

Grado en Ingeniería de computadores

# Trabajo Fin de Grado



Autor: Álvaro Gallego Lucendo

Tutor: Salvador Otón Tortosa

**2019**

**Desarrollo de Aplicaciones con Elastic Search**

# **GRADO EN INGENIERÍA DE COMPUTADORES**

Trabajo Fin de Grado

Desarrollo de aplicaciones con Elastic Search

**Autor:** Álvaro gallego Lucendo

**Tutor/es:** Salvador Otón Tortosa

**TRIBUNAL:**

**Presidente:**

**Vocal 1º:**

**Vocal 2º:**

**FECHA:**



# Índice

1. Resumen.....	4
2. Resume.....	5
3. Palabras clave / Keywords .....	6
4. Resumen Extendido .....	7
5. Marco teórico .....	9
5.1 Bases de datos .....	9
5.1.1 Orígenes de las bases de datos .....	9
5.1.2 Historia del almacenamiento.....	9
5.1.3 Tipos de bases de datos .....	14
5.2 ELK.....	22
5.2.1 Beats .....	23
5.2.2 Logstash .....	24
5.2.3 Elastic Search .....	25
5.2.4 Kibana .....	31
6. Aplicaciones de Elastic Search .....	33
7. Marco práctico .....	34
7.1 Simulación 1.....	34
7.1.1 Instalación y recogida de logs .....	34
7.1.2 Kibana .....	43
7.2 Simulación 2.....	52
Descarga y configuración.....	52
Querys .....	58
8. Costes .....	76
8.1 Coste software.....	76
8.2 Coste hardware.....	77
8.3 Coste humano.....	78
9. Conclusión.....	79
10. Bibliografía .....	80

## 1. Resumen

En este proyecto, nuestro objetivo principal será demostrar lo útil e interesante que puede llegar a ser la utilización de bases de datos.

Vamos a realizar un estado del arte de Elastic Search, explicando los orígenes de las bases de datos, desde los diferentes dispositivos de almacenamiento que surgieron y como se emplearon, ya que estos son los pilares de las bases de datos, hasta los diferentes usos que se les puede llegar a dar hoy en día.

Además, dentro de este mundo, comprobaremos que aplicaciones tiene Elastic Search, un motor de búsqueda basado en Java, con el cual podremos agilizar nuestras consultas sobre las bases de datos.

## **2. Resume**

In this project, our main objective will be to demonstrate how useful and interesting the use of databases can be.

We are going to make a state of the art Elastic Search, explaining the origins of databases, from the different storage devices that emerged and how they were used, as these are the pillars of databases, to the different uses that can be given today.

In addition, within this world, we will see that applications have Elastic Search, a search engine based on Java, with which we can streamline our queries about databases.

### 3. Palabras clave / Keywords

- **Node:**  
Equipo encargado de realizar una tarea.  
Computer in charge of carrying out a task.
- **Cluster:**  
Conjunto de nodos que realizan un trabajo específico.  
Set of nodes performing a specific job.
- **Elastic Stack:**  
Paquete de herramientas compuesto por Beats, Logstash, Elastic Search y Kibana.  
Toolkit consisting of Beats, Logstash, Elastic Search and Kibana.
- **Beats:**  
Módulo encargado de la recolección de logs.  
Module in charge of collecting logs.
- **Logstash:**  
Módulo avanzado encargado de recolectar y procesar logs.  
Advanced module in charge of collecting and processing logs.
- **Elastic Search:**  
Motor de búsqueda y tratamiento de logs.  
Search engine and log processing.
- **Kibana:**  
Interfaz gráfica para mostrar logs procesados.  
Graphical interface to display processed logs.

## 4. Resumen Extendido

Nuestro objetivo principal es hacer un estado del arte de Elastic Search, una tecnología que forma parte de ELK.

Explicaremos ampliamente como funcionan todos los módulos dentro de ELK, para que veamos que utilidad pueden tener cada uno de ellos, así como la parte en la que nos vamos a centrar nosotros, Elastic Search.

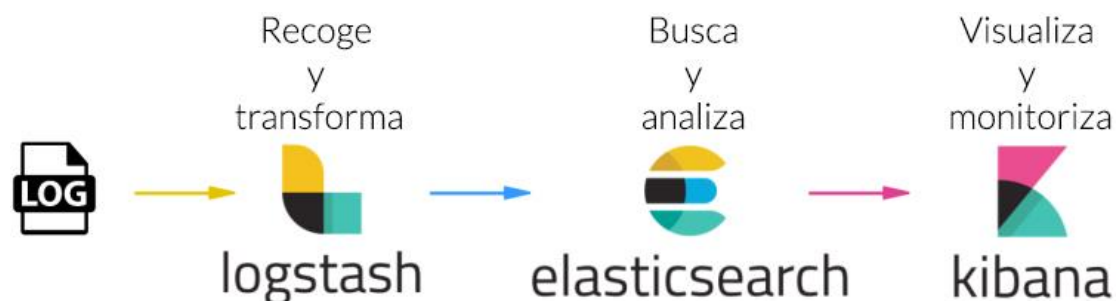
Como breve introducción, vamos a explicar estos ejemplos, aunque ya los desarrollaremos más adelante.

Dentro del ELK nos podemos encontrar 3 módulos: **Logstash**, una herramienta para administrar los logs de eventos y aplicaciones, recolectar y preprocesar información, además de poder guardarla para futuras aplicaciones.

Elastic Search, un servidor de búsqueda desarrollado en Java, con el que podemos obtener grandes ventajas a la hora de realizar búsquedas y análisis: la velocidad a la hora de recorrer datos, su compatibilidad con plataformas Java, su buena adaptación a sistemas distribuidos además del fácil acceso al software, ya que es open source.

Finalmente, la última parte de ELK será **Kibana**, la parte visual, donde tendremos los dashboard con la información que podremos procesar y analizar para futuros estudios o análisis, en los que podremos alojar las bases de datos que tengamos asociadas a Elastic Search.

Como resumen se adjunta esta imagen en la que se muestra perfectamente la unión de todos estos componentes.



*Figura 1 – Flujo de datos de ELK*

Dicho esto, en este proyecto vamos a centrarnos en varios objetivos fundamentales dentro de las bases de datos, a partir de los cuales, según vayamos explicando la diferente terminología, iremos desarrollando diferentes temas en función de las dudas que vayan surgiendo.

El primer objetivo será hacer un estado del arte de la tecnológica Elastic Search, mostrando que había antes de surgir este tipo de herramientas, que tipos de almacenamiento habían, como surgirán, que tipos de bases de datos nos encontraremos y para qué sirven.

También haremos un análisis de ventajas y desventajas, puesto que tenemos un amplio camino a investigar, ya que, al ser una tecnología de código abierto, tendrá multitud de usos y aplicaciones.



Por otro lado, tras la parte teórica, realizaremos un par de aplicaciones prácticas, para las que tenemos pensado realizar varias simulaciones, empleando diferentes módulos que nos proporciona Elastic Stack, comprobando como podemos recoger logs, como podemos procesarlos, representarlos con Kibana...

También comprobaremos como funciona la API de Elastic, viendo que plugin podemos conectar, como puede ser Excelastic y una parte importante será el análisis del lenguaje de programación, o QUERYS que podemos realizar con este tipo de tecnología.

## 5. Marco teórico

### 5.1 Bases de datos

#### 5.1.1 Orígenes de las bases de datos

El origen de las bases de datos es bastante lejano, para hacernos una idea, tendríamos que partir de la antigüedad, donde se almacenaba todo tipo de información en pergaminos y papel, en diversas bibliotecas.

El termino base de datos se escucharía por primera vez en 1963, en una Conferencia celebrada en California. Este término fue empleado para dar nombre a todo ese conjunto de información que podemos encontrar de forma ordenada, estructurada y a disposición de los usuarios.

Como tal, una base de datos es una colección de información, fácilmente accesible, estructurada y actualizada, normalmente bajo un sistema de gestión específico, capaz de realizar la comunicación entre información y usuario.

Pero ¿Dónde se almacena esta información, y que mecanismos se encargan de ello?

#### 5.1.2 Historia del almacenamiento

Antes de nada, trataremos de poner un contexto a todo, para que así las explicaciones sean más sencillas y podamos entender los orígenes del almacenamiento de la información.

Las bases de datos surgieron de la necesidad de almacenar la información, para poder disponer de ella de manera clasificada, ordenada y rápida. Siguiendo todos esos pasos, se consigue una mayor eficiencia y productividad, ya que, por ejemplo, realizar una consulta para obtener algún dato, o información concreta, se podía convertir en un incordio con los medios que se disponían en el momento.

Todo esto nos lleva a la conclusión de que era muy importante el factor espacio, así como su gestión, ya que durante mucho tiempo la única manera de almacenar la información era hacerlo mediante documentos en papel, cosa que era ineficiente, difícil de consultar y mantener, ya que con el tiempo su estado se degradaba.

El primer elemento que permitió poder almacenar archivos fue la **tarjeta perforada**, surgida en el año 1725, por Basille Bouchon y Jean-Baptiste.

En un su inicio, surgieron para llevar a cabo el control de los telares, aunque más tarde, en el siglo 20, (concretamente en 1890) Herman Hollerit las utilizaría como método de almacenamiento en las computadoras, teniendo como primer uso realizar el censo de los estados unidos en 1890.

Años después Herman fundaría la empresa Tabulating Machine Company, la cual más tarde sería una de las tres empresas que se unieron para formar la Computing Tabulating Recording Corporation, o lo que hoy conocemos como IBM.

Las tarjetas perforadas, eran cartulinas con pequeños orificios, que contenían información, representada mediante la presencia o ausencia de agujeros en posiciones concretas (código binario).

Años más tarde aparecerán las **cintas magnéticas**. Estas eran un medio de almacenamiento que se grababa en pistas sobre una banda de plástico, sobre un material magnético.

A pesar de que los principios de la grabación magnética surgieron en 1878, por Oberlin Smith, no sería hasta 1979 cuando se empezarían a emplear como almacenamiento de datos.

En 1949 Edvac sería el primer ordenador en utilizar este tipo de dispositivos como método de almacenamiento principal de datos, procesando un sistema binario, y con un sistema de grabación de cinta magnética.

Con esto fue posible trasladar la información de las tarjetas perforadas a las cintas magnéticas, suponiendo una revolución del sector industrial, científico, además de diferentes motivos gubernamentales.

En 1951 aparecería el primer ordenador personal, el Eckert-Mauchly, utilizando como método de grabación la cinta magnética.

Más tarde, en 1956, surge un nuevo dispositivo de almacenamiento, las unidades de **disco duro**, desarrolladas por IBM, concretamente el modelo 350.

Los discos duros podían almacenar menos 5 MB de información, con una velocidad de transferencia de 8.8 KBPS, teniendo como curiosidad su peso, ya que pesaban más de una tonelada (medía 1.52 metros de largo por 1.72 de alto).

Con los años irían apareciendo nuevos modelos, como en 1962, con el 1301, con una capacidad de almacenaje de 28 MB, con una velocidad de transferencia 10 veces superior al primer dispositivo surgido en 1956.

Y así continuó la evolución en los siguientes años, en 1965 surgiría el modelo 2310, añadiendo los primeros almacenamientos desmontables (primer disco flexible), en 1966 el 2314, con cabezales de lectura de ferrita o en 1973, donde surgiría el revolucionario Winchester 3340, destacado por su tamaño y peso, convirtiéndose en el nuevo estándar de almacenamiento de acceso directo, con 30 MB de capacidad.

El desarrollo continuará con el IBM 3380 y Seagate ST-506, en el año 1980.

Los ordenadores seguían en constante evolución y a su vez el software, ya que los programas requerían de cada vez más almacenamiento, por lo que el 3380 supuso un gran avance, ya que su capacidad era de 1GB.

A su vez, Seagate, o como en ese momento se llamaba, Shugart Technology, lanzaría el ST-506, el primer disco con el formato estándar 5.25, llegando el primer disco duro disponible no solo para empresas con un importante poder económico, sino para el público general y los ordenadores al uso, manteniéndose hasta la actualidad.

Esto convertiría a Seagate en uno de los fabricantes más importantes del mundo.

En 1967, la empresa de moda IBM desarrollaría un nuevo sistema: barato, sencillo y a la altura de todo tipo de usuarios, el **Disquete o floppy disk**.

El disquete sería lanzado de forma oficial en 1971, y serían los primeros discos de almacenamiento en volverse masivos, con una capacidad de 1.2 MB o 1.4 MB, dependiendo del tamaño.

Fueron muy populares y aclamados en su momento, en gran parte por la aparición de la industria de los videojuegos, especialmente Nintendo.

Pero con el tiempo apareció un problema, haría falta más almacenamiento.

Para solucionar este problema aparecerá el Compac disc, o como comúnmente denominamos, **CD-ROM**. Creados en 1979, pero lanzados de forma oficial en Japón el 1982, el CD surgía para resolver estos problemas de espacio, pudiendo almacenar entre 650-700 MB de información y pesando menos de 30 gramos, por lo que dejaron en desuso al disquete.

Tras él, en 1990 aparecerá el **DVD**, un dispositivo más avanzado con una capacidad de 4.7 GB, un método de almacenamiento que vendría a sustituir al cd.

Las dos propuestas desarrolladas fueron el Multimedia Compac disc (MMCD), desarrollado por Sony y Philips, y el Super Density Disc (SD), por Toshiba y Warner Home Entertainment, además de otras empresas. Tras disputas, el modelo que triunfaría sería el Super Density Disc.

Como curiosidad, los DVD fueron desarrollados para tener una capacidad de 5 GB, pero la implementación de la tecnología “push-pull”, (encargada de facilitar rápidamente la transición de escenas) y la adopción del sistema de Philips “EFMplus”, un sistema de codificación alternativo al que propuso Toshiba, el cual además añadía grandes resistencias a daños en el disco, disminuyeron su capacidad a 4.7 GB.

Entre los años 1995-1996 aparece un novedoso método de almacenamiento, las unidades flash, o memorias USB.

Las memorias flash eran derivados de las memorias EEPROM (Pequeñas memorias ROM programables), y eran pequeños dispositivos con gran capacidad de almacenaje, además de ser portátiles.

Serian y son muy importantes, ya que tienen un bajo consumo, son portables como ya hemos comentado y actualmente nos los podemos encontrar con altas capacidades (Kingston ha lanzado actualmente su último modelo con 2TB de capacidad, con altas velocidades de transferencia de archivos), son muy versátiles, resistentes a golpes y caídas, además de que, si son codificados correctamente, pueden llegar a ser dispositivos muy seguros.

Dentro de las unidades flash, podemos encontrarnos diversos tipos:

- SmartMedia: tarjetas con una memoria EEPROMM, con capacidades de hasta 128 MB. Empleadas en dispositivos electrónicos, especialmente en cámaras de fotos.
- IBM Microdrive: basados en los discos duros (con un gran parecido, ya que son discos duros de una pulgada de tamaño), llegaron a tener gran superioridad sobre el resto de las unidades flash, llegando a los 8 GB de capacidad, teniendo como estructura unos sistemas de archivos, que soportara sus capacidades. Un problema era su sensibilidad a golpes y vibraciones, ya que contenían gran cantidad de componentes mecánicos fácilmente estropeables.
- Tarjetas SD (Secure Digital): dispositivos portátiles, empleados en multitud de dispositivos electrónicos, con grandes capacidades (pueden llegar hasta los 2 TB). Una característica para destacar es que poseen un mecanismo de encriptación, el cual permite insertar datos codificados, además de un mecanismo de seguridad
- Tarjetas MMC (Multimedia Card): prácticamente iguales a las tarjetas SD, con la desventaja de que están no tienen un mecanismo de seguridad/encriptación.

En la actualidad llegan a tener hasta 512 GB de capacidad. Están basadas.

- Compact Flash: lanzado por SanDisk en 1994, con un controlador ID integrado, lo que le permite al dispositivo leer sin problema tarjetas de hasta 4 GB de capacidad.
- XD Picture Card: desarrollada por Olympus y Fuji, con capacidades de hasta 8GB.
- Memory Stick: esta memoria flash fue lanzada por Sony en 1999. En la actualidad, llegan a los 2 TB de capacidad, y son empleadas para dispositivos portátiles, como pueden ser videoconsolas, cámaras o portátiles.
- Memoria Flash USB (Pendrive): el dispositivo flash por excelencia. El pendrive sería lanzado en marzo de 1996 por las empresas IBM y Trek Technology. Cuando surgieron, requerían de pequeñas pilas o baterías para ser alimentados, pero en la actualidad les basta la alimentación procedente de un USB. Son muy versátiles, con capacidades de hasta 2TB de capacidad, altas velocidades de transferencia y lectura pueden llegar a ser residentes al agua, pueden ser leídos y soportados por casi cualquier sistema operativo y son asequibles a cualquier tipo de usuario.

Finalmente, llegamos a unos de los inventos más importantes dentro del almacenamiento de datos, y sobre todo la base de una de las posibles implementaciones que podríamos realizar con Elastic Search, los discos virtuales, o como comúnmente hemos oído hablar, **la nube**.

A pesar de que no sabemos muy bien cuando empezó este desarrollo, se estima que el almacenamiento en la nube nace en 2006, a través de los avances de grandes empresas dentro de sus modelos de negocio y estructura empresarial, como por ejemplo Amazon o Google.

Este sistema, se caracteriza por una serie de servidores, alojados en diversas ubicaciones, los cuales se encargan de atender todas las peticiones de la red en cualquier momento.

Al estar la información alojada en estos servidores, un usuario cualquiera puede disponer de ella cuando desee en cualquier lugar del mundo, de manera rápida, segura y en cualquier dispositivo.

Algunas de las características que destacan a este tipo de tecnología pueden ser su alto rendimiento, ya que los diferentes tipos de proveedores de servicios se encargan de controlar y a optimizar los recursos de forma automática, la escalabilidad, porque fácilmente se puede adaptar a cualquier tipo de mejora/desarrollo sin suponer gran trabajo, su bajo coste, ya que inicialmente se puede requerir de una inversión importante en este tipo de almacenamiento, pero vamos a evitar el tener que ocupar espacio físico para el almacenamiento requerido en el negocio, así como su mantenimiento. Por otro lado, dentro del mantenimiento también tendremos ventajas, ya que al no tener “físicamente” ese almacenamiento nosotros, no vamos a necesitar acciones por nuestra parte, ya que estamos accediendo remotamente a la información, por lo que el proveedor también se encarga de este aspecto.

También es importante la portabilidad.

Como hemos comentado, yo puedo mantener mi información alojada de unos de estos servidores externos, irme a otro país, y con mi dispositivo móvil poder acceder a ella, sin tener ningún tipo de problema ni preocuparme por tener que llevar dispositivos de almacenamiento, por lo que no tenemos límites.

Dicho esto, vamos ahora a ver de forma global las ventajas y desventajas que nos proporciona el almacenamiento en la nube.

Ventajas:

- Portabilidad y escalabilidad de los recursos.
- Mantenimiento automático de los servidores.
- No requiere un consumo extra de energía, siendo esta más eficiente.
- Altas capacidades, en función del tipo de usuario.
- Fácilmente integrable a casi cualquier tipo de software.
- Rápida implementación y migración de datos.
- Optimizamos el resto de los recursos de nuestra empresa, al liberarnos de la carga que supone alojar grandes cantidades de información, lo que mejora nuestros equipos.
- Ahorro en costes.
- Variedad de proveedores y servicios.
- Podemos tener copias de nuestra información en la red, a modo de Backup.

Desventajas:

- Sobrecargas en servidores debido a un alto número de peticiones.
- La continua actualización de estos sistemas conlleva una constante actualización de las interfaces de las aplicaciones.
- Necesitamos conexión a la red, no como con los dispositivos de almacenamiento convencionales.
- A nivel de seguridad, dependemos del proveedor, ya que dejamos nuestra información en sus servidores.
- En una tecnología aun en crecimiento, y a nivel empresarial, todavía se necesita una adaptación a los usuarios en este tipo de técnicas.
- La escalabilidad se puede dar a largo plazo, ya que, a más usuarios, más posibilidades hay de sobrecargas en los servicios.

Dicho esto, queda cerrado el capítulo de las diferentes tipologías de almacenamiento.

Nuestro objetivo, como comentamos, es que el usuario tenga un contexto sobre la “historia de la información”, para que cuando profundicemos en los tipos de recursos para gestionar esta información (como es Elastic Search), tengamos un mayor contexto y nos sea más fácil comprender como funciona este tipo de tecnología.

### 5.1.3 Tipos de bases de datos

#### 1. Bases de datos relacionales

Inventadas por Edgar Frank Codd en 1970, las bases de datos relacionales fueron el primer tipo de bases de datos que surgió. Éstas, son un conjunto de información ordenada, la cual queda alojada en tablas para su posterior consulta o constante actualización

Una de las grandes ventajas de este tipo de bases de datos es que son muy intuitivas y fáciles de entender, y una vez desarrolladas, cualquier persona puede realizar consultas (QUERYS) muy sencillas sin tener conocimientos de programación.

Pero ¿Por qué se denominan bases de datos relacionales? La respuesta es simple: las tablas (o relaciones) se componen de una serie de registros o entidades (filas), a los que denominaremos tuplas. Estas tuplas contendrán información relacionada con el tipo elegido en cada atributo (columna).

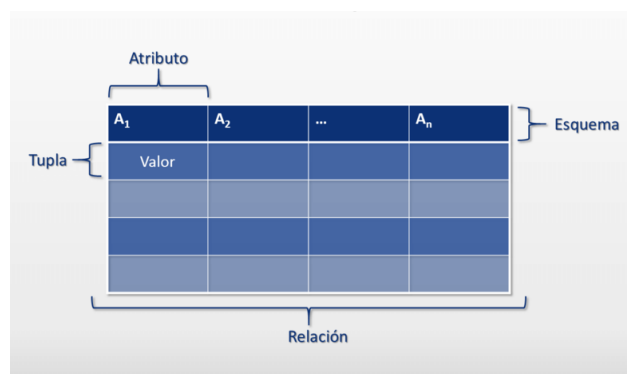


Figura 2 – Estructura de base de datos relacional

Los atributos, tendrán un tipo concreto, el cual deben respetar todos los registros si no queremos tener errores a la hora de insertar información.

Para ver esto de manera mas sencilla, proponemos el estudio del siguiente ejemplo:

<b>ID*</b> (Numeric)	<b>Nombre</b> (Varchar 20)	<b>Teléfono</b> (Numeric)	<b>DNI</b> (Varchar 9)	<b>Mail</b> (Varchar 100)
1	Alvaro	650356039	09061977K	Alvaro.gallego@edu.uah.es
2	Juan	685423659	08469547L	Juan@gmail.com
3	Andrea	918862547	95468762P	Andrea1234@empresa.eu
4	José	699569874	64981331T	JoseGar@ejemplo.es

La siguiente tabla, a la que denominaremos tabla usuarios, nos servirá para explicar levemente cómo funciona la estructura de una tabla común.

Por un lado, tendremos como hemos comentado antes, los diferentes atributos, como son el ID, Nombre, Teléfono etc. Si nos hemos fijado, el campo ID este marcado por un asterisco, el cual queremos dar a entender que es una Primary Key. La Primary Key es el atributo mas importante de la tabla, ya que es el campo que identifica a la tabla entera. Esta clave es única, y no puede haber dos registros con la misma clave primaria.

Tiene gran importancia porque a la hora de hacer una inserción en una tabla, está podrá comprobar si existe por la Primary Key, dándose varias condiciones: si existe, simplemente la información no se insertará (como mucho se actualizará), a gusto del usuario, y si no existe el registro se introducirá con éxito.

Como detalle, en una misma tabla podremos definir mas de una Primary Key.

También tenemos otro tipo de clave, como es la clave foránea, la cual se encargará de relacionar varias tablas existentes, o la clave índice que servirán para tener un acceso más rápido a información.

Por otro lado, otro detalle importante que tenemos que respetar es el tipo de los atributos, el cual en la tabla ejemplo está reflejada por la letra que tenemos en cursiva.

La definición de tipos, juntos a los diferentes tipos de claves es crucial en una base de datos, ya que marcará la estructura final que contendrá la información. Por ejemplo, el campo DNI, es de tipo Varchar, equivalente en el resto de los lenguajes de programación a un String, o cadena de texto, en este caso de tamaño 9 que coincide con la longitud numérica de un DNI estándar.

Cada vez que insertemos una nueva tupla a la base de datos, esta primero hará las comprobaciones de la Primary Key, y continuación comprobará que el tipo de dato que metemos para cada tipo de atributo es correcto. Por ejemplo, si metemos un DNI que contenga mas de 9 caracteres, tendremos un error en la inserción, ya que sobrepasamos el tamaño del atributo, sin embargo, si metiésemos un DNI con 5 letras, si podría ser insertado, ya que el tamaño del tipo indica la capacidad máxima que podemos insertar, no la obligatoria. El mismo fallo tendríamos sin por ejemplo queremos introducir un numero (tipo Numeric) en un campo que solo acepta Varchar, por eso es muy importante la definición de la base de datos.

Pero, esto nos provoca una duda: ¿Es posible modificar el tipo de un atributo?

La respuesta es sí, pero con ciertas limitaciones. Si tomamos como ejemplo el campo DNI, si queremos hacer una modificación en el tipo, y transfórmalo a tipo Numeric, nos encontraríamos con problemas, ya que un DNI obligatoriamente tiene una letra en su composición. Para hacer esto, tendríamos que borrar toda la columna, y reinsertar el dato sin la letra, cosa que a veces puede ser muy costoso y complicado de realizar. Mas fácil seria, por ejemplo, aumentar la capacidad del campo transformándolo de Varchar 9 a Varchar 20, ya que como la información contenida en el atributo respeta el nuevo tipo, esta no causara problemas.

La estructura por lo tanto queda definida a gusto del usuario, y es responsabilidad suya llevar a cabo una buena definición de tipos para evitar problemas estructurales en un futuro, así como las diferentes restricciones, atributos, dominios...

Finalmente, y para poner un punto final a esta breve explicación de las bases de datos relacionales, tenemos que ver el lenguaje empleado para el desarrollar, mantener y consultar información, que en este caso es el denominado lenguaje de consultas estructuradas, más conocido como SQL.

El SQL es un lenguaje muy conocido en la comunidad informática, ya que todos hemos oído alguna vez la palabra QUERY, método de consulta sobre las diferentes tablas, con las que podemos comprobar cualquier información que contengan.

Este tipo de lenguaje no acaba aquí, sino que cada tipo de diferentes bases de datos (relacionales y no relacionales), emplearán lenguajes derivados de este para realizar las consultas sobre las tablas que generen sus estructuras para procesar la información.

Algunas de las características a destacar dentro de este tipo de bases de datos serán, por lo tanto:



- Sencillez.
- Respetando tipos, evitamos duplicación de la información y datos inconsistentes.
- Fácil relación entre tablas.
- SQL, accesible a todo tipo de usuarios.
- Beneficia la normalización.

Finalmente, listaremos un conjunto de herramientas con los que podremos desarrollar/gestionar tablas:

- Mi SQL
- PostgreSQL
- DB2
- Oracle Database
- MariaBD
- Microsoft SQL Server

## 2. Bases de datos no relacionales

También denominadas bases de datos NOSQL. Rompen con el modelo relacional aportando flexibilidad, alto rendimiento y su variedad.

Para que entendamos su diferencia con las bases de datos relacionales, éstas no siguen un esquema en el que un identificador relaciona unos conjuntos de datos con otros, sino que la información queda organizada en documentos.

Este tipo de base de datos esta en auge, ya que son fáciles de desarrollar, y no requieren altos conocimientos para realizar una implantación, además, una de sus características mas importantes es que en este tipo de bases de datos no tenemos que predefinir una estructura y seguir un modelo para almacenar la información.



*Figura 3 – Ejemplos BBDD NOSQL*

A continuación, mostraremos los diferentes tipos que podemos encontrarnos dentro de estas bases de datos:

### 2.1. Orientadas a documentos

Aquí encontramos unos de los modelos más interesantes que ha surgido dentro de las bases de datos. Dentro de este tipo se encuentra Elastic Search, tema fundamental en este proyecto.

Las bases de datos orientados a documentos son bases de datos no relacionales formadas por documentos en vez de datos estructurados, que facilitaran a los usuarios la consulta y tratamiento de la información.

El formato de almacén serán documentos de tipo JSON.

Posee grandes ventajas sobre el resto, ya que, si queremos actualizar el modelo de datos, o simplemente cambiar, no tendremos problemas esquemáticos ni tendremos que cambiar la estructura, simplemente tendremos que actualizar los documentos que sean precisos. Además, es importante destacar su alta velocidad de lectura y consulta.

Ejemplos de estos tipos de datos son:

- MongoDB
- Elastic Search
- Couchbase
- Cassandra

## 2.2. Valor-clave

Una base de datos de tipo valor clave, es aquella que emplea metodologías de clave/valor, como su propio nombre indica, para alojar la información.

Comúnmente podemos conocer a este tipo valor/clave como Hash. Un hash es una serie de caracteres aleatorios, fruto de una función criptográfica o proceso de encriptación.

Utilizara una especie de mapa, donde cada clave tendrá asociado su valor dentro de una colección de datos.

Pongamos un ejemplo para verlo más claro:

Clave	Valor
Málaga	844-9631
Madrid	844-9840
Porto	844-6844

Como podemos ver, tenemos un campo clave, que va a ser el mas importante y el cual no podremos repetir, ya que es el único de puede identificar y acceder al valor que disponga asociado.

Son bases de datos algo complejas, pero también tienen ventajas, como su escalabilidad o que no necesitan tener definido un esquema concreto para almacenar información, como ocurre con las bases de datos orientadas a documentos.

Ejemplos son:

- Redis
- Aerospike
- OracleNoSQL
- Voldemorte

## 2.3. Orientadas a objetos

Estas bases de datos son muy similares a los lenguajes de programación orientados a objetos, ya que información se almacena en ella mediante objetos con información encapsulada.

Se componen de los ODBMS, es decir “object database management System” o sistemas de gestión de bases de datos orientadas a objetos, en las que tendremos integrado un lenguaje de programación, como puede ser C#, C++ o Java a la base de datos.

Ejemplos de estos ODBMS son:

- Informix
- Db40
- GemStone

Una funcionalidad extra que tienen este tipo de bases de datos es que se les puede definir operaciones sobre los datos, como parte de definición de la propia base de datos.

Albergan ventajas como una buena capacidad de encapsulación, polimorfismo y además pueden trabajar con leyes de herencia como ocurre con los lenguajes de programación.

Podemos dividirlos en tres grupos, en función de su uso:

- Abiertas: relacionadas con la programación, el diseñador puede poner su parte de trabajo.
- Mandatorias: obligatorias.
- Opcionales: se encargan de mejorar el sistema, aunque no son obligatorias.

Por otro lado, si lo comparamos con otras bases de datos, podemos encontrar un mayor rendimiento, ya que almacenan el producto generado a partir de un lenguaje de programación compatible, pero esta a la vez es su limitación, ya que nos será complicado introducir elementos de otra forma.

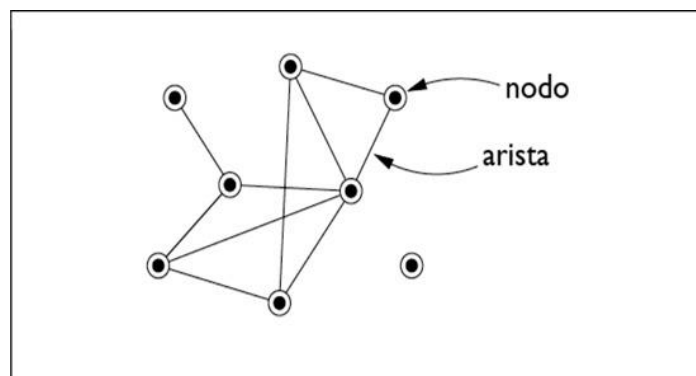
Otra curiosidad es que no se guarda relación de filas y columnas, como pasa en la mayor parte de las bases de datos, sino que los objetos entran y quedan alojados como si la base de datos fuera un almacén de relaciones, cosa que podemos considerar una desventaja, ya que no hay agrupaciones de objetos.

Para acceder a los datos se utilizarán punteros vinculados a los objetos.

#### 2.4. Orientadas a grafos

Las bases de datos orientadas a grafos son uno de los tipos de bases de datos más interesantes para analizar, ya que están compuestas por una serie de nodos interconectados entre sí, dando lugar al grafo.

Por lo que tenemos que diferenciar entre dos elementos, el nodo, que corresponderá con las diferentes entidades y las aristas que representarán las relaciones.



*Figura 4 – Explicación del grafo*

Están compuestas a su vez por una única columna por cada tabla disponible, por lo que cada arista unirá dos tablas, que a su vez serán dos columnas.

Respecto otros tipos de bases de datos, podemos encontrarnos ventajas como:

- Jerarquización de la base de datos, podemos recorrerla fácilmente saltando de nodo en nodo.
- No tenemos que definir un tamaño específico para el tipo de dato alojado en cada tabla.
- Las diferentes tablas no tienen porque tener el mismo tamaño, puede ser variable.
- Mejor rendimiento que las bases datos relacionales, ya que el aumento de consultas no sobrecarga la base de datos, como si pasa con el resto.
- Flexibles y escalables.
- Garantizamos la integridad de los datos.
- Casi cualquier lógica se puede modelar de manera sencilla mediante grafos dirigidos.

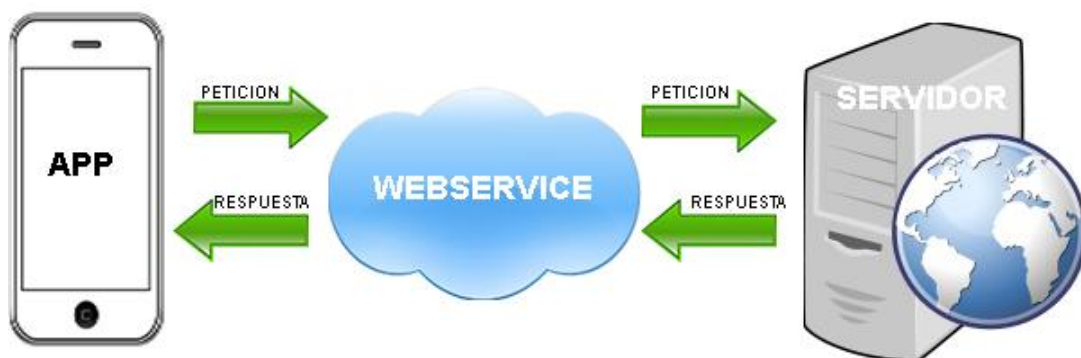
Algunos ejemplos que nos encontramos en el mercado son:

- Neo4j
- Infinite graph
- ArangoDB
- Horton
- Filament

### 3. Bases de datos As a Service

El futuro de las bases de datos probablemente se encuentre dentro de esta categoría, ya que las bases de datos As a Service, o como comúnmente nos las podemos encontrar DBaaS, son las bases de datos alojadas en la nube, ofreciéndonos servicios bajo demanda.

Es decir, nosotros contratamos un servicio en el que tendremos alojada la base de datos en el proveedor, no en nuestra propia infraestructura, por lo que él se encargará de la gestión y mantenimiento de esta.



*Figura 5 – Proceso petición-respuesta en Bases de datos As a Service*

Dentro de este tipo de bases de datos, nos podemos encontrar dos tipos, que casualmente coinciden con las dos ramas que hemos explicado antes: bases de datos SQL y bases de datos NoSQL

- SQL: suelen dar mas complicaciones, porque su diseño original no contemplaba desarrollos en la nube, cosa que le provoca una baja escalabilidad.
- NoSQL: estas si son mas recomendables a la hora de crear bases de datos en la nube, ya que, sí que están diseñadas para estos fines, por lo que si son escalables. Sirven para realizar altas cargas de trabajo, tanto para lectura como escritura, además son fáciles de migrar de integrar a diferentes lenguajes.

Algunas de las ventajas que nos proporcionan estos tipos de bases de datos, en función de la elección que hagamos dentro de sus distintos tipos, serán, por lo tanto:

- Escalabilidad y flexibilidad.
- Fácil migración de la información.
- Ahorro considerable al no tener que implementar la infraestructura.
- Fácil integración de lenguajes de programación.
- Accesibilidad a la información, ya que al estar en servidores de empresas que proporcionan el servicio, podemos acceder a ella desde cualquier parte.
- Ahorro de trabajo, ya que las labores de seguridad y mantenimiento las realiza el proveedor.

Ejemplos de implementaciones de estos tipos de bases de datos son:

- Xaas
- Oracle Cloud
- Azure SQL Database

## 5.2 ELK

Una vez explicada la parte más teórica, para poder tener un contexto sobre el origen de la información, almacenamiento y los diferentes tipos de bases de datos, vamos a proceder a explicar que es ELK, modulo del que forma parte Elastic Search, tema de este proyecto.

ELK surgió como fruto de la combinación de tres elementos, Elastic Search, Logstash y Kibana.

Su unión creo este potente proyecto, el cual alberga gran potencial gracias a su poder de adquisición, procesado, búsqueda y motorización de la información.

Recientemente, ELK ha cambiado su nombre a Elastic Stack, por la incorporación de un nuevo elemento, Beats, encargada de recoger logs.

Este proyecto esta ideado para en la monitorización, analítica de logs, explotación de la información, todas dependientes de sus módulos individuales, que hemos comentado antes.

Es totalmente versátil, teniendo algunas características como su almacenamiento distribuido, nos sirve como gestor de datos con el podemos procesar y monitorizar la información, comprobar su estado en tiempo real además de otras muchas más funcionalidades.

También cabe destacar que es Open Source, por lo que está a disposición de cualquier tipo de usuarios, además de ser un modulo importante para diversos desarrollos.

Algunas de sus ventajas son:

- Escalabilidad
- Fácilmente adaptable as sistemas distribuidos
- Grandes velocidades gracias a los índices inversos.
- Sencillo de configurar, a través de ficheros yaml.
- Fácil de invocar, gracias al trabajo con el formato JSON, lo que proporciona rápidas respuestas.
- Multiplataforma, pudiendo trabajar en cualquier sistema operativo.
- Open source.
- Desarrollado en java, por lo que la incorporación de nuevas herramientas por parte de la comunidad es casi continua.

Por otro lado, también tenemos una serie de desventajas, como:

- El formato JSON, a pesar de ser una ventaja, puede limitar ciertos lenguajes de programación, al ser este el único tipo de respuesta aceptado.
- A pesar de ser gratuito, por su característica Open Source, puede tener una importante inversión inicial emplear esta tecnología, ya que, al estar en desarrollo no hay muchos profesionales que entiendan Elastic Search completamente, y, por otro lado, la situación ideal dentro de Elastic, es tener diferentes equipos actuando como nodos para los diferentes módulos que poseen, ya que sino puede llegar a convertirse en una tecnología lenta y insegura, pudiendo producirse perdidas de información.

Dicho esto, vamos a proceder a explicar los distintos módulos por separado, ya que es la forma mas sencilla de descomponer la complejidad que Elastic Stack abarca.

### 5.2.1 Beats

Beats es una de las últimas incorporaciones de ELK, además es el motivo por el que el proyecto ha pasado de llamarse EKL, a Elastic Stack.

Estos módulos se encargan de la adquisición y alimentación de Elastic Stack, pudiendo realizar multitud de personalizaciones.

Con anterioridad, este trabajo lo realizaba Logstash, pero quedo liberado de carga de trabajo, dejándolo como un filtro para después.

Beats por su parte, además de ocupar menos espacio, nos permiten utilizar menos recursos de los que emplea Logstash, instalándolo en nuestros servidores para enviar los tipos de datos que deseemos.

La familia de Beats es amplia, pudiéndonos encontrar los siguientes módulos, además de otros que proporciona la comunidad de usuarios, no olvidemos que es un proyecto Open Source:

- **Filebeat:** es uno de los bits más difíciles de configurar dentro del equipo de Beats, pudiendo realizar multitud de tareas. Su labor principal es el análisis de log, ya sea creados de manera automática por diferentes aplicaciones, como pueden ser logs de Apache o realizar análisis de diversos tipos de performances.
- **Metricbeat:** es uno de los Beats más interesantes, pudiéndonos proporcionar métricas de todo tipo, como pueden ser eventos en sistemas operativos, diferentes logs arrojados por aplicaciones, HTTP, diferentes bases de datos etc. Toda la información la deja lista para que sea recogida por Elastic Search y visualizada por Kibana.
- **Packetbeat:** centrado en la monitorización del tráfico de red de los servidores, soportando todo tipo de protocolos: TLS, HTTP, IPC, Redis, DNS...
- **Winlogbeat:** empleado para monitorizar el sistema operativo Windows, viendo todos sus eventos. Sera el que empleemos en nuestra simulación.
- **Auditbeat:** es prácticamente idéntico al anterior, con la diferencia que el sistema operativo al que dedican el análisis de eventos es Linux.
- **Heartbeat:** nos permite consultar el estado de los diferentes servicios en funcionamiento dentro de nuestro servidor/infraestructura, comprobando el estado de las diferentes bases de datos (indicándonos por ejemplo si se ha caído alguna), el estado del sistema, podemos comprobar anomalías etc. Es realmente útil ya que se puede encargar de la monitorización del sistema, sustituyendo a otras aplicaciones dedicadas a ello, como pueden ser Icinga, Sengu, Nagio, Paessler o Zabbix.
- **Functionbeat:** este es uno de los últimos Beats desarrollado y en uso ya que se encarga de recoger eventos generados por servicios en la nube, enviándoselos a Elastic Search para procesarlos y Kibana para visualizarlos.



### 5.2.2 Logstash

Logstash es el otro método de recolección del paquete Elastic Stack. Con el podremos administrar, recolectar, procesar y guardar los diferentes logs que genere nuestro sistema o aplicaciones, transformándolo al formato que nosotros deseemos.

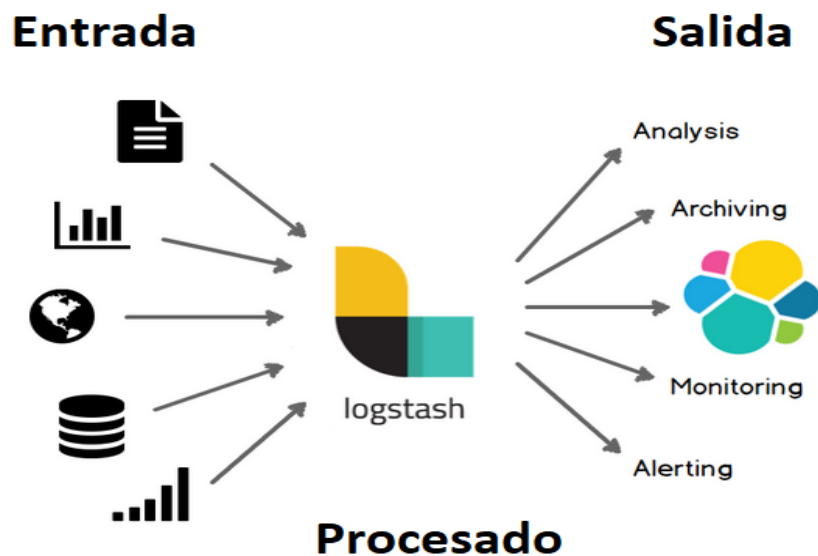
Tras la transformación, pasará la información a Elastic para que pueda tratarla.

Mientras que el resto de los módulos tienen labores más concretas y son más sencillos de manejar, Logstash destaca tanto por su polivalencia como por su complejidad.

El porqué de esta polivalencia, y característica que le diferencia de beats, es la capacidad de procesar prácticamente cualquier tipo de entrada, como pueden ser bases de datos, servidores de correo, logs del sistema, servicios en la nube...

Cuando este mensaje es adquirido por logstash, él se encargará de moldearlo a nuestra forma, sacando todas las variables que queramos, dejando más limpio el código, ya que por defecto los ficheros json de los que se suelen nutrir todas las salidas de elastic por defecto, a menudo son difíciles y no tan cómodas de ver.

Esta información procesada, podrá salir de logstash en el mismo formato en el que entró, como pueden ser nuevos logs, correos electrónicos, cargas en la nube, servicios redis...



*Figura 6 – Estructura entrada-procesado-salida de Logstash*

En la figura 6 podemos ver el funcionamiento del pipeline de Logstash, con su entrada de datos, su posterior fase de procesado donde se realizarán todos los filtros y diferentes condicionales que le apliquemos, y su salida, ya sea para el análisis, archivo, monitorización, alertas...

### 5.2.3 Elastic Search

Para empezar, tenemos que entender que es Elastic Search, su estructura y funcionamiento, por lo que vamos a explicar por partes como trabaja esta tecnología.

Como Elastic es el corazón de este proyecto, vamos a tratar de explicarlo de manera más desarrollada, para que podamos entender todo perfectamente, y a la hora de hacer las simulaciones veamos mejor que estamos haciendo.

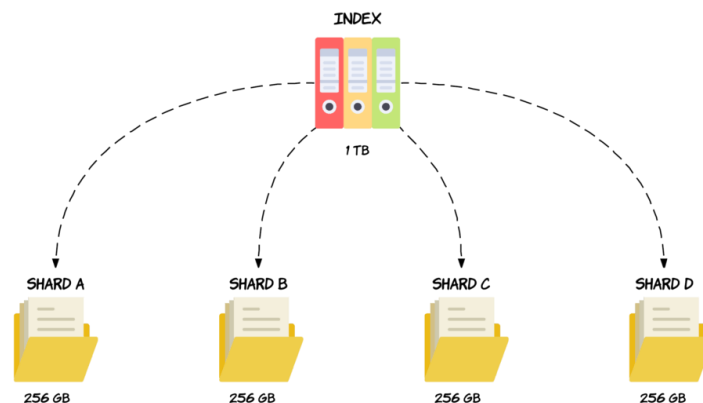
- **Estructura nodal**

Lo primero es definir que es un nodo. Un nodo para que nos entendamos, puede ser un equipo de trabajo, el cual tenemos configurado como nosotros deseemos, como por ejemplo a modo de servidor.

Estos nodos a su vez están formados por una estructura compleja, la cual se encargará de albergar la información y funcionalidad.

Las partes principales de un nodo son:

- **Index:** los índices van a ser las zonas en las que se van a alojar los diferentes tipos de documentos, al igual que, por ejemplo, dentro de una base de datos común, podríamos decir que en una tabla llamada automóviles, vamos a tener motos los coches, camiones, ...  
Alojaran la información, que más tarde podremos tratar.  
Elasticsearch asignara de forma aleatoria en que nodo se asignara cada índice.
- **Shard:** a su vez, los índices mostrados anteriormente estarán compuestos por shards. Estos shards son pequeños fragmentos que forman los propios índices, y los cuales están repartidos por los diferentes nodos, de ahí su complejidad e importancia de la correcta coordinación entre todos.  
El número de shards por defecto en Elastic es 5, aunque es el usuario el que puede configurarlo como desee.
- **Replica:** son copias de los shards, por defecto Elastic crea uno por nodo, pero también es configurable.



*Figura 7 – Ejemplo de distribución de Shards a partir del índice.*

- **Organización y tipos de nodos**

Cuando varios nodos cooperan entre si forman un Cluster, es decir, es un grupo de trabajo en el que cada parte tiene una funcionalidad concreta y vital en el sistema para llegar a un fin común.

Elastic Search, se basa en el trabajo del Cluster, aunque este lo componga un único nodo (la situación ideal es tener mínimo un nodo por módulo de Elastic Stack utilizado).

Aunque ya hemos mostrado los conceptos en el punto anterior, vamos a definir varios términos, ya que a partir de ahora estos cobrarán más importancia.

Elastic Search	BBDD relacionales
Index	Database/Table
Document	Row

Cada vez que mencionemos los índices, tendremos que pensar que son el equivalente de las bases de datos, los tipos serán las tablas y los documentos que añadimos son el equivalente a las filas que tiene nuestra tabla en las bases de datos relacionales.

Familiarizados con estos términos, ahora vamos a ver los diferentes tipos de nodos que nos podemos encontrar en el Cluster:

- **Master:** el nodo maestro se encarga de comprobar el estado del Cluster, y asegurar su integridad.  
También puede realizar otras tareas como la creación de índices, gestión de documentos en los otros nodos etc.  
Solo puede haber un master por Cluster, aunque si este tuviese un problema, cualquier otro nodo del Cluster ocupara su lugar (Elastic Tiene preconfigurado que cualquier nodo pase a ser master ante un problema en el principal).  
Es el nodo más importante, por lo que siempre tiene que estar en correcto funcionamiento, y como detalle, cualquier nodo de nuestro Cluster puede optar a ser un nodo maestro, aunque la asignación es aleatoria.
- **Data:** albergan los documentos (los shards y replicas que explicamos en puntos anteriores) y la mayor parte de trabajo, además se encargan de realizar las búsquedas. Todos los nodos que vamos incorporando al Cluster, de serie, pueden ser nodos de Datos, de manera similar a lo que ocurre con los Masters.
- **Client/Coord:** son menos utilizados, se encargan de balancear la información, enrutando peticiones al Cluster de forma externa, redirigiendo las peticiones que reciba por parte de cliente a las zonas correctas.
- **Ingest:** los nodos de ingesta se encargan de procesar la información, de manera similar a lo que hacen tanto Beats como Logstash. Conviene tenerlos separados ya que al igual que ocurre cuando utilizamos Logstash, pueden consumir una alta cantidad de recursos, por lo que se recomienda tener nodos potentes para albergar los nodos de ingesta.

- **Tribe:** su funcionalidad es muy similar a los nodos cliente. Se encargan de, una vez reciben la petición, coordinar la búsqueda no solo dentro de ese Cluster, sino entre diferentes Clusters dentro de Elastic Search para buscar la respuesta correcta.

- **Indexación**

Elastic Search es un motor de búsqueda cuya función es almacenar información, indexarla y luego más tarde poder consultarla.

Está basada en índices invertidos, estructura clave utilizada para recuperar datos dentro de altos volúmenes de información.

Por lo que el primer escenario que vamos a encontrarnos es el proceso de indexación.

En el proceso de indexación, Elastic se encargará de estructurar la información, en función de la configuración que establezca el usuario, para más tarde darle un sentido, y poder interpretarlo ya sea con Kibana o con la herramienta de análisis que deseemos emplear.

Como primer paso, tenemos la tokenización. Como puede resultar algo completo de ver, vamos a explicarlo con algunos ejemplos para que sea más fácil de entender.

Por ejemplo, si tenemos un campo “Descripción” en la tabla que queremos consultar, Elastic partiría la información de la siguiente forma:

Descripción	Documento
“Mesa rectangular de color marrón”	Documento 1
“Sillas de color azul, con decorados marrones”	Documento 2
“Mesas rectangulares procedentes de Suiza”	Documento 3

Elastic, procederá a realizar una fase de tokenización, que nosotros podremos configurar (si lo deseamos), dando lugar a algo parecido a esto:

Token	Documento
Mesa	1,2
Rectangular	1,3
Color	1,2
Azul	2
Marrón	1,2
Silla	2
Suiza	3
(...)	(...)

De esta manera, a la hora de realizar una QUERY, esta se realizará de forma más rápida cuando procedamos a buscar una palabra clave, porque Elastic no buscará en todos los documentos, sino dentro de los que haga match el token que busquemos, devolviéndonos el documento completo.

Además, a no ser que indiquemos lo contrario, no solo nos encontrará la palabra deseada, sino derivadas de ella. Ejemplo es la palabra “rectangular”, Elastic nos encontrará también la palabra “rectangulares” y todas sus derivadas. Todo esto es totalmente personalizable, podríamos filtrar por singular-plural, idioma, genero... todo lo que se nos ocurra y deseemos programar (la mayor parte de personalización radica en Logstash).

Podemos encontrarnos varios tipos de sistemas de tokenizacion en función del tipo de búsqueda que queramos hacer. El ejemplo propuesto tokenizaría por espacios en blanco, pero también podríamos crear distintos formatos en función del tipo de entrada que recibamos, como separaciones por puntos y comas (como los CSV's), mayúsculas o minúsculas, pipes etc.

Por eso la elección del tokenizar es algo muy importante en Elastic Search para buscar correctamente lo que queramos.

Tras la fase de tokenizacion, procedemos a dar significado a los tokens encontrados, pasando a la fase de análisis en la que podremos dar significado propio a las búsquedas, o crear clasificaciones.

Por ejemplo, podríamos clasificar los colores, buscar sinónimos, traducir a otro idioma o ignorar diferentes elementos del lenguaje como mostramos en el siguiente ejemplo:

Token	Análisis
Azul, Marrón	Color
Suiza	País
Decorados	Adornados
Mesa	Table
de	
(...)	(...)

Todo esto está orientado a la búsqueda, por eso Elastic Search es un motor tan potente.

Pero podemos encontrar problemas. Por ejemplo, si nosotros queremos saber en que documentos aparece el token azul, sería muy sencillo, ya que la función de tokenizacion de Elastic nos devolvería fácilmente donde esta alojada, pero si por otro lado quisiéramos saber que contiene el documento 1, Elastic tendría un problema, ya que tendría que recorrer todos los tokens de todos los documentos, siendo esto lento y poco eficiente.

Como solución a este problema, Elastic nos propone un nuevo concepto: Doc Values.

Esta nueva técnica, en vez de aplicar el índice invertido, aplicaría una ordenación de los tokens por documentos. Para que la explicación sea mas sencilla, mostraremos el siguiente ejemplo, partiendo del ejemplo inicial:

- Para la tabla:

Descripción	Documento
“Mesa rectangular de color marrón”	Documento 1
“Sillas de color azul, con decorados marrones”	Documento 2
“Mesas rectangulares procedentes de Suiza”	Documento 3

Recordamos que el proceso de tokenización nos reportaría el siguiente resultado:

Token	Documento
Mesa	1,2
Rectangular	1,3
Color	1,2
Azul	2
Marrón	1,2
Silla	2
Suiza	3
(...)	(...)

Pero con Doc Values, conseguiríamos ordenar la información tal que:

Tokens	Documento
mesa, rectangular, color, marrón	1
sillas, color azul, decorados, marrones	2
mesas, rectangulares, procedentes, Suiza	3

De esta manera, la estructura de tokens nos permite buscar rápidamente por documentos, evitando recorrer todos los tokens (como comentamos antes, artículos y preposiciones se obvian, aunque se podrían añadir, depende de la configuración que escoja cada usuario)

Elastic Search, cuando va a llevar a cabo su proceso de indexación, y como acabamos de explicar, iniciará el proceso de tokenización, analizará el contenido, construirá el índice invertido y además nos generará el Doc Values,

Como podemos entender, esto hace que este tipo de tecnología sea muy potente y versátil.

Finalmente listaremos el conjunto de características y ventajas que nos proporcionará este tipo de herramienta:

- Velocidad: Elasticsearch es capaz de analizar y encontrar velozmente coincidencias para todos aquellos elementos que consultemos, gracias a su peculiaridad, el índice invertido.
- Fácil de usar: Elasticsearch ofrece una API potente, una interfaz HTTP simple además de utilizar documentos JSON sin esquemas, lo que facilita su indexar, buscar y consultar datos.
- Disposición de la información en tiempo real.
- Formato de respuesta JSON, característico de las bases de datos orientadas a documentos.
- Multilenguaje: acepta multitud de lenguaje para el desarrollo y complemento de nuestras bases de datos: Python, PHP, Java, JavaScript.

- No posee una estructura concreta, ni tipos para los atributos, Elastic será capaz de almacenar información heterogénea.
- API REST intuitiva, con toda la funcionalidad explicada, en ella nos basaremos para hacer las consultas en la simulación.
- Distribuido: Ejecuta escalado de forma dinámica.
- Elastic Stack: la combinación de las 4 herramientas: Beats, Logstash, Elastic Search y Kibana lo convierten en una herramienta completísima y potente.

## 5.2.4 Kibana

Finalmente, tras haber recogido log con Beats o Logstash y haberlos procesado con Elastic Search, llegamos a la explicación del ultimo modulo del paquete Elastic Stack, Kibana.

Kibana es la herramienta que utilizaremos para poder dar una representación grafica a toda esa información que hemos recogido, pudiendo explorar los diferentes datos indexados per Elastic Search.

Dentro de la herramienta, la cual podremos previsualizar desde el dominio que indiquemos en su fichero de configuración .yaml, podremos crear diferentes dashboards, donde podremos ver la información, como podemos apreciar en la imagen 8.

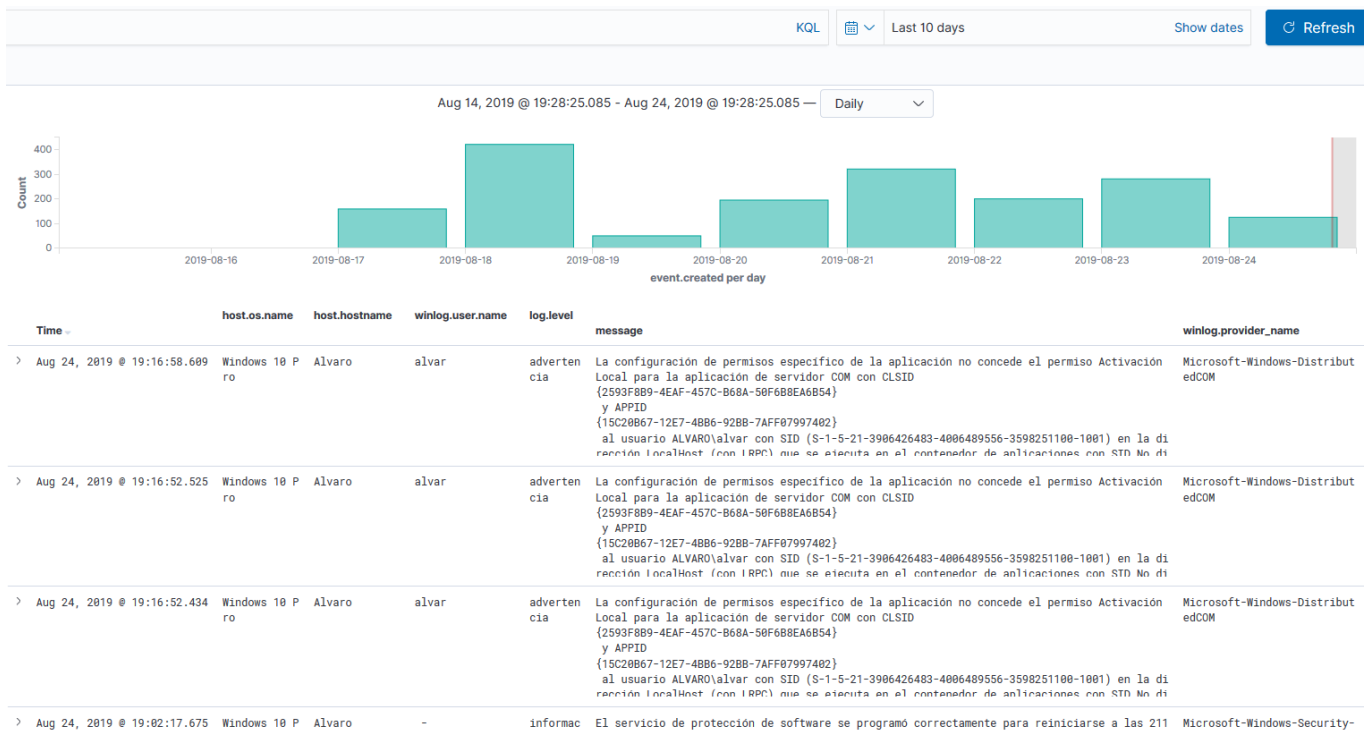


Figura 8 – Dashboard de Kibana

Podremos crear filtros de tiempo, para ver los logs dentro del rango de fechas que deseemos (como en nuestro ejemplo que estamos filtrando por los últimos 10 días), hacer subfiltros para mostrar en cada cuanto tiempo deseamos partir la información (en el ejemplo hemos puesto que sea a diario) y una vez seleccionadas las variables temporales, podremos añadir a nuestra vista los diferentes atributos que queremos analizar, como ocurre en nuestro ejemplo, que queremos mostrar el nombre del host tanto a nivel de usuario como de sistema operativo, el usuario que se ha conectado, la clasificación del mensaje dentro del nivel de logs, el tipo de mensaje que se ha producido...

También podremos crear diferentes tipos de gráficos y herramientas que nos proporciona la interfaz de Kibana(lo mostraremos en la parte de simulación) que podremos utilizar para el análisis y posterior tratamiento de la información.



Kibana, es una plataforma en auge, ya que su trabajo se tenía que desarrollar con otro tipo de herramientas antes de que esta surgiera, y esta, sin embargo, esta al estar conectada a Elastic Search, esta constantemente actualizada por lo que no hay que hacer nada, simplemente actualizar la información, suponiendo esto una gran ventaja dentro del mundo del análisis de datos, como ocurre dentro del business intelligence y business analytics.

Podemos resumir las ventajas de Kibana como:

- Fácil disposición de la información, podemos crear estructuras, gráficos y diferentes tipos de visualización que simplemente quedaran guardados y se actualizaran cuando llegue nuevo flujo de información.
- Open source, al igual que el resto del paquete de Elastic Stack, por lo que ahorramos considerablemente en software dedicado a la generación de informes, como puede ocurrir con la herramienta Power BI.
- Accesible a cualquier tipo de usuario, ya que la complejidad radica en la codificación del proyecto (la cual tampoco es muy complicada), una vez configurados los módulos, cualquier tipo de usuario podría disponer de un acceso a Kibana, con su usuario y contraseña para disponer la información al instante.
- Disposición de la información en tiempo real, al estar todos los módulos normalmente interconectados por una red de nodos o Cluster, la información estará constantemente en actualización, ya que un nodo recogerá logs con beat o Logstash (o con ambos a la vez, ya que son compatibles), otro nodo procederá a procesar e indicar la información para dejarla lista con Elastic Search, y otro nodo podrá contener Kibana para enlazarse con Elastic y poder visualizar la información.

Como podemos ver, Elastic Stack supone una gran evolución dentro del mundo del big data y el tratamiento y análisis de la información, ya que con sus herramientas podemos realizar proyectos cosas de manera muy sencilla.

## 6. Aplicaciones de Elastic Search

Tras entender el funcionamiento de las bases de datos y comprender el funcionamiento del paquete Elastic Stack, vamos a explicar algunos de los usos que podríamos dar a este conjunto de herramientas.

A nivel empresarial, cada vez resulta más complicado tratar tanta cantidad de información.

Para esto tendremos Elastic Search, para tratar de separar “el grano de la paja” en este gran granero al que conocemos con el mundo del big data.

Algunas de las aplicaciones más comunes que se llevan a cabo con esta herramienta son:

- Análisis: todos sabemos que el big data es el futuro. A través de los diferentes puestos dentro del análisis, como el Business Intelligence o Business Analytics, se puede llegar a sacar grandes conclusiones del estado del negocio, además de poder estudiar que soluciones tomar para llevar un camino más exitoso, analizando el pasado para mejorar el futuro, y analizando las tendencias futuras para mejorar en las actualidad.  
Con Elastic podremos realizar estudios fácilmente, viendo nuestros puntos fuertes o débiles mediante la creación de reportes a través de los diferentes documentos indexados en Elastic o llevar a cabo análisis de gráficos que podemos implementar con Kibana, donde podremos tratar, recopilar y procesar la información con el fin de optimizar procesos.  
Un ejemplo podría ser el análisis de campañas de correos, mandados por una empresa. Con Elastic podríamos sacar reportes fácilmente para ver sus estadísticas, como por ejemplo el número de mails enviados, el número de visitas a estos mails, el número de clics, con lo que podríamos comprobar fácilmente que campañas funciona mejor que otras.
- Almacén: con Elastic podemos tener almacenada toda la información que deseemos, pudiendo crear nuestra lógica de negocio dentro de ella para su posterior análisis o incluso pudiendo crear Backup, para tener protegida la información y disponible en caso de pérdidas.
- Monitorización: podremos realizar monitorizaciones de equipos, aplicaciones o sistemas. Para este caso, podemos ver un ejemplo real: en mi puesto de trabajo, se asignó al equipo comprobar si el login de los usuarios a una aplicación desarrollada por la empresa era correcto. Con Elastic, se puede procesar el tiempo que tarda un usuario en logarse, y fácilmente podemos comprobar si el estado de la aplicación en diferentes países es el mismo. Por ejemplo, para clientes alojados en España, Rusia, Portugal y Alemania en tiempo medio de login eran aproximadamente de 5 segundos, mientras que en Francia era de 19. Con esta herramienta se pudo detectar fácilmente que había un problema, el cual finalmente fue en problema de servidores.
- Alerta: podemos llevar el control de diferentes recursos, como pueden ser equipos, nodos, servidores o aplicaciones, para saber en todo momento que estos funcionan correctamente, y no han sufrido caídas o ataques. Un ejemplo sería el análisis de logins en un equipo, viendo que usuarios han accedido a él.

## 7. Marco práctico

Para la parte práctica, se ha decidido hacer varias simulaciones, para mostrar la amplia gama de aplicaciones que podemos dar a Elastic Search.

En la primera veremos un ejemplo de monitorización y muestra de resultados por Kibana y en la segunda mostraremos como subir bases de datos ya creadas, a través de herramientas compatibles con Elastic Stack, y su posterior lenguaje de Querys.

### 7.1 Simulación 1

#### 7.1.1 Instalación y recogida de logs

En esta primera simulación, vamos a realizar una monitorización interconectando un módulo de Beats, pasando como output la información a Elastic Search, y finalmente visualizándolo con Kibana.

Lo primero es escoger que módulo de Beats vamos a utilizar. En mi caso, como estoy trabajando sobre Windows, he decidido utilizar el módulo **Winlogbeat**.

Este modulo se encargara de recolectar log eventos de Windows, ya sea a nivel de aplicación, seguridad, instalación o sistema, como podemos observar en el visor de eventos:

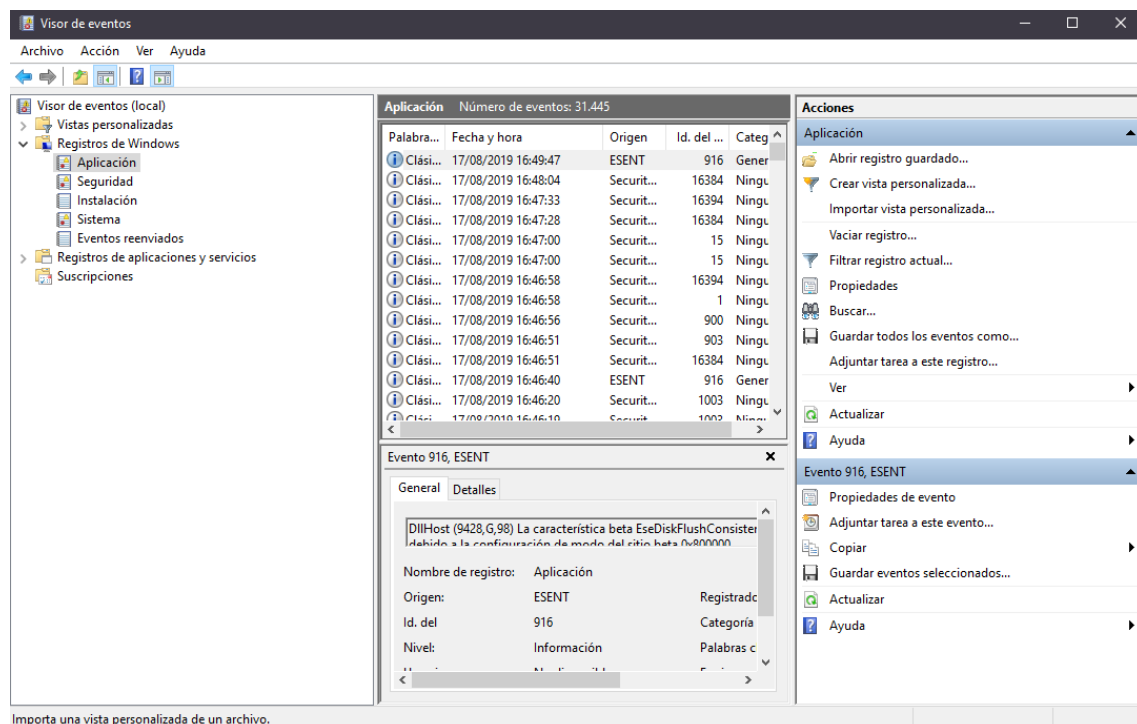


Figura 9 – Visor de eventos de Windows

Todos esos eventos que podemos ver serán los que podamos consultar una vez escojamos que deseamos ver.

Explicado esto, procedemos a descargar todos los módulos que vamos a necesitar, como son: **Winlogbeat, Elastic Search y Kibana,**

Para ello, basta con dirigirse a la web de Elastic Stack, en la que nos ofrecen la amplia gama de productos de Elastic:

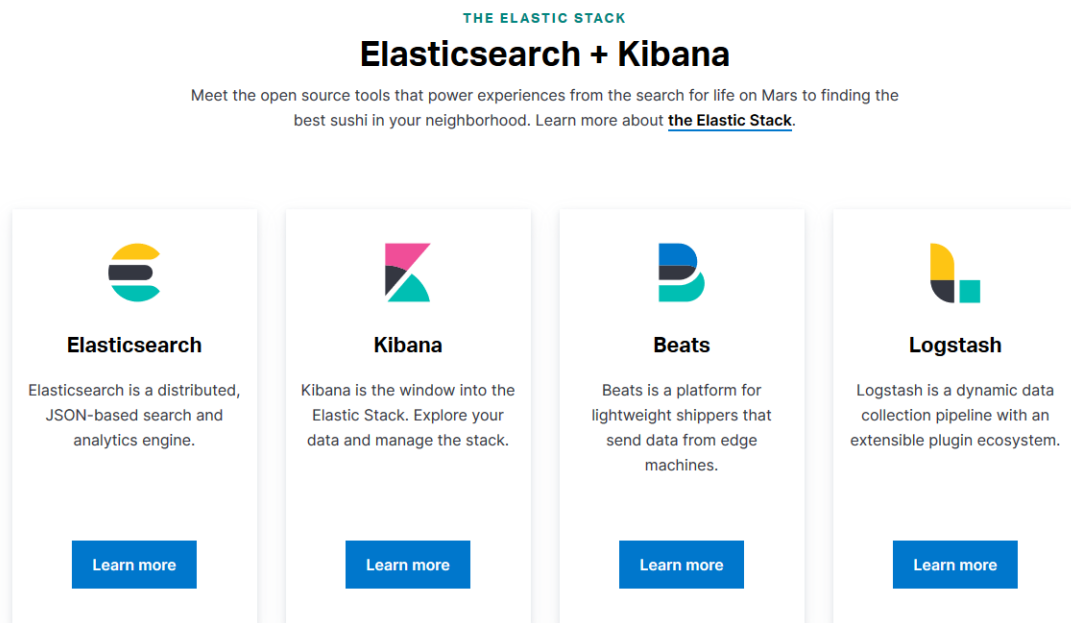


Figura 10 – Paquete Elastic Stack

Primero descargaremos el modulo Beats, encargado de recoger toda la información.

Como podemos ver (y ya explicamos de manera teórica) tendremos diversas opciones:

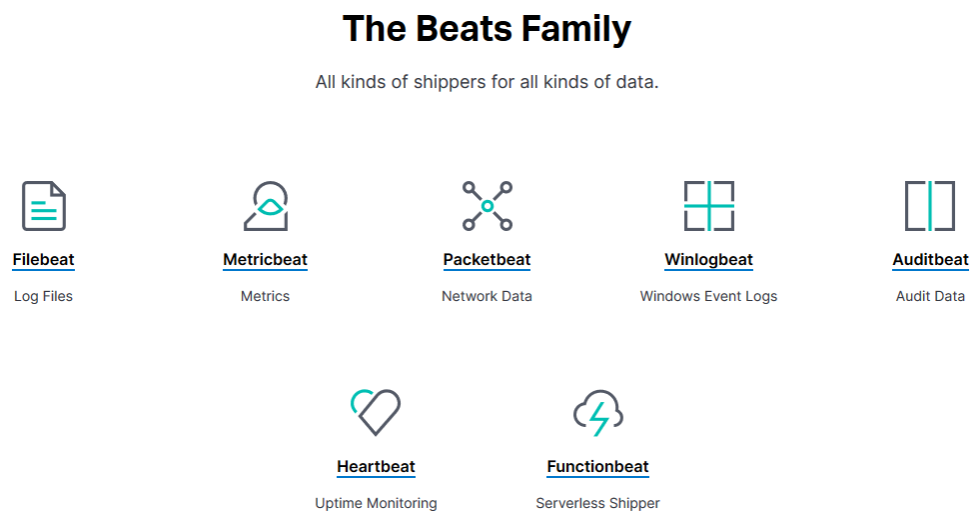


Figura 11 – Familia Beats

Seleccionamos Winlogbeat, y descargamos el paquete de datos.

Para el resto de los módulos tendremos que hacer lo mismo, seleccionando el tipo de datos en función del sistema operativo en el que vayamos a trabajar:

# Download Elasticsearch

Want it hosted? Deploy on Elastic Cloud. [Get Started »](#)

Version: 7.3.0

Release date: July 31, 2019

License: **Elastic License**

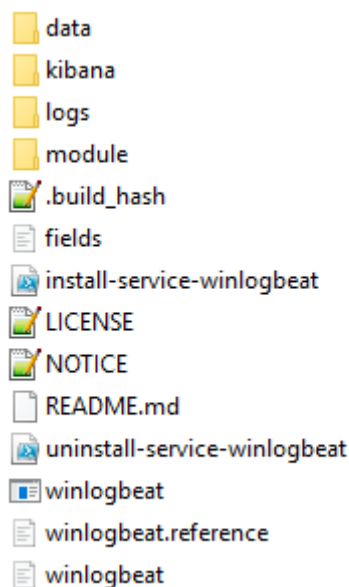
Downloads: [📄 WINDOWS sha asc](#) [📄 MACOS sha asc](#)  
[📄 LINUX sha asc](#) [📄 DEB sha asc](#)  
[📄 RPM sha asc](#) [📄 MSI \(BETA\) sha asc](#)

*Figura 12 – Descarga de Elastic Search*

En mi caso, la descarga de todos los módulos será para la plataforma Windows.

Una vez completada la descarga de Beats, Elastic, y Kibana, procederemos a descomprimirlos y a configurarlos.

Empezaremos con la configuración de Beats. Una vez descomprimido, nos quedara una estructura de archivos similar a esta:



*Figura 13 – Ramificación de archivos de Elastic Search*

Una gran ventaja que nos proporciona el paquete Elastic, es que nos da preconfigurados casi todos los módulos, teniendo que modificar nosotros simplemente la entrada y salida de la información, y sobre que queremos trabajar. Para ello, utilizaremos los ficheros .yaml, en los que guardaremos las configuraciones de lanzamiento.

En este caso configuraremos el archivo **Winlogbeat.yml**. Para que la explicación sea mas sencilla, dividiremos la explicación del código fuente en pequeños módulos:

```
Beats > WinLogBeat > ! winlogbeat.yml
1  winlogbeat.event_logs:
2    - name: Application
3      ignore_older: 72h
4
5    - name: System
6
7    - name: Security
8      processors:
9        - script:
10          lang: javascript
11          id: security
12          file: ${path.home}/module/security/config/winlogbeat-security.js
13
14    - name: Microsoft-Windows-Sysmon/Operational
15      processors:
16        - script:
17          lang: javascript
18          id: sysmon
19          file: ${path.home}/module/sysmon/config/winlogbeat-sysmon.js
20
```

Figura 14 – Configuración de Winlogbeat.yml, parte I

Este primer fragmento de código muestra la configuración de los event logs, es decir, la información del sistema que vamos a analizar.

Las variables mas importantes son las denominadas name, en las que indicaremos que eventos recolectar:

- **Application:** incluye los eventos generados por las aplicaciones del sistema.
- **System:** este evento registra todos los eventos relacionados con el propio sistema, así con todo lo relacionado con su funcionamiento.
- **Security:** aquí tendremos registros con las auditorias dentro de inicio de sesión, que usuarios se han conectado a nuestro sistema, si han puesto bien las contraseñas...
- Eventos personalizados

También podremos seleccionar diversos path donde alojar los logs, cosa que en un futuro nos vendría muy bien si tenemos que realizar una migración o actualización del sistema.

```
#===== Elasticsearch template settings =====
setup.template.settings:
  index.number_of_shards: 1
  #index.codec: best_compression
  #_source.enabled: false
```

Figura 15 – Configuración de Winlogbeat.yml, parte II

Aquí indicaremos el número de shards (que como explicamos anteriormente, son fragmentos en los que descomponemos los índices, para no queden 100% alojados en un único nodo) que tendrán nuestros nodos, en mi caso, al disponer de un único nodo solamente voy a poner un shard.

```
#===== Kibana =====

# Starting with Beats version 6.0.0, the dashboards are loaded via the Kibana API.
# This requires a Kibana endpoint configuration.
setup.kibana:

  # Kibana Host
  # Scheme and port can be left out and will be set to the default (http and 5601)
  # In case you specify an additional path, the scheme is required: http://localhost:5601/path
  # IPv6 addresses should always be defined as: https://[2001:db8::1]:5601
  #host: "localhost:5601"

  # Kibana Space ID
  # ID of the Kibana Space into which the dashboards should be loaded. By default,
  # the Default Space will be used.
  #space.id:
```

Figura 16 – Configuración de Winlogbeat.yml, parte III

También dispone de un módulo de configuración de Kibana, que es de gran utilidad, ya que no haría falta tener a Elastic Search de intermediario si empleamos este código.

Yo he decidido comentarlo para poder utilizar Elastic Search como puente entre Beats y Kibana, para demostrar la interconexión de los tres módulos.

```
#----- Elasticsearch output -----
output.elasticsearch:
  # Array of hosts to connect to.
  hosts: ["localhost:9200"]

  # Optional protocol and basic auth credentials.
  #protocol: "https"
  #username: "elastic"
  #password: "changeme"
```

Figura 17 – Configuración de Winlogbeat.yml, parte IV

Este último módulo se encarga de enlazar Beats con Elastic Search, matcheándolo por la dirección en la que tenemos alojado el Elastic. Eso es muy interesante, ya que, como es recomendable, podríamos tener distintos nodos para las distintas funcionalidades.

También se puede ver que podemos indicar el tipo de protocolo a utilizar, así como usuario y contraseña para una mayor encriptación de la información, cosa que en mi simulación no es necesaria, ya que solo lo voy a utilizar yo.

Además de estos submódulos que se acaban de explicar, existen otras muchas funcionalidades que se pueden aplicar dentro del fichero .yml, recordemos que Elastic destaca entre otras cosas por ser un paquete de código abierto, por lo que hay multitud de desarrollos que podemos indexar a nuestro proyecto, como diferentes dashboards para Kibana, diferentes visualizaciones, renombramiento de nodos etc.

Hasta aquí, tenemos la preconfiguración de Beats, que como hemos podido observar, ha sido muy sencilla.

Antes de explicar cómo lanzar beat, vamos explicaremos cómo configurar los .yml de los otros dos módulos, ya que son igual de fáciles de configurar.

Para Elastic Search, solo tendremos que localizar el fichero dentro de su carpeta de configuración (carpeta *config*, y archivo *elasticsearch.yml*).

Localizado, simplemente tendremos que configurar aquellos parámetros que deseemos. En mi caso, muchos campos han quedado por defecto y solo he realizado modificaciones de parámetros iguales que las de Beats, ya que Elastic lo deja preconfigurado, solo tenemos que indicar el host y puerto de Kibana para hacer el enlace.

Si tuviéramos que configurar varios nodos para funcionalidades distintas, elaborar la configuración sí que sería más completo, ya que trabajaríamos con Clusters y seguramente tendríamos que utilizar herramientas como Docker.

Como detalle, he añadido algunos parámetros que fueron necesarios en el fichero de configuración, ya que al principio tuve errores que no permitían a la aplicación iniciarse:

```
cluster.routing.allocation.disk.threshold_enabled: true
cluster.routing.allocation.disk.watermark.flood_stage: 5gb
cluster.routing.allocation.disk.watermark.low: 8gb
cluster.routing.allocation.disk.watermark.high: 8gb
```

*Figura 18 – Configuración del Cluster en Elastic Search*

Esta información se localizó en foros de Elastic Search ya que el problema era muy común en usuarios y es que el “watermark.low” siempre tiene que tener un valor más bajo o igual para el Cluster que el “watermark.high”, y Elastic por defecto nos pone superior la que no debe (a veces sucede a modo de Bug), por lo que haciendo los cambios de la figura 18 ya tendríamos solucionado el problema.

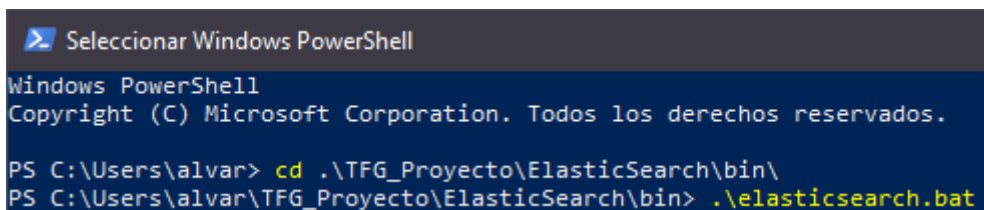
Y finalmente para Kibana tenemos la configuración también preestablecida por defecto, si no queremos añadir funcionalidad extra no debemos tocar nada, solo las mismas modificaciones que el fichero de Elastic.

Una vez listos los ficheros de configuración, procedemos a lanzar todas las aplicaciones, para empezar a recoger, procesar y visualizar los logs.

Antes de lanzar nada, tenemos que pensar en que orden debemos iniciar los módulos. El primero en lanzarse será Elastic Search, ya que los otros dos dependen de él, y si los lanzamos antes, por un lado, Kibana nos marcara errores al no poder conectarse a Elastic, y Beats estará creando logs que nadie puede recibir.

Para ello, abrimos un terminal de Powershell, y nos posicionamos sobre el fichero bin dentro de la carpeta que descargamos con Elastic Search.

Acto seguido, lanzamos Elastic Search (en Windows es el fichero *elasticsearch.bat*, aunque en otros sistemas bastaría con poner *.\elasticsearch*), como mostramos a continuación en la figura 18:



```
Seleccionar Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

PS C:\Users\alvar> cd .\TFG_Proyecto\ElasticSearch\bin\
PS C:\Users\alvar\TFG_Proyecto\ElasticSearch\bin> .\elasticsearch.bat
```

*Figura 18 – Comandos para el lanzamiento de Elastic Search*

Tras esto, Elastic quedará lanzado (puede tardar unos minutos en establecer conexión), quedándose a la espera de los otros módulos.



El lanzamiento del modulo Winlogbeat es algo diferente, ya que además de lanzar la aplicación deberemos pasarle por referencia el fichero de configuración .yaml que hemos explicado anteriormente.

Para ello tendremos que abrir una nueva PowerShell (dejando trabajar libremente a Elastic Search en su terminal) y bastará con posicionarse en la carpeta de Winlogbeat, y ejecutar las siguientes instrucciones:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

PS C:\Users\alvar> cd .\TFG_Proyecto\Beats\WinLogBeat\
PS C:\Users\alvar\TFG_Proyecto\Beats\WinLogBeat> .\winlogbeat.exe -c .\winlogbeat.yaml
```

Figura 19 – Comandos para el lanzamiento de Winlogbeat

Si todo ha ido bien, Winlogbeat comenzara a trabajar, y comenzaran a llegar logs al módulo de Elastic Search:

```
[2019-08-17T16:48:05.921][INFO][o.e.n.Node][ALVARO] started
[2019-08-17T16:48:07.223][INFO][o.e.c.r.a.AllocationService][ALVARO] Cluster health status changed from [RED] to [YELLOW] (reason: [shards started [[.kibana_task_manager][0], [.kibana_1][0]] ...]).
[2019-08-17T16:53:23.500][INFO][o.e.c.m.MetaDataMappingService][ALVARO] [winlogbeat-7.3.0-2019.08.17-000001/cv7eY1RBRCEqJwelmCSAA] update_mapping [_doc]
[2019-08-17T16:53:24.165][INFO][o.e.c.m.MetaDataMappingService][ALVARO] [winlogbeat-7.3.0-2019.08.17-000001/cv7eY1RBRCEqJwelmCSAA] update_mapping [_doc]
[2019-08-17T16:53:24.784][INFO][o.e.c.m.MetaDataMappingService][ALVARO] [winlogbeat-7.3.0-2019.08.17-000001/cv7eY1RBRCEqJwelmCSAA] update_mapping [_doc]
[2019-08-17T16:53:25.426][INFO][o.e.c.m.MetaDataMappingService][ALVARO] [winlogbeat-7.3.0-2019.08.17-000001/cv7eY1RBRCEqJwelmCSAA] update_mapping [_doc]
[2019-08-17T16:53:25.974][INFO][o.e.c.m.MetaDataMappingService][ALVARO] [winlogbeat-7.3.0-2019.08.17-000001/cv7eY1RBRCEqJwelmCSAA] update_mapping [_doc]
[2019-08-17T16:53:26.483][INFO][o.e.c.m.MetaDataMappingService][ALVARO] [winlogbeat-7.3.0-2019.08.17-000001/cv7eY1RBRCEqJwelmCSAA] update_mapping [_doc]
[2019-08-17T16:53:27.566][INFO][o.e.c.m.MetaDataMappingService][ALVARO] [winlogbeat-7.3.0-2019.08.17-000001/cv7eY1RBRCEqJwelmCSAA] update_mapping [_doc]
[2019-08-17T16:53:27.955][INFO][o.e.c.m.MetaDataMappingService][ALVARO] [winlogbeat-7.3.0-2019.08.17-000001/cv7eY1RBRCEqJwelmCSAA] update_mapping [_doc]
[2019-08-17T16:53:28.346][INFO][o.e.c.m.MetaDataMappingService][ALVARO] [winlogbeat-7.3.0-2019.08.17-000001/cv7eY1RBRCEqJwelmCSAA] update_mapping [_doc]
[2019-08-17T16:53:28.811][INFO][o.e.c.m.MetaDataMappingService][ALVARO] [winlogbeat-7.3.0-2019.08.17-000001/cv7eY1RBRCEqJwelmCSAA] update_mapping [_doc]
[2019-08-17T16:53:29.176][INFO][o.e.c.m.MetaDataMappingService][ALVARO] [winlogbeat-7.3.0-2019.08.17-000001/cv7eY1RBRCEqJwelmCSAA] update_mapping [_doc]
[2019-08-17T16:53:29.615][INFO][o.e.c.m.MetaDataMappingService][ALVARO] [winlogbeat-7.3.0-2019.08.17-000001/cv7eY1RBRCEqJwelmCSAA] update_mapping [_doc]
[2019-08-17T16:53:29.918][INFO][o.e.c.m.MetaDataMappingService][ALVARO] [winlogbeat-7.3.0-2019.08.17-000001/cv7eY1RBRCEqJwelmCSAA] update_mapping [_doc]
[2019-08-17T16:53:30.270][INFO][o.e.c.m.MetaDataMappingService][ALVARO] [winlogbeat-7.3.0-2019.08.17-000001/cv7eY1RBRCEqJwelmCSAA] update_mapping [_doc]
[2019-08-17T16:53:30.624][INFO][o.e.c.m.MetaDataMappingService][ALVARO] [winlogbeat-7.3.0-2019.08.17-000001/cv7eY1RBRCEqJwelmCSAA] update_mapping [_doc]
[2019-08-17T16:53:30.925][INFO][o.e.c.m.MetaDataMappingService][ALVARO] [winlogbeat-7.3.0-2019.08.17-000001/cv7eY1RBRCEqJwelmCSAA] update_mapping [_doc]
[2019-08-17T16:53:31.206][INFO][o.e.c.m.MetaDataMappingService][ALVARO] [winlogbeat-7.3.0-2019.08.17-000001/cv7eY1RBRCEqJwelmCSAA] update_mapping [_doc]
[2019-08-17T16:53:31.455][INFO][o.e.c.m.MetaDataMappingService][ALVARO] [winlogbeat-7.3.0-2019.08.17-000001/cv7eY1RBRCEqJwelmCSAA] update_mapping [_doc]
[2019-08-17T16:53:31.699][INFO][o.e.c.m.MetaDataMappingService][ALVARO] [winlogbeat-7.3.0-2019.08.17-000001/cv7eY1RBRCEqJwelmCSAA] update_mapping [_doc]
[2019-08-17T16:53:31.957][INFO][o.e.c.m.MetaDataMappingService][ALVARO] [winlogbeat-7.3.0-2019.08.17-000001/cv7eY1RBRCEqJwelmCSAA] update_mapping [_doc]
[2019-08-17T16:53:32.288][INFO][o.e.c.m.MetaDataMappingService][ALVARO] [winlogbeat-7.3.0-2019.08.17-000001/cv7eY1RBRCEqJwelmCSAA] update_mapping [_doc]
[2019-08-17T16:53:32.677][INFO][o.e.c.m.MetaDataMappingService][ALVARO] [winlogbeat-7.3.0-2019.08.17-000001/cv7eY1RBRCEqJwelmCSAA] update_mapping [_doc]
[2019-08-17T16:53:33.068][INFO][o.e.c.m.MetaDataMappingService][ALVARO] [winlogbeat-7.3.0-2019.08.17-000001/cv7eY1RBRCEqJwelmCSAA] update_mapping [_doc]
[2019-08-17T16:53:33.487][INFO][o.e.c.m.MetaDataMappingService][ALVARO] [winlogbeat-7.3.0-2019.08.17-000001/cv7eY1RBRCEqJwelmCSAA] update_mapping [_doc]
[2019-08-17T16:53:33.689][INFO][o.e.c.m.MetaDataMappingService][ALVARO] [winlogbeat-7.3.0-2019.08.17-000001/cv7eY1RBRCEqJwelmCSAA] update_mapping [_doc]
[2019-08-17T16:53:39.792][INFO][o.e.c.m.MetaDataMappingService][ALVARO] [winlogbeat-7.3.0-2019.08.17-000001/cv7eY1RBRCEqJwelmCSAA] update_mapping [_doc]
[2019-08-17T16:53:39.825][INFO][o.e.c.m.MetaDataMappingService][ALVARO] [winlogbeat-7.3.0-2019.08.17-000001/cv7eY1RBRCEqJwelmCSAA] update_mapping [_doc]
```

Figura 20 – Muestra de indexación de logs por parte de Elastic Search, tras recogerlos Winlogbeat.

Para poder comprobar que Elastic Search se ejecuta de marea correcta, basta con abrir un explorador, llamando al localhost en el puerto 9200, o donde lo tengamos configurado.

Como podemos comprobar en la figura 21, Elastic esta desplegado correctamente:

JSON	Datos sin procesar	Cabeceras
Guardar	Copiar	Contraer todo Expandir todo <input type="text" value="Filtrar JSON"/>
<pre>name: "ALVARO" cluster_name: "elasticsearch" cluster_uuid: "pcQXQFbaRXK1ZmWGS7bueA" version:   number: "7.3.0"   build_flavor: "default"   build_type: "zip"   build_hash: "de777fa"   build_date: "2019-07-24T18:30:11.767338Z"   build_snapshot: false   lucene_version: "8.1.0"   minimum_wire_compatibility_version: "6.8.0"   minimum_index_compatibility_version: "6.0.0-beta1" tagline: "You Know, for Search"</pre>		

Figura 21 – Pantalla principal de Elastic Search

Podemos comprobar si el documento se ha creado bien, añadiendo al localhost la función de la API cat (que explicaremos más adelante) la siguiente sentencia **\_cat/índices**, quedando el enlace de la siguiente manera: [http://localhost:9200/\\_cat/indices](http://localhost:9200/_cat/indices)

Como resultado, podremos ver los diferentes documentos creados, y confirmar que el módulo de Beats está funcionando correctamente:

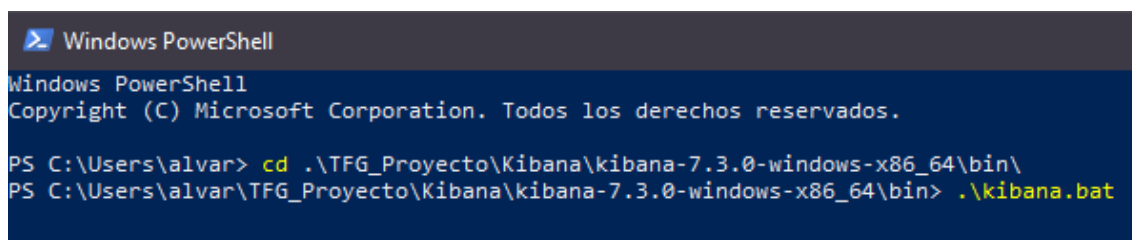
```
green open .kibana_task_manager QrtwN8ltRreey5nytkrbZQ 1 0 2 0 13kb 13kb
yellow open winlogbeat-7.3.0-2019.08.17-000001 Cv7eY1RBRCiEqJWelmCSAA 1 1 145 0 308.9kb 308.9kb
green open .kibana_1 ls_fLU9OTnOwSfAdNqdUqA 1 0 28 3 154.5kb 154.5kb
```

Figura 22 – Muestra de índices alojados en Elastic Search

Y efectivamente, los módulos están bien comunicados.

El ultimo modulo que nos queda por conectar es el encargado de mostrar gráficamente la información, como es Kibana.

Su lanzamiento es el mismo que Elastic Search, tendremos que posicionarnos en la carpeta bin, dentro de la carpeta que descargamos, y ejecutar la siguiente orden:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

PS C:\Users\alvar> cd .\TFG_Proyecto\Kibana\kibana-7.3.0-windows-x86_64\bin\
PS C:\Users\alvar\TFG_Proyecto\Kibana\kibana-7.3.0-windows-x86_64\bin> .\kibana.bat
```

Figura 21 – Pantalla principal de Elastic Search

Tendremos que tener paciencia, ya que el módulo de Kibana suele ser uno de los que mas tarda en conectar, pero tras la espera, deberíamos obtener algo similar a lo siguiente:

```
log [15:54:07.131] [info][status][plugin:xpack_main@7.3.0] Status changed from yellow to green - Ready
log [15:54:07.134] [info][status][plugin:graph@7.3.0] Status changed from yellow to green - Ready
log [15:54:07.138] [info][status][plugin:searchprofiler@7.3.0] Status changed from yellow to green - Ready
log [15:54:07.140] [info][status][plugin:ml@7.3.0] Status changed from yellow to green - Ready
log [15:54:07.144] [info][status][plugin:tilemap@7.3.0] Status changed from yellow to green - Ready
log [15:54:07.146] [info][status][plugin:watcher@7.3.0] Status changed from yellow to green - Ready
log [15:54:07.149] [info][status][plugin:grokdebugger@7.3.0] Status changed from yellow to green - Ready
log [15:54:07.151] [info][status][plugin:logstash@7.3.0] Status changed from yellow to green - Ready
log [15:54:07.154] [info][status][plugin:beats_management@7.3.0] Status changed from yellow to green - Ready
log [15:54:07.157] [info][status][plugin:index_management@7.3.0] Status changed from yellow to green - Ready
log [15:54:07.159] [info][status][plugin:index_lifecycle_management@7.3.0] Status changed from yellow to green - Ready
log [15:54:07.161] [info][status][plugin:rollup@7.3.0] Status changed from yellow to green - Ready
log [15:54:07.162] [info][status][plugin:remote_clusters@7.3.0] Status changed from yellow to green - Ready
log [15:54:07.163] [info][status][plugin:cross_cluster_replication@7.3.0] Status changed from yellow to green - Ready
log [15:54:07.165] [info][status][plugin:file_upload@7.3.0] Status changed from yellow to green - Ready
log [15:54:07.168] [info][status][plugin:snapshot_restore@7.3.0] Status changed from yellow to green - Ready
log [15:54:07.172] [info][kibana-monitoring][monitoring] Starting monitoring stats collection
log [15:54:07.181] [info][status][plugin:maps@7.3.0] Status changed from yellow to green - Ready
log [15:54:08.053] [warning][reporting] Generating a random key for xpack.reporting.encryptionKey. To prevent pending reports from failing on restart, please set xpack.reporting.encryptionKey in Kibana.yml
log [15:54:08.061] [info][status][plugin:reporting@7.3.0] Status changed from uninitialized to green - Ready
log [15:54:08.748] [info][listening] Server running at http://localhost:5601
log [15:54:08.766] [info][server][Kibana][http] http server running
log [15:54:08.780] [info][status][plugin:spaces@7.3.0] Status changed from yellow to green - Ready
```

Figura 22 – Estado de lanzamiento de Kibana

Y para comprobar que el modulo esta en funcionamiento, bastara con conectarse al puerto 5601 del local host, y si la conexión ha sido exitosa, debería mostrarse la interfaz gráfica de Kibana:

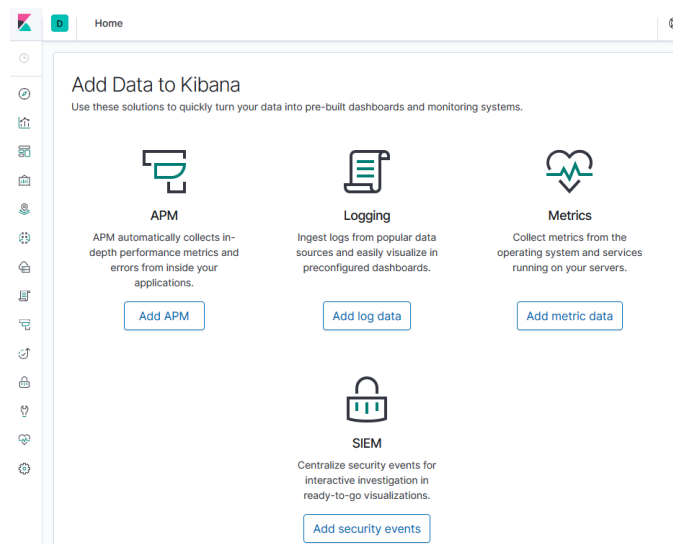


Figura 22 – Pantalla principal de Kibana

Tas esta explicación de la instalación y lanzamiento de los distintos módulos, vamos a centrarnos ahora en todo el flujo de información que se ha producido, desde que Beats obtuvo la información, Elastic la organizo y Kibana la tiene lista para ser analizada mediante su interfaz gráfica.

Para ello, se ha estado aproximadamente un mes recogiendo logs a través de Winlogbeat, tratándolos a su vez con Elastic Search, que se ha encargado de pasar la información a su índice correspondiente, por lo que para llevar a cabo esta labor se ha tenido que estar lanzando a diario las dos aplicaciones, (en una empresa esto podría ser fácilmente automatizable dejándolo lanzado sin cortes, pero para un usuario normal no se puede ya que una vez apagamos el equipo las herramientas dejan de funcionar).

Para comprobar el estatus del proyecto, simplemente abrimos Kibana, y nos posicionamos sobre su pestana “Discover”.

Una vez aquí, seleccionamos el índice que queremos visualizar, en nuestro caso Winlogbeat\* (El asterisco nos evitará tener que poner el nombre del índice al completo). En mi caso, voy a mostrar los últimos 30 días (last 30 days) ordenados de forma diaria (Daily), como apreciamos en la figura 23:

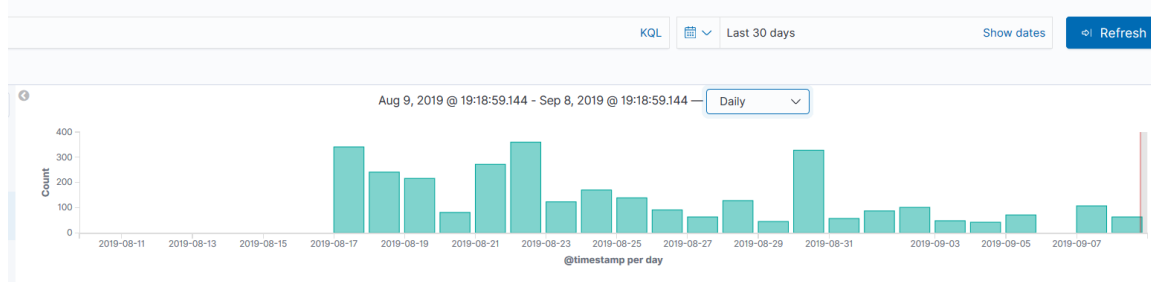


Figura 23 – Grafico de Kibana con los logs almacenados disponibles

Una curiosidad apreciable, es la línea roja al final del gráfico, que indica el estado de procesamiento de los logs. Al no estar completa, significa que aun esta “escuchando” en búsqueda de nuevos logs, además de que es posible que aún no se hayan procesado todos.

## 7.1.2 Kibana

Una vez instalado correctamente el paquete de herramientas, y llevado a cabo el largo proceso de recolección de logs, vamos a mostrar cómo podríamos visualizar esta información en Kibana,

La parte más sencilla, puede ser mostrar los campos mas importantes, en un flujo de tiempo determinado. Para ello, nos posicionamos en la última parte que vimos, la pestaña Discover, y aplicamos los mismos filtros que aplicamos por entonces, Last 60 days y agrupación Daily.

Si todo es correcto, tendremos algo similar a lo siguiente:

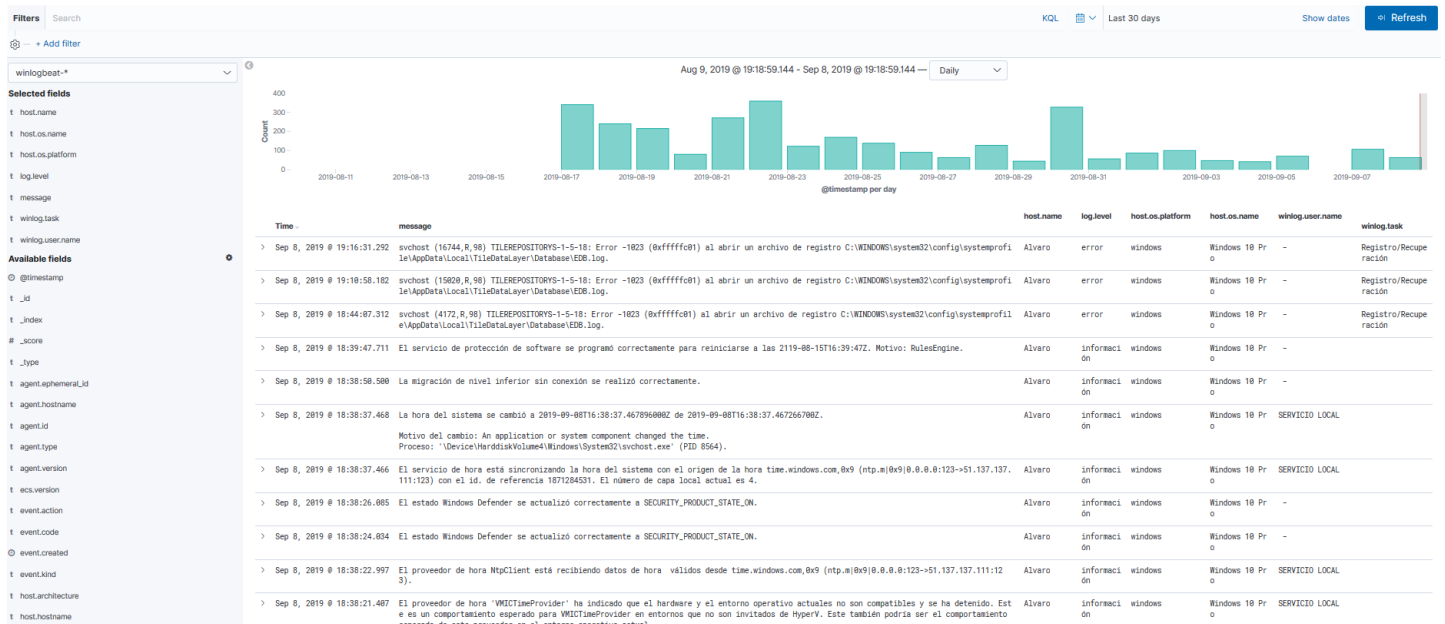


Figura 24 – Pestaña “Discover” de Kibana

La parte superior nos mostrará un pequeño gráfico con la cantidad de logs recogidos en el tiempo. Por otro lado, tenemos varias secciones. En la parte de la izquierda tenemos dos subdivisiones: “Available fields” y “selected fields”. Este primero mostrara todos aquellos campos que hemos recogido en los logs, en mi caso hay un gran numero (cerca de 80), ya que he tratado de registrar toda la información posible del sistema. Y por otro lado en “selected fields” tendremos los campos que mostraremos en la parte inferior al gráfico, donde se verá toda la información disponible, en formato json.

Si clicamos en un campo cualquiera y pulsamos “visualize”, nos saldrá un pequeño gráfico mostrando la variedad de elementos que hay dentro de él:

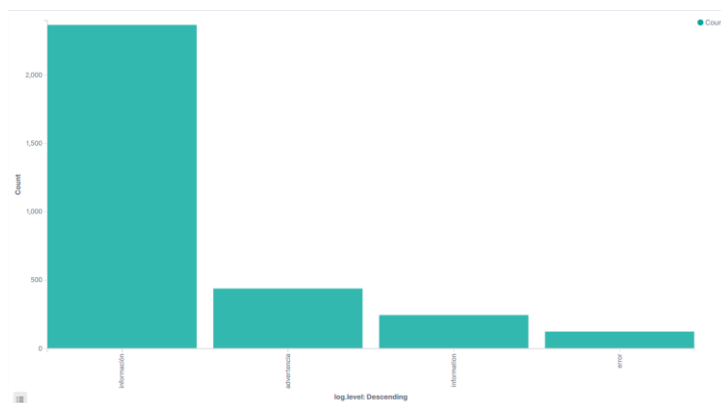


Figura 25 – Ejemplo de visualización de campo en Kibana

Como vemos en la figura 25, tenemos la agrupación de los mensajes por niveles en Windows, siendo estos: información, advertencia, information, los mensajes de error. También nos saldrá el recuento de mensajes por cada nivel.

Podríamos por lo tanto ver las estadísticas de cualquier campo simplemente clicando en la pestaña de visualización, pero esto es simplotte una preview, ya que a continuación mostraremos como visualizarlo correctamente.

## 1. Visualizaciones

Pero si lo que queremos es crear nuestros gráficos personalizados, tendremos que dirigirnos a la pestaña Visualize, alojada en el menú de la derecha dentro de Kibana.

Una vez posicionados, podemos clicar en create new visualización, y proceder a crear el nuestro.

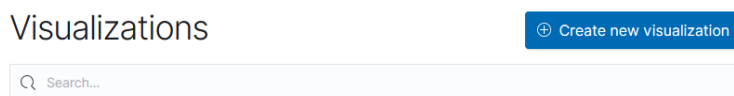


Figura 26 – Creación de la visualización

Dentro, nos encontraremos diferentes tipos de visualizaciones, a gusto del usuario, como podemos observar en la figura 27.

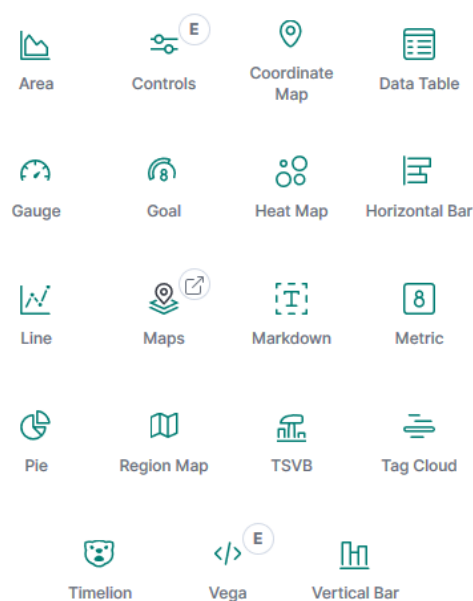


Figura 27 – Tipos de visualizaciones disponibles

- **Diagrama Pie**

Para este primer ejemplo, vamos a crear un simple diagrama de tarta o de pastel, por lo que clicamos en pie y seleccionamos el índice a analizar, es este caso los logs que recogimos con Winlogbeat.

Para poder seleccionar el campo que queremos visualizar, tendremos que irnos a Buckets y crearemos un Split slices.

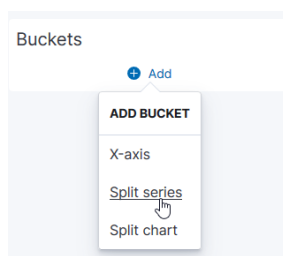


Figura 28 – Añadir nuevo Split

Ahí tendremos varios parámetros configurables:

- Aggregation: serán las agrupaciones que realizaremos en función del campo seleccionado (esto es muy importante ya que todas las visualizaciones de Kibana se basan en agregaciones, y más adelante veremos cómo realizarlas con Elastic Search). Tendremos multitud de tipos, como pueden ser agregaciones por términos, por rango, filtros etc.
- Field: este será el campo para visualizar.

Y el resto de los campos son menos importantes, ya que son para meros detalles. Como curiosidad, podríamos crear diferentes Split slices para introducir más campos a nuestro diagrama.

En nuestro ejemplo al ser un diagrama de queso, no sería necesario, pero si utilizáramos por ejemplo un diagrama de donut (también configurable dentro del diagrama de tipo tarta, solo tendremos que ir a opciones y activar el check donut) podríamos mostrar varios campos con unos en función de otros.

Una vez seleccionados los parámetros deseados, podremos crear el gráfico pulsando el botón de play.

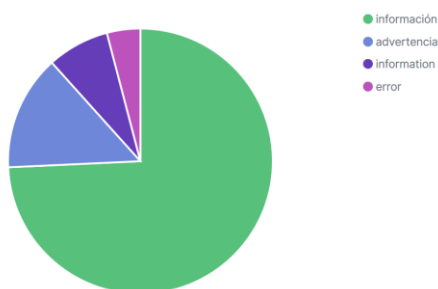


Figura 29.1 – Gráfico tipo pie I

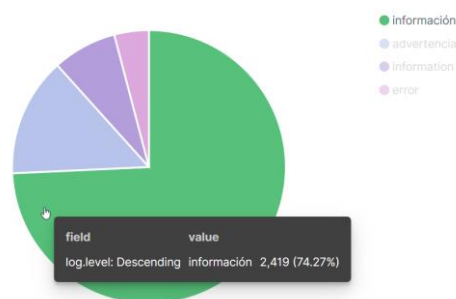


Figura 29.2 – Gráfico tipo pie I, junto a sus stats

Como vemos en este ejemplo, tenemos organizados todos los mensajes recogidos por los logs, organizados por su nivel de tipología: información, advertencia, information (son otro tipo de log de información de Windows 10) y errores. Si ponemos el cursor encima, como vemos en la figura 29.2, obtendremos el total de registros por sección.

En la figura 30 podemos ver otro ejemplo de este tipo de visualización, en este caso con los diferentes proveedores de servicios de Windows 10.

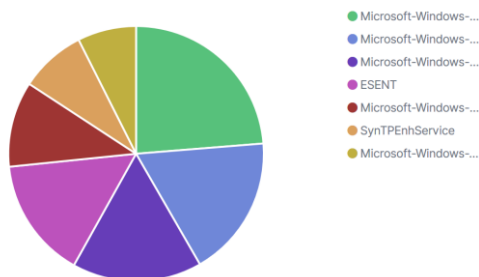


Figura 30 – Gráfico tipo pie II

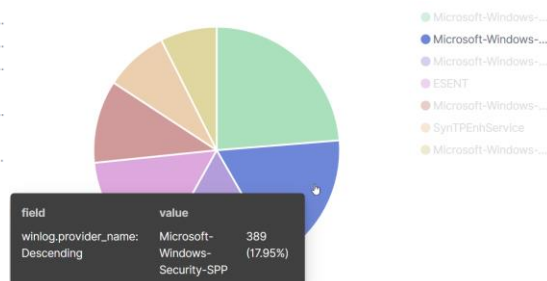


Figura 31 – Gráfico tipo pie II, junto a sus stats

Un ejemplo del gráfico tipo donut, en el que podemos añadir los usuarios de los que procede el mensaje, podría ser el siguiente:

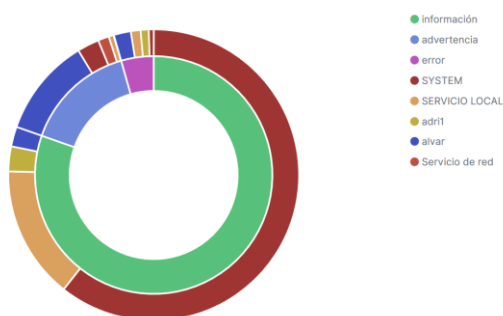


Figura 32 – Gráfico tipo pie donut

### • Diagrama Gauge

Casi todos los tipos de visualizaciones son igual de fácil de configurar, ya que los parámetros de configuración son los mismos. Queda a elección del usuario con cual prefiere trabajar para mostrar la información. El tipo gauge o medidor esta orientado para graficas relacionadas con tiempos. Por ejemplo, si queremos analizar el tiempo medio de Login de un usuario en una aplicación, este tipo de visualización seria el perfecto.

Aunque en realidad podemos usarlo para cualquier tipo de recuento.

Para nuestro ejemplo, vamos a probar otra funcionalidad, que veremos también más adelante en profundidad ya que aquí solo queremos mostrar como podemos visualizar la información tratada, y es la zona superior del menú, donde podemos insertar Querys y filtros manuales.

Lo primero que vamos a hacer es filtrar todos los mensajes dentro del nivel de advertencia, para ello simplemente realizamos la siguiente sentencia:

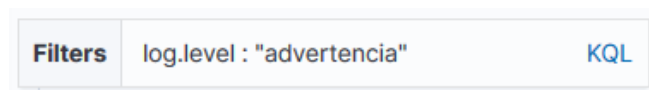


Figura 33 – Zona reservada a filtros y Querys



Como observamos, solo tenemos que poner el nombre del campo y el criterio. Si nos fijamos bien, a la derecha nos podemos encontrar la palabra “KQL” que viene a ser el lenguaje sql facilitado por Kibana para realizar estas consultas. En un principio (y podemos seleccionarlo si lo deseamos solo clicando en la palabra) estaba desarrollado para consultas orientadas a Lucene, pero al ser algo engorroso para muchos usuarios, Kibana desarrollo este tipo simple de lenguaje para que nos sea mas sencillo hacer filtros. Al igual que yo he puesto un filtro simple, podríamos elaborar sin ningún problema una query compleja.

Una vez seleccionado, creamos al igual que en el ejemplo del diagrama de tipo tarta, un Split slices con el campo sobre el que queremos contabilizar el numero de mensajes de tipo advertencia.

Realizado todo esto, simplemente tenemos que ejecutar y obtendremos el grafico:



Figura 34 – Grafico tipo Gauge

Como podemos ver en la figura 34 la mayor parte de registros de este tipo provienen de mi usuario, ya que soy el que normalmente maneja el equipo todos los días. Estos mensajes de advertencia pueden provenir de numerosos lugares, ya sea de actualizaciones, instalación de programas, fallos al introducir contraseñas etc.

No hace falta ni decir, que todos estos diagramas los podemos guardar, y según evolucione la información y Elastic se nutra de más log, más rica será nuestra visualización.

- **Diagramas de barras**

Estos son los gráficos que muestra por defecto Kibana cuando estamos en la pestaña principal de Discover.

Al igual que en el resto que hemos explicado, podremos crear Querys o diferentes agrupaciones la información, para poder tratarla.

Un ejemplo rápido es el siguiente:

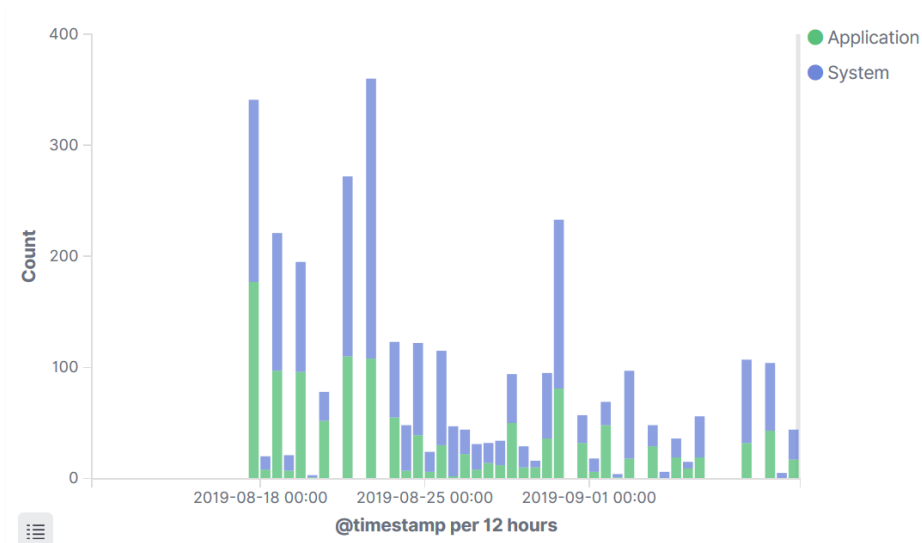


Figura 35 – Grafico tipo Gauge



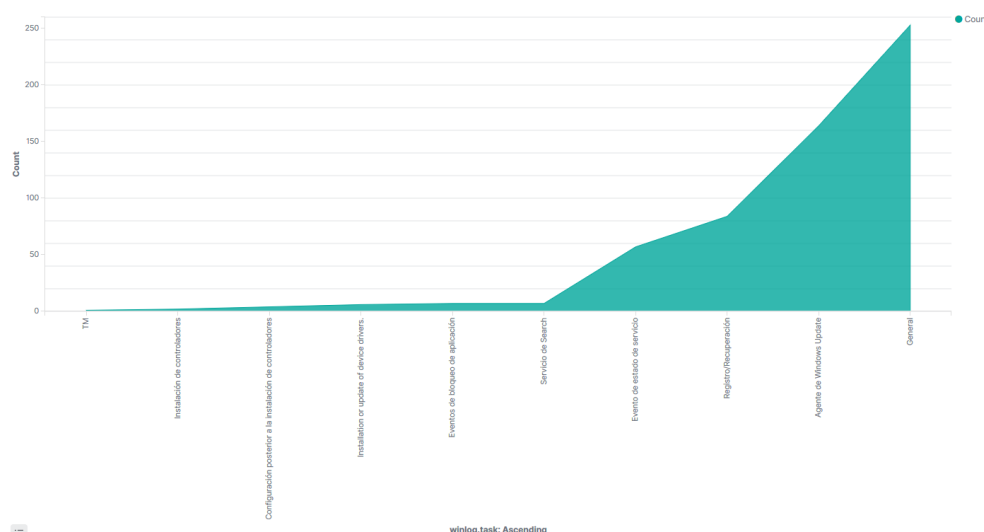
Aquí tenemos el número de eventos registrados en el tiempo, filtrando por canal: Ampliación o Sistema.

- **Diagrama de área**

Estos diagramas son similares a los diagramas de líneas de toda la vida, con la única diferencia que rellenan la parte inferior a esta con un determinado color.

Como siempre, el método de funcionamiento es simple, creando la visualización desde el menú, seleccionando desde el selector de Querys los filtros que deseemos, en el caso de que queramos poner alguno y añadiendo tantas Split Series dentro de los Buckets como campos deseemos graficar.

En el caso de ejemplo, vamos a graficar el top 10 tareas ejecutadas en Windows en el periodo de recolección de log, por lo que el eje de abscisas representara las diferentes tareas, y el eje de ordenadas representara el conteo de estas:



*Figura 36 – Grafico tipo Área*

Como podemos ver en la figura 36 la mayoría de las tareas realizadas han sido tareas de propósito general (tareas básicas de usuario, como por ejemplo realizar una búsqueda en internet) y la segunda tarea mas repetida han sido las actualizaciones a través del agente de Windows.

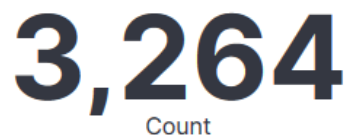
- **Otras tipologías**

Como hemos comprobado, crear visualizaciones a partir de los logs almacenados es algo relativamente sencillo, y el procedimiento de creación es prácticamente el mismo para todos los demás, por eso es un poco innecesario verlos todos.

Para cerrar esta parte de la simulación, vamos a comentar aquellos que son interesantes.

Por un lado, tenemos todos los relacionados con los mapas y geolocalización. Con Kibana, si tenemos algún tipo de variable que guarde la ubicación (no es nuestro caso, sino lo hubiéramos probado) podríamos crear mapas con las diferentes interacciones de los usuarios, pudiendo crear mapas de calor con zonas calientes y zonas frías.

Otra característica de visualización puede ser el de métricas, que simplemente nos muestran números con la información que deseamos, como este pequeño ejemplo que muestra el número de eventos.



*Figura 37 – Grafico tipo métrica*

También podremos sacar la información normal, como si fuera una query, con las visualizaciones de tablas de datos, como por ejemplo con el típico ejemplo que ya hemos realizado varias veces con diferentes visualizaciones, mostrando los diferentes niveles de organización de los logs:

Log Levels ↕	Count ↕
información	2,419
advertencia	464
information	247
error	134

*Figura 38 – Grafico tipo tabla*

Todo esto hace que el análisis de información sea mucho mas sencillo, ya que una vez creamos los modelos que deseamos tener, nunca más tendremos que volver a configurarlos, a no ser que queramos añadir parámetros claro.

Aquí podemos ver la versatilidad y potencia de esta herramienta, como hace que sea sencillo tratar la información, sin tener que ser necesariamente programador, analista o cualquier relación con el ámbito informático, quedando al alcance de todos.

## 2. Dashboard

Otra funcionalidad dentro de Kibana son los dashboards. Aquí podremos crear nuestro un panel de control personalizado, donde podremos insertar estas visualizaciones que hemos creado anteriormente.

Así podremos monitorizar fácilmente el comportamiento del sistema obteniendo una visión más global de todo.

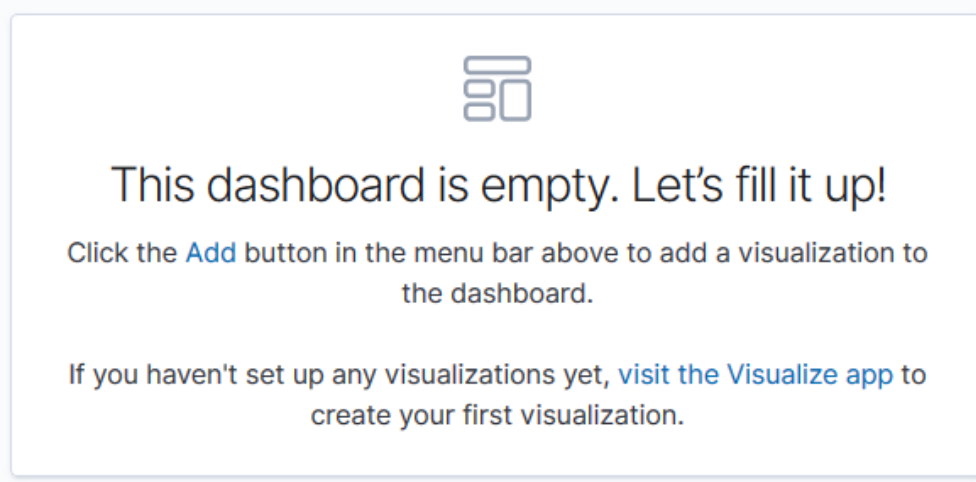
Para acceder a esta característica, tendremos que situarnos en el menú de la izquierda de Kibana, donde hemos visto ya la pestaña Discover y la pestaña de visualizaciones. Esta vez accederemos a las de “Dashboards”.

Posicionados, tendremos que crear un nuevo dashboard, como indica la figura 39.



*Figura 39 – Creación de dashboard*

Tras esto, tendremos nuestro panel de control listo para agregar visualizaciones:



*Figura 40 – Nuevo dashboard, sin visualizaciones.*

En mi caso, tengo preparados todos los anteriores que explicamos, junto a alguno extra que he ido realizando en diversas pruebas. Para añadirlos simplemente clicaremos en menú superior de la izquierda en add:

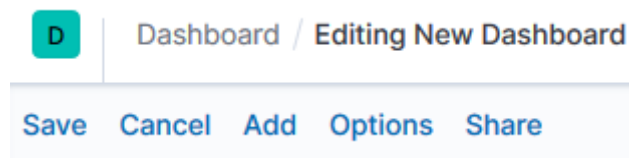


Figura 41 – Acciones extra en el Dashboard

y procederemos a añadir nuestras visualizaciones o búsquedas guardadas

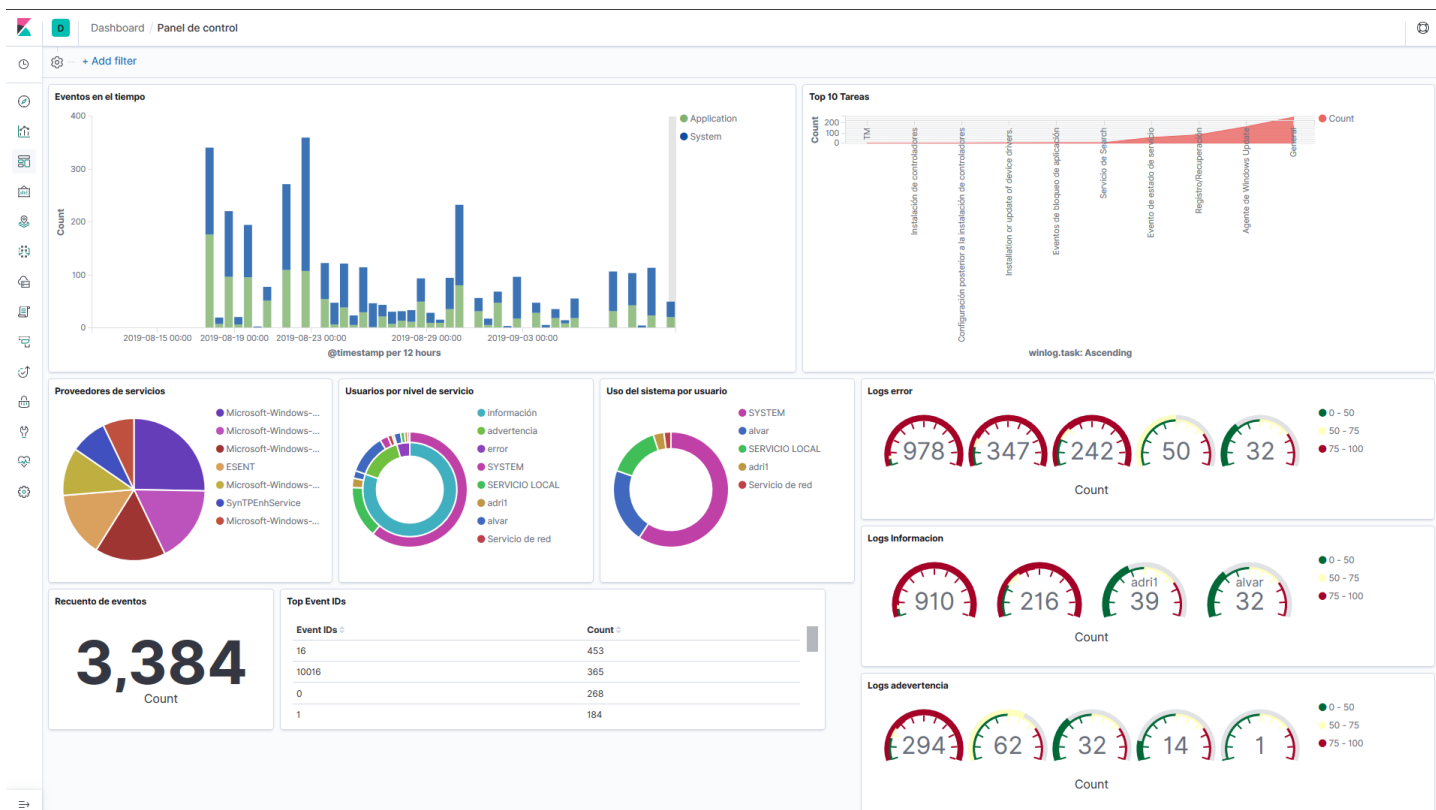


Figura 42 – Nuevo dashboard, sin visualizaciones.

El resultado final de la simulación, y como vemos en la figura 42, obtenemos un panel de mando con toda la información relativa al sistema, siendo monitorizada en tiempo real.

Si posicionamos el cursor sobre cualquier diagrama nos mostrará las estadísticas por punto, y si lo deseamos, podremos ponerlo a pantalla completa para poder analizarlo

Como vemos, tenemos estadísticas sobre el numero de logs que hemos introducido en el tiempo, el top 10 tareas utilizadas por nuestro equipo, los diferentes proveedores de servicios, dentro de los diferentes niveles de mensaje, cuales se producen en que usuario, también podemos ver la ocupación de cada usuario en el sistema operativo, diferentes diagramas de medidor, indicándonos que tipos de logs son errores, de información o de advertencia en función del usuario (no se ve el nombre del usuario porque he hecho a propósito pequeños los diagramas, para poder ver todos), también tendremos el recuento de eventos y para finalizar una pequeña tabla que muestra el top 5 task id del sistema, por si queremos ver que tareas son las más frecuentadas.

## 7.2 Simulación 2

En esta segunda simulación, vamos a mostrar el funcionamiento de Elastic Search a nivel de consulta, viendo las diferentes herramientas que podemos utilizar, y que nos proporcionará la API de Elastic Search.

Para que podamos desarrollar Querys mas completas, y poder explicar mejor cada tipo, vamos a partir de una base de datos ya completa que hemos conseguido de un repositorio de bases de datos para uso libre.

Concretamente hemos elegido una base de datos de bienes inmuebles, con campos como: referencia, fecha de alta, tipo, operación, provincia, superficie, precio de venta...

Pero tenemos una pega, esta base de datos no está disponible en ningún índice sino en formato .xls. ¿Podremos importarla a Elastic Search, para poder procesarla con Kibana?

La respuesta es sí, debido a la multitud de herramientas que complementan a Elastic Search, al ser como muchas veces hemos mencionado, una tecnología Open Source.

Para llevar a cabo esta tarea, vamos a utilizar Excelastic. Excelastic es una herramienta capaz de procesar ficheros, para que Elastic pueda crear la estructura de índices y poder importarla fácilmente a Kibana.

### Descarga y configuración

Para poder descargar nuestra herramienta, solo tendremos que ir al repositorio correspondiente de GitHub: <https://github.com/codingchili/excelastic/releases>

Una vez posicionados en la página, procederemos a descargar Excelastic, bajándonos el .jar más reciente.




 <a href="#">excelastic-1.3.6.jar</a>	29.3 MB
 <a href="#">Source code (zip)</a>	
 <a href="#">Source code (tar.gz)</a>	

Figura 43 – Repositorio de descarga de Excelastic

Para poder lanzar la herramienta, necesitaremos crear un fichero de configuración, de tipo .JSON, al igual que el resto de las herramientas del paquete Elastic Stack.

Excelastic también es una herramienta basada en una interfaz gráfica, por lo que también se lanzará a modo de Localhost (o la IP que deseemos), por lo que este será uno de los parámetros a configurar. El resto de los parámetros serán el puerto sobre el que trabaja Elastic Search, el puerto sobre el que queremos que trabaje la herramienta y mecanismos de autenticación en el caso de que deseemos añadirlos.

```
{} config.json •
Excelastic > {} config.json > ...
1  {
2    "web_port": 8200,
3    "elastic_port": 9200,
4    "elastic_host": "localhost",
5    "elastic_tls": false,
6    "authentication": false,
7    "basic": "username:password"
8  }
```

Figura 44 – Fichero de configuración de Excelastic

Una vez configurado, procedemos a iniciar la simulación.

Lo primero, será lanzar Elastic Search, para que este escuchando nuevas peticiones:

```
Windows PowerShell
PS C:\Users\alvar\TFG_Proyecto\ElasticSearch\bin> .\elasticsearch.bat
```

Figura 45 – Lanzamiento de Elastic Search

Una vez comprobado que Elastic esta listo, procedemos a lanzar Excelastic para poder generar nuestro índice a través del Excel que tenemos.

Por lo que tendremos que abrir un nuevo terminal, posicionarnos sobre la carpeta en la que hemos alojado el archivo .jar que hemos descargado con anterioridad y lanzar la orden: alojado el archivo .jar que hemos descargado con anterioridad y lanzar la orden: **java -Xmx2g -jar Excelastic-1.3.6.jar**

```
Windows PowerShell
PS C:\Users\alvar\TFG_Proyecto\Excelastic> java -Xmx2g -jar .\excelastic-1.3.6.jar
sep 01, 2019 4:18:03 PM com.codingchili.logging.ApplicationLogger startupMessage
INFORMACIÖN: Starting excelastic 1.3.6..
```

Figura 46 – Lanzamiento de Excelastic

Si todo el proceso ha transcurrido correctamente, si nos posicionamos en un navegador en el host y IP que indicamos en el fichero de configuración de Excelastic (en nuestro caso localhost y puerto 200), tendríamos que tener algo similar a la siguiente ventana:

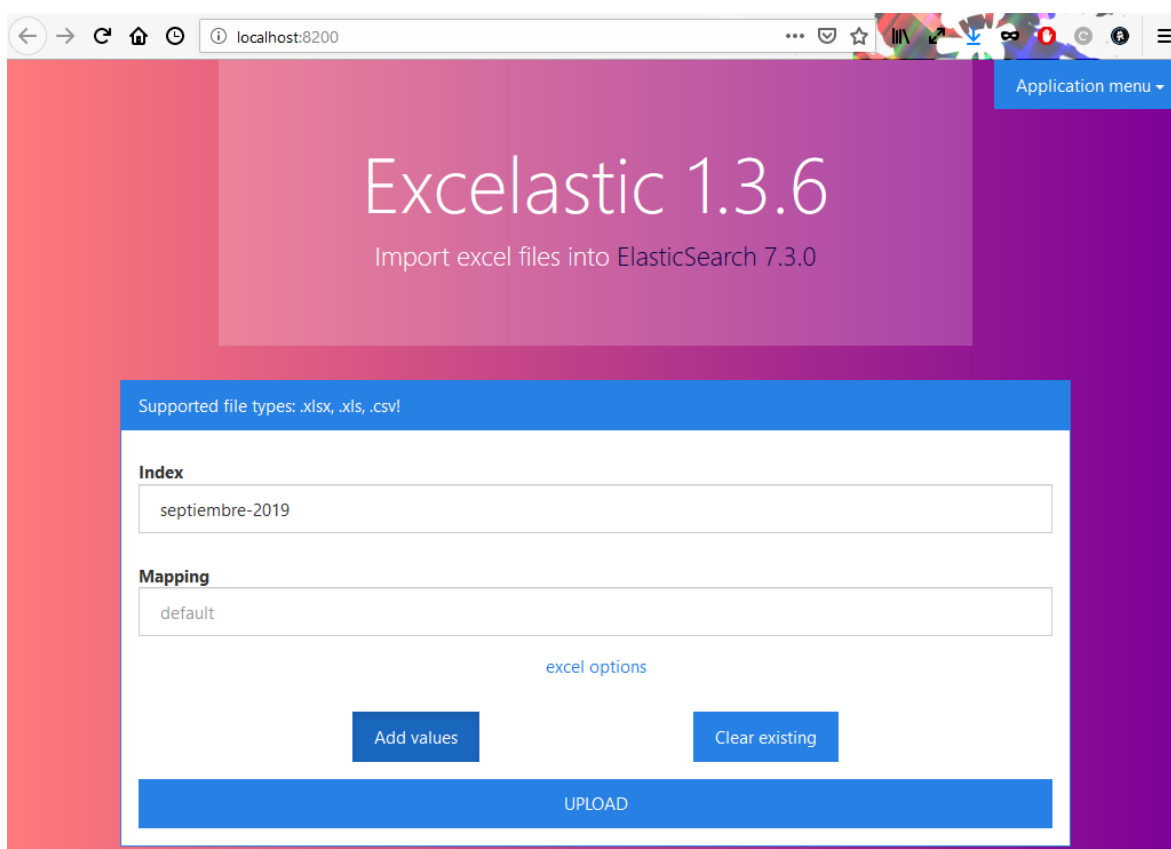


Figura 47 – Interfaz principal de Excelastic

Como podemos observar, la interfaz gráfica de Excelastic es muy sencilla, ya que solo tenemos que indicar el nombre del índice, y cargar un documento en formato xlsx, xls o csv.

En nuestro caso, vamos a poner de índice el nombre bienes\_inmuebles y procedemos a cargar el archivo.

Si todo ha ido bien, nos saldrá el mensaje:

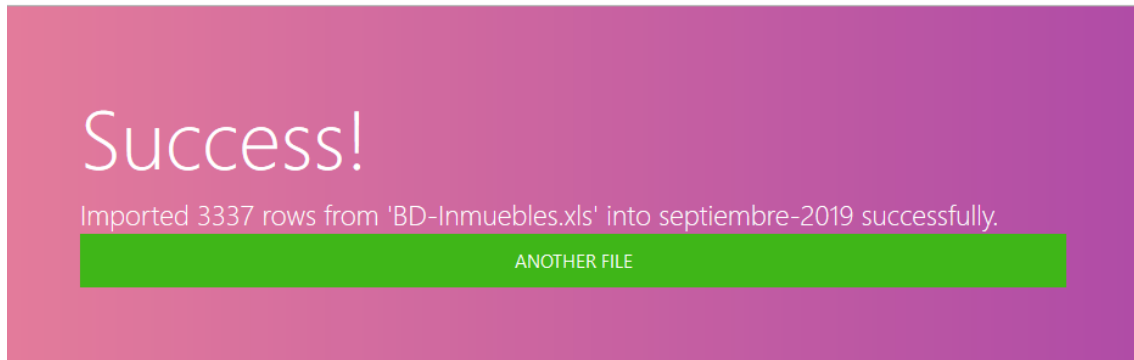


Figura 48 – Subida completada, en Excelastic.

Hay que tener en cuenta que esta herramienta, trata los nombres de los campos como el resto de las bases de datos, es decir, no podremos tener nombres de columnas con caracteres extraños o espacios, ya que la herramienta nos lanzaran un error sintáctico.

También podremos ver en Elastic Search si el índice ha sido creado correctamente, con es el caso:

```
[2019-09-01T16:33:29,619][INFO ][o.e.c.m.MetadataCreateIndexService] [ALVARO] [bienes_inmuebles] creating index, ca
use [auto(bulk api)], templates [], shards [1]/[1], mappings []
[2019-09-01T16:33:29,934][INFO ][o.e.c.m.MetadataMappingService] [ALVARO] [bienes_inmuebles/QhX3nHrvTpqEh6k0vRRJ_A]
create_mapping [default]
```

Figura 49 – Elastic Search, muestra de la creación de índice

Al igual que en Excelastic:

```
Windows PowerShell
INFORMACIEN: Submitted items [2175 -> 2303] of 3337 with result [200] OK into 'bienes_inmuebles' [69,0%]
sep 01, 2019 4:33:32 PM com.codingchili.logging.ApplicationLogger onImportedBatch
INFORMACIEN: Submitted items [2303 -> 2431] of 3337 with result [200] OK into 'bienes_inmuebles' [72,9%]
sep 01, 2019 4:33:32 PM com.codingchili.logging.ApplicationLogger onImportedBatch
INFORMACIEN: Submitted items [2431 -> 2559] of 3337 with result [200] OK into 'bienes_inmuebles' [76,7%]
sep 01, 2019 4:33:32 PM com.codingchili.logging.ApplicationLogger onImportedBatch
INFORMACIEN: Submitted items [2559 -> 2687] of 3337 with result [200] OK into 'bienes_inmuebles' [80,6%]
sep 01, 2019 4:33:32 PM com.codingchili.logging.ApplicationLogger onImportedBatch
INFORMACIEN: Submitted items [2687 -> 2815] of 3337 with result [200] OK into 'bienes_inmuebles' [84,4%]
sep 01, 2019 4:33:32 PM com.codingchili.logging.ApplicationLogger onImportedBatch
INFORMACIEN: Submitted items [2815 -> 2943] of 3337 with result [200] OK into 'bienes_inmuebles' [88,2%]
sep 01, 2019 4:33:33 PM com.codingchili.logging.ApplicationLogger onImportedBatch
INFORMACIEN: Submitted items [2943 -> 3071] of 3337 with result [200] OK into 'bienes_inmuebles' [92,1%]
sep 01, 2019 4:33:33 PM com.codingchili.logging.ApplicationLogger onImportedBatch
INFORMACIEN: Submitted items [3071 -> 3199] of 3337 with result [200] OK into 'bienes_inmuebles' [95,9%]
sep 01, 2019 4:33:33 PM com.codingchili.logging.ApplicationLogger onImportedBatch
INFORMACIEN: Submitted items [3208 -> 3336] of 3337 with result [200] OK into 'bienes_inmuebles' [100,0%]
sep 01, 2019 4:33:33 PM com.codingchili.logging.ApplicationLogger onImportedBatch
INFORMACIEN: Submitted items [3208 -> 3336] of 3337 with result [200] OK into 'bienes_inmuebles' [100,0%]
sep 01, 2019 4:33:33 PM com.codingchili.Controller.Website lambda$onComplete$9
INFORMACIEN: Imported file 'BD-Inmuebles.xls' successfully into 'bienes_inmuebles'.
```

Figura 50 – Excelastic, muestra de la importación correcta

Con estos sencillos pasos tendremos indexado completamente el fichero, pudiendo realizarlo todas las veces que deseemos, o incluso, si lo que queremos es actualizarlo, bastara con realizar de nuevo la importación poniendo el mismo nombre al índice generado.

Como ultima comprobación, podemos irnos al host y puerto sobre el que tenemos Elastic Search, y hacer una pequeña consulta para ver los índices disponibles. Para ello abrimos un navegador con los parámetros que tengamos configurados, y añadimos la orden `_cat/índices`, con la que simplemente usamos la librería `cat` para mostrar los índices disponible

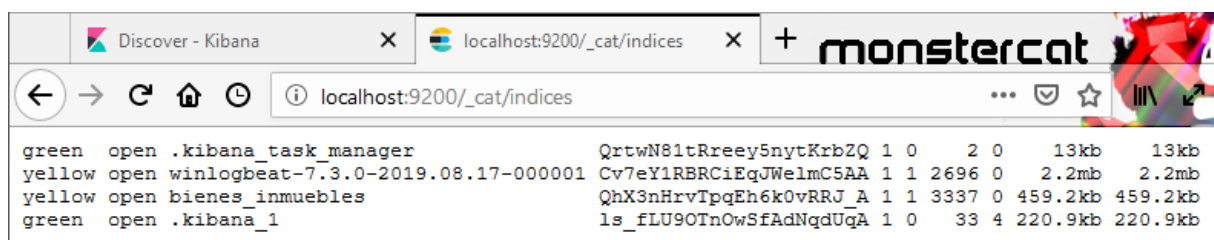


Figura 51 – Panel de control de Excelastic, junto al nuevo índice “bienes\_inmuebles” creado

Podemos comprobar que están correctamente los dos índices creados para las simulaciones. Como detalle, podemos observar que ambos nodos están en yellow esto es debido a que Elastic Stack recomienda tener más de un nodo por índice, por seguridad, pero para nuestra simulación basta con 1, por lo que no importa el estado.

Una vez tratada la información, vamos a llevarla a Kibana para poder trabajar con ella de manera más sencilla.

Para ello, abrimos Kibana a través de un nuevo terminal:

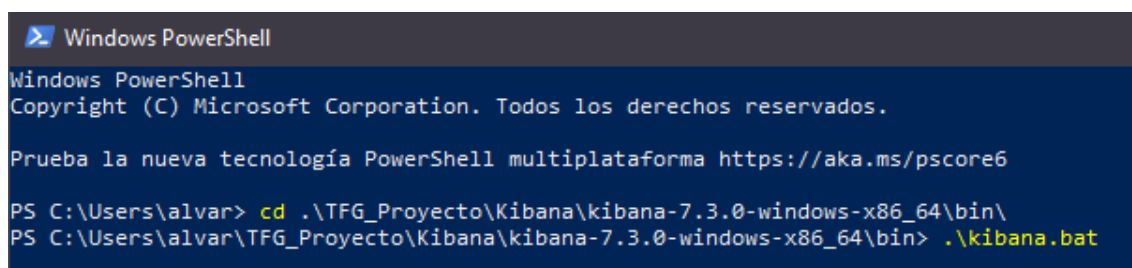


Figura 52 – Lanzamiento de Kibana

Cuando este listo, abrimos un navegador con el host y puerto correspondiente, y nos posicionamos en la pestaña “Management” dentro del menú que nos proporciona Kibana.

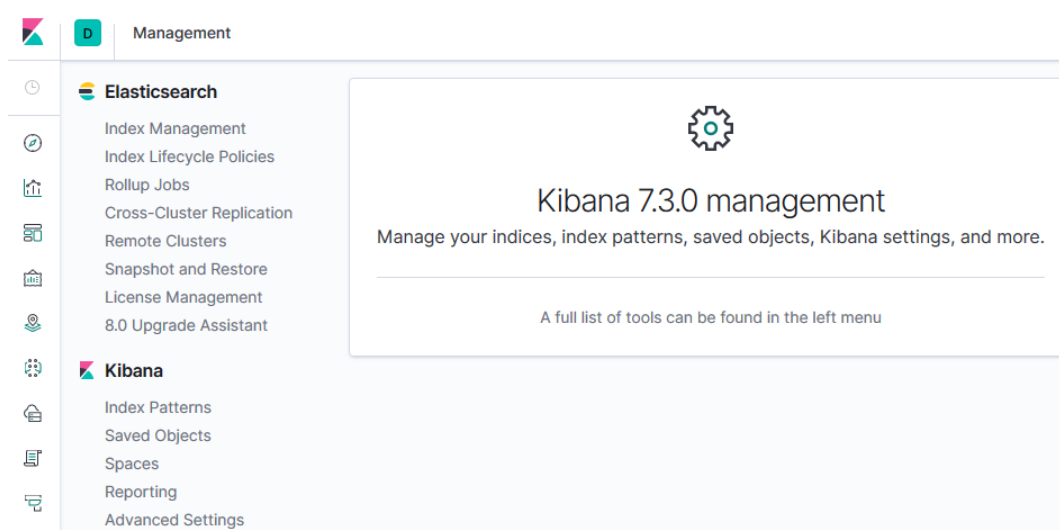


Figura 53 – Panel principal de Kibana



Dentro del apartado Kibana, iremos a Index Patterns, para configurar esta nueva tabla que hemos añadido, por lo que clicaremos sobre Create Index Patterns y seleccionaremos nuestra tabla:

## 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

bienes\_inmuebles\*

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

> Next step

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

**bienes\_inmuebles**

Rows per page: 10 ▾

Figura 54 – Creación del nuevo patrón de índices

Para mostrar la información de la tabla en los diferentes cronogramas, nos recomendamos poner un campo que tenga formato fecha, por lo que nosotros escogeremos la fecha de alta del servicio:

## Create index pattern

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

☐ Include system indices

### Step 2 of 2: Configure settings

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

Time Filter field name [Refresh](#)

Fecha\_Alta ▾

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 55 – Selección de campo “tiempo” a indexar

Como podemos comprobar en el figura 56, tendremos tras esto la selección de tipos de los diferentes campos que tenemos en la tabla, tal cual venían en el Excel origen que tomamos.

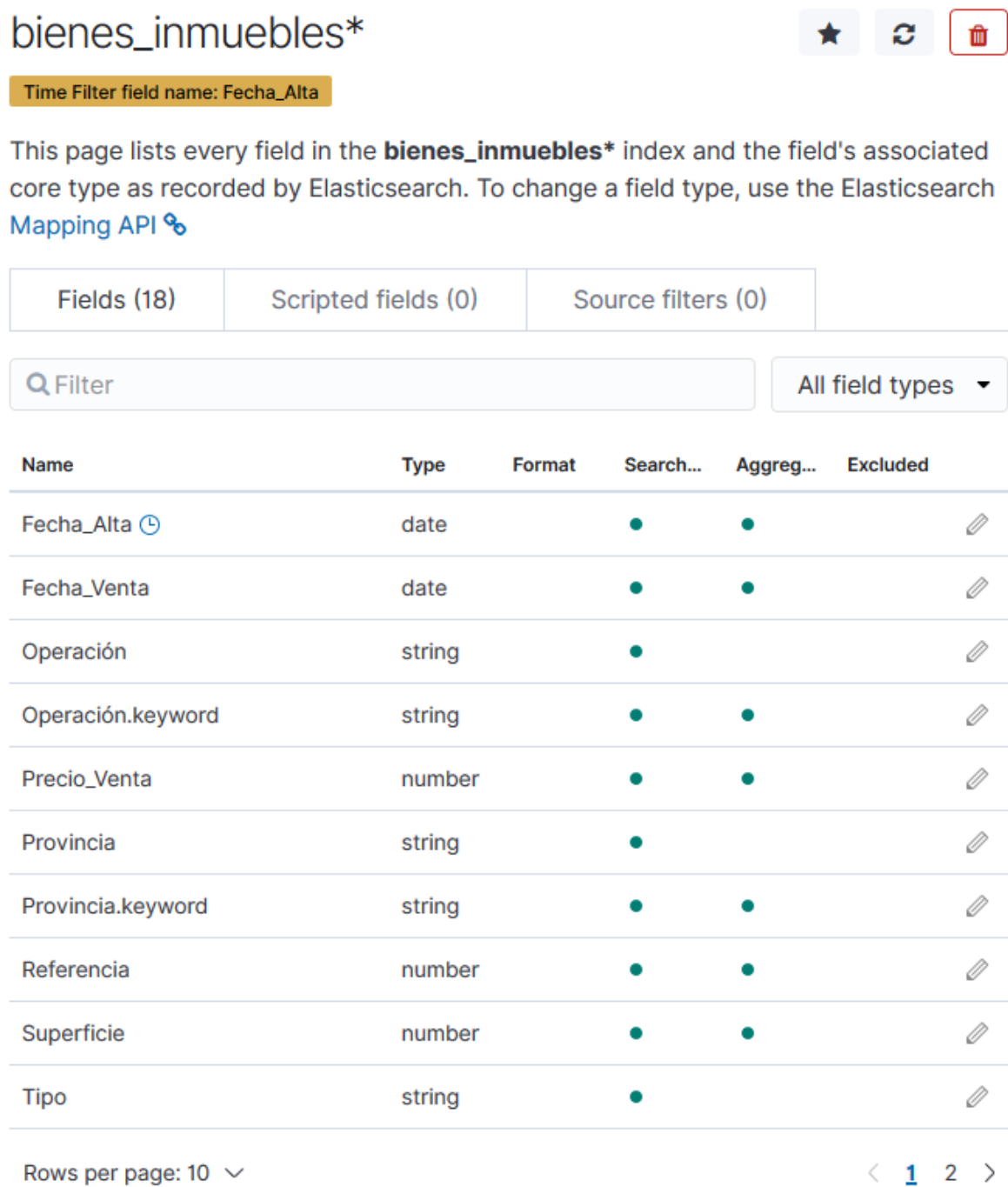


Figura 56 – Muestra de patrón de índices creado

Por lo que tenemos ya nuestra tabla lista.

## Queryys

El fin de esta simulación, como hemos explicado al inicio, es mostrar como funciona el lenguaje de consulta de Elastic Search.

Para poder realizar las comunes QUERYYS, podríamos realizar dos caminos. El menos atractivo, seria realizar las QUERYYS desde la interfaz de Elastic Search, pero a menudo resulta incómodo y poco útil, por lo que vamos a optar por la segunda opción, que es una opción dentro de la interfaz de Kibana, las Dev Tools.

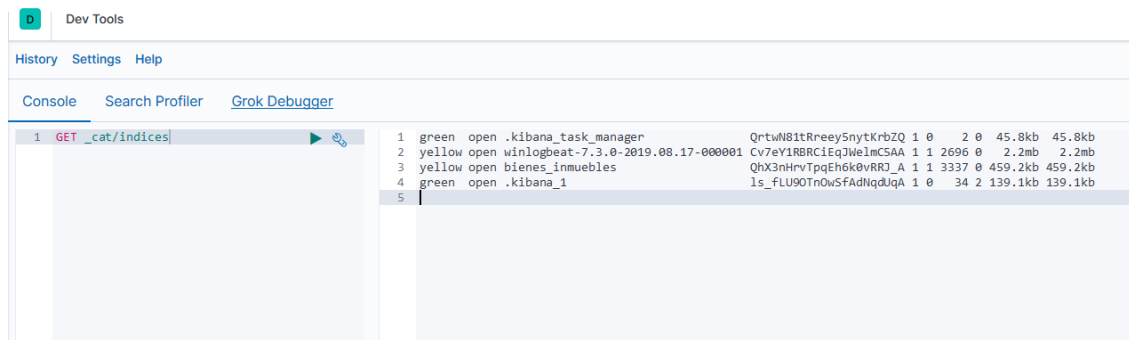


Figura 56 – Muestra de las dev tools (Interfaz de Kibana para tratar Queryys de Elastic Search)

Las Dev Tools son las herramientas que nos proporciona Elastic Search junto a Kibana para poder realizar nuestras consultas, como por ejemplo la que vemos en la imagen , en la que podemos ver los diferentes índices que tiene Elastic Search.

Algunos de los tipos mas importantes de Queryys que nos podemos encontrar son:

### 7.2.1 Query DSL

- **Query por matcheo**

Las Queryys de tipo DSL (Domain Specific Lenguaje), realizaran peticiones GET con formato .json para lanzar a nuestros índices y obtener una respuesta.

También son conocidas como Full Text Queryys, en la que la Query es analizada en función del mapping del campo sobre el que hacemos la Query. Es decir, analizara el texto que introducimos en nuestra Query, y la buscara por los diferentes tokens que encuentre en la tabla (Tokens que se generan tras su paso por Elastic Search), en el campo indicado

Como ejemplo, procedemos a realizar una Query que busque dentro de la tabla cuantas de las ventas de vienes e inmuebles se han producido en Barcelona.

Para ello, realizaremos la siguiente consulta:

```
GET /bienes_inmuebles/_search/
{
  "query": {
    "match": {
      "Provincia": "Barcelona"
    }
  }
}
```

Indicamos por lo tanto sobre qué índice buscar (si no indicamos el índice la búsqueda se realizará sobre todos los índices disponibles), y cuál es el match para analizar, en este caso Barcelona.

Como salida obtendremos un JSON con varias partes, el número de impactos, en este caso 837 como se aprecia en la imagen

```
{
  "took" : 1,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 837,
      "relation" : "eq"
    }
  }
}
```

*Figura 57 – Resultado de Query por matcheo, impactos*

Y además todos aquellos los documentos en los que aparece la palabra que hemos buscado:

```

    "max_score" : 1.3827058,
    "hits" : [
      {
        "_index" : "bienes_inmuebles",
        "_type" : "default",
        "_id" : "4vI-7WwBzIHqyCBxowc",
        "_score" : 1.3827058,
        "_source" : {
          "Referencia" : 13.0,
          "Fecha_Alta" : "2004-01-03T23:00:00Z",
          "Tipo" : "Parking",
          "Operación" : "Venta",
          "Provincia" : "Barcelona",
          "Superficie" : 300.0,
          "Precio_Venta" : 2937300.0,
          "Fecha_Venta" : "2004-11-03T23:00:00Z",
          "Vendedor" : "Jesús"
        }
      },
      {
        "_index" : "bienes_inmuebles",
        "_type" : "default",
        "_id" : "5_I-7WwBzIHqyCBxowc",
        "_score" : 1.3827058,
        "_source" : {
          "Referencia" : 18.0,
          "Fecha_Alta" : "2004-01-05T23:00:00Z",
          "Tipo" : "Suelo",
          "Operación" : "Venta",
          "Provincia" : "Barcelona",
          "Superficie" : 283.0,
          "Precio_Venta" : 1679605.0,
          "Fecha_Venta" : "2004-06-05T22:00:00Z",
          "Vendedor" : "Carmen"
        }
      },
      {
        "_index" : "bienes_inmuebles",
        "_type" : "default",
        "_id" : "6fI-7WwBzIHqyCBxowc",
        "_score" : 1.3827058,

```

Figura 58 – Resultado de Query por matcheo, resultados

Dentro de este tipo de consultas, también podremos realizar multi-match, para poder hacer búsquedas de un mismo dato sobre más de un campo, como, por ejemplo:

```

GET /bienes_inmuebles/_search/
{
  "query": {
    "multi_match": {
      "query": "70",
      "fields":
        ["Referencia","Superficie"]
    }
  }
}

```

Aquí buscaremos a la vez, la referencia 70, y aquellos inmuebles cuya superficies sean 70 metros (no es una consulta con lógica, simplemente es para mostrar la funcionalidad multi-match), por lo que obtendremos aquellos resultados que tengan un campo u otro con ese valor, o ambos a la vez si los hubiese.

```

{
  "_index" : "bienes_inmuebles",
  "_type" : "default",
  "_id" : "8PI-7WwBzIHyqhCBxowc",
  "_score" : 1.0,
  "_source" : {
    "Referencia" : 27.0,
    "Fecha_Alta" : "2004-01-10T23:00:00Z",
    "Tipo" : "Local",
    "Operación" : "Alquiler",
    "Provincia" : "Tarragona",
    "Superficie" : 70.0,
    "Precio_Venta" : 659330.0,
    "Fecha_Venta" : "2004-12-22T23:00:00Z",
    "Vendedor" : "Pedro"
  }
},
{
  "_index" : "bienes_inmuebles",
  "_type" : "default",
  "_id" : "G_I-7WwBzIHyqhCBxo0c",
  "_score" : 1.0,
  "_source" : {
    "Referencia" : 70.0,
    "Fecha_Alta" : "2004-01-27T23:00:00Z",
    "Tipo" : "Casa",
    "Operación" : "Alquiler",
    "Provincia" : "Tarragona",
    "Superficie" : 179.0,
    "Precio_Venta" : 1647695.0,
    "Fecha_Venta" : "2004-07-11T22:00:00Z",
    "Vendedor" : "María"
  }
}

```

Figura 58 – Resultado de Query por multi-match, resultados

- **Term Query**

Este es otro tipo de extracción, complementaria de las Querys por matcheo. Prácticamente tienen funcionamientos similares, pero con una gran excepción. Mientras que las Match Querys van a buscar una palabra concreta, y van a realizar la búsqueda entre los diferentes tokens generados a partir de las tablas, las Term Querys van a buscar literalmente el termino, no el token.

Pongamos un ejemplo:

Si deseamos buscar la palabra “Error” con las match Querys podríamos encontrar todos aquellos documentos que contienen la palabra dentro de una cadena de texto, o simplemente la palabra sin más.

Sin embargo, si buscamos la misma frase empleando Term Querys, solo tendríamos resultados cuando el campo es literalmente esa frase:

- Ejemplo de tabla:

Usuario	Acción	Estado	Fecha
Alvaro	Login	Correcto.	21/04/19
Juan	Login	Error, contraseña incorrecta.	12/08/18
Laura	Login	Error	28/07/18
Andrés	Login	Error, usuario desconocido.	09/03/19
Pepe	Login	Error.	11/04/18

- Salida, en función de la Query elegida

Campo	Búsqueda	Match Query	Term Query
Usuario	"Error"	<p>Usuario: Juan Acción: Login Estado: <b>Error</b>, contraseña incorrecta Fecha: 12/08/18</p> <p>Usuario: Laura Acción: Login Estado: <b>Error</b> Fecha: 28/07/18</p> <p>Usuario: Andrés Acción: Login Estado: <b>Error</b>, usuario desconocido Fecha: 09/03/19</p> <p>Usuario: Pepe Acción: Login Estado: <b>Error</b> Fecha: 11/04/18</p>	<p>Usuario: Laura Acción: Login Estado: <b>Error</b> Fecha: 28/07/18</p> <p>Usuario: Pepe Acción: Login Estado: <b>Error</b> Fecha: 11/04/18</p>

Como lamentablemente la base de datos de bienes inmuebles no contiene ninguna cadena de texto, no vamos a poder realizar la simulación sobre ella, para poder ver este tipo de Query.

Pero, como en la tabla que utilizamos en la primera simulación, donde monitorizamos el sistema operativo si contiene campos que sean cadenas de texto, vamos a desplazarnos hasta ella para realizar la simulación:

Para ello, vamos a utilizar el campo "message", que almacena los mensajes que recogen los log, como pueden ser errores, actualizaciones, instalación de programas...

- Probemos la búsqueda con el tipo Match Query

```
GET /bienes_inmuebles/_search/
{
  "query": {
    "match": {
      "Provincia": "Barcelona"
    }
  }
}
```

Si comprobamos el .JSON de salida de la Query, podemos comprobar resultados como los siguientes:

```
"message" : "Error en la notificación de
perfil del evento Create para el
componente {2c86c843-77ae-4284-9722
-27d65366543c}; el código de error es
No implementado\n. ",
```

Figura 59 – Resultado de Term Query, resultados parte I

```
"message" : "\"\"ShellExperienceHost
(15576,P,98) TILEREPOSITORYS-1-5-21
-3906426483-4006489556-3598251100-1003
: Al intentar abrir el dispositivo con
el nombre '\\.\C:\" que contiene \"C:\",
se produjo un error del sistema 5
(0x00000005): \"Acceso denegado. \". La
operación se cerrará con el error
-1032 (0xfffffbf8).\"\"\",
```

Figura 60 – Resultado de Term Query, resultados parte II

```
"message" : "ShellExperienceHost (15576
,P,98) TILEREPOSITORYS-1-5-21
-3906426483-4006489556-3598251100-1003
: Al intentar abrir el dispositivo con
el nombre \"\"\"\"\\.\C:\" que contiene \"C
:\", se produjo un error del sistema 5
(0x00000005): \"Acceso denegado. \". La
operación se cerrará con el error
-1032 (0xfffffbf8).\"\"\",
```

Figura 61 – Resultado de Term Query, resultados parte III

Como observamos, todos han encontrado la palabra error dentro del campo mensaje, pero no solo contienen esta, sino muchas mas palabras que forman el mensaje completo.

Sin embargo, si realizamos la misma Query, pero de tipo Term Query o búsqueda por termino:

```
GET /winlogbeat*/_search/
{
  "query":{
    "term":{"message":"Error"}
  }
}
```



No nos reportaría nada, ya que no hay ningún mensaje dentro de todos los logs recogidos que solo contenga la palabra error,

```
{
  "took" : 0,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 0,
      "relation" : "eq"
    },
    "max_score" : null,
    "hits" : [ ]
  }
}
```

Figura 62 – Resultado de Term Query tipo búsqueda

Para que encontráramos resultados, tendríamos que hacer una búsqueda más específica:

```
GET /winlogbeat*/_search/
{
  "query":{
    "term":{"message":"Error en la notificación de perfil del evento Create para el
componente {2c86c843-77ae-4284-9722-27d65366543c}; el código de error es No
implementado\n. "}
  }
}
```

```
"message" : "Error en la notificación de
perfil del evento Create para el
componente {2c86c843-77ae-4284-9722-
27d65366543c}; el código de error es
No implementado\n. ",
"winlog" : {
  "event_id" : 1534,
  "user" : {
    "name" : "SYSTEM",
    "domain" : "NT AUTHORITY",
    "type" : "Well Known Group",
    "identifier" : "S-1-5-18"
  },
  "event_data" : {
    "Component" : "{2c86c843-77ae-4284-
9722-27d65366543c}",
    "Error" : "No implementado",
    "Event" : "Create"
  }
},
```

Figura 63 – Resultado de Term Query tipo búsqueda específica

Ahora si encontraríamos resultados, al hacer una búsqueda literal de la cadena.

- **Range Query**

Este tipo de Query es bastante sencilla. Con ella podremos realizar operaciones relacionales.

- IT: menor que
- ITE: menor o igual que
- GT: mayor que
- GTE mayor o igual que

Como ejemplo veremos aquellos registros que cuya superficie sea mayor que 295 m, para ello realizaremos la QUERY:

```
GET
/bienes_inmuebles/_search/
{
  "query":{
    "range":{
      "Superficie":{
        "gte":295
      }
    }
  }
}
```

Como resultado tendremos, como apreciamos en la figura 64, 70 registros que cumplen la condición, mostrando como ejemplo alguno de estos registros.

Esta Query es muy útil siempre que queramos trabajar con rangos, ya que no solo podemos realizar las operaciones de mayor que y menor que, también podremos combinarlos para crear una lógica mas precisa, pudiendo ser por ejemplo la superficie mayor que 100 metros y menos que 200.

```
{
  "took" : 1,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 70,
      "relation" : "eq"
    },
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "bienes_inmuebles",
        "_type" : "default",
        "_id" : "3_I-7WwBzIHqyqCBxowc",
        "_score" : 1.0,
        "_source" : {
          "Referencia" : 10.0,
          "Fecha_Alta" : "2004-01-03T23:00:00Z",
          "Tipo" : "Parking",
          "Operación" : "Venta",
          "Provincia" : "Lleida",
          "Superficie" : 299.0,
          "Precio_Venta" : 2042768.0,
          "Fecha_Venta" : "2004-10-05T22:00:00Z",
          "Vendedor" : "Joaquín"
        }
      },
      {
        "_index" : "bienes_inmuebles",
        "_type" : "default",
        "_id" : "4vI-7WwBzIHqyqCBxowc",
        "_score" : 1.0,
        "_source" : {
          "Referencia" : 13.0,
          "Fecha_Alta" : "2004-01-03T23:00:00Z",
          "Tipo" : "Parking",
          "Operación" : "Venta",
          "Provincia" : "Barcelona",
          "Superficie" : 300.0,
          "Precio_Venta" : 2937300.0,
          "Fecha_Venta" : "2004-10-05T22:00:00Z",
          "Vendedor" : "Joaquín"
        }
      }
    ]
  }
}
```

Figura 64 – Resultado de Term Query tipo range

### 7.2.2 Elasticsearch SQL

Tras explicar las Querys DSL, procedemos a explicar las Elasticsearch SQL, el tipo de QUERY más solicitada por los desarrolladores dentro de Elastic Search.

El por qué es sencillo, con este tipo de consulta podremos llevar el lenguaje de la Query al lenguaje SQL, es decir, que podremos hacer consultas como las que podemos hacer con cualquier gestor SQL.

Siguiendo con el índice de bienes inmuebles, vamos a realizar un nuevo ejemplo.

Pongamos que queremos consultar todas las operaciones realizadas por el vendedor Pedro, siempre y cuando estas tengan más de 150 metros de superficie y se hayan realizado en Girona (No mostraremos todos los campos de la tabla para que no ocupe demasiado espacio en el documento).

La Query para realizar esto, sería algo como esto:

```
POST _sql
{
  "query":
  "SELECT Fecha_Venta, Precio_Venta, Provincia, Referencia, Superficie, Tipo,
  Vendedor
  FROM bienes_inmuebles
  WHERE Vendedor='Pedro' AND Provincia= 'Girona' and Superficie > 150 "
}
```

Antes de ver la salida, cabe destacar que en este ejemplo hemos usado la orden POST en vez de GET ya que en el cuerpo del mensaje es donde va a ir la consulta que vamos a enviar, junto al end-point que en ese caso es SQL.

Una vez realizamos la Query, solo tenemos que ver la salida, la cual tiene un problema:

Como podemos ver en la figura 65 la Query nos devuelve la información en un documento json, lo que a menudo es difícil de interpretar...

Esto es solucionable, añadiendo dentro de la sentencia post el formato en el que queremos que nos devuelva el resultado de la consulta.

Los formato que la API de Elastic Search nos proporciona, para la salida de la Query pueden ser: csv, json, tsv, txt, yaml, chor o smile.

En nuestro caso vamos a seleccionar el formato más fácil de interpretar a simple vista como es el txt, aunque si quisiéramos exportar la información, por ejemplo, a un Excel el más recomendable podría ser el formato csv (delimitado por comas).

```
"rows" : [
  [
    "2003-12-31T23:00:00.000Z",
    "2004-04-18T22:00:00.000Z",
    "Venta",
    1945424.0,
    "Girona",
    2.0,
    199.0,
    "Local",
    "Pedro"
  ],
  [
    "2004-01-03T23:00:00.000Z",
    "2004-11-28T23:00:00.000Z",
    "Alquiler",
    820336.0,
    "Girona",
    15.0,
    176.0,
    "Industrial",
    "Pedro"
  ],
  [
    "2004-01-15T23:00:00.000Z",
    "2004-07-08T22:00:00.000Z",
    "Venta",
    1160572.0,
    "Girona",
    37.0,
    229.0,
    "Local",
    "Pedro"
  ],
]
```

Figura 65 – Resultado Query tipo elasticsearch sql, ejemplo I

Explicado esto, la Query nos quedaría tal que:

```
POST _sql?format=txt
{
  "query":
  "SELECT Fecha_Venta, Precio_Venta, Provincia, Referencia, Superficie, Tipo,
  Vendedor
  FROM bienes_inmuebles
  WHERE Vendedor='Pedro' AND Provincia= 'Girona' and Superficie > 150 "
}
```

Obteniendo una salida de la siguiente forma:

1	Fecha_Venta	Precio_Venta	Provincia	Referencia	Superficie	Tipo	Vend
2							
3	2004-04-18T22:00:00.000Z	1945424.0	Girona	2.0	199.0	Local	Pedro
4	2004-11-28T23:00:00.000Z	820336.0	Girona	15.0	176.0	Industrial	Pedro
5	2004-07-08T22:00:00.000Z	1160572.0	Girona	37.0	229.0	Local	Pedro
6	2004-07-23T22:00:00.000Z	2685816.0	Girona	90.0	292.0	Suelo	Pedro
7	2004-07-29T22:00:00.000Z	1461356.0	Girona	123.0	179.0	Local	Pedro
8	2004-10-01T22:00:00.000Z	1403595.0	Girona	155.0	185.0	Suelo	Pedro
9	2004-06-30T22:00:00.000Z	1561432.0	Girona	156.0	284.0	Casa	Pedro
10	2004-08-13T22:00:00.000Z	720882.0	Girona	198.0	174.0	Oficina	Pedro
11	2004-11-08T23:00:00.000Z	1050985.0	Girona	201.0	247.0	Parking	Pedro
12	2004-07-12T22:00:00.000Z	1219158.0	Girona	215.0	214.0	Parking	Pedro
13	2004-11-19T23:00:00.000Z	1079449.0	Girona	275.0	193.0	Local	Pedro
14	2004-07-27T22:00:00.000Z	2699136.0	Girona	297.0	288.0	Suelo	Pedro
15	2005-03-21T23:00:00.000Z	1171894.0	Girona	302.0	211.0	Industrial	Pedro
16	2004-08-11T22:00:00.000Z	977740.0	Girona	321.0	190.0	Industrial	Pedro
17	2004-08-18T22:00:00.000Z	1707944.0	Girona	357.0	196.0	Suelo	Pedro
18	2004-09-12T22:00:00.000Z	830760.0	Girona	386.0	184.0	Parking	Pedro
19	2005-03-03T23:00:00.000Z	1164375.0	Girona	393.0	225.0	Piso	Pedro
20	2004-10-01T22:00:00.000Z	1159470.0	Girona	448.0	234.0	Local	Pedro
21	2005-05-31T22:00:00.000Z	1962380.0	Girona	463.0	214.0	Casa	Pedro
22	2005-04-01T22:00:00.000Z	1316073.0	Girona	497.0	209.0	Oficina	Pedro
23	2005-01-15T23:00:00.000Z	1313526.0	Girona	517.0	174.0	Industrial	Pedro
24	2004-10-30T22:00:00.000Z	2215002.0	Girona	551.0	237.0	Piso	Pedro
25	2005-05-09T22:00:00.000Z	1816437.0	Girona	632.0	201.0	Piso	Pedro
26	2005-03-27T22:00:00.000Z	1153173.0	Girona	648.0	267.0	Oficina	Pedro
27	2005-04-19T22:00:00.000Z	1520627.0	Girona	665.0	163.0	Local	Pedro
28	2005-08-03T22:00:00.000Z	1392168.0	Girona	674.0	152.0	Parking	Pedro
29	2005-08-26T22:00:00.000Z	1122240.0	Girona	764.0	210.0	Oficina	Pedro
30	2005-08-28T22:00:00.000Z	2543625.0	Girona	891.0	255.0	Casa	Pedro
31	2005-07-17T22:00:00.000Z	1653456.0	Girona	901.0	294.0	Parking	Pedro
32	2005-05-16T22:00:00.000Z	1962360.0	Girona	939.0	216.0	Industrial	Pedro
33	2005-09-12T22:00:00.000Z	1447551.0	Girona	946.0	243.0	Piso	Pedro
34	2005-07-06T22:00:00.000Z	1725597.0	Girona	948.0	237.0	Piso	Pedro
35	2005-03-10T23:00:00.000Z	2153852.0	Girona	950.0	233.0	Piso	Pedro
36	2005-11-29T23:00:00.000Z	2273762.0	Girona	981.0	278.0	Local	Pedro
37	2005-06-23T22:00:00.000Z	2808008.0	Girona	1066.0	287.0	Suelo	Pedro
38	2005-12-13T23:00:00.000Z	1339044.0	Girona	1122.0	228.0	Piso	Pedro
39	2005-12-01T23:00:00.000Z	1966272.0	Girona	1152.0	224.0	Local	Pedro
40	2006-01-21T23:00:00.000Z	1136502.0	Girona	1168.0	206.0	Piso	Pedro
41	2005-11-03T23:00:00.000Z	1599233.0	Girona	1273.0	283.0	Piso	Pedro
42	2006-04-08T22:00:00.000Z	1702590.0	Girona	1299.0	290.0	Local	Pedro
43	2005-07-19T22:00:00.000Z	2589312.0	Girona	1307.0	264.0	Industrial	Pedro
44	2005-07-13T22:00:00.000Z	1059680.0	Girona	1311.0	185.0	Casa	Pedro

Figura 66 – Resultado Query tipo elasticsearch sql, formato txt

Como vemos, la información con el formato txt es mucho más sencilla de leer para un usuario común, por lo que, si simplemente queremos realizar una consulta puntual, este formato es el idóneo.

Para ver que efectivamente este tipo de Query es una de las más recomendables si tenemos experiencia en el uso de SQL, vamos a realizar otra consulta para ver como este tipo de herramienta acepta casi cualquier tipo de instrucción.

En la siguiente consulta vamos a buscar todos aquellos vendedores, registrados (por lo que vamos a obviar todas aquellas operaciones que no tienen asignado un vendedor), y vamos a contar las operaciones que han realizado en las diferentes ciudades.

Para ello, escribimos la siguiente Query:

```
POST _sql?format=txt
{
  "query":
  "SELECT Vendedor, COUNT(Provincia) as VentasPorProvincia
  FROM bienes_inmuebles
  where Vendedor is not null
  GROUP BY Vendedor,Provincia "
```

Como resultado de esta, obtendremos:

1	Vendedor	VentasPorProvincia
2		
3	Carmen	105
4	Carmen	124
5	Carmen	117
6	Carmen	132
7	Jesús	126
8	Jesús	104
9	Jesús	103
10	Jesús	117
11	Joaquín	130
12	Joaquín	93
13	Joaquín	98
14	Joaquín	116
15	Luisa	129
16	Luisa	132
17	Luisa	108
18	Luisa	117
19	María	113
20	María	136
21	María	98
22	María	103
23	Pedro	95
24	Pedro	129
25	Pedro	118
26	Pedro	116
27		

Figura 67 – Resultado Query tipo elasticsearch sql, ejemplo II

Si quisiéramos ver todos los tipos de funciones sql compatibles dentro de Elastic, solo tendríamos que dirigirnos a su api y comprobarlo, ya que en la documentación viene explicado perfectamente cada comando, y que hace cada uno.

Como vemos Elastic Search tiene todo tipo de consultas disponibles, incluso las típicas que todos hemos realizado alguna vez en SQL, convirtiéndola por lo tanto en una herramienta muy potente y versátil, con la que podemos ahorrarnos mucho trabajo.

Finalmente, y a modo de curiosidad, este tipo de query tiene una rara funcionalidad, ya que es posible transformar la consulta de tipo Elasticsearch SQL a su consulta equivalencia dentro de las consultas de tipo DSL.

Para ello, y reutilizando la query anterior, solo tendremos que pasar como argumento la sentencia /traslate:

```
POST _sql/translate
{
  "query":
  "SELECT Vendedor, COUNT(Provincia) as VentasPorProvincia
  FROM bienes_inmuebles
  where Vendedor is not null
  GROUP BY Vendedor,Provincia "
}
```

Como salida obtendríamos la query DSL Equivalente:

```
1 {
2   "size" : 0,
3   "query" : {
4     "exists" : {
5       "field" : "Vendedor",
6       "boost" : 1.0
7     }
8   },
9   "source" : false,
10  "stored_fields" : "_none_",
11  "aggregations" : {
12    "groupby" : {
13      "composite" : {
14        "size" : 1000,
15        "sources" : [
16          {
17            "699" : {
18              "terms" : {
19                "field" : "Vendedor.keyword",
20                "missing_bucket" : true,
21                "order" : "asc"
22              }
23            }
24          },
25          {
26            "693" : {
27              "terms" : {
28                "field" : "Provincia.keyword",
29                "missing_bucket" : true,
30                "order" : "asc"
31              }
32            }
33          }
34        ]
35      },
36      "aggregations" : {
37        "693" : {
38          "filter" : {
39            "exists" : {
40              "field" : "Provincia",
41              "boost" : 1.0
42            }
43          }
44        }
45      }
46    }
47  }
48 }
```

Figura 68 – Resultado Query tipo elasticsearch sql, creando query DSL equivalente

### 7.2.3 Agregaciones

Las agregaciones nos van a permitir crear agrupaciones a partir de las salida que nos van a dar las diferentes QUEYS en base a unos criterios específicos.

Hay multitud de tipos de agregaciones, factor que ha hecho destarar a Elastic Search respecto al resto de herramientas dentro de la monitorización y analítica de logs.

Alguno de los muchos tipos de agregaciones que nos podemos encontrar son:

- Metrics Aggregations:
- Bucket Aggregations:
- Pipeline Aggregations:
- Matrix Aggregations:
- Caching heavy aggregations:
- Returning only aggregation results:
- Aggregation Metadata:
- Returning the type of the aggregation:

A pesar de que Kibana es el sitio ideal para realizar más fácilmente agregaciones de forma simple y automática, vamos a hacer unos ejemplo para entender que son y como funcionan.

Partiendo, por ejemplo, de la query que utilizamos cuando explicamos las range Querys, podríamos elaborar la siguiente consulta:

```
GET /bienes_inmuebles/_search/
{
  "query":{
    "range":{
      "Superficie":{
        "gte":295
      }
    }
  },
  "aggs": {
    "Vendors": {
      "terms": {
        "field": "Vendedor.keyword"
      }
    }
  }
}
```

Esta QUERY va a buscar dentro de nuestro índice de bienes inmuebles, aquellas operaciones que se hayan realizado (indiferentemente si son ventas o alquileres), y nos las va a agrupar.

Esta agrupación se realiza con la sentencia “aggs”, acompañada del nombre que vamos a dar a la agrupación, como en es nuestro caso la palabra vendors, seguido del tipo de búsqueda que vamos a hacer en la agrupación, en este caso realizaremos una agregación por termino, analizando el vendedor como indicamos al final del bloque.

A pesar de que a simple vista parece complejo, no va más allá que seleccionar el tipo de agregación, y sobre que agregar, por lo que la metodología es sencilla.

Una vez ejecutamos la consulta obtenemos:

```
{
  "_index": "bienes_inmuebles",
  "_type": "default",
  "_id": "3_I-7WmBzIHqhcBxowc",
  "_score": 1.0,
  "_source": {
    "Referencia": 10.0,
    "Fecha_Alta": "2004-01-03T23:00:00Z",
    "Tipo": "Parking",
    "Operación": "Venta",
    "Provincia": "Lleida",
    "Superficie": 299.0,
    "Precio_Venta": 2042768.0,
    "Fecha_Venta": "2004-10-05T22:00:00Z",
    "Vendedor": "Joaquín"
  }
},
{
  "_index": "bienes_inmuebles",
  "_type": "default",
  "_id": "4vI-7WmBzIHqhcBxowc",
  "_score": 1.0,
  "_source": {
    "Referencia": 13.0,
    "Fecha_Alta": "2004-01-03T23:00:00Z",
    "Tipo": "Parking",
    "Operación": "Venta",
    "Provincia": "Barcelona",
    "Superficie": 300.0,
    "Precio_Venta": 2937300.0,
    "Fecha_Venta": "2004-11-03T23:00:00Z",
    "Vendedor": "Jesús"
  }
},
}
```

Figura 69 –Agregación, parte de datos

```
"aggregations": {
  "Vendors": {
    "doc_count_error_upper_bound": 0,
    "sum_other_doc_count": 0,
    "buckets": [
      {
        "key": "Carmen",
        "doc_count": 11
      },
      {
        "key": "Joaquín",
        "doc_count": 11
      },
      {
        "key": "María",
        "doc_count": 10
      },
      {
        "key": "Luisa",
        "doc_count": 9
      },
      {
        "key": "Pedro",
        "doc_count": 9
      },
      {
        "key": "Jesús",
        "doc_count": 8
      }
    ]
  }
}
```

Figura 70 –Agregación, parte de agregación

Por un lado obtenemos el resultado normal de la query, como muestra la figura 69 con los diferentes resultados que encontramos (mostramos solo un fragmento de la salida) y como podemos ver en la figura 70 podemos comprobar las agregaciones que acabamos de añadir, en las que nos muestra por ejemplo, que la vendedora Carmen ha aparecido 11 veces en la query, Joaquín 11 también o Jesús con 8, pudiendo agrupar por lo tanto y saber el numero de hits fácilmente.

Otros parámetros interesantes que podemos pasar a este tipo de query podría ser la sentencia size, que nos mostrará ese top de registros a buscar, que mas coincidencias tienen, en nuestro ejemplo, si hiciéramos el top dos tendríamos a Carmen y Joaquín.

Las agregaciones no terminan aquí, no solo vamos a poder realizar búsquedas para agrupar los resultados, también podemos, en el caso de que deseemos hacerlo, realizar operaciones aritméticas, para tener cálculos sobre nuestras QUERYS.

Para ello simplemente tendremos que modificar el tipo de agregación, antes realizamos una agregación de tipo termino, pues ahora, por ejemplo, realizaremos uno de tipo suma, para ver cuanto suman las ventas de todos aquellos bienes cuya superficie es mayor a 295 metros, y además otra de tipo recuento o como comúnmente se denomina "count" para saber cuantos bienes e inmuebles han sido vendidos bajo esas mismas condiciones.



Para ello, realizamos la siguiente query:

```
GET /bienes_inmuebles/_search/
{
  "query":{
    "range":{
      "Superficie":{
        "gte":295
      }
    }
  },
  "aggs": {
    "SumaVentas": {
      "sum": {
        "field": "Precio_Venta"
      }
    },
    "CuentaVentas":{
      "value_count": {
        "field": "Precio_Venta"
      }
    }
  }
}
```

Como salida, obtendríamos por un lado los mismos resultados que la figura 71, ya que la información es la misma, pero dentro de la agregación obtendríamos:

```
"aggregations" : {
  "CuentaVentas" : {
    | "value" : 70
  },
  "SumaVentas" : {
    | "value" : 1.52015751E8
  }
}
```

*Figura 71 –Salida agregación por operaciones*

Observamos que, por un lado, el número de bienes inmuebles efectuados tras las condiciones propuestas han sido 70, además la cuantiosa cantidad que se ha conseguido sacar por su venta/alquiler.

Esto tampoco acaba aquí,

ya que vamos a poder realizar cualquier tipo de operación, ya sea sumar, restar, calcular medias, sumatorios, máximos y mínimos, casi cualquier tipo de operación matemática.

La mas interesante a mi parecer, dentro de todas las operaciones, es una sentencia especial que nos proporciona Elastic, estamos hablando de “stats”

Con stats obtendremos un conjunto de operaciones predefinidas, sobre el campo que indiquemos.

Hagamos un nuevo ejemplo, basándonos como siempre sobre la query inicial:

```
GET /bienes_inmuebles/_search/
{
  "query":{
    "range":{
      "Superficie":{
        "gte":295
      }
    }
  },
  "aggs": {
    "Estadisticas": {
      "stats": {
        "field": "Precio_Venta"
      }
    }
  }
}
```

Lanzada la query, obtenemos este conjunto de operaciones, como puede ser el recuento, el valor mínimo, el valor máximo, la media y el sumatorio total de ventas. Esta característica es realmente útil si queremos ahorrarnos un puñado de líneas de código, y mostrar de golpe todas las estadísticas del campo a analizar.

```
"aggregations" : {
  "Estadisticas" : {
    "count" : 70,
    "min" : 1251315.0,
    "max" : 2977318.0,
    "avg" : 2171653.5857142857,
    "sum" : 1.52015751E8
  }
}
```

Figura 72 –Salida agregación por stats

Siendo con las agregaciones, ahora vamos a probar un nuevo tipo, como son los filtros. Con los filtros no solo vamos a tener las diferentes operaciones explicadas anteriormente, sino que además vamos a poder poner ciertas condiciones sobre lo que queremos mostrar.

Como siempre, y para esta nueva simulación, vamos a realizar un nuevo ejemplo.

Crearemos una nueva QUERY, que buscara dentro de la base de datos exclusivamente las operaciones que han sido ventas, sin tratar los alquileres. Una vez encontrados, realizaremos un filtro de todas aquellas que hayan sido vendidas entre 1 millón y 2 millones, denominándolo “VentasIntermedias” otro con las ventas inferiores a 1 millón, denominándolo “VentasInferiores” y finalmente otro filtro con las ventas que superan los dos millones, nombrando a este como “VentasSuperiores”.

Para ello, elaboramos la siguiente consulta:

```
GET /bienes_inmuebles/_search/
{
  "query":{
    "term": {
      "Operación.keyword": {
        "value": "Venta"
      }
    }
  },
  "aggs": {
    "filtroVentas":{
      "filters": {
        "filters": {
          "VentasIntermedias": {"range": {
            "Precio_Venta": {
              "gte": 1000000,
              "lte": 2000000
            }
          }},
          "VentasInferiores": {"range": {
            "Precio_Venta": {
              "lte": 1000000
            }
          }},
          "VentasSuperiores": {"range": {
            "Precio_Venta": {
              "gte": 2000000}
          }}
        }
      }
    }
  }
}
```

Reportándonos las estadísticas que podemos ver a continuación:

```
{
  "aggregations" : {
    "filtroVentas" : {
      "buckets" : {
        "VentasInferiores" : {
          "doc_count" : 713
        },
        "VentasIntermedias" : {
          "doc_count" : 758
        },
        "VentasSuperiores" : {
          "doc_count" : 219
        }
      }
    }
  }
}
```

Figura 73 –Salida agregación por filtro

Como hemos estado comprobado, podemos hacer casi cualquier tipo de agrupación con Elastic Search, y también se habrá podido observar que hemos puesto más importancia en este punto. Esto es porque Kibana, la plataforma encargada de la parte visual dentro del Stack, se nutre de las agrupaciones o como dentro de la herramienta las encontramos, agregaciones, para poder realizar sus histogramas, gráficos, estadísticas, que más tarde podrán trabajar todos los analistas.

Además, a la hora de crear la visualización de la simulación 1, podemos introducir este tipo de agregaciones y consultas que hemos estado realizando en el selector de Querys, y nos mostrarían los datos de manera sencilla, teniendo que seleccionar solo aquello que queremos ver.

Con esto damos por finalizada esta última simulación dentro del ámbito práctico del proyecto. Hemos explicado todo el funcionamiento del lenguaje de consulta dentro de Elastic, mostrando la cantidad de recursos disponibles que hay dentro de él, ya sea con la Querys de tipo DSL, con las cuales vamos a poder buscar tanto por términos concretos, como por coincidencias literales (Gracias a los índices invertidos), también podremos realizar operaciones matemáticas para consultar entre rangos, como hicimos con las range Querys, o un sinfín de tipologías que podremos encontrar dentro de las Querys DSL, ya que hay multitud de desarrollos y aquí hemos mostrado los más utilizados. Por otro lado, también explicamos las Querys SQL o Elasticsearch SQL Querys, las cuales utilizan el propio lenguaje sql para realizar sus consultas, una práctica muy utilizada por los diferentes desarrolladores que no quieren utilizar otro tipo de Querys. Y finalmente mostramos con agrupar toda la información, ya sea de forma simple por términos, agrupaciones con operaciones aritméticas o combinar todo esto anterior y además crear filtros para tener la información más estructurada y tratable para cuando se quiera exportar a Kibana (aunque recordemos que ya estamos en Kibana, ya que nos proporciona las dev tools para realizar este tipo de trabajos).

## 8. Costes

### 8.1 Coste software

Como ya venimos explicando en apartados anteriores, Elastic Stack es un paquete Open source. Esto significa, que es una fuente abierta de información, con libre acceso a todo tipo de usuarios, para que estos puedan desarrollar sus propias aplicaciones o que los propios usuarios desarrollen sus propias mejoras o plugins, las cuales dejan acceso al resto de usuarios, creando una comunidad rica en alternativas.

Por lo que, por esta parte, tanto Beats, Logstash, Elastic Search y Kibana no supondrán coste alguno.

Sin embargo, si que podemos encontrarnos software de pago relacionado con Elastic Search, como por ejemplo el Amazon Elastic Search Service.

Este servicio es una alternativa que nos proporciona Amazon, a la instalación y configuración del resto de módulos. Es un servicio a nivel de administración, que se encargará de la implementación, operación y escalado de los diferentes clústeres que contratemos en la nube de los Amazon Web Service.

Si por ejemplo tenemos caídas en los nodos, Amazon automáticamente lo gestionara, balanceando el Cluster y sustituyéndolo por otro que este operativo.

Aunque lo hemos puesto en el apartado software, también supondrá un ahorro hardware ya que todos los nodos que emplearemos estarán en la parte del proveedor.

Las tarifas de Amazon variaran en función del servicio a contratar, ya sea bajo demanda, anual o cada tres años. En la *imagen 9* tenemos un ejemplo del tarifario por parte de Amazon.

#### Precios de instancias bajo demanda

Región:	UE (París) *			
	vCPU	Memoria (GiB)	Almacenamiento de instancias (GB)	Precio por hora
<b>Uso general – Generación actual</b>				
t2.small.elasticsearch	1	2	Solo EBS	0,038 USD por hora
t2.medium.elasticsearch	2	4	Solo EBS	0,077 USD por hora
m5.large.elasticsearch	2	8	Solo EBS	0,165 USD por hora
m5.xlarge.elasticsearch	4	16	Solo EBS	0,33 USD por hora
m5.2xlarge.elasticsearch	8	32	Solo EBS	0,661 USD por hora
m5.4xlarge.elasticsearch	16	64	Solo EBS	1,322 USD por hora
m5.12xlarge.elasticsearch	48	192	Solo EBS	3,965 USD por hora
<b>Con optimización para informática – Generación actual</b>				
c5.large.elasticsearch	2	4	Solo EBS	0,149 USD por hora
c5.xlarge.elasticsearch	4	8	Solo EBS	0,298 USD por hora
c5.2xlarge.elasticsearch	8	16	Solo EBS	0,596 USD por hora
c5.4xlarge.elasticsearch	16	32	Solo EBS	1,192 USD por hora
c5.9xlarge.elasticsearch	36	72	Solo EBS	2,682 USD por hora
c5.18xlarge.elasticsearch	72	144	Solo EBS	5,363 USD por hora

Figura 74 – Tarifa Amazon para instancias bajo demanda.

## 8.2 Coste hardware

En este punto, es seguramente donde tengamos que realizar una inversión inicial superior.

Cuando explicamos el funcionamiento de Elastic Search, apareció un concepto muy interesante, el Cluster. Un Cluster, es un conjunto de nodos u equipos que trabajan para llegar a un fin común.

Para que el paquete Elastic Search funcione correctamente, nos bastaría con un único nodo, aunque ya sabemos que esto no es nada eficiente (En la simulación realizada en el proyecto, hemos empleado un único nodo ya que es del único equipo que se disponía).

Pero, para que tengamos un funcionamiento ideal, y eficiente, necesitaríamos uno, o dos nodos mínimo por cada módulo de Elastic Stack que estamos utilizando (A más nodos, más optimo será nuestro sistema).

Por ejemplo, si vamos a emplear Logstash, Elastic Search y Kibana, lo ideal sería tener varios nodos trabajando recolectados logs, con Logstash implementado en sus memorias, otros nodos realizando los procesos de tratamiento e indexación con Elastic Search, y varios nodos para el procesamiento y visualización desde Kibana. Esto nos ahorraría problemas, ya que, si por algún motivo algún nodo cae, tendríamos otro que pase a realizar su labor. Además, ya vimos que Elastic tiene implementado un sistema de autogestión de los diferentes nodos, ya sean maestros, de datos o de ingesta, para que, si uno se cae, automáticamente otro candidato pase a su puesto a seguir con las tareas.

Esto por ejemplo no lo podemos ver en la práctica, ya que, al disponer sol de un nodo, su caída provocaría errores.

Para la gestión del Cluster, podríamos utilizar otra herramienta gratuita llamada Cerebro. Cerebro es un sistema de monitorización que nos permite comprobar en tiempo real como se encuentran los diferentes nodos en el Cluster. Esta herramienta es muy interesante, y normalmente se emplea en la mayoría de los desarrollos junto a Elastic Search.

En la *imagen 10* podemos ver un ejemplo de la interfaz gráfica de cerebro

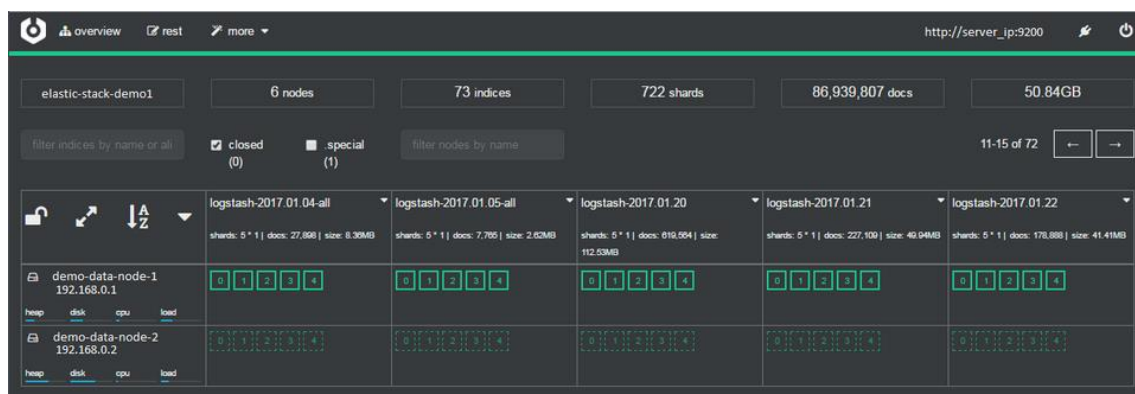


Figura 75 – Interfaz gráfica de cerebro

Podemos ver claramente la información de cada nodo, su capacidad, los documentos que alberga, el numero de shards disponibles...

Por lo que los costes a nivel de hardware van a quedar a función del desarrollo y presupuesto de cada empresa, recomendándose un mayor número de nodos para una mayor velocidad y seguridad en el sistema.

### **8.3 Coste humano**

Finalmente, los costes humanos. Esto también depende de muchos factores, si vamos a implementar nosotros todo el sistema, vamos a necesitar profesionales especialistas en bases de datos, por lo que los costes quedaran en función del presupuesto que tenga la empresa para contratar estas personas.

Si, por otro lado, vamos a delegar la implementación a empresas externas como ocurre con los servicios de Amazon, es posible que necesitemos menos personal avanzado, simplemente una o dos personas para asegurarse del mantenimiento correcto de los procesos y monitorización de los clústeres.

Como al final, todo acaba siendo visualizaciones en la interfaz gráfica de Kibana, cualquier tipo de usuario podría ser capaz de aprender rápidamente como ver la información, entenderla y exportarla a plataformas que desee, como por ejemplo Excel.

Por lo que para este último tipo de usuarios solo necesitaríamos dar varias formaciones al personal para ponerla al día en este tipo de tecnología.

## 9. Conclusión.

Finalmente nos encontramos ante la conclusión.

Creo que hasta este punto hemos podido analizar ampliamente el origen y evolución de la información, así como su almacenamiento, técnicas y usos. Como venimos comentando durante todo el proyecto, uno de los objetivos finales es que cualquier usuario sea capaz de entender como se ha ido desarrollando todo y comprender como funcionan las bases de datos.

Pasamos desde los primeros dispositivos de almacenamiento, como eran las cintas, tarjetas de memoria, discos duros hasta llegar a la nube.

Mas tarde entendimos que tipo de bases de datos existen y como han surgido, ya que antes de utilizar Elastic Search nos viene muy bien tomar un poco de contexto.

Una vez visto esto, procedimos a explicar como funciona el paquete Elastic Stack, viendo como trabajan sus recolectores de logs, como son Beats y Logstash, explicamos en profundidad y entendimos como funcionaba Elastic Search, el encargado de indexar y consultar información y finalmente pasando por Kibana, la interfaz grafica en la que podemos ver todo.

También realizamos varias simulaciones, en las que aprendimos como monitorizar un sistema operativo, como es el caso de Windows 10, mediante la conexión entre Winlogbeat, Elastic Search y Kibana, viendo en el dashboard final todas las estadísticas.

Por otro lado, también vimos como utilizar alguna de las herramientas compatibles con Elastic Search, para comprobar su función de almacén y consulta, concretamente Excelastic, con el que podemos importar bases de datos ya creadas. Igualmente, sobre esta misma simulación explicamos los principales tipos de consultas que nos ofrece la API de Elastic, así como las agregaciones, muy útiles para mas tarde crear si lo deseamos visualizaciones.

A mi parecer, Elastic Search es el futuro dentro de las monitorización de eventos, ya que proporciona una potente API y un conjunto de herramientas y componentes en constante desarrollo, que nos hacen más sencillo el análisis de datos, así como el tratamiento de la información.



## 10. Bibliografía

### - Discos duros

- Article title:** Unidad de disco duro  
**Website title:** Es.wikipedia.org  
**URL:** [https://es.wikipedia.org/wiki/Unidad\\_de\\_disco\\_duro](https://es.wikipedia.org/wiki/Unidad_de_disco_duro)
- Article title:** Historia del disco duro  
**Website title:** CCM  
**URL:** <https://es.ccm.net/contents/228-historia-del-disco-duro>
- Author** Lisandro Pardo  
**Article title:** Historia de los discos duros - NeoTeo  
**Website title:** NeoTeo  
**URL:** <https://www.neoteo.com/historia-de-los-discos-duros/>

### - Disquete

- Article title:** Definición de disquete — Definicion.de  
**Website title:** Definición.de  
**URL:** <https://definicion.de/disquete/>

### - Memorias flash

- Article title:** Qué es una memoria Flash - Culturación  
**Website title:** Culturación  
**URL:** <http://culturacion.com/que-es-una-memoria-flash/>

### - Dispositivos de almacenamiento

- Author** Jonathan Delgado  
**Article title:** Breve historia de los dispositivos de almacenamiento  
**Website title:** Es.slideshare.net  
**URL:** [https://es.slideshare.net/jonathan\\_delgado/dispositivos-de-almacenamiento-27263661](https://es.slideshare.net/jonathan_delgado/dispositivos-de-almacenamiento-27263661)

### - Nube

- Article title:** Ventajas y beneficios de la nube  
**Website title:** Solmicro.com  
**URL:** [https://www.solmicro.com/blog/cloud/la\\_nube\\_ventajas\\_beneficios](https://www.solmicro.com/blog/cloud/la_nube_ventajas_beneficios)

## - BBDD

**Article title:** Bases de datos ¿qué son? ¿qué tipos existen? Lo que necesitas saber como profesional  
**Website title:** platzi.com  
**URL:** <https://platzi.com/blog/bases-de-datos-que-son-que-tipos-existen/>

**Article title:** Tipos de bases de datos y las mejores bases de datos del 2016  
**Website title:** Pandora FMS - The Monitoring Blog  
**URL:** <https://pandorafms.com/blog/es/tipos-de-bases-de-datos-y-las-mejores-bases-de-datos-del-2016/>

**Article title:** Tipos de Bases de Datos: Modelos, Usos y Beneficios  
**Website title:** Tecnologias-informacion.com  
**URL:** <https://www.tecnologias-informacion.com/basesdedatos.html>

## - BBDD relacionales

**Article title:** ¿Qué es Base de datos relacional? - Definición en WhatIs.com  
**Website title:** SearchDataCenter en español  
**URL:** <https://searchdatacenter.techtarget.com/es/definicion/Base-de-datos-relacional>

**Author:** Cuestiones técnicas  
**Article title:** Bases de datos relacionales  
**Website title:** 1&1 Digitalguide  
**URL:** <https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/bases-de-datos-relacionales/>

**Article title:** Base de datos relacional  
**Website title:** Es.wikipedia.org  
**URL:** [https://es.wikipedia.org/wiki/Base\\_de\\_datos\\_relacional](https://es.wikipedia.org/wiki/Base_de_datos_relacional)

## - BBDD orientadas a documentos

**Article title:** ¿Qué es una base de datos de documentos?  
**Website title:** Amazon Web Services, Inc.  
**URL:** <https://aws.amazon.com/es/nosql/document/>

## - BBDD orientadas a objetos

**Article title:** Base de datos orientada a objetos  
**Website title:** Es.wikipedia.org  
**URL:** [https://es.wikipedia.org/wiki/Base\\_de\\_datos\\_orientada\\_a\\_objetos](https://es.wikipedia.org/wiki/Base_de_datos_orientada_a_objetos)

**Article title:** Conceptos sobre base de datos orientada a objetos  
**Website title:** El blog de Kyocera: soluciones para digitalizar tu negocio  
**URL:** <https://smarterworkspaces.kyocera.es/blog/conceptos-base-datos-orientada-objetos/>

#### - **BBDD orientadas a grafos**

**Article title:** Base de datos orientada a grafos  
**Website title:** Es.wikipedia.org  
**URL:** [https://es.wikipedia.org/wiki/Base\\_de\\_datos\\_orientada\\_a\\_grafos](https://es.wikipedia.org/wiki/Base_de_datos_orientada_a_grafos)

#### - **BBDD as a Service**

**Author** Grupo Garatu  
**Article title:** DBaaS: ¿Que son las bases de datos en la nube? - Grupo Garatu TICs - IT  
**Website title:** Grupo Garatu  
**URL:** <https://grupogaratu.com/dbaas-bases-datos-la-nube/>

**Article title:** Base de datos en la nube  
**Website title:** Es.wikipedia.org  
**URL:** [https://es.wikipedia.org/wiki/Base\\_de\\_datos\\_en\\_la\\_nube](https://es.wikipedia.org/wiki/Base_de_datos_en_la_nube)

#### - **Índice invertido**

**Author** Juan Rodríguez  
**Article title:** Recuperación de información (information retrieval) y búsqueda en la Web: Índices invertidos  
**Website title:** Pdl.n.blogspot.com  
**URL:** [http://pdl.n.blogspot.com/2013/05/recuperacion-de-informacion-information\\_15.html](http://pdl.n.blogspot.com/2013/05/recuperacion-de-informacion-information_15.html)

#### - **Logstash**

**Article title:** Logstash - Adictos al trabajo  
**Website title:** Adictos al trabajo  
**URL:** <https://www.adictosaltrabajo.com/2015/09/17/logstash/>

#### - **Kibana**

**Article title:** Introducción a Kibana - Adictos al trabajo  
**Website title:** Adictos al trabajo  
**URL:** <https://www.adictosaltrabajo.com/2015/12/27/introduccion-a-kibana/>

#### - **ELK**

**Article title:** Elasticsearch  
**Website title:** Es.wikipedia.org  
**URL:** <https://es.wikipedia.org/wiki/Elasticsearch>

## - Beats

**Article title:** Beats: Data Shippers for Elasticsearch | Elastic  
**Website title:** Elastic.co  
**URL:** <https://www.elastic.co/es/products/beats>

## - Nodo/Cluster

**Article title:** Funcionamiento de los nodos en un clúster  
**Website title:** Publib.boulder.ibm.com  
**URL:** [https://publib.boulder.ibm.com/tividd/td/TSMCW/GC32-0784-02/es\\_ES/HTML/anrws52121.htm](https://publib.boulder.ibm.com/tividd/td/TSMCW/GC32-0784-02/es_ES/HTML/anrws52121.htm)

**Article title:** Administración de Elasticsearch - Adictos al trabajo  
**Website title:** Adictos al trabajo  
**URL:** <https://www.adictosaltrabajo.com/2016/06/01/administracion-de-elasticsearch/>

## - Shards

**Article title:** Basic Concepts | Elasticsearch Reference [6.2] | Elastic  
**Website title:** Elastic.co  
**URL:** [https://www.elastic.co/guide/en/elasticsearch/reference/6.2/\\_basic\\_concepts.html](https://www.elastic.co/guide/en/elasticsearch/reference/6.2/_basic_concepts.html)

## - Costes Amazon Elastic Search Service

**Article title:** Precios de Amazon Elasticsearch Service  
**Website title:** Amazon Web Services, Inc.  
**URL:** <https://aws.amazon.com/es/elasticsearch-service/pricing/>

## - Simulaciones prácticas

**Article title:** Kibana Guide [7.3] | Elastic  
**Website title:** Elastic.co  
**URL:** <https://www.elastic.co/guide/en/kibana/current/index.html>  
**URL:** <https://www.elastic.co/guide/en/kibana/current/discover.html>  
**URL:** <https://www.elastic.co/guide/en/kibana/current/visualize.html>  
**URL:** <https://www.elastic.co/guide/en/kibana/current/dashboard.html>  
**URL:** <https://www.elastic.co/guide/en/kibana/current/xpack-logs.html>  
**URL:** <https://www.elastic.co/guide/en/kibana/current/devtools-kibana.html>  
**URL:** <https://www.elastic.co/guide/en/kibana/current/api.html>

