

PRÁCTICA ELASTICSEARCH

Documentación de la prueba.

Este documento contiene en el mejor sentido al detalle cada paso realizado para alcanzar los objetivos de la prueba sobre la plataforma Elasticsearch.

Como primer punto del documento se describe brevemente la herramienta Elasticsearch y a la REST API. Después, para los primeros dos puntos de la prueba, la documentación abarca una explicación breve del comando y un link a la documentación consultada para generarlo, con la finalidad de que cualquier persona ajena a Elasticsearch y a los métodos de REST API pueda generarlos por su cuenta.

0. ELASTICSEARCH Y REST API.

Como primer punto de la prueba fue necesario conocer esta plataforma, de los elementos que componen su “stack” y las REST API. En el siguiente párrafo describo un poco de lo que aprendí hasta ahora acerca de las herramientas mencionadas:

0.1 Elasticsearch

0.2 REST API

1. CREACIÓN DE CUENTA EN ELASTICSEARCH

Para crear una cuenta provisional o de prueba se siguieron los pasos siguientes, ya que en la guía de prueba muestra contenido desactualizado. El primer link de acceso es:

<https://www.elastic.co/es/>

En esta página se hizo *scroll down* hasta la sección con el título “Los productos que dieron origen a todo”. Aquí seleccioné la opción “Conocer Más” que nos dirigire al link:

<https://www.elastic.co/es/elastic-stack/>

En esta página, se observa en la segunda sección la palabra “*Elasticsearch*” con la opción “Conoce más” que nos redirige al link:

<https://www.elastic.co/es/elasticsearch/>

En esta última página seleccionaremos la opción “PRUÉBALO AHORA” que te llevará a crear la cuenta gratuita. Al crear la cuenta es probable que observes que no llega un correo de verificación (por lo menos no hasta el momento de entregar este documento) pero si un correo

de bienvenida. Posteriormente a la creación de cuenta se inicia la sesión con tus credenciales en el link:

<https://cloud.elastic.co/home>

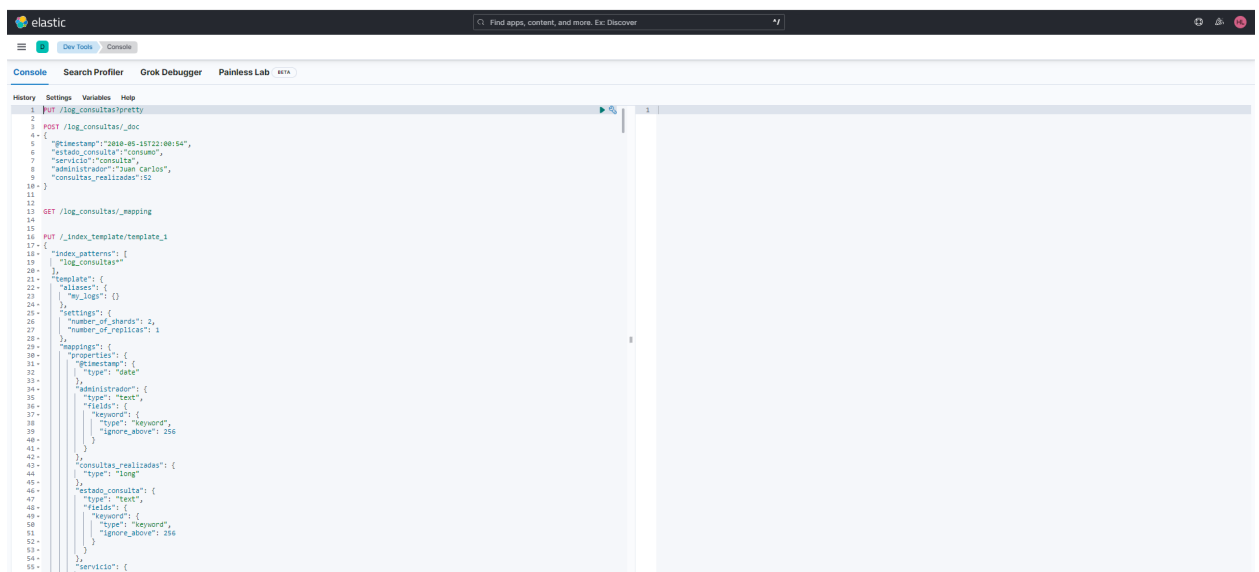
Al entrar a tu cuenta tendrás acceso a una interfaz o portal donde la primera opción más relevante y que se muestra casi en la parte central de tu pantalla es la opción para crear un *deployment*. Para la configuración se ingresaron los parámetros que vienen en la guía (relacionados al nombre, región, optimización y versión) y se esperó aproximadamente 5-6 minutos para la completación de este paso. Al terminar este tiempo copie las credenciales (en un archivo texto para posterior acceso a la cuenta) y se presiona la opción “Continuar”.

Una nueva página aparecerá que te llevará directamente al portal del servicio en la nube de Elasticsearch donde se reconocieron las opciones de *DevTools*, *Management*, *Visualize* y *Dashboards*, en la parte izquierda del menú desplegable que aparece en la parte posterior izquierda. Con este fácil acceso se pudo omitir la parte de la guía que menciona hacer el “Launch” de Kibana.

2. Creando documentos en DevTools (preguntas 1 y 2 de la prueba).

2.1 Creación de documentos y templates

La herramienta de *DeVTools* abre dos interfaces o consolas, divididas por la mitad (véase Figura inmediata). La parte de la izquierda es la parte que permite escribir el código donde podemos crear nuevos documentos o acceder a algunas ya existentes usando las API.



La parte de la derecha permite ver las salidas del código que se ejecuta a la izquierda. Para conocer un poco más de DevTools pueden consultar los dos siguientes links:

<https://www.elastic.co/guide/en/kibana/current/console-kibana.html>
<https://www.geeksforgeeks.org/rest-api-introduction/>

En este sentido, el método PUT nos permite crear un nuevo recurso/documento, especificando si se requiere un “*body*”, el cual contiene nuestros campos o “*field names*” y sus valores. En esta ocasión primero cree el objeto con el comando:

PUT /log_consultas?pretty

La parte /log_consultas define el nombre de tu nuevo documento, y ?pretty permite una especie de impresión más adecuada o fácil de leer, pero es un argumento que puedes omitir. Para agregar nuestras nuevas columnas a nuestro objeto se utilizó el método POST, el cual agrega estos campos al ya creado objeto log_consultas. El comando utilizado fue:

```
POST /log_consultas/_doc
{
  "@timestamp":"2010-05-15T22:00:54",
  "estado_consulta":"consumo",
  "Servicio":"consulta",
  "administrador":"Juan Carlos",
  "Consultas_realizadas":52
}
```

Observamos que la parte derecha de la consola muestra una especie de objeto que te hará saber si tu comando fue efectivo o si hubo algún error, en este caso por la buena ejecución del anterior comando podrás observar un resultado parecido al siguiente, remarco en amarillo los campos de interés:

```
{
  "_index": "log_consultas",
  "_id": "xXIIoMBT2yw0q5V3F xv",
  "_version": 1,
  "result": "created",
  "_shards": {
    "total": 2,
    "successful": 2,
    "failed": 0
  },
  "_seq_no": 177,
  "_primary_term": 1
}
```

Si el código no funciona correctamente tendrás un resultado en la forma anterior pero que te especificará el error. Los más comunes son errores en la sintaxis de la creación de tu objeto, por lo que debes verificar espacios, comas y el correcto cierre con los corchetes.

El mapeo es el proceso de definir cómo se almacenan e indexan un documento y los campos que contiene. Para hacer esto el método es agregando `_mapping` al final de nuestro objeto `/log_consultas` y utilizando la API GET. GET se usa para leer el recurso creado. El comando queda de la siguiente manera:

GET `/log_consultas/_mapping`

El resultado en la parte derecha de la consola te especificará los campos que componen tu objeto y el tipo de dato que tiene (texto, número, etc). El ejemplo de la salida del comando anterior es este, con lo que podrás verificar si cada campo es correcto en cuanto a su tipo de dato:

```
{
  "log_consultas": {
    "mappings": {
      "properties": {
        "@timestamp": {
          "type": "date"
        },
        "administrador": {
          "type": "text",
          "fields": {
            "keyword": {
              "type": "keyword",
              "ignore_above": 256
            }
          }
        },
        "consultas_realizadas": {
          "type": "long"
        },
        "estado_consulta": {
          "type": "text",
          "fields": {
            "keyword": {
              "type": "keyword",
              "ignore_above": 256
            }
          }
        }
      }
    }
  },
}
```

```

    "servicio": {
      "type": "text",
      "fields": {
        "keyword": {
          "type": "keyword",
          "ignore_above": 256
        }
      }
    }
  }
}
}
}
}
}
}
}

```

El index template se crea posteriormente a la definición del index. Para consultar más información de los templates se puede consultar:

<https://www.elastic.co/guide/en/elasticsearch/reference/current/index-templates.html> o <https://opster.com/guides/elasticsearch/data-architecture/index-composable-templates/>

El template funcionará para que al crear un nuevo documento o añadir nuevos objetos al documento que existe pueda tener una idea la recarga de qué campos y que tipos de datos asignar a las nuevas entradas. ¿Cómo sabe el programa que cuando cargues nuevos documentos deberá usar el *template* creado? Pues especificarás un *index_pattern* en tu comando de creación, y cuando hagas una recarga, si el nombre de tu nuevo objeto contiene este patrón, automáticamente sabrá que debe ser con tu *template* (se resaltó en amarillo esta particularidad). Para crear el template utilicé:

PUT /_index_template/template_1

```

{
  "index_patterns": [
    "log_consultas*"
  ],
  "template": {
    "aliases": {
      "my_logs": {}
    },
    "settings": {
      "number_of_shards": 2,
      "number_of_replicas": 1
    },
    "mappings": {
      "properties": {
        "@timestamp": {

```

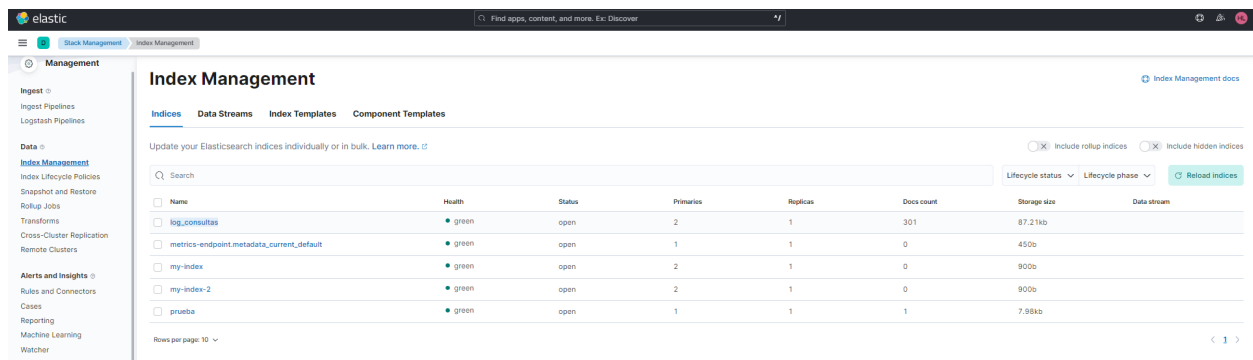

En esta ocasión sugiero que se vea a detalle documentación de como crear un comando con curl, para que puedas exportar un archivo JSON de manera rápida. El código anterior presenta errores ya que no fue posible validar el correcto link para conectarse a DevTools y ejecutar el comando. Para sentido práctico de esta prueba recomiendo que se copie y pegue la *data* del archivo Registros.json después del comando:

POST /log_consultas/_bulk

Toda data debe tener la forma:

```
{"index":{"_index":"log_consultas","_id":28}}
{"@timestamp":"2010-05-15T07:08:09","estado_consulta":"consumo","servicio":"consulta","administrador":"Juan Carlos","consultas_realizadas":65}
```

La primera línea especifica a qué documento pertenece o al cual deberá ser agregado, y una segunda línea donde vienen los campos a agregar. Si tu documento no tiene esta forma es probable que tengas un error en el comando POST. Para saber si se han agregado correctamente tus documentos puedes ir a la sección de Management, y luego en Index Management (link https://sps-practica-eea57b.kb.us-east-1.aws.found.io:9243/app/management/data/index_management/indices). Ahí podrás ver que log_consultas tiene 300 documentos (en la Figura inmediata se observa 301 pero porque corrí el comando POST inicial una vez más).



The screenshot shows the Elastic Index Management interface. On the left is a sidebar with navigation links. The main area is titled 'Index Management' and contains a table of indices. The table has columns for Name, Health, Status, Primaryes, Replicas, Docs count, Storage size, and Data stream. The first index listed is 'log_consultas' with a health of 'green', status of 'open', 2 primaryes, 1 replica, 301 documents, and a storage size of 87.21kb. Other indices include 'metrics-endpoint.metadata.current.default', 'my-index', 'my-index-2', and 'prueba'.

Name	Health	Status	Primaryes	Replicas	Docs count	Storage size	Data stream
<input type="checkbox"/> log_consultas	green	open	2	1	301	87.21kb	
<input type="checkbox"/> metrics-endpoint.metadata.current.default	green	open	1	1	0	450b	
<input type="checkbox"/> my-index	green	open	2	1	0	900b	
<input type="checkbox"/> my-index-2	green	open	2	1	0	900b	
<input type="checkbox"/> prueba	green	open	1	1	1	7.96kb	

2.2 Consultar tus documentos con _search

El método `_search` te permitirá consultar tu documento y obtener solo lo que quieres saber, sin ver los más de 301 documentos que hemos creado hasta ahora. Para conocer más acerca de la sintaxis básica de `_search` puedes consultar el video de <https://www.youtube.com/watch?v=UStLBiVQ-M0>.

En este sentido tus comandos deberán contener GET (porque quieres leer tu documento) seguido del nombre de tu documento (log_consultas) finalizando con el método `_search`. Este podrá tener un argumento adicional que iniciará con un signo de interrogación,

una q, y posteriormente tu consulta, donde podrás especificar el campo y el valor sobre el cual quieres “filtrar”. Para la primer pregunta de la guía el comando quedó:

```
GET /log_consultas/_search?q=estado_consulta:(error OR consumo)
```

Como puedes ver tiene un GET, el nombre de tu documento, el método utilizado y el query. En este caso queremos un query que nos traiga todos los objetos del documento donde el valor de estado_consulta sea igual a error o igual a consumo. Para la segunda pregunta de la guía el comando quedó:

```
GET /log_consultas/_search?q=administrador:"Juan Lara"
```

En este caso el valor texto Juan Lara se colocó entre comillas, ya que si no lo haces de esta manera, te traerá objetos que contienen un administrador con nombre Juan solamente o Lara como apellido y nunca el de Juan Lara. En la pregunta anterior pudimos colocar “error” y “consumo” también pero en este caso no hay más entradas que puedan corresponder a la categoría de estado_consulta. Para la tercer pregunta el comando fue:

```
GET /log_consultas/_search?q=estado_consulta%Informativo+servicio%borrado
```

Se nos pedía en este caso 2 restricciones del query, la forma de hacerlo es bajo la misma intuición de los dos comandos anteriores, solo que para tener una segunda restricción debemos agregar un signo “+”, además de que los : cambian por un %. Entonces, este query nos dice que nos traerá los objetos que tienen estado_consulta igual a “informativo” pero también un servicio igual a “borrado”, ambas condiciones deben cumplirse (como una especie de AND en otros lenguajes).

Por último tenemos una pregunta que nos pide hacer una suma del valor de consultas realizadas pero con la necesidad de filtrar primero, sumando entonces solamente las consultas de donde el estado de consulta es igual a “error”. En este caso ya no se utilizó la sintaxis básica anterior y tendremos que definir un “cuerpo” a nuestro comando GET /log_consultas/_search haciéndolo más complicado. Para saber más de esta manera de hacerlo puedes consultar el link:

<https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-multi-match-query.html>

El comando usado fue:

```
GET /log_consultas/_search?
{
  "query":{
    "match": {
      "estado_consulta": "error"
    }
  }
}
```



```

},
"aggs": {
  "suma de consultas realizadas":{
    "sum":{
      "field":"consultas_realizadas"
    }
  }
}
}
}
}

```

Como puedes ver se define primero en el *"body"* un objeto que realiza un query y por lo tanto filtra la información dejando solo los objetos que tienen la consulta en estado de error. Sobre estos resultados entonces podrás crear un nuevo objeto *"aggs"*, que deberá contener otro con el nombre de tu función de agregación, después otro con el método de agregación (en este caso *"sum"*, y finalmente el campo de tu documento (en este caso *consultas realizadas*). Este mismo comando puedes replicarlo y cambiar solamente los parámetros variables.

La salida o respuesta de cada método ya no se anexaron pero tendrás resultados parecidos al primer ejemplo de salida, donde podrás ver el número de entradas que corresponden a tu query.

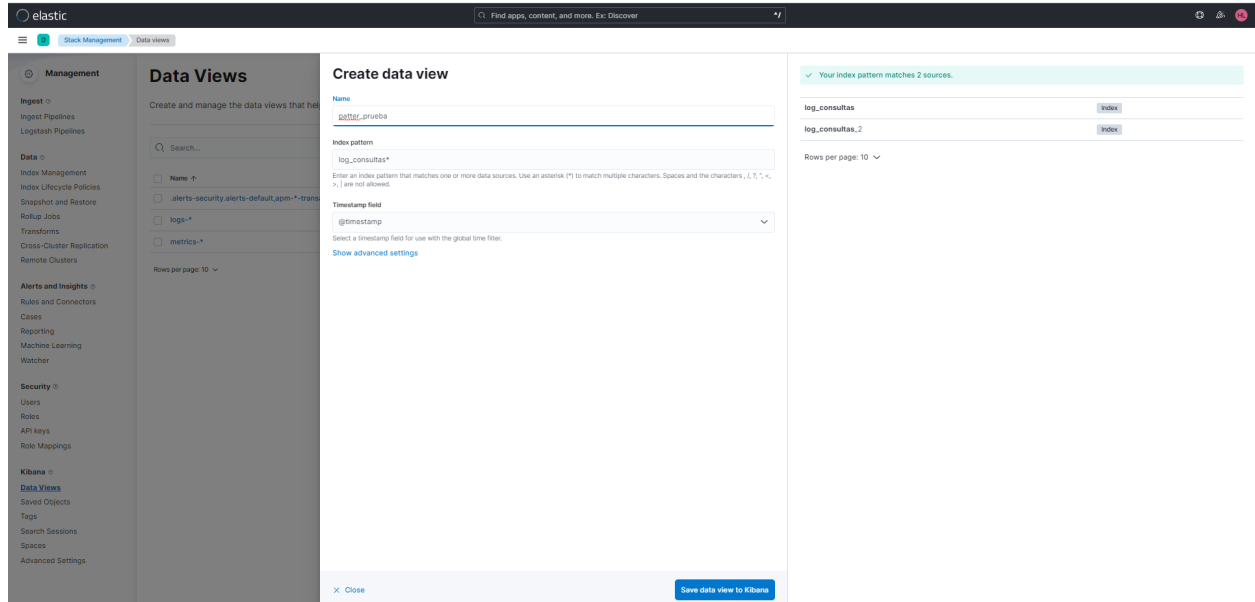
3. Realizar un tablero para visualizar información de empleados.

3.1 Gráfico Heat Map

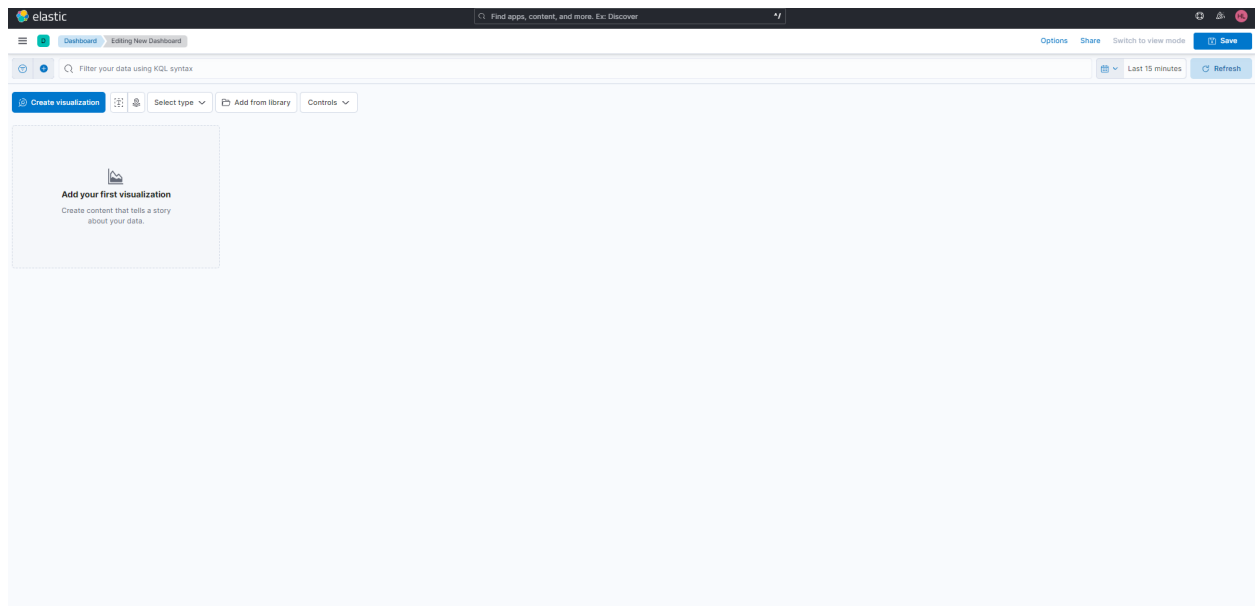
Para la tercera pregunta se visitó la nueva documentación para Kibana donde menciona que el menú de Kibana ha sustituido el nombre de *"Index Pattern"* a *"Data Views"*. El link consultado para la documentación fue:

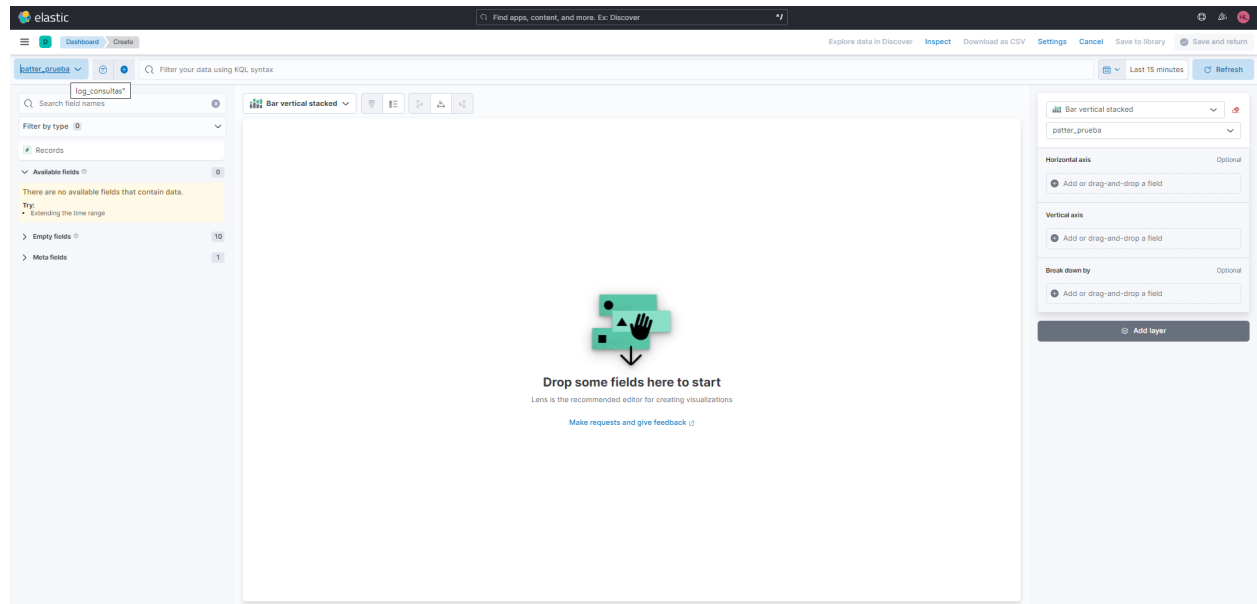
<https://www.elastic.co/guide/en/kibana/current/index-patterns.html>

Se accede a la opción *"Data Views"* y después en la esquina superior derecha se selecciona *"Create Data View"*. Esto abrirá un panel lateral derecho, aquí solamente colocamos el name de nuestro *"pattern"* (ya que no está especificado en la guía yo puse *"pattern_prueba"*), en *"Index Pattern"* escribimos *log_consultas*. Notarás que no se puede seleccionar el documento con clic desde la parte derecha pero en medida que escribes el nombre irás haciendo match con los nombres disponibles. La última opción permitirá seleccionar el campo *@timestamp* (véase la Figura inmediata).

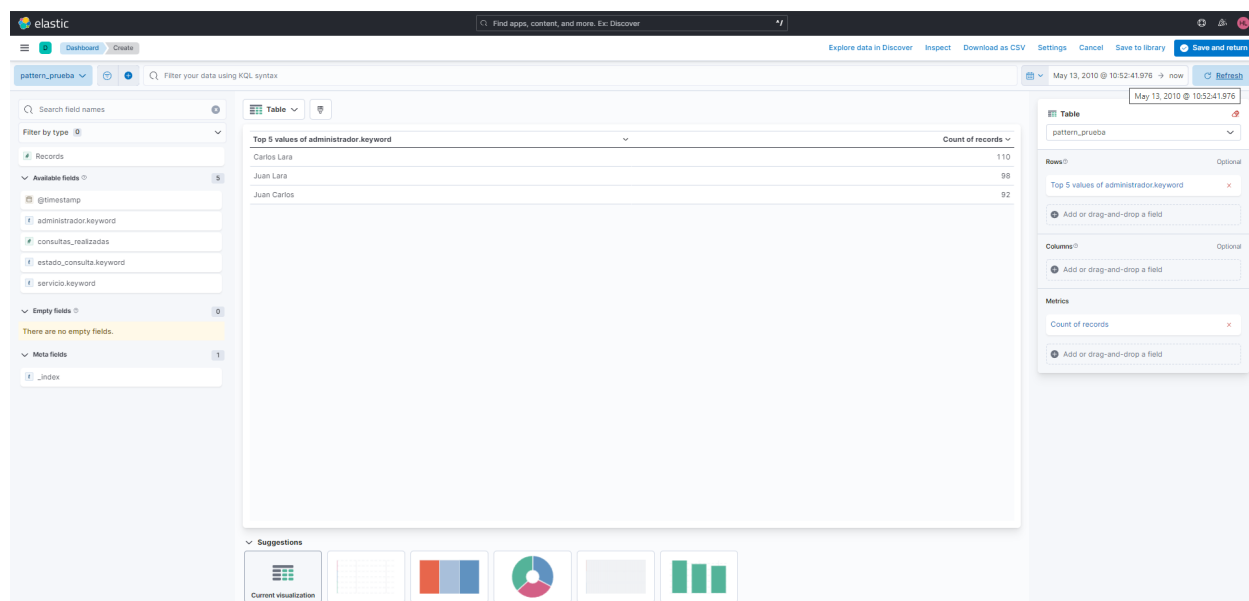


La opción “*Visualize*” ya no existe, por lo tanto para crear una visualización, la opción adecuada es “*Dashboard*” encontrada dentro de “*Analytics*”. Se abrirá una nueva ventana (véase la Figura inmediata) donde seleccionaremos “*Create Visualization*”. Esta opción abrirá otra ventana (véase segunda Figura inmediata). En la parte superior izquierda de esta segunda ventana seleccionaremos nuestro “*Index Pattern*” creado en pasos anteriores, denominado “pattern_prueba”. Después seleccionaremos nuestro tipo de gráfico (para el primer ejercicio es el tipo de gráfico “*heat map*”).

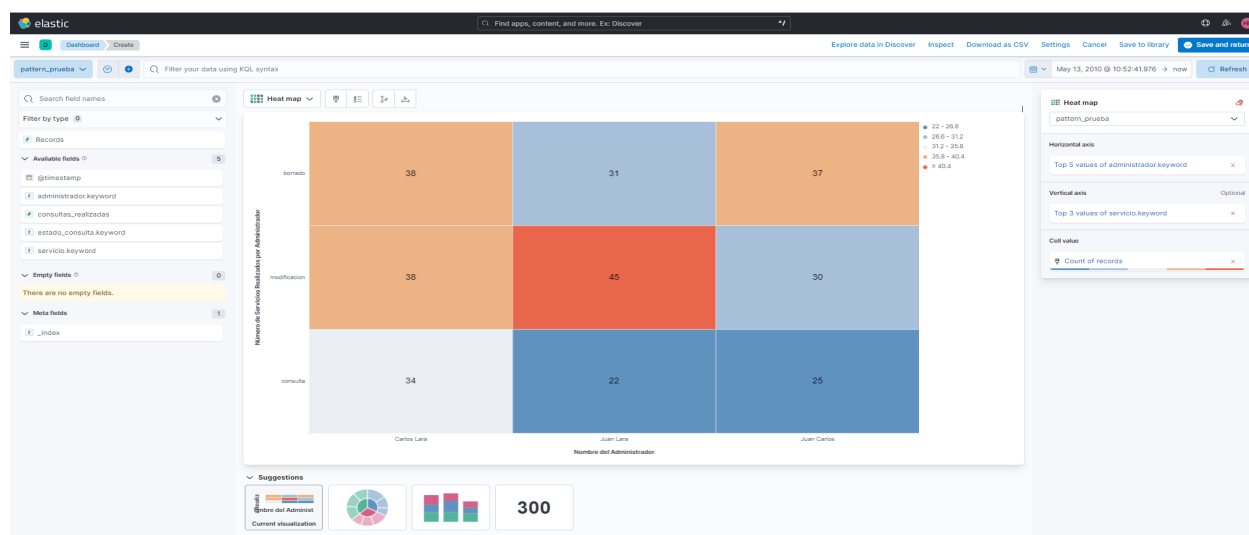




En esta sección hay que tener en cuenta la ventana de tiempo que estamos visualizando. Es probable que al asignar tus variables en los ejes aparezca el mensaje al centro del gráfico que menciona “no se han encontrado resultados”. Si es así, en la parte superior derecha podremos modificar la ventana de tiempo. En el caso de la prueba se observó que se estaba visualizando una ventana de tiempo de los últimos 15 minutos. Para corregirlo, debes mover la ventana a la que corresponden tus datos, en este ejemplo los 300 documentos cargados en log_consultas corresponden a una fecha del 15 de Mayo del 2010 (véase la Figura inmediata). Una vez actualizada la fecha, se procede a asignar los valores para los ejes. se procede a seleccionar nuestros ejes. En la primera solicitud de la guía tendremos que elegir a administrador en el eje horizontal y servicios realizados en el eje vertical. Nota también que los valores mostrados corresponden a los “Top” valores, pero puedes elegir si mostrar esta opción, un filtro o algún rango.



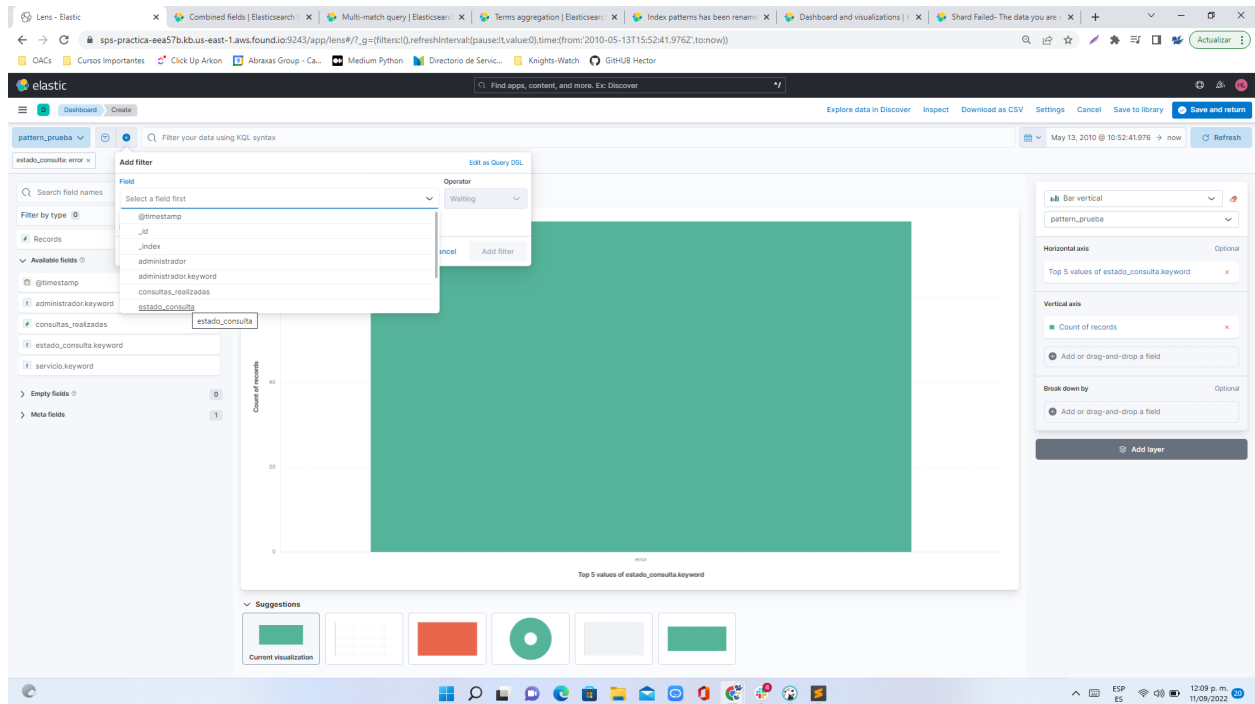
Nuestro primer gráfico del tipo heatmap debería verse más o menos como el *siguiente screenshot*. Esta gráfica se puede guardar con la opción “*Save and Return*”.



3.1 Gráfico de Barras

El gráfico de barras se realizó solamente cambiando el tipo de gráfico a “*Bar Vertical*” y cambiando los valores de los ejes a estado_consulta en el eje horizontal y # Records en el eje vertical. También muestra por *default* el “*Top*” de valores. En este caso se nos solicitó visualizar solamente el número de entradas para el estado_consulta que sea igual a la palabra “error”. Para esto se agrega un filtro (véase Figura inmediata). Este se selecciona en la parte central y superior de la pantalla del gráfico, aquí seleccionamos filtrar por estado_consulta, y que el valor sea igual (*where value “is”*) al texto “error”. El gráfico mostrará una sola barra mostrando que

hay 78 records con estado_consulta igual a error. Esta gráfica se puede guardar con la opción “Save and Return”.



4. Gráficos Extra.

Nuestro *Dashboard* puede visualizarse con estos dos gráficos terminados. Puedes filtrar la visualización dando clic sobre la barra o sobre alguna combinación en el *Heat Map* haciendo esto dinámico, como lo es por ejemplo un dashboard en PowerBI.

