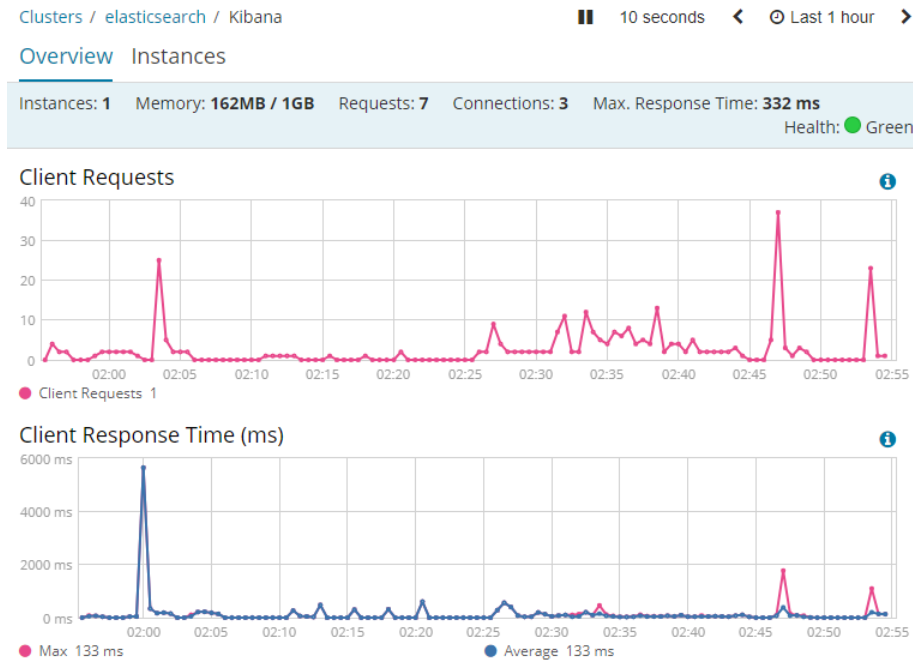


Figura 64. Estadístiques de resposta de Kibana

D'aquesta figura podríem identificar atacs contra el servidor Kibana, com per exemple DDOS, segons els pics de peticions i respostes.

Unauthorized

This server could not verify that you are authorized to access the document requested. Either you supplied the wrong credentials (e.g., bad password), or your browser doesn't understand how to supply the credentials required.

Apache/2.4.25 (Debian) Server at kibanaccess.ddns.net Port 443

Figura 65. Missatge d'error inici de sessió incorrecte

Quan s'intenta accedir a Kibana a partir de <https://kibanaccess.ddns.net/> sense autorització, es mostra el pertinent missatge d'error. Amb els fitxers de log d'Apache també podem rastrejar el tràfic per detectar possibles intrusions:

```
84.78.18.137 - - [06/Jun/2018:03:42:10 +0200] "GET / HTTP/1.1" 401 919 "-" "Mozilla/5.0 (Linux; Android 5.1; HUAWEI TAG-L21 Build/HUAWEITAG-L21) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.158 Mobile Safari/537.36"
84.78.18.137 - - [06/Jun/2018:03:42:14 +0200] "GET / HTTP/1.1" 401 766 "-" "Mozilla/5.0 (Linux; Android 5.1; HUAWEI TAG-L21 Build/HUAWEITAG-L21) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.158 Mobile Safari/537.36"
84.78.18.137 - - [06/Jun/2018:03:42:17 +0200] "GET / HTTP/1.1" 401 766 "-" "Mozilla/5.0 (Linux; Android 5.1; HUAWEI TAG-L21 Build/HUAWEITAG-L21) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.158 Mobile Safari/537.36"
84.78.18.137 - - [06/Jun/2018:03:42:18 +0200] "GET / HTTP/1.1" 401 766 "-" "Mozilla/5.0 (Linux; Android 5.1; HUAWEI TAG-L21 Build/HUAWEITAG-L21) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.158 Mobile Safari/537.36"
84.78.18.137 - - [06/Jun/2018:03:42:20 +0200] "GET / HTTP/1.1" 401 766 "-" "Mozilla/5.0 (Linux; Android 5.1; HUAWEI TAG-L21 Build/HUAWEITAG-L21) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.158 Mobile Safari/537.36"
```

Figura 66. Apache access.log

6.1.4 Monitoratge Fallides/Split Brain

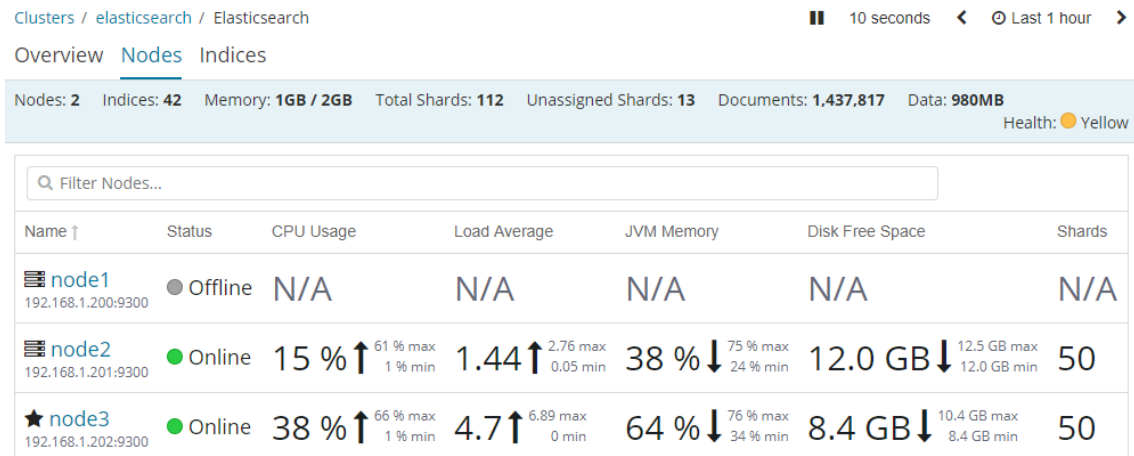


Figura 67. Simulació de fallida del node1

Simulant la fallida del node1, en aquest exemple podem veure que el clúster segueix disponible perquè la majoria de nodes continuen actius, però l'estat canvia *Yellow* perquè el clúster està a punt de fallar completament:

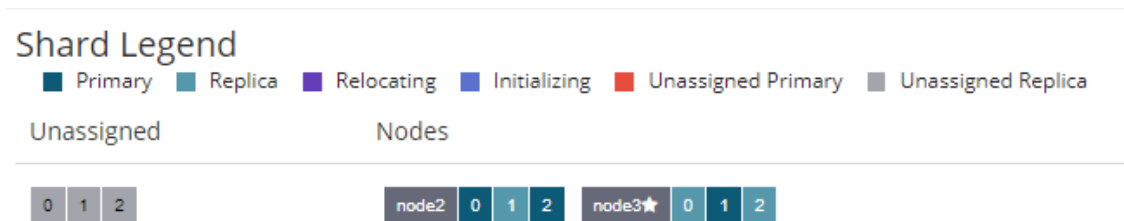


Figura 68. Manegament automàtic de shards després de la fallida

Es pot observar com Elasticsearch ha elevat la categoria del shard 2 del node2, de secundari a primari, respecte la figura 61.

Clusters / elasticsearch / Elasticsearch

10 seconds

<

⌚ Last 1 hour

>

Overview

Nodes

Indices

Nodes: 2

Indices: 42

Memory: 898MB / 2GB

Total Shards: 112

Unassigned Shards: 13

Documents: 1,435,256

Data: 989MB

Health: ● Yellow

🔍 Filter Indices...

☐ Show system indices

Name	Status ↓	Document Count	Data	Index Rate	Search Rate	Unassigned Shards
events_qlikviewsrv	● Yellow	19.5k	4.9 MB	5.41 /s	0 /s	3
incidents	● Yellow	14.8k	14.5 MB	3.66 /s	0.25 /s	3
performance_qlikviewsrv	● Yellow	203.5k	71.6 MB	56.54 /s	0 /s	3
sessions_qlikviewsrv	● Yellow	8.9k	7.7 MB	2.47 /s	0 /s	3

Figura 69. Estats dels índexs amb consistència eventual

L'estat dels índexs canvien a *Yellow* perquè falta una rèplica de cada *shard*. Si ara també fem fallar el node2, Kibana continuarà disponible al node3 però el clúster deixa d'estar actiu per evitar *split brain*.

Status: Red

kibanaccess.ddns.net

Heap Total

178.25 MB

Heap Used

124.87 MB

Load

0.48, 2.18, 2.67

Response Time Avg

139.00 ms

Response Time Max

139.00 ms

Requests Per Second

0.20

Status Breakdown

ID	Status
plugin:kibana@6.2.4	✔ Ready
plugin:elasticsearch@6.2.4	⚠ Request Timeout after 30000ms
plugin:xpack_main@6.2.4	✔ Ready
plugin:searchnrofiler@6.2.4	✔ Ready

Figura 70. Prevenió d'inici del clúster per evitar split brain

6.2 Tractament de Fitxers de Log

Dashboard amb mètriques útils per al gestor:

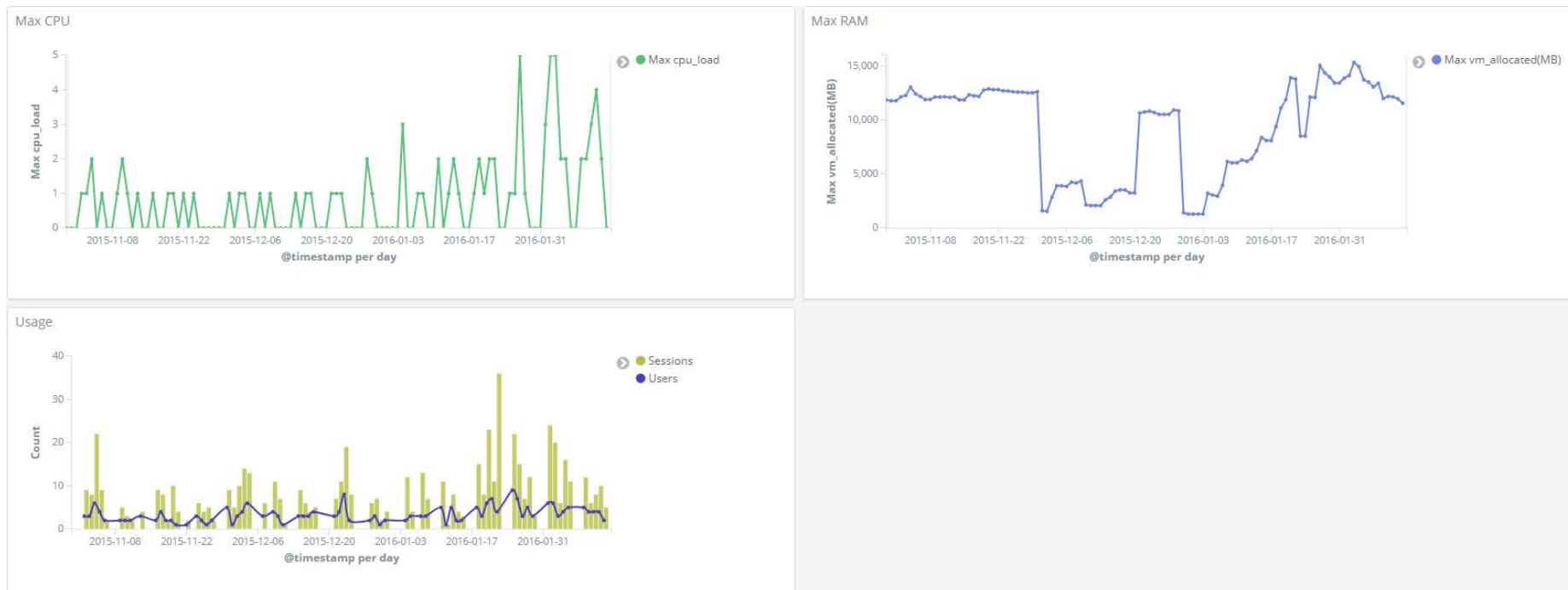


Figura 71. Mètriques del servidor

Informació: Consum màxim de CPU i RAM utilitzada diàriament pel servidor QlikView. Quantes sessions i usuaris únics hi ha hagut diàriament al servidor

Millors:

- Els recursos del servidor estan saturats? Potser caldria actualitzar el hardware o escalar.
- Per què hi ha hagut un pic en les sessions el dia X? Potser va haver-hi algun problema i els usuaris van haver de reiniciar les sessions.

usage_docs

Doc	Sessions	Unique Users	Average Selections	Avg Duration (s)	Avg Duration (m)	MBytes Sent + Received	Session Calls	Last Accessed
C:\QLIKVIEW\PROJECT\GENERAL\QIA.qvw	170	11	5.735	2,184.071	36.401	116,657	13,464	February 12th 2016, 12:19:29.000
C:\QLIKVIEW\PROJECT\CONTROL DWH\TIL_ST.qvw	136	19	7.846	2,685.934	44.766	63,753	7,662	February 11th 2016, 14:22:08.000
C:\QLIKVIEW\PROJECT\FARMACIA\WAT_Sagessa_Farmacia.qvw	105	21	10.648	2,658.219	44.304	142,124	14,125	February 11th 2016, 09:26:17.000
C:\QLIKVIEW\PROJECT\URGENCIES\Quadre Urgencies.qvw	85	12	10.106	2,410.894	40.182	115,227	12,253	February 11th 2016, 14:32:01.000
C:\QLIKVIEW\DEVELOPMENT\DEVELOPINGAPPS\AMBULATORIES\Ambulatories.qvw	41	8	6.39	2,189.049	36.484	75,504	4,947	February 12th 2016, 12:53:21.000
C:\QLIKVIEW\PROJECT\CONTROL DWH\BICTR.qvw	38	1	2.921	1,807.421	30.124	6,777	2,494	February 11th 2016, 10:10:00.000
C:\QLIKVIEW\DEVELOPMENT\DEVELOPINGAPPS\AMBULATORIES\Ambulatories_19012016.qvw	35	9	3.057	2,639	43.983	64,981	3,719	February 1st 2016, 10:15:16.000
C:\QLIKVIEW\PROJECT\URGENCIES\WAT_Sagessa_Urgencies.qvw	3	2	18	2,158.333	35.972	4,757	526	November 3rd 2015, 17:13:02.000

Export: [Raw](#) [Formatted](#)

Figura 72. Ús dels fitxers

Informació: Informació sobre l'ús donat pels usuaris dels fitxers del servidor

Milliores:

- El fitxer X fa molt que no s'utilitza. Potser ja no cal tenir-ho.
- El fitxer X s'utilitza molt i durant molt de temps. Potser s'hauria de fer una còpia de seguretat d'aquest fitxer per si de cas, ja que sembla important.

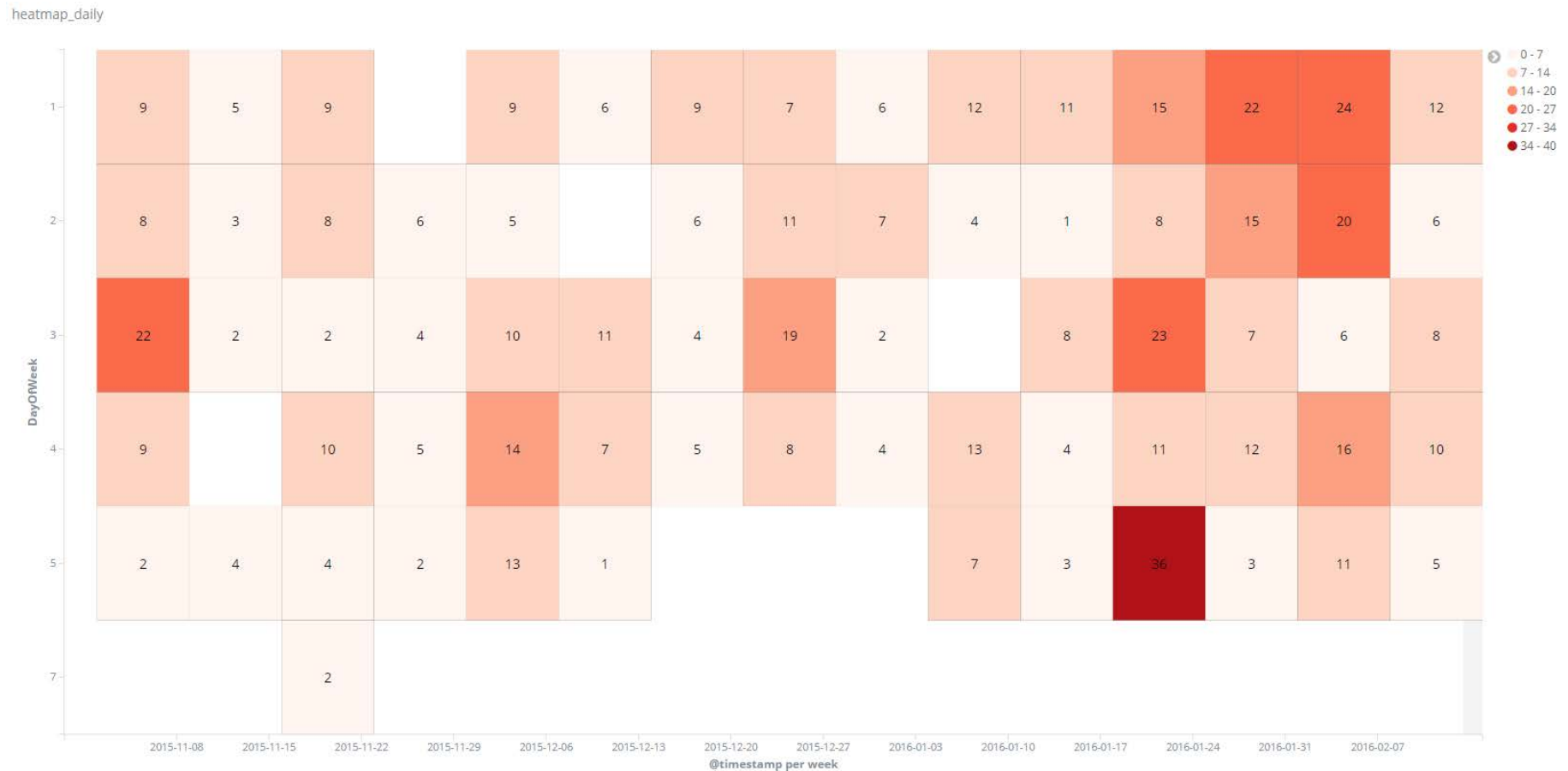


Figura 73. Heatmap amb les sessions per dia

Informació: Nombre de sessions per dia de la setmana per setmanes.

Millores:

- Hi ha un ús constant del servidor cada dilluns. Potser treball acumulat per dia festiu? Podríem posar el servidor en mode de consum energètic alt només els dilluns per atendre millor les peticions dels usuaris. Mentre que els diumenges ho ficarem en mode estalvi.

usage_user

User	Sessions	Unique Documents	Average Selections	Avg Duration (s)	Avg Duration (m)	MBytes Sent + Received	Session Calls	Last Accessed
GRUPSGS\mgonzale	210	8	8.086	1,966.976	32.783	196,097	19,456	February 12th 2016, 12:53:21.000
GRUPSGS\legil	86	6	4.36	2,090.244	34.837	50,115	4,757	February 10th 2016, 18:36:29.000
GRUPSGS\19989	78	7	8.654	2,415.346	40.256	70,014	9,776	February 8th 2016, 11:43:10.000
GRUPSGS\oros	49	6	4	2,611.796	43.53	31,339	3,074	February 12th 2016, 10:56:46.000
GRUPSGS\rrmtomas	25	5	6	2,447.24	40.787	41,078	3,676	February 9th 2016, 10:39:33.000
GRUPSGS\17797	21	2	24.238	3,316.429	55.274	46,872	5,268	February 11th 2016, 09:26:17.000
GRUPSGS\lmontoya	19	5	4.579	3,162.263	52.704	39,463	2,216	February 8th 2016, 14:06:16.000
GRUPSGS\21051	17	2	14.176	2,375.412	39.59	32,786	3,440	February 11th 2016, 14:32:01.000
GRUPSGS\alatorre	14	2	7.571	2,372.286	39.538	6,778	848	February 9th 2016, 19:02:49.000
GRUPSGS\Administrador	12	2	0.833	494.5	8.242	1,485	151	February 5th 2016, 15:39:29.000

Export: [Raw](#) [Formatted](#)

1 2 3 »

Figura 74. Estadístiques dels usuaris

Informació: Resum de l'activitat dels usuaris al servidor

Millores:

- L'usuari X quasi no interactua amb el servidor. Hi hauria d'interactuar per fer les seves tasques com treballador de l'empresa. Potser caldria demanar una reunió per aclarir aquest fet.

6.3 Tractament i Anàlisi d'Incidències

Per comprovar la qualitat del classificador, cal comprovar la qualitat de l'algorisme que s'ha dissenyat. Cal indicar que la qualitat dels resultats depèn també de les paraules claus (*keywords*) que s'utilitzin per a etiquetar les incidències.

Les incidències que s'utilitzen en aquest apartat són només en català i prèviament etiquetades manualment pel personal del servei d'informàtica.

6.3.1 Qualitat de l'Algorisme amb Dades Controlades

Donat que hi pot haver incidències mal etiquetades en el conjunt inicial (errors humans) obtingut des de l'hospital, per comprovar que l'algorisme funciona correctament hem de reduir el nombre d'incidències a un subconjunt de categories controlades i validades. Per això farem el següent:

- Escollir 20 incidències de cada categoria arrel definida a **tags.test.json**.
- Revisar que cada incidència estigui categoritzada correctament pel camp CLASSE.
- Guardar aquestes incidències a **Incidents.test.json** i indexar-les
- Calcular F1 score per cada categoria arrel.

La qualitat del nostre algorisme serà validada contrastant la categoria obtinguda pel classificador, amb el camp CLASSE del document en qüestió:

- Vertader positiu (TP): Si el TAG1 coincideix amb la CLASSE
- Fals positiu (FP): Si el TAG1 no coincideix amb la CLASSE

Un cop executat el procés de classificació, podem calcular la precisió i el *recall* del nostre algorisme per cada categoria i calcular la mètrica F1 que representa la qualitat dels resultats (definit a p. 28).

A continuació presentem els *tags* i *keywords* de prova:

```
{
  "tags": [
    {
      "Alta Usuari": "alta usuari accedir compte account
contrasenya pass password"
    },
    {
      "HN_Traducció": "traduir traducció idioma",
      "tags": [
        {
          "castellà": "castellà"
        },
        {
          "català": "català"
        }
      ]
    }
  ],
}
```

```

    {
      "HW_PC": "ordinador pc ventilador soroll ratolí teclat driver
equip torre"
    },
    {
      "HW_PRN": "impressora imprimir fotocopiar escàner",
      "tags": [
        {
          "Tinta": "tinta"
        },
        {
          "Tóner": "tóner"
        },
        {
          "Cartutx": "cartutx"
        },
        {
          "Paper": "paper"
        },
        {
          "Epson": "epson"
        },
        {
          "HP": "hp"
        }
      ]
    },
    {
      "SW_antivirus": "antivirus virus worm trojan ransomware
spyware infectat tallafocs seguretat alerta intrusió hack",
      "tags": [
        {
          "McAfee": "McAfee"
        },
        {
          "Trend Micro": "Trend Micro"
        }
      ]
    },
    {
      "SW_Internet": "www com cat org net pàgina plana web connexió
internet navegador navegar browser cookies facebook dns gencat",
      "tags": [
        {
          "Google": "Google",
          "tags": [
            {
              "Google Chrome": "Chrome"
            }
          ]
        },
        {
          "Internet Explorer": "Explorer"
        },
        {
          "Mozilla Firefox": "Mozilla Firefox"
        }
      ]
    }
  ]
}

```

```

        }
    ],
    },
    {
        "SW_Email": "adreça correu missatge safata bústia mail email
pop3 smtp",
        "tags" : [
            {
                "Outlook": "Outlook"
            },
            {
                "Exchange": "Exchange"
            },
            {
                "Gmail": "gmail"
            },
            {
                "Thunderbird": "thunderbird"
            }
        ]
    },
    {
        "SW_Pacs": "synapse imatge visualització RX mamografia
radiografia radiologia ecografia x-ray raigs pacs infopacs"
    }
]
}

```

Per aquesta prova cal incloure les següents variables globals a **classifier.py**:

```

tags_file = "tags.test.json"

total = {
    "Alta Usuari": 20,
    "HN_Traducció": 5,
    "HW_PC": 20,
    "HW_PRN": 20,
    "SW_antivirus": 20,
    "SW_Internet": 20,
    "SW_Email": 20,
    "SW_Pacs": 20
}

all = 145

```

Es necessita saber quants documents de cada categoria hi ha a l'índex i la suma total d'aquestes, per calcular les mètriques posteriors. Es declaren només les categories principals de **tags*.json** per així evitar una absorció de categories no definides, evitant mètriques distorsionades per l'increment de falsos positius.

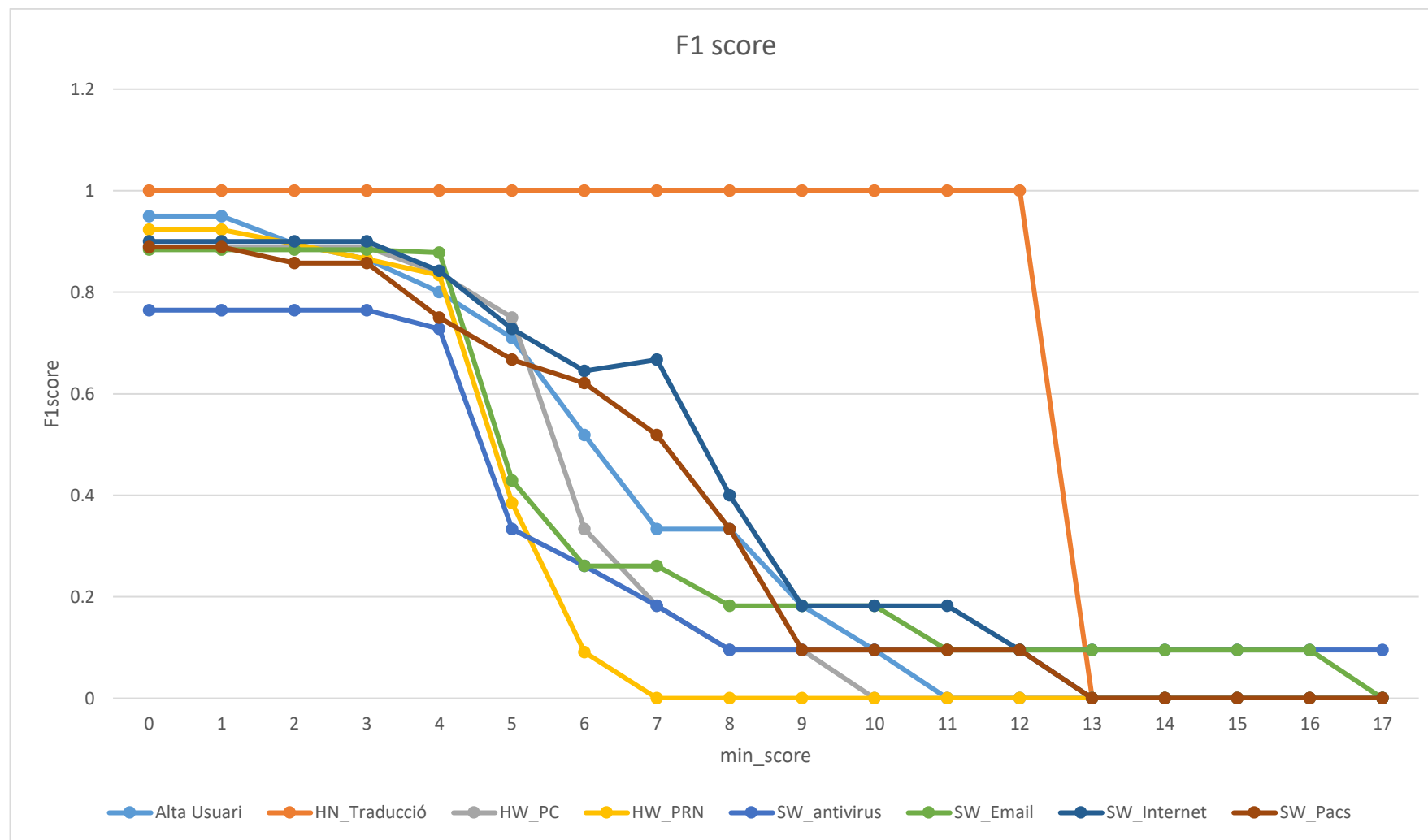


Figura 75. Resultats Incidents.test.json

La qualitat F1 score es mostra en funció del min_score. Cada document retornat per Elasticsearch, després de fer la cerca en *classifier.py*, conté una puntuació en el camp *_score* indicant com de bé s'ajusta el document a la cerca. Si volem ser més estrictes i tenir en compte només documents que superin un cert llindar de similitud, deixem de banda els documents que tinguin *_score* < min_score. El fet de descartar documents segons min_score, fa que *precision* i *recall* variïn, propagant així el canvi a F1 score.

A vegades pot ser d'interès acceptar documents que només superin un cert llindar de similitud. Segons la figura 73, el nostre algorisme seria útil amb un rang min_score entre 0 i 4. A partir de 5, la qualitat del classificador cau dràsticament. Valors més alts que 8, pràcticament no tindrien sentit. També veiem que la categoria HN_Traducció es comporta de forma diferent perquè té molt pocs documents a la BD, per tant presenta resultats menys fiables, ja que el sistema etiqueta sempre correctament aquests 5 documents, per tant, sembla que en aquest cas sigui més fàcil identificar-la que en els altres.

La propera taula mostra detalls sobre la rellevància dels documents retornats amb min_score = 3. Se selecciona aquest valor perquè representa un bon compromís entre llindar de similitud i qualitat de classificació:

<i>Categories</i>	<i>Vertader Positiu</i>	<i>Fals Positiu</i>	<i>Vertader Negatiu</i>	<i>Fals Negatiu</i>	<i>Recall</i>	<i>Precision</i>	<i>F1 score</i>
<i>Alta Usuari</i>	16	1	124	4	0.8	0.94	0.86
<i>HN_Traducció</i>	5	0	140	0	1	1	1
<i>HW_PC</i>	20	5	120	0	1	0.8	0.89
<i>HW_PRN</i>	16	1	124	4	0.8	0.94	0.86
<i>SW_antivirus</i>	13	1	124	7	0.65	0.93	0.76
<i>SW_Email</i>	19	4	121	1	0.95	0.83	0.88
<i>SW_Internet</i>	18	2	123	2	0.9	0.9	0.9
<i>SW_Pacs</i>	15	0	125	5	0.75	1	0.86

Taula 3. Resultats Incidents.test.json amb min_score=3

Per millorar la qualitat de l'algorisme, es poden fer proves amb el codi del classificador, o es pot millorar la qualitat dels *keywords* a *tags.test.json*. Aprofitant el potencial de Kibana, podem investigar *keywords* per incorporar-les al fitxer de *tags*.

Per exemple, a la taula 3 es pot observar que hi ha més falsos negatius que falsos positius. Si volem investigar els falsos negatius, només hem de cercar els documents que compleixin les següents condicions:

- CLASSE = categoria a investigar
- TAG1 = diferent de la categoria a investigar

Si volem investigar els 7 falsos negatius de SW_antivirus, podem aplicar 2 filtres amb les condicions esmentades en la pestanya *Discover* de Kibana:

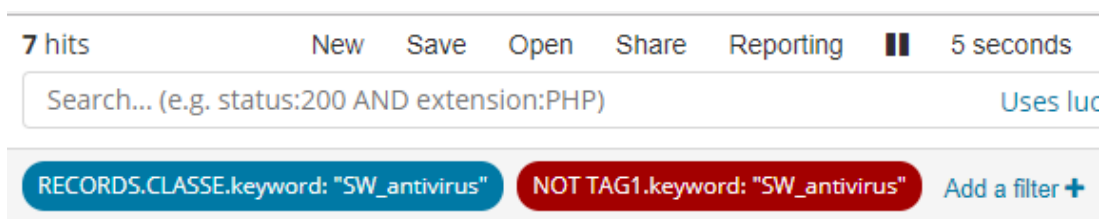


Figura 76. Filtres barra superior

Primer de tot, podem veure amb quines categories SW_antivirus entra en conflicte:

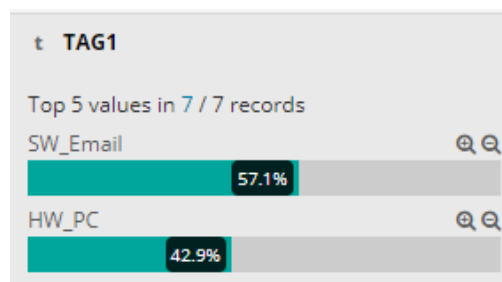


Figura 77. Filtres barra lateral

Inspeccionant els documents manualment, efectivament, arribem a la conclusió que hi ha conflictes pels següents motius:

SW_antivirus vs SW_Email:

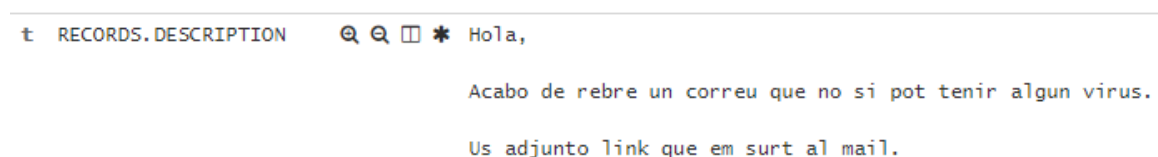


Figura 78. Exemple de conflicte

Es tracta d'usuaris preocupats perquè han rebut un e-mail sospitós de virus. Ara bé, es tracta d'un problema de correu o de virus? Els e-mails infectats estan més relacionats amb problemes de seguretat que amb problemes d'e-mails? Això només ho pot decidir el gestor, ja que com expert i usuari del sistema DSS, el seu criteri seria el correcte. No obstant això, a partir del camp CLASSE, podem deduir que els e-mails infectats formen part de la categoria SW_antivirus, potser perquè els antivirus són els encarregats de bloquejar-los.

Una forma senzilla de solucionar aquests falsos negatius, és donant més pes a la categoria SW_antivirus:

```
{
    "SW_antivirus": "antivirus virus worm trojan ransomware
spyware infectat tallafocs seguretat alerta intrusió hack",
    "tags": [
        {
            "McAfee": "McAfee"
        },
        {
            "Trend Micro": "Trend Micro"
        },
        {
            "SW_antivirus": "antivirus virus worm trojan
ransomware spyware infectat tallafocs seguretat alerta intrusió hack"
        }
    ]
},
```

Amb aquesta modificació en *tags.test.json*, li diem al classificador que preferim l'etiqueta SW_antivirus sobre les categories arrels, ja que el classificador proporciona un pes major com major sigui la distància de la subcategoria al node arrel. Tenint en compte la següent línia de *classifier.py*:

```
"boost": "" + str(lvl) + "",    //(Per al camp DESCRIPTION)
"boost": "" + str(lvl * 3) + "", //(Per al camp TITLE)
```

Ara els *keywords* de SW_antivirus del segon nivell, col·laboraran en els resultats amb les mateixes *keywords* del nivell arrel:

```
"boost": 2    //(Per al camp DESCRIPTION)
"boost": 5    //(Per al camp TITLE)
```

Així doncs, tornem a classificar per obtenir els següents resultats:

<i>Categories</i>	<i>TP</i>	<i>FP</i>	<i>TN</i>	<i>FN</i>	<i>F1 score</i> <i>nou</i>	<i>F1 score</i> <i>anterior</i>
<i>Alta Usuari</i>	16	1	124	4	0.86	0.86
<i>HN_Traducció</i>	5	0	140	0	1	1
<i>HW_PC</i>	20	3	122	0	0.93	0.89
<i>HW_PRN</i>	16	1	124	4	0.86	0.86
<i>SW_antivirus</i>	17	1	124	3	0.89	0.76
<i>SW_Email</i>	19	2	123	1	0.93	0.88
<i>SW_Internet</i>	18	2	123	2	0.9	0.9
<i>SW_Pacs</i>	15	0	125	5	0.86	0.86

Taula 4. Resultats després de millora

La millora en HW_PC, SW_antivirus i SW_Email era d'esperar, ja que el fet de donar preferència a SW_antivirus ha disminuït el nombre de falsos negatius i ha disminuït el nombre de falsos positius en HW_PC i SW_Email, millorant així el F1 score de tots tres. El fet de donar més preferència a una categoria pot comportar efectes negatius col·laterals. Però en aquest cas SW_antivirus realment tenia més preferència, resolent així els conflictes.

Aquest ha sigut un exemple satisfactori de com millorar la qualitat de la classificació a partir de canvis en *tags.test.json*. Hem pogut introduir aquesta millora gràcies a tractar un conjunt reduït de dades validades. Per estadística, és possible que les millores introduïdes en una mostra de la població, siguin extrapolables al conjunt sencer.

A banda de possibles millores, els resultats de les figures i taules d'aquest apartat demostren que el classificador té bona qualitat perquè els valors F1 ronden 0.9.

6.3.2 Qualitat de l'Algorisme amb Dades no Controlades

Per comprovar si els resultats anteriors són vàlids també per la resta de registres d'incidències que tenim, comprovem els resultats a gran escala. Ara els passos seran els següents:

- Indexar totes les incidències d'**Incidents.json**
- Utilitzar una definició de categories més ampla **tags.json**. Aquest fitxer conté fins a 3 nivells de profunditat.
- Calcular F1 score.
- Demostrar inserció de *tags*.

En aquest cas estem informats que les categories assignades al camp CLASSE poden ser errònies perquè s'han assignat de forma manual i sense revisar. Tot i això, es creu que els errors són poc freqüents i per tant, els resultats de les mètriques d'avaluació poden ser considerats com prou bons.

Igual que abans, s'han de definir algunes variables globals a *classifier.py*:

```
tags_file = "tags.json"

total = {
    "HW_PRN": 712,
    "SW_Email": 610,
    "HN_HC3-RE": 536,
    "HW_PC": 477,
    "SW_Genomi": 405,
    "SW_antivirus": 291,
    "ST_Storage": 260,
    "SW_Ofimatica": 244,
    "SW_Hnet": 239,
    "SW_ECAP": 219,
    "Alta Usuari": 148,
    "SW_Omifas": 122,
    "SW_Pacs": 121,
    "HW_Telefonia": 120,
    "SW_VDI": 115,
    "SW_Internet": 88,
    "SW_Omiap": 75,
    "HW_TFT": 43,
    "HW_Electromedicina": 38,
    "HN_Traducció": 4
}

all = 0
for v in total.values():
    all += v
```

Aquest cop es tindran en compte totes les 7442 incidències de l'índex *incidents*.

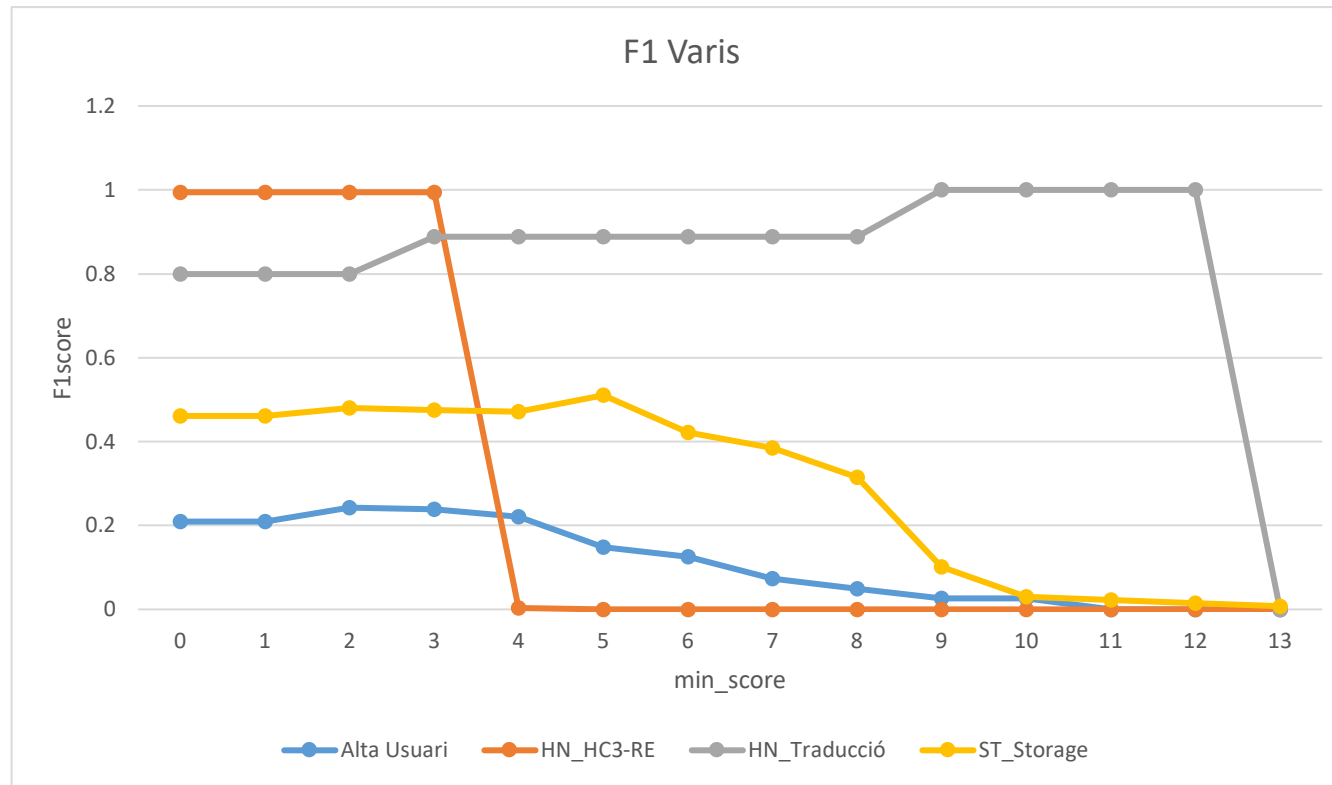


Figura 79. Resultats amb Incidents.json (1)

Observem un comportament força diferent en aquestes 4 categories. HN_HC3-RE funciona perfecte mentre min_score sigui menor que 4, perquè els seus documents són automàtics. Després passa a funcionar molt malament perquè 3 és l'score màxim que pot donar de si el *keyword* “HC3” en els documents generats automàticament. El cas de HN_Traducció és també curiós perquè augmenta el seu F1score. Això passa perquè en cap dels casos té falsos negatius, i per tant, en augmentar el llindar, redueix els falsos positius i millora el seu indicador global. En la resta de casos veurem que F1 sol decreix. Podem veure que amb un llindar de 3, aquesta categoria funciona ja prou bé (F1=0.8). Finalment, hi ha dues categories que tenen un valor de qualitat molt baix. En aquests casos no hem aconseguit classificar de forma correcta. Cal fer un estudi dels *keywords* d'aquestes categories o de la mena de frases que s'escriuen quan es vol una alta d'usuari o un es registra un esdeveniment ST_storage.

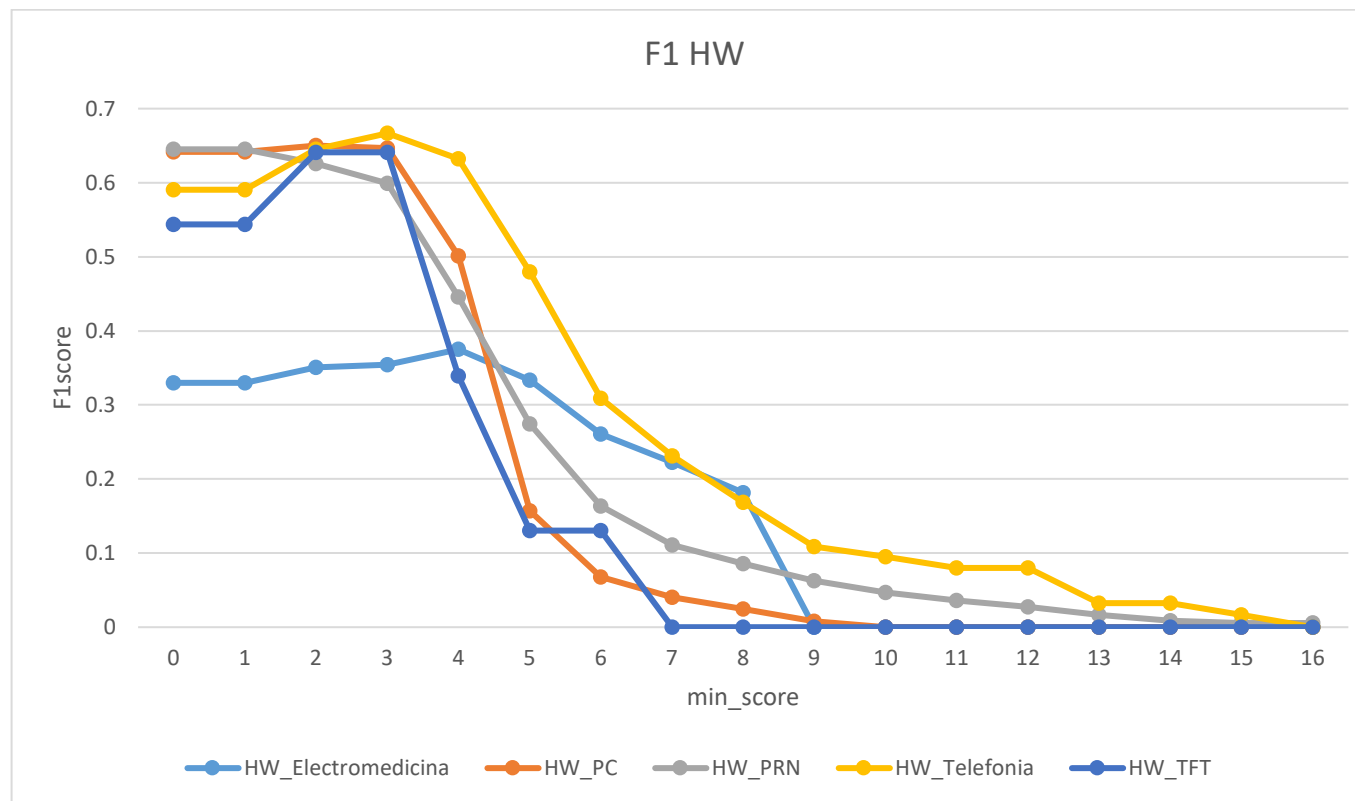


Figura 80. Resultats amb Incidents.json (2)

Les classificacions HW_PC, HW_PRN, HW_Telefonia i HW_TFT són bastant acceptables i tenen un comportament similar. Això pot ser degut al domini lingüístic hardware que utilitzen, específic de cada categoria. HW_Electromedicina es manté més estable però amb una qualitat baixa.

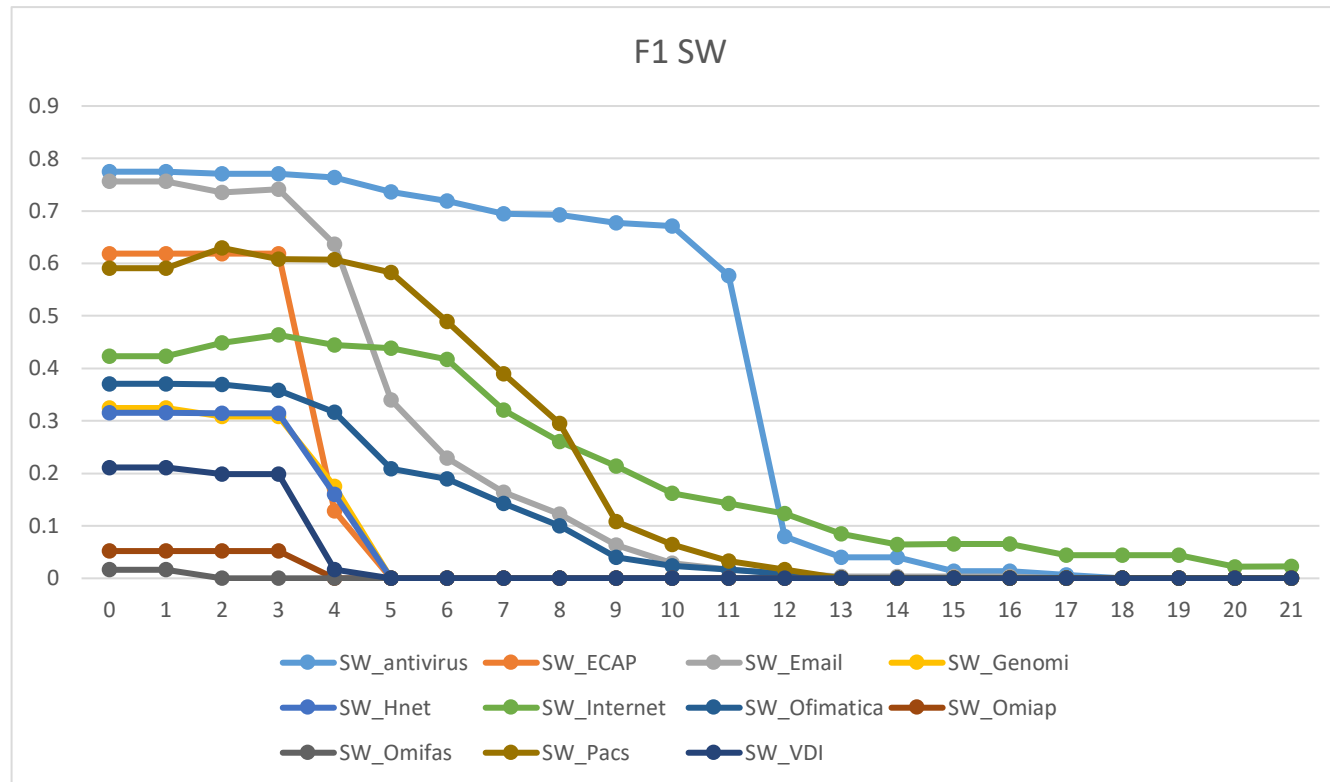


Figura 81. Resultats amb Incidents.json (3)

En aquest gràfic podem agrupar les categories en:

- Qualitat notable: SW_antivirus, SW_Email
- Qualitat acceptable: SW_ECAP, SW_Pacs
- Qualitat dolenta: SW_Internet, SW_Ofimatica, SW_Genomi, SW_Hnet, SW_VDI
- Qualitat nefasta: SW_Omiap, SW_Omifas

La tendència confirma que com menys detallat o específic siguin les categories, pitjor resultats obtenim. En les gràfiques F1 de la secció 6.3 podem veure com a regla general que $\text{min_score} = 4$ és un punt d'inflexió per l'algorisme que proposem.

La millor manera de representar els *tags* afegits als documents d'Elasticsearch, és precisament fent ús de la visualització de tipus *tag cloud*:

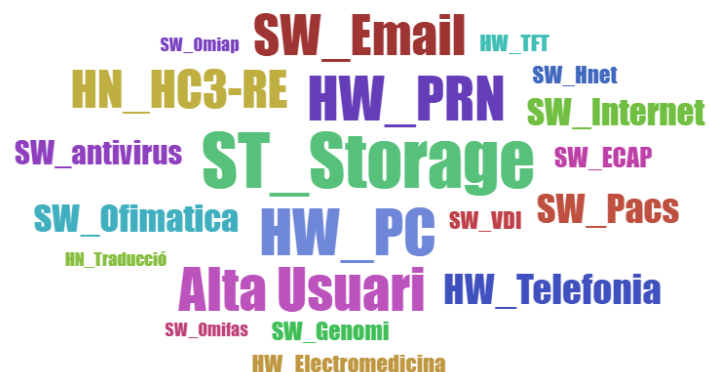


Figura 82. Tag Cloud TAG1

La mida dels tags representen la seva importància relativa als altres tags, segons el nombre de documents que té cada *tag*.

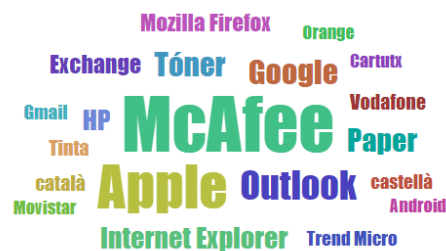


Figura 83. Tag Cloud TAG2

Com que només hi ha el tag “Google Chorme” definit al nivell 3, s’omet la visualització del camp TAG3.

7. Conclusions i Treball Futur

7.1 Entorn de Desenvolupament

Amb les figures de l'apartat 6.1, es demostra la tolerància a particions del clúster simulat, la seguretat de les dades i la gran quantitat d'informació que podem obtenir sobre Elastic Stack i els components que el formen, en quasi temps real. Aquesta informació pot ser molt útil a l'hora d'optimitzar el rendiment dels servidors que formen Elastic Stack per obtenir un DSS més ràpid i estable.

7.2 Tractament de Fitxers de Log

Tenir documents correctament indexats en Elasticsearch és suficient per poder utilitzar Kibana una vegada connectat a algun node del clúster.

Amb les funcions predefinides podem preparar nombrosos tipus de visualitzacions de dades estructurades per obtenir la informació que vulguem de forma instantània. Kibana també permet personalitzar visualitzacions posant més esforç. Pels desenvolupadors, Kibana ofereix unes eines molt pràctiques com ara una consola per comunicar-se amb Elasticsearch de forma més senzilla que *curl*, o *grok debugger* per fer proves de les definicions de *pipelines*.

7.3 Tractament i Anàlisi d'Incidències

Pel que hem vist a la secció 6.3, el classificador és capaç de realitzar bones classificacions si les categories a *tags.json* estan ben definides.

El balanç entre *precision* i *recall* no es pot evitar. L'únic que es pot fer és trobar l'algorisme o el mètode perquè el gap entre tots dos sigui molt petit sense *min_score*, i que *recall* i *precision* tinguin valors alts.

La solució proposada és una de les moltes que es podien haver implementat, sobretot amb l'àmplia gamma de solucions que Elasticsearch ofereix per a l'anàlisi de text. Podríem provar manualment moltes configuracions i definicions de categories per veure quina dóna millors resultats, però la casuística és tan gran que el següent pas seria utilitzar tècniques de Machine Learning per trobar la configuració més òptima per cada categoria que volguéssim trobar.

És a dir, el fet de definir les categories en *tags.json* per classificar els documents estalvia molta feina, ja que ningú ha de classificar-los a mà. Però l'altra forma de solucionar el problema seria partir d'un *dataset* molt gran amb categories correctes on s'hauria invertit molt de temps classificant el *dataset* manualment per després aplicar tècniques de Machine Learning o el mètode descrit a <https://www.elastic.co/blog/text-classification-made-easy-with-elasticsearch>. Per una banda, el temps de classificació del *dataset* seria menor si moltes persones el realitzessin paral·lelament. Per altra banda el nostre classificador continua sent més flexible que aquest mètode perquè només caldria canviar 4 línies, però potser mai arribaríem al nivell d'optimització aconseguit per les tècniques amb *datasets*.

En el cas d'aquest projecte, cal algú expert en el domini per definir les categories. En general, en la construcció de tot sistema de suport a la decisió cal coneixement precís i ampli del problema per tal que les eines automàtiques puguin extreure coneixement de les dades. En aquest projecte ha estat molt útil l'aportació del Sr. González, que com a cap del servei d'informàtica de l'Hospital de Reus ens ha pogut assessorar i guiar en el procés de definició de les eines de classificació dels documents.

El sistema de classificació que s'ha dissenyat en aquest treball fi de grau és molt flexible. En comptes de definir només categories de les incidències, podríem definir el tipus de document com per exemple incidència, petició, queixa. O fins i tot afegir categories relacionades amb paraules que fan referència a sentiments. El fet de tenir-ho estructurat en una taxonomia que es guarda en un fitxer JSON, fa que es puguin afegir fàcilment altres conceptes en qualsevol nivell per a que automàticament el programa els tingui en compte quan tracti nous documents que arribin al sistema.

Gràcies a Elasticsearch obtenim un anàlisi automàtic una vegada configurat l'analitzador al *template*. Desgraciadament no hem aconseguit que el filtre per idioma i la indexació de les accions siguin automàtiques.

7.4 Valoració Personal del TFG

Aquest projecte m'ha servit per completar la carrera amb coneixement sobre *Big Data* i *Text Mining* i aprendre com a mínim els seus conceptes bàsics. La part més difícil del projecte ha sigut aprendre a dominar Elastic Stack des de zero i utilitzar aquest coneixement per acabar el projecte en gairebé tres mesos. La part que més m'ha agradat ha sigut el fet de manipular *Big Data* real i adonar-me de les enormes possibilitats d'extreure diferents tipus d'informació útil.

8. Recursos Utilitzats

<https://www.udemy.com/elasticsearch-6-and-elastic-stack-in-depth-and-hands-on/>

<https://www.elastic.co/webinars/gdpr-compliance-and-elasticsearch>

Elasticsearch: The Definitive Guide 2.X

<https://urv.on.worldcat.org/discovery>

Elasticsearch Reference 6.2

Ordinador de taula personal

<https://www.draw.io/>

Elastic Stack 6.2.4

Windows 10

VirtualBox

Debian

Acrònims

NLP:	Natural Language Processing
NoSQL:	Not only SQL / No SQL
RAM:	Random Access Memory
CPU:	Central Processing Unit
JSON:	JavaScript Object Notation
SQL:	Structured Query Language
ACID:	Atomicity Consistency Isolation Durability
DSS:	Decision Support System
BASE:	Basic Availability Soft-state Eventual consistency
BD:	Base de Dades
IA:	Intel·ligència Aritficial
GDPR:	General Data Protection Regulation
OOP:	Object-Oriented Programming
ES:	Elasticsearch
CRUD:	Create Read Update Delete
I/O:	Input Output
HTTP:	Hypertext Transfer Protocol
REST:	Representational State Transfer
API:	Application Programming Interface
DFR:	Divergence From Randomness model
DFI:	Divergence From Independence model
IB:	Information Based model
TF/IDF:	Term Frequency Inverse Document Frequency
DSL:	Domain Specific Language
SMP:	Symmetric Multi-Processing
APU:	Accelerated Processing Unit
DDR3:	Double Data Rate 3
GB:	Gigabyte
MB:	Megabyte
SO:	Sistema Operatiu
DNS:	Domain Name System
DHCP:	Dynamic Host Configuration Protocol
MAC:	Media Access Control
IP:	Internet Protocol
JVM:	Java Virtual Machine
YAML:	YAML Ain't Markup Language
UUID:	Universally Unique Identifier
NAT:	Network Address Translation

DDNS: Dynamic DNS
SSL: Secure Sockets Layer
TCP: Transmission Control Protocol
URL: Uniform Resource Location
CA: Certification Authority
SAN: Subject Alternative Name

Referències¹¹

- [1] Rajkumar Buyya, Rodrigo N. Calheiros, Amir Vahid Dastjerdi, Big Data Principles and Paradigms, Morgan Kauffman, 2016
- [2] Hrushikesha Mohanty, Prachet Bhuyan, Deepak Chenthati, Big Data A Primer, Springer India, 2015
- [3] Stuart J. Russell, Peter Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, 2009
- [4] <https://help.qlik.com/es-ES/qlikview/November2017/Content/what-is.htm>
- [5] Luis Joyanes Aguilar, Big Data. Análisis de grandes volúmenes de datos en organizaciones. S.A Marcombo, 2014
- [6] <https://www.udemy.com/elasticsearch-6-and-elastic-stack-in-depth-and-hands-on/>
- [7] <https://db-engines.com/en/ranking/search+engine>
- [8] <https://www.elastic.co/products>
- [9] https://es.wikipedia.org/wiki/Reglamento_General_de_Protecci%C3%B3n_de_Datos
- [10] <https://www.elastic.co/webinars/gdpr-compliance-and-elasticsearch>
- [11] Ralph Stair, George Reynolds, Principles of Information Systems, Cengage Learning, 2016
- [12] <https://www.elastic.co/guide/en/elasticsearch/guide/current/document.html>
- [13] https://www.elastic.co/guide/en/elasticsearch/guide/current/_scale_horizontally.html
- [14] https://www.elastic.co/guide/en/elasticsearch/guide/current/_coping_with_failure.html
- [15] <https://www.elastic.co/guide/en/elasticsearch/guide/current/inverted-index.html>
- [16] <https://www.elastic.co/guide/en/elasticsearch/reference/current/index-modules-similarity.html>
- [17] <https://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-unranked-retrieval-sets-1.html>
- [18] <http://www.chioka.in/differences-between-roc-auc-and-pr-auc/>
- [19] <https://www.elastic.co/guide/en/logstash/current/first-event.html>
- [20] <https://www.elastic.co/guide/en/beats/metricbeat/current/view-kibana-dashboards.html>
- [21] <http://www.oracle.com/technetwork/articles/servers-storage-admin/f2-1907150.gif>
- [22] <https://conceptdraw.com/a1778c3/p1/preview/640/pict--router-cisco-routers---vector-stencils-library.png--diagram-flowchart-example.png>
- [23] <http://simpleicon.com/wp-content/uploads/computer-5.png>
- [24] <https://www.elastic.co/subscriptions>
- [25] <https://www.elastic.co/guide/en/elasticsearch/reference/6.2/update-license.html>
- [26] https://serverfault.com/questions/845766/generating-a-self-signed-cert-with-openssl-that-works-in-chrome-58?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa
- [27] <https://kemptechnologies.com/blog/reverse-proxy-for-skype-for-business/>
- [28] https://httpd.apache.org/docs/2.4/howto/reverse_proxy.html
- [29] <http://unicode.org/reports/tr29/>
- [30] <https://github.com/apache/lucene-solr/blob/master/lucene/analysis/common/src/resources/org/apache/lucene/analysis/ca/stopwords.txt>

¹¹ Links comprovats 05/06/2018