9. Lucene y Elasticsearch

9.1. Introducción y objetivos

Existen múltiples tipos de almacenamiento de información además de las bases de datos relacionales clásicas. Podemos encontrar sistemas de almacenamiento altamente especializados en ciertas funciones, como pueden ser las búsquedas por texto.

La mayoría de las bases de datos tradicionales relacionales e incluso una gran mayoría de las no relacionales adolecen este tipo de búsquedas donde se requiere de una precisión excepcional para llevarlas a cabo. Algunos ejemplos serían las búsquedas que realizamos en plataformas de compra *online,*en servicios de *streaming* de películas o series o en aplicaciones que operan sobre una gran cantidad de *logs* producidos por servidores en ejecución.

Una de las opciones que podemos tener en cuenta para implementar este tipo de operaciones es Elasticsearch. Se trata de un excelente motor de búsqueda distribuido para una gran variedad de tipos de datos. En este tema exploraremos qué es esta tecnología y cómo podemos aprovecharla para implementar buscadores en nuestras aplicaciones web que cuenten con un excelente rendimiento y eficiencia.

<https://github.com/>

El código de apoyo que usaremos en este tema es el siguiente:

* [Aplicación de inventario – Productos](https://github.com/UnirCs/back-end-inventory-products-elasticsearch) (versión con Elasticsearch).
* [Inventario de pizzas e ingredientes](https://github.com/UnirCs/back-end-pizza-catalogue).

9.2. Apache Lucene

Para entender qué es Elasticsearch es necesario conocer qué es Lucene, ya que el primero está desarrollado a partir del segundo.

Apache Lucene es una librería *open source* escrita totalmente en Java que implementa un motor de búsqueda de alto rendimiento.

Este motor de búsqueda es ideal para las aplicaciones que demandan trabajos con datos estructurados,con búsquedas dentro de grandes textos o que requieran correcciones o sugerencias de escritura, entre otras cosas. Como podrás imaginar, al tratarse de una librería escrita en Java está planteada para utilizarse con dicho lenguaje.

No obstante, en los últimos años, y debido a su éxito, han surgido adaptaciones de Lucene para otros lenguajes de programación. Este programa es capaz de llegar mucho más allá que las bases de datos tradicionales, en cuanto a las búsquedas de texto, ya que es capaz de indexar y operar sobre cualquier tipo de información que se pueda representar de forma textual.

El elemento principal y central en Lucene es el índice inverso. Estos índices difieren de los que nos encontramos en la mayoría de las soluciones de almacenamiento, donde la clave es el valor de la clave primaria o valor de ordenación en disco y el valor son los datos del registro. En este caso, las claves hacen referencia a las diferentes partes que componen un documento (unidad de información de Lucene) y el valor es el ID del documento, de forma que la búsqueda por valores nos llevará a uno o varios identificadores de documento.

Un índice contendrá múltiples documentos y estos documentos tendrán asociados metadatos, que podrán ser útiles a la hora de realizar búsquedas. Es importante tener en cuenta que los datos que introduzcamos en el índice tienen que ser tratados previamente de alguna forma para asegurarnos de que cumplan con la estructura del índice. De otra forma, no sería posible garantizar la fiabilidad de las consultas que realicemos posteriormente. Una vez indexados los datos, todo lo que hay que hacer es realizar las búsquedas que consideremos a través de la API (interfaz de programación de aplicaciones) que Lucene ofrece.

Lucene utiliza los analizadores y los *tokenizadores* para analizar los documentos que recibe y partirlos en diferentes fragmentos, que serán los que se almacenen en los índices inversos.

Consideremos los siguientes documentos de ejemplo, que son simplemente texto: «El Real Madrid gana la Champions» y «Aitor vive en Madrid». Lo primero que hará Lucene será analizar ese texto y usar el analizador y el *tokenizador* por defecto para partir esos documentos en trozos. Los trozos que se recuperarían serían: «el», «real», «madrid», «gana», «la», «champions», «aitor», «vive», «en» y «Madrid». Estos trozos, que denominaremos *tokens* se almacenan en el índice inverso que se haya creado de la siguiente forma (se muestran solo algunos registros):

A table with numbers and letters

Description automatically generated

Tabla 1. Índice inverso de Lucene. Fuente: elaboración propia.

De esta forma, una consulta al índice con el término «madrid» devolvería los dos documentos y una consulta con el término «aitor» devolvería uno de ellos.

Una vez entendido lo que es un índice inverso, es importante hacer hincapié en la variedad de consultas sobre texto que nos ofrece Lucene:

* Búsqueda puntuada. Los documentos obtienen una puntuación y los mejor puntuados son mostrados en primer lugar.
* Consultas sobre frases.
* Consultas con *wildcards*.
* Consultas sobre los metadatos de los documentos.

Por tanto, a la luz de lo indicado, podemos identificar dos fases claras a la hora de tratar con esta librería y con todos los productos que están construidos sobre ella (como Elasticsearch). Estas dos fases se ilustran a continuación:

A diagram of a software application

Description automatically generated

Figura 1. Fases de trabajo con Apache Lucene. Fuente: elaboración propia.

A screenshot of a computer

Description automatically generated

Tabla 2. Fases de Lucene. Fuente: elaboración propia.

Apache Lucene como base de datos

Hemos presentado Lucene como una librería que implementa un potente motor de búsqueda, el cual nos permite realizar consultas muy potentes sobre texto estructurado. Hace uso de índices para almacenar la información, por lo que no sería extraño que nos preguntásemos a continuación: ¿puede ser Lucene una base de datos?

Esta pregunta no está exenta de polémica. En primer lugar, si consideramos una respuesta afirmativa, ¿sería relacional o no relacional? Hemos de recordar que, aunque trabajamos con datos estructurados, la indexación se lleva a cabo utilizando índices inversos y no tablas, por lo que no hay una representación ni almacenamiento tabular de la información. Además, no hay ningún tipo de relación lógica entre los diferentes índices por lo que no podríamos considerar a Lucene como una base de datos relacional.

Sí que encajaría, por el contrario, como un almacenamiento no relacional orientado a documentos. Y, de hecho, la información que indexamos en Lucene se queda ahí hasta que decidimos borrarla. Por tanto, es técnicamente posible utilizar Lucene como una base de datos no relacional (y también todos aquellos productos construidos sobre esta librería, como Elasticsearch).

No obstante, el hecho de que sea técnicamente posible hacerlo no implica que se trate de una buena práctica. Todo dependerá del propósito por el que queremos recurrir a este tipo de motores de búsqueda. No debemos olvidar que Lucene nace con el objetivo de ser un motor de búsqueda y no una base de datos, por lo que, aunque la información almacenada en el interior de sus índices pueda permanecer allí, es recomendable usar esta librería para lo que fue concebida originalmente.

Es decir, sería adecuado utilizar un motor de búsqueda de estas características para realizar búsquedas sobre grandes cantidades de logs, productos, textos, etc. Sin embargo, no es el mecanismo que elegiríamos para almacenar a los usuarios de nuestra aplicación (con sus contraseñas, datos personales, etc.), ya que para eso existen soluciones más adecuadas.

A continuación, se presenta un ejemplo para ilustrar esto:

A diagram of a software application

Description automatically generated

Figura 2. Uso de Lucene como motor de búsqueda. Fuente: elaboración propia.

Esta figura bien podría pertenecer a un negocio online de venta de productos de todo tipo. Este negocio dispondría de dos tablas dentro de una base de datos relacional MySQL, una para los usuarios, y otra para los productos. Es importante recalcar que, para este tipo de negocios, los productos que venden son uno de sus principales activos y por ello su almacenamiento debe dejarse en manos de tecnologías especialmente diseñadas para ofrecer la correcta persistencia de los datos.

No obstante, una base de datos relacional MySQL no es capaz de compararse con Lucene en cuanto a las consultas sobre texto. La capacidad de Lucene es muy superior en este sentido y es muy probable que los usuarios que busquen productos no sepan el nombre exacto de lo que quieren o los busquen utilizando expresiones inadecuadas o incompletas. Lucene complementa muy bien esa carencia de las bases de datos.

Por lo tanto, esta empresa ha decidido implementar Lucene creando un proceso de indexación que, diariamente, se encarga de indexar todos los productos de la tabla relacional en un índice de Lucene.Posteriormente, las aplicaciones clientes realizarán sus búsquedas contra el índice de Lucene y no contra la base de datos, lo que permite aprovechar la eficiencia y el rendimiento del motor de búsqueda, sin sacrificar en ningún momento la correcta persistencia de los datos.

9.3. Elasticsearch y Elastic Stack

Elasticsearch es un motor de analítica de datos y de búsqueda desarrollado a partir de Apache Lucene, totalmente distribuido y que es capaz de gestionar datos:

* Textuales.
* Numéricos.
* Geoespaciales.
* Estructurados.
* No estructurados.

Elasticsearch vio la luz por primera vez en 2010 y desde entonces se ha convertido en un producto muy recomendado a la hora de implementar cualquier tipo de buscador o de motor de analítica y análisis.Debido a su naturaleza distribuida, Elasticsearch es capaz de escalar horizontalmente y de trabajar a gran velocidad ofreciendo a los programas clientes tiempos de respuesta muy bajos.

Elasticsearch es parte de Elastic Stack (ELK Stack), un conjunto de productos orientados al procesamiento y la analítica de datos. Este conjunto incluye los siguientes productos:

* Logstash: un *pipeline* de procesamiento de datos que se encarga de recoger la información de múltiples fuentes para después procesarla y enviarla a Elasticsearch.
* Elasticsearch: el motor de búsqueda y analítica.
* Kibana: permite visualizar los datos indexados en Elasticsearch a través de cuadros y gráficos.

A group of logos with text

Description automatically generated

Figura 3. ELK (Elastic Stack). Fuente: Elastic, 2022.

En este tema, nos centraremos únicamente en estudiar Elasticsearch.

Dado que Elasticsearch está desarrollado a partir de Lucene, todas las características de la librería que vimos en el apartado anterior aplican a este producto. Sin embargo, es importante tener en cuenta algunas diferencias sustanciales entre Lucene y Elasticsearch.

Lucene es una librería de Java, que puede ser incluida dentro de un proyecto escrito en dicho lenguaje y hacer referencia a ella mediante su API (funciones de librería). Por otro lado, Lucene no es un sistema distribuido, de hecho, este concepto es desconocido para esta librería.

Elasticsearch es un producto. Se trata de un sistema distribuido construido utilizando Lucene y almacenando los documentos en formato JSON. La forma de interactuar con Elasticsearch es únicamente a través de una API REST basada en JSON, por lo que no es necesario conocer la forma de hablar con Lucene, Elasticsearch abstrae esa complejidad por nosotros. Además, Elasticsearch proporciona herramientas de monitoreo de los nodos/clústeres que indexan la información.

Si bien es posible utilizar Elasticsearch de manera gratuita, lo más probable será que se tengan que pagar ciertas licencias si queremos disponer de este motor de búsqueda en los sistemas productivos de una aplicación real. No obstante, existe una forma de crear un clúster de Elasticsearch totalmente gratuito con el que podremos interactuar para desarrollar nuestras aplicaciones.

Muchos autores califican a Elasticsearch como una base de datos no relacional y, aunque técnicamente puede comportarse como tal, Elasticsearch es, en palabras de sus creadores, un motor de analítica y análisis distribuido.

Toda la información referente a Elasticsearch que encontrarás en este tema es de la versión 7.10.2. Esta será también la versión que se utilizará en las prácticas de la asignatura relacionadas con esta tecnología. Existen versiones más recientes pero utilizaremos esta por ser la versión con la que trabaja Bonsai.io, aplicación web que pondrá a nuestra disposición un clúster de Elasticsearch gratuito y público (ya desplegado).

9.4. Tipos de datos en Elasticsearch

En Elasticsearch, todos los campos o atributos de un documento deben tener un tipo concreto. El tipo de dato sirve para indicar la naturaleza de la información que se está almacenando y, por ende, el uso que esperamos hacer de él. A continuación, se indicarán los tipos de datos más utilizados (existen más y pueden ser usados si el caso de uso así lo requiere).

Tipos de datos para texto

Dado que Elasticsearch se especializa en las búsquedas por texto, no es extraño ver que cuente con varios tipos de datos para indexar el texto.

* keyword : se utiliza cuando queremos almacenar contenido estructurado, como podría ser un identificador personal, un e-mail, un número de teléfono, etc. Permiten búsquedas exactas sobre el texto y se pueden utilizar para construir agregaciones y realizar ordenación.
* text : se utiliza para almacenar contenido desestructurado. Por ejemplo, la descripción completa de un producto, el prólogo de un libro, etc. Si no se especifica un método de análisis concreto, Elasticsearch utilizará el analizador por defecto que indexará el texto dividido en tantos *tokens* como palabras contenga. Esto permitirá realizar búsquedas por esas palabras concretas, pero no por partes de ellas. Por ejemplo, si almacenamos como  text  la cadena «Mi perro es grande», Elasticsearch indexará internamente cuatro tokens, «mi», «perro», «es» y «grande». Podremos realizar una búsqueda por el token «perro» (completo), pero no por «gran» (incompleto).
* search as you type : sin duda uno de los tipos de dato más utilizados para ofrecer al cliente correcciones o sugerencias y búsquedas full-text. Este tipo de dato es una especialización del anterior. Internamente, crea al menos 3 atributos en el documento con los nombres *nombreDato,nombreDato.\_2gram* y *nombreDato.\_3gram.* Esto quiere decir que los *tokens* que se crearán (por defecto) para la palabra perro serían los siguientes.



Tabla 3. Diferentes campos y valores creados internamente por el tipo de dato search as you type. Fuente: elaboración propia.

Tipos de datos numéricos

* long : números correspondientes al tipo de dato  long  de los lenguajes de programación modernos.
* integer : números enteros.
* short : números correspondientes al tipo de dato  short  de los lenguajes de programación modernos.
* double : números correspondientes al tipo de dato  double  de los lenguajes de programación modernos.
* float : números correspondientes al tipo de dato  float  de los lenguajes de programación modernos.

Tipos de datos booleanos

* boolean : almacena los valores  true  y  false .

Tipos de datos para objetos

* object : se utiliza cuando queremos almacenar objetos (siempre en formato JSON). Si se utiliza este tipo de dato, se perderá la relación entre los subcampos del JSON.
* flattened : se utiliza para almacenar un objeto JSON como único valor de un campo (aplanado).
* nested : versión especializada de  object . Se utiliza para almacenar objetos JSON preservando la relación entre los subcampos de este, lo que permite consultas sobre los atributos de los niveles inferiores.

Además de los indicados, existen muchos más tipos de datos en Elasticsearch, aunque en la mayoría de los casos quedan relegados a situaciones o casos de uso muy particulares.

Te recomendamos ver el siguiente vídeo, *Tipos de datos en Elasticsearch,* donde repasamos los conceptos principales de los tipos de datos que se han indicado y se muestran algunos detalles más para tener en cuenta.

No obstante, te recomendamos que profundices en el estudio de estos tipos de datos, ya que te será muy útil de cara a las prácticas que realizarás. Para ello, en la sección A fondo, en Tipos de datos en Elasticsearch, dispones de un enlace a la documentación oficial de Elasticsearch sobre todos los tipos de datos disponibles en la versión 7.10. Recuerda que, siempre que utilices la documentación de Elasticsearch en línea, debes asegurarte de que estás buscando en la documentación de la versión correcta (7.10).

9.5. Operaciones con Elasticsearch

A través de la API REST que expone Elasticsearch podemos realizar cualquier tipo de operación de las que se nos ofrecen con la instalación del producto. Existen diversas API, cada una de ellas enfocadas en operaciones o conjuntos de operaciones concretas.

En la sección A fondo, dentro de Las API REST en Elasticsearch, dispones de un enlace a todas las APIs de la versión 7.10 que utilizaremos.

En esta sección, nos centraremos en estudiar las operaciones más comunes que realizarás con tu clúster de Elasticsearch. Estas operaciones son:

* Creación de índices y eliminación de índices.
* Asignación de un alias a un índice.
* Inserción, modificación y eliminación de documentos.
* Búsquedas.
  + Basadas en clave.
  + Basadas en textos completos.
  + Basadas en campos de tipo  search   as   you   type .
* Agregaciones.

Conociendo esas operaciones podremos hacer un uso más que decente del clúster de Elasticsearch que crearemos en la siguiente sección. Todos los tipos de operaciones que se explicarán a continuación se encuentran recogidos en el siguiente vídeo, *Tipos de operaciones en Elasticsearch,* donde realizaremos las operaciones en directo sobre un clúster ya creado.

Creación y eliminación de Índices

Como hemos venido diciendo a lo largo de este tema, los índices son elementos fundamentales tanto de Elasticsearch como de Lucene. Dentro de un índice guardaremos todos los documentos relacionados con él y podremos utilizarlo para búsquedas o análisis.

Para crear un índice es necesario utilizar una petición *PUT* a nuestro clúster. La dirección del clúster será algo con este formato:



Figura 4. Dirección del clúster. Fuente: elaboración propia.

A continuación, puedes ver un ejemplo de petición de creación de índice:



Tabla 4. Creación de índice. Fuente: elaboración propia

Si queremos borrar un índice (operación muy común), podemos hacerlo con la siguiente operación:



Tabla 5. Borrar índice. Fuente: elaboración propia

Obviamente, es necesario que el índice que queremos borrar exista previamente. En caso contrario, recibiremos un error indicando que el índice no existe.

Creación de alias

En algunos casos es muy útil asignar un alias a un índice para referirse a él a la hora de hacer búsquedas. Esta práctica está muy ligada a lo que se conoce como *index rolling.*A continuación, veremos como asignar y desasignar alias.

A white sheet with black text

Description automatically generated

Tabla 6. Operaciones con alias. Fuente: elaboración propia.

En la sección A fondo, tienes un enlace, en *Index rolling strategy,* en el que podrás aprender como implementar este tipo de estrategias cuando los índices se crean y se destruyen constantemente y necesitas un nombre común con el que referirte a ellos.

Inserción, modificación y eliminación de documentos

Una vez hemos creado uno o varios índices en nuestro clúster de Elasticsearch, es el momento de comenzar a ingresar datos en él. Para ello una vez más haremos uso de la API de Elasticsearch que nos permitirá insertar, modificar y eliminar los documentos.

Las operaciones relacionadas con documentos son las siguientes:

A screenshot of a computer

Description automatically generated

Tabla 7. Operaciones con documentos. Fuente: elaboración propia.

La diferencia entre las versiones *PUT* y *POST* es que en la segunda no es necesario indicar el ID del documento, Elasticsearch lo generará automáticamente.

Búsquedas

Ahora que los índices creados disponen de datos, podemos pasar a realizar búsquedas sobre ellos. El tipo de búsqueda que podremos realizar estará determinado en su totalidad por el tipo de dato de los campos del índice. En esta sección cubriremos los tres tipos de búsqueda más habituales.

Búsquedas sobre campos cuyo valor exacto conocemos. *Term query*

Es decir, se refiere a aquellas búsquedas sobre los datos cuyo tipo de dato es *keyword*.

A screenshot of a computer code

Description automatically generated

Tabla 8. Búsqueda sobre *keywords*. Fuente: elaboración propia.

Búsquedas sobre campos que contienen una gran cantidad de texto. *Match query*

En este tipo de búsquedas queremos encontrar ocurrencias a partir de un conocimiento incompleto (por ejemplo, sabemos cuál es una de las palabras del título de una película). Normalmente se realizan sobre campos de tipo *text*.

A screenshot of a computer

Description automatically generated

Tabla 9. Búsqueda *full text*. Fuente: elaboración propia

Esta búsqueda nos devolvería todos aquellos resultados que tuviesen la palabra barco en su sinopsis.

Búsquedas orientadas a sugerencias o correcciones. *Multi match query*

Estas búsquedas las realizaremos cuando queramos implementar algún tipo de mecanismo de sugerencia en el buscador. No será necesario conocer una palabra completa del texto (solo sabremos cómo empieza o acaba una palabra, por ejemplo).

A screenshot of a computer code

Description automatically generated

Tabla 10. Búsqueda sobre campos  search as you type . Fuente: elaboración propia.

Si el campo nombre fuese de tipo search as you type, se habrían generado automáticamente dos campos más con los sufijos \_2gram y \_3gram. Para realizar una búsqueda sobre todos los campos a la vez es necesario usar una *query* de tipo *multi-match,* que nos permite definir un *array* de campos sobre los que efectuar la consulta.

Agregaciones

Las agregaciones nos permiten resumir el contenido de las consultas que lanzamos con el objetivo de obtener métricas, estadísticas u otros datos de carácter analítico. Se conocen también como *facetas*.

Presta atención a la siguiente imagen de una conocida web de venta de productos electrónicos online en España, PCcomponentes:

A screenshot of a phone

Description automatically generated

Figura 5. Web de venta de productos electrónicos. Fuente: PCcompnentes

Sin duda, te resultará familiar ver esos números entre paréntesis que hacen referencia a una serie de filtros preestablecidos que la página ha realizado por ti. No son otra cosa que agrupaciones o facetas. En Elasticsearch es sencillo realizar agregaciones con la siguiente sintaxis:

A screenshot of a computer code

Description automatically generated

Tabla 11. Agregación. Fuente: elaboración propia.

Las agregaciones pueden ser, principalmente, de dos tipos:

* Las métricas sirven para realizar agrupaciones mediante cálculos, como sumas o medias, entre otras. Siempre deben usarse en campos de tipo numérico.
* Las *bucket* devuelven un conjunto de los diferentes valores que se pueden encontrar en base a un campo de tipo *keyword* o numérico. En la Tabla 11 vemos un ejemplo de este tipo, ya que nos devolvería todas las categorías de películas que se encuentren y el número para cada una.

A partir de este momento, dejaremos de ver el detalle de las llamadas a la API de Elasticsearch. Esto es porque, en la práctica, utilizaremos un SDK para facilitar esta tarea; no obstante, siempre es bueno conocer cómo trabaja la API del producto que utilizamos.

9.6. Creación de un clúster con Elasticsearch

De cara a la realización de las prácticas y para poder poner en práctica todo lo que hemos aprendido en este tema, vamos a crear un clúster propio con Elasticsearch para poder trabajar con él. Si lo prefieres, puedes ver el siguiente vídeo, *Creación de un clúster de Elasticsearch,* donde realizaremos en directo la creación del clúster e introduciremos datos de prueba.

Para esto usaremos [bonsai.io](https://bonsai.io/), una aplicación web que nos permite crear un clúster en modo *sandbox* de baja capacidad para realizar pruebas de concepto. Lo primero que haremos será dirigirnos a esta aplicación web y crear una cuenta haciendo clic en el botón de *sign up.*

A screenshot of a website

Description automatically generated

Figura 6. Bonsai.io. Fuente: Bonsai, 2022.

Como habrás podido observar, esta aplicación web también tiene un plan de pago que te permitirá crear más de un clúster y que las capacidades de estos sean mayores. No obstante, nosotros elegiremos la opción gratuita donde obtendremos un clúster con la versión 7.10.2 de Elasticsearch instalada.

Una vez tengamos nuestra cuenta creada y estemos dentro de la zona de usuarios de la aplicación, veremos algo similar a esto:

A blue text box with white text

Description automatically generated

Figura 7. *Home* en bonsai.io. Fuente: Bonsai, 2022.

Haremos clic en *create a free sandbox cluster* para crear nuestro clúster. Veremos el siguiente formulario:

A screenshot of a computer

Description automatically generated

Figura 8. Formulario de creación de clúster en bonsai.io. Fuente: Bonsai, 2022.

Dado que no estamos usando la versión paga, las características que podemos elegir son limitadas. Entre ellas, la región donde se desplegará el clúster. Bonsai utiliza internamente *AWS (Amazon Web Services)*para alojar el clúster, por lo que nuestra *sandbox* se encontrará en una de sus regiones. Por defecto, se alojará en US East. Esto nos afectará en el tiempo de respuesta, ya que toda petición al clúster que hagamos deberá ir a EE. UU. y volver.

Haremos clic en *Create Cluster* y Bonsai comenzará a crear un nuestro clúster. Este proceso tardará unos minutos.

Durante estos minutos, Bonsai estará creando y aprovisionando máquinas virtuales en la nube de Amazon Web Services, en las cuales instalará Elasticsearch. Una vez el proceso haya terminado, veremos algo similar a esto:

A screenshot of a computer

Description automatically generated

Figura 9. Clúster creado correctamente en Bonsai. Fuente: Bonsai, 2022.

Nótese que, si en el momento de crear el clúster, la aplicación te permite seleccionar una región de AWS más cercana a tu ubicación (por ejemplo, Europa), no dudes en elegirla. La elección de una u otra región afectará únicamente en el tiempo de respuesta de las peticiones que realizaremos contra Elasticsearch.

Si entramos a la página de administración de nuestro clúster a través del nombre de este, veremos algo como esto:

A screenshot of a computer

Description automatically generated

Figura 10. Panel de administración del clúster en Bonsai. Fuente: Bonsai, 2022.

Existen varios puntos a destacar viendo este panel:

* Elasticsearch es distribuido y esto podemos verlo en la sección *Data Allocation.* El espacio total de almacenamiento disponible (125 MB de datos) se encuentra repartido en tres máquinas diferentes, donde cada una puede guardar hasta un máximo de 41,7 MB.
* El número máximo de documentos que podremos almacenar en el clúster será 35 000, como vemos en la sección *Usage.*
* Tenemos por otro lado los “shards”. Como ya sabemos, los datos en Elasticsearch se organizan en índices. Un índice puede estar compuesto por uno o más shards. Un shard es una instancia de un índice de Lucene, es decir, un motor de búsqueda en sí mismo que se encargará de indexar y gestionar las consultas que se realicen para un subconjunto de datos concreto en un clúster de Elasticsearch. Tenemos un límite de 10 en la versión gratuita. Cuánto más grande es el tamaño de los shards, peor es el rendimiento de Elasticsearch, por eso siempre se debe tratar de tener shards de un tamaño razonable, menor a los 50 GB.
* Podemos ver que la versión de Elasticsearch instalada en nuestro clúster es la 7.10.2.
* No veremos ningún tipo de *log* ni métrica de rendimiento, ya que aún no hemos realizado ninguna operación con Elasticsearch.
* En el menú lateral, si accedes a *Data*->*Console* verás una consola interactiva en la cual podrás indicar operaciones a realizar con Elasticsearch. Te recomendamos que la utilices usando algunas de las operaciones que se han visto en la sección anterior.
* En el menú lateral, si accedes a *Access* -> *Credentials* verás las credenciales que puedes usar para conectar una aplicación con el clúster de Elasticsearch. Las utilizaremos en la siguiente sección para integrar una aplicación creada con Spring con el clúster que acabamos de construir.
* Tenemos acceso a una instalación gratuita de Kibana, pero esta herramienta no se cubrirá en esta asignatura.

9.7. Integración de Elasticsearch con Spring

Cualquier interacción que queramos realizar directamente con Elasticsearch pasa por usar su API REST. Sin duda esto es una ventaja, ya que es una de las mejores formas de integración (recordemos que los microservicios la utilizan).

No obstante, cuando trabajamos con otro tipo de fuentes de datos para realizar consultas (bases de datos, por ejemplo) no acostumbramos a usar API, sino que nos conectamos directamente a la base de datos o utilizamos algún tipo de librería o SDK que nos facilite el trabajo.

En este caso, la situación es la misma. Usaremos un SDK para Spring y Java que nos facilitará en gran medida la integración de nuestras aplicaciones con el clúster que hemos creado en la sección anterior. La ventaja de haber estudiado previamente algunas de las operaciones de la API de Elasticsearch es que sabemos qué está haciendo el SDK por debajo. De otra forma, estaríamos ciegos y no sabríamos realmente como están sucediendo las cosas.

Usaremos Spring Data Elasticsearch para indexar documentos en nuestro clúster y construir consultas de forma dinámica.

En la sección de A fondo, *Documentación oficial de Spring Data Elasticsearch,* tienes un enlace a la documentación oficial de este proyecto.

Para integrar nuestra aplicación hecha en Spring con el clúster en Elasticsearch solo hay que conectarlos. Para ello es necesario incluir en nuestro *pom* la librería de Spring Data Elasticsearch. Una vez hecho esto, crearemos un nuevo paquete llamado *config* (si no existe ya en el proyecto) y dentro de él crearemos una nueva clase que llamaremos *ElasticSearchConfiguration*.

Esta clase contendrá todo lo necesario para conectar con nuestro clúster y registrará un *bean* que utilizaremos para realizar operaciones con él. Puede ser similar a la que se muestra a continuación:

@Configuration

@EnableElasticsearchRepositories(basePackeages = “es.app.repository”)

public class ElasticSearchConfiguration {

@Value(“${elasticsearch.endpoint}”)

private String clusterEndpoint;

@Value(“${elasticsearch.credentials.user}”)

private String username;

@Value(“${elasticsearch.credentials.password}”)

private String password;

@Bean

public ElasticsearchOperations template() {

final CredentialsProvider credentialsProvider = new BasicCredentialsProvider();

credentialsProvider.setCredentials(

AuthScope.ANY,

new UsernamePasswprdCredentials(username, password);

return new ElasticsearchRestTemplate (

new RestHighLevelClient(

RestClient.builder(

new HttpHost(

clusterEndpoint,

443,

“https”))

.setHttpClientConfigCallback(new RestClientBuilder.HttpClientConfigCallback() {

@Override

public HttpAsyncClientBuilder customizeHttpClient(HttpsAsyncClientBuilder httpClientBuilder) {

return httpClientBuilder.setDefaultCredentialsProvider;

}

})));

}

Como puedes observar, utilizamos properties para almacenar los valores del *host*, usuario y contraseña del clúster. Estos datos los hemos obtenido de la consola de administración de Bonsai en la sección anterior. Se podrían almacenar de forma aún más segura a través de variables de entorno. Esta clase de configuración registrará un *bean* del tipo *ElasticsearchOperations,* que podremos usar en cualquier otro componente de Spring para trabajar con nuestro clúster de Elasticsearch.

Te recomendamos mirar el siguiente vídeo, *Integración de Spring con Elasticsearch,* donde podrás obtener información sobre cómo utilizar este SDK para realizar consultas e indexar documentos.

Por último, no olvides acceder al repositorio de la asignatura para practicar todo lo que has aprendido en este tema.

<https://github.com/UnirCs>