

Práctica en Elasticsearch para SPS.

Creación del proyecto general	2
Actividad 1. Creación de un índice	3
Actividad 2. Realizar búsquedas sobre el índice	8
Actividad 3. Realizar un tablero para visualizar información de empleados	12
Bibliografía	23

Creación del proyecto general

1. Configura el entorno con las características deseadas.


Create your first deployment

A deployment includes Elasticsearch, Kibana, and other Elastic Stack features, allowing you to store, search, and analyze your data.

Name

sps_practica

Settings

Cloud provider	 Amazon Web Services	▼
Region	us N. Virginia (us-east-1)	▼
Hardware profile ⓘ	Storage optimized	▼
Version ⓘ	8.3.3 (latest)	▼

> [Advanced settings](#)

[Show configuration](#) ^

Create deployment

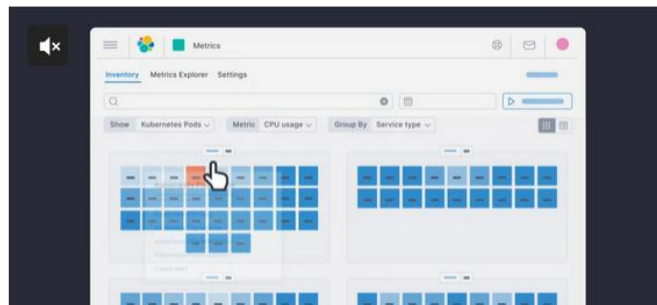


Creating your deployment (takes about five minutes)

Continue

While you're waiting,

Discover what you can do with Elastic



Nota: Guardar las credenciales proporcionadas (solo se muestran una vez).

Actividad 1. Creación de un índice

1. Crear un índice con el nombre de “log_consultas” con el archivo JSON proporcionado.

Se utiliza “POST” para indicar la creación de un índice, el formato es “*POST nombreArchivo/_doc/*”.

```

1 POST log_consultas/_doc/
2 {
3   "@timestamp":"2010-05-15T22:00:54",
4   "estado_consulta":"consumo",
5   "servicio":"consulta",
6   "administrador":"Juan Carlos",
7   "consultas_realizadas":52
8 }
9

```

- Se visualizará en consola, así como el tiempo de ejecución.
- En la línea 5 mostrará como resultado “created” lo que indica que se ha creado con éxito.
- En la línea 8 se muestra un “1” como sinónimo de exitoso en el proceso de creación.

201 - 576 ms

```

1 {
2   "_index": "log_consultas",
3   "_id": "JI1pt4IBqCeIhfoGOKW2",
4   "_version": 1,
5   "result": "created",
6   "_shards": {
7     "total": 2,
8     "successful": 1,
9     "failed": 0
10  },
11   "_seq_no": 0,
12   "_primary_term": 1
13 }

```

2. Obtén el mapping del índice anterior y genera un template a partir de este índice haciendo uso del API de plantillas (TEMPLATE). El patrón para el índice debe ser: “log_consultas*”. Verifica los tipos de datos hagan sentido con la información almacenada.

2.1. Generación de mapping

Se utiliza GET (puede ser considerado como sinónimo de “traer” “mandar a llamar”) seguido del nombre del archivo y después el mapping (que es lo que necesitamos).

```
10
11 GET /log_consultas/_mapping
```

- Nos mostrará en consola el mapeo solicitado:

```
1 - {
2 -   "log_consultas": {
3 -     "mappings": {
4 -       "properties": {
5 -         "@timestamp": {
6 -           "type": "date"
7 -         },
8 -         "administrador": {
9 -           "type": "text",
10 -          "fields": {
11 -            "keyword": {
12 -              "type": "keyword",
13 -              "ignore_above": 256
14 -            }
15 -          }
16 -        },
17 -         "consultas_realizadas": {
18 -           "type": "long"
19 -         },
20 -         "estado_consulta": {
21 -           "type": "text",
22 -           "fields": {
23 -             "keyword": {
24 -               "type": "keyword",
25 -               "ignore_above": 256
26 -             }
27 -           }
28 -         },
29 -         "servicio": {
30 -           "type": "text",
31 -           "fields": {
32 -             "keyword": {
33 -               "type": "keyword",
34 -               "ignore_above": 256
35 -             }
36 -           }
37 -         }
38 -       }
39 -     }
40 -   }
41 - }
```

Nota: El mapping (o mapeo) nos permite generar una *definición* de cómo se encuentra almacenado y que campos contiene, es decir, podemos entenderlo como un grupo o colección donde cada uno tiene su propio tipo de dato, nos genera una lista donde es más fácil detectar que contiene cada campo para poder relacionarlas o asociarlas con distintos campos en el futuro.

2.2.Creación de template

Utilizaremos el mapping creado en el apartado 2.1, después usaremos el método “PUT” (esto nos ayuda a actualizar o insertar recursos), posterior le indicaremos lo siguiente “con el método PUT vas a usar un template de index y lo vas a nombrar “template_consultas”.

- En código veríamos lo siguiente (Este código NO está completo, solo es para entender la explicación anterior):

```
13  
14 PUT _index_template/template_consultas
```

Para completar el código y crear un template es necesario indicar de donde estamos tomando nuestra base, es decir, que campos predeterminados tendrá, por lo que deberemos indicar el origen que es “log_consultas*” (línea 17).

- De la línea 16 a la 21 indicamos las características generales de nuestro template¹.
- Posterior a la línea 21, usamos el mapping generado anteriormente para el template.
- El código completo para crear un template es el siguiente:

¹ Para comprender mejor el proceso véase la documentación en el siguiente link: <https://www.elastic.co/guide/en/elasticsearch/reference/current/indices-put-template.html#put-index-template-api-prereqs>

```

13
14 PUT _index_template/template_consultas
15 {
16   "index_patterns": [
17     "log_consultas*"
18   ],
19   "template": {
20     "settings": {
21       "number_of_shards": 1
22     },
23     "mappings": {
24       "properties": {
25         "@timestamp": {
26           "type": "date"
27         },
28         "administrador": {
29           "type": "text",
30           "fields": {
31             "keyword": {
32               "type": "keyword",
33               "ignore_above": 256
34             }
35           }
36         },
37         "consultas_realizadas": {
38           "type": "long"
39         },
40         "estado_consulta": {
41           "type": "text",
42           "fields": {
43             "keyword": {
44               "type": "keyword",
45               "ignore_above": 256
46             }
47           }
48         },
49         "servicio": {
50           "type": "text",
51           "fields": {
52             "keyword": {
53               "type": "keyword",
54               "ignore_above": 256
55             }
56           }
57         }
58       }
59     }
60   }
61 }

```

- Una vez ejecutado el código se visualizará en consola lo siguiente:

```

1 {
2   "acknowledged": true
3 }

```

Esto nos indica que se ha creado con éxito.

3. Definido el template se cargará una serie de documentos en el índice utilizando el archivo JSON que se ha proporcionado, se utilizará API (BULK).

Utilizaremos el método bulk para agregar los archivos JSON a nuestro índice².

```

62
63
64 POST log_consultas/_bulk
65 {"index":{"_index":"log_consultas","_id":1}}
66 {"@timestamp":"2010-05-15T22:00:54","estado_consulta":"consumo","servicio"
   : "consulta","administrador":"Juan Carlos","consultas_realizadas":52}
67 {"index":{"_index":"log_consultas","_id":2}}
68 {"@timestamp":"2010-05-15T12:55:04","estado_consulta":"consumo","servicio"
   : "modificacion","administrador":"Juan Lara","consultas_realizadas":10}
69 {"index":{"_index":"log_consultas","_id":3}}
70 {"@timestamp":"2010-05-15T14:56:48","estado_consulta":"consumo","servicio"
   : "consulta","administrador":"Juan Lara","consultas_realizadas":20}
71 {"index":{"_index":"log_consultas","_id":4}}
72 {"@timestamp":"2010-05-15T22:33:34","estado_consulta":"error","servicio"
   : "modificacion","administrador":"Juan Carlos","consultas_realizadas":65}
73 {"index":{"_index":"log_consultas","_id":5}}
74 {"@timestamp":"2010-05-15T18:36:57","estado_consulta":"consumo","servicio"
   : "consulta","administrador":"Carlos Lara","consultas_realizadas":5}
75 {"index":{"_index":"log_consultas","_id":6}}

```

Nota: El código continua, sin embargo, por cuestiones prácticas se omitió el código faltante.

- Se visualizará en consola lo siguiente, lo que indica que se ha realizado el proceso:

```

1 {
2   "took": 70,
3   "errors": false,
4   "items": [
5     {
6       "index": {
7         "_index": "log_consultas",
8         "_id": "1",
9         "_version": 1,
10        "result": "created",
11        "_shards": {
12          "total": 2,
13          "successful": 2,
14          "failed": 0
15        },
16        "_seq_no": 1,
17        "_primary_term": 1,
18        "status": 201
19      }
20    ]
21  }

```

² Esta forma de hacerlo NO se encuentra optimizada, debido a que no es eficiente escribir demasiadas líneas de código para utilizar el método bulk, sin embargo, fue la solución al problema presentado.

Es de destacar que es necesario encontrar una mejor solución para este problema.

Del mismo modo, utilizamos como base la documentación en el siguiente link:
<https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-bulk.html>

Actividad 2. Realizar búsquedas sobre el índice

1. Obtener el número de registros con estado_consulta igual a error y consumo

Utilizamos GET para poder realizar la consulta. El formato es: “log_consultas/_search”³. Donde le estamos diciendo que en el índice log_consultas va a buscar (o realizar una consulta) donde la consulta sea igual nuestra query.

- En código se verá de la siguiente manera:

```
666 |
667 GET /log_consultas/_search
668 {
669   "query":{
670     "bool":{"filter":{"
671       "bool":{"
672         "must":[
673           {"term":{"estado_consulta":"error"}},
674           {"term":{"estado_consulta":"consumo"}}
675         ]
676       }}
677     }}
678   }
679 }
680
```

- El resultado en la consola será el siguiente.
- El resultado de la consulta es 0 (**línea 12**), esto debido a que el estado de consulta está condicionado a un valor único, y al realizar la consulta con la conjunción “y” dará como resultado 0 ya que ningún campo contiene ambos valores⁴.

³ Para una mejor comprensión para hacer las consultas nos basamos en la documentación del siguiente link: <https://www.elastic.co/guide/en/elasticsearch/client/java-api/7.9/java-compound-queries.html>

⁴ Para ejemplificar mejor podemos poner el siguiente ejemplo: realizar una consulta al campo edad donde el valor sea 18 y 19. El resultado será 0 debido a que edad solo puede tener un valor. Si la consulta fuera “o” mostraría los resultados de 18 y 19.


```

1 {
2   "took": 0,
3   "timed_out": false,
4   "_shards": {
5     "total": 1,
6     "successful": 1,
7     "skipped": 0,
8     "failed": 0
9   },
10  "hits": {
11    "total": {
12      "value": 0,
13      "relation": "eq"
14    },
15    "max_score": null,
16    "hits": []
17  }
18 }

```

2. Obtener el número de registros realizados por el administrador Juan Lara

La siguiente consulta se realiza con el método search, sin embargo, se utiliza “match_phrase” para encontrar todas las coincidencias que contengan “Juan Lara”.

- El código es el siguiente:

```

681
682 GET /log_consultas/_search
683 {
684   "query":{
685     "match_phrase": {
686       "administrador": "Juan Lara"
687     }
688   }
689 }
690

```

- El resultado de la consulta se encuentra en la línea 12 con un valor de 98 coincidencias para Juan Lara.
- El resultado en la consola es el siguiente:

```

1 ▾ {
2   "took": 9,
3   "timed_out": false,
4 ▾   "_shards": {
5     "total": 1,
6     "successful": 1,
7     "skipped": 0,
8     "failed": 0
9   },
10 ▾  "hits": {
11 ▾    "total": {
12      "value": 98,
13      "relation": "eq"
14    },
15    "max_score": 0.8246293,
16 ▾    "hits": [
17 ▾      {
18        "_index": "log_consultas",
19        "_id": "2",
20        "_score": 0.8246293,
21 ▾        "_source": {
22          "@timestamp": "2010-05-15T12:55:04",
23          "estado_consulta": "consumo",

```

3. Obtener el número de registros con estado_consulta igual a informativo y servicio igual a borrado

Para realizar esta consulta se replicó el query 1, sin embargo, en esta ocasión, al ser campos distintos y estar condicionado por la conjunción “y” dará los resultados que tengan ambos valores en los dos campos.

- El resultado de la consulta se encuentra en la línea 12, con un valor de 52 coincidencias. El resultado en consola es el siguiente:

```

1 {
2   "took": 0,
3   "timed_out": false,
4   "_shards": {
5     "total": 1,
6     "successful": 1,
7     "skipped": 0,
8     "failed": 0
9   },
10  "hits": {
11    "total": {
12      "value": 52,
13      "relation": "eq"
14    },
15    "max_score": 0,
16    "hits": [
17      {
18        "_index": "log_consultas",
19        "_id": "6",
20        "_score": 0,
21        "_source": {
22          "@timestamp": "2010-05-15T11:21:05",
23          "estado_consulta": "informativo",

```

4. Obtener la suma de los valores en consultas_realizadas con estado_consulta igual a error

Para esta consulta se utilizó directamente lenguaje SQL, se utiliza GET para mandar a llamar el método SQL, donde se le dice que el tipo de formato será en texto, es decir, en forma de consulta como si fuera en lenguaje SQL⁵.

- El código es el siguiente:

```

707 GET /_sql?format=txt
708 {
709   "query": "SELECT SUM(consultas_realizadas) FROM log_consultas WHERE
710             estado_consulta='error'"
711 }

```

⁵ Esta consulta fue realizada así debido a su facilidad, sin embargo, también se puede desarrollar sin lenguaje SQL, podría optimizarse para que sea acorde a las demás consultas, aunque de esta forma también cumple su función.

- Las consultas realizadas que se han hecho donde el estado de la consulta es igual a error es de 2,865 (se encuentra en la línea 3).

```

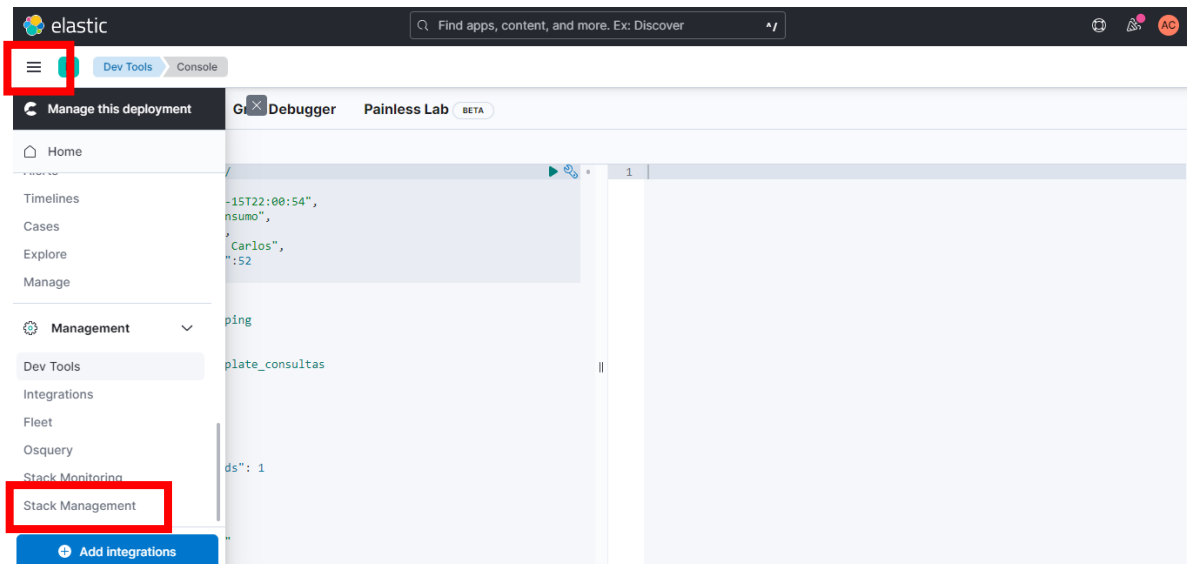
1 SUM(consultas_realizadas)
2 -----
3 2865
4

```

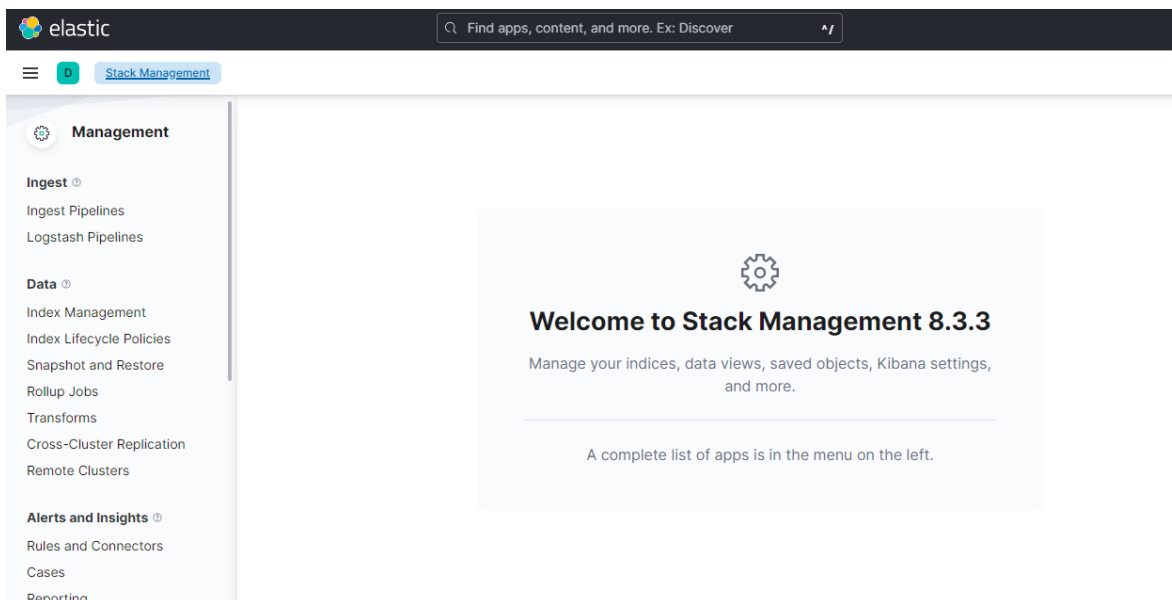
Actividad 3. Realizar un tablero para visualizar información de empleados

1. Crea un patrón de índice en Kibana, dirígete a la opción de Management y selecciona la opción Kibana >Index Patterns y selecciona el índice que creaste en los pasos anteriores log_consultas tomando como Time Filter el campo de @timestamp.
 - a. Vista de heat map, se debe mostrar el número de servicios realizados por administrador.
 - b. Vista de barras, donde se grafique el número de registros con estado_consulta igual a error a través del tiempo.

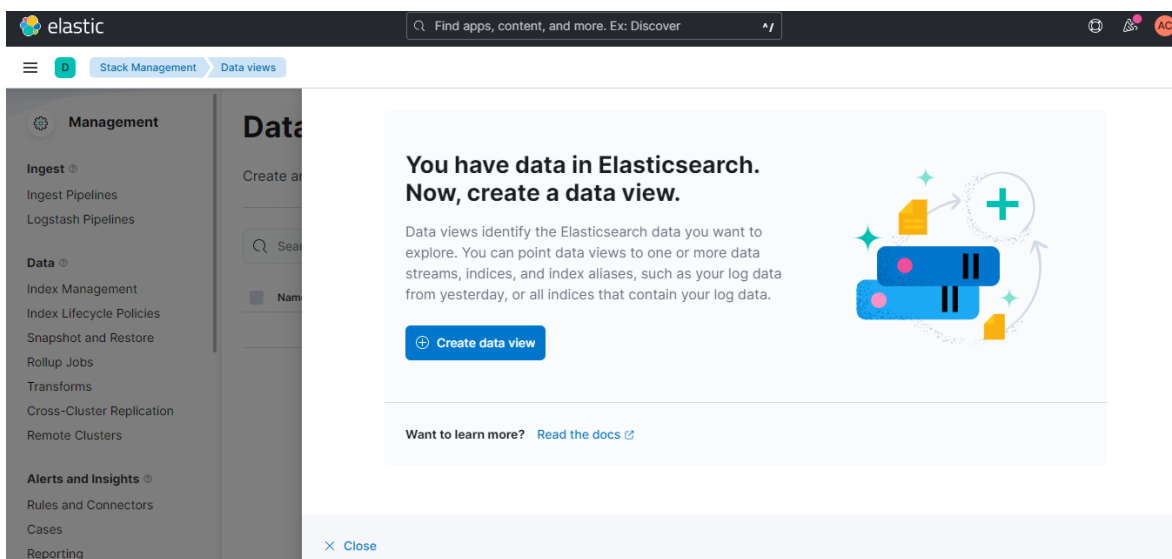
Nos dirigiremos al menú > Deslizaremos hacia la parte de abajo > Seleccionamos la opción de “Stack Management”



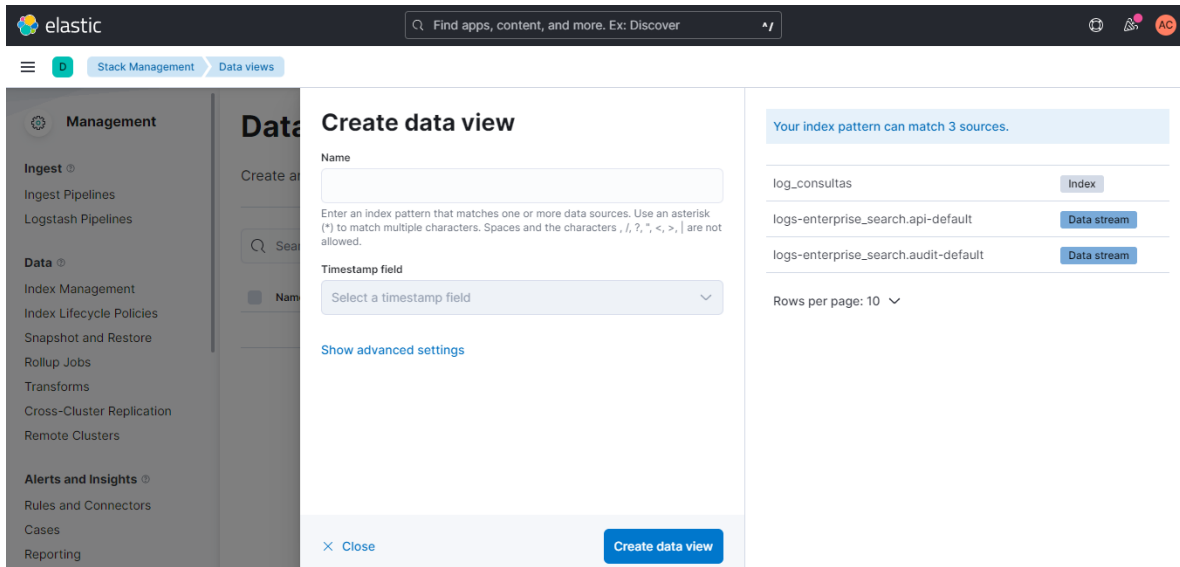
Se visualizar la siguiente pantalla:



Deslizamos hacia abajo > Buscamos la opción de “Kibana” > Clic en “Data Views” > Se visualizará la siguiente pantalla:

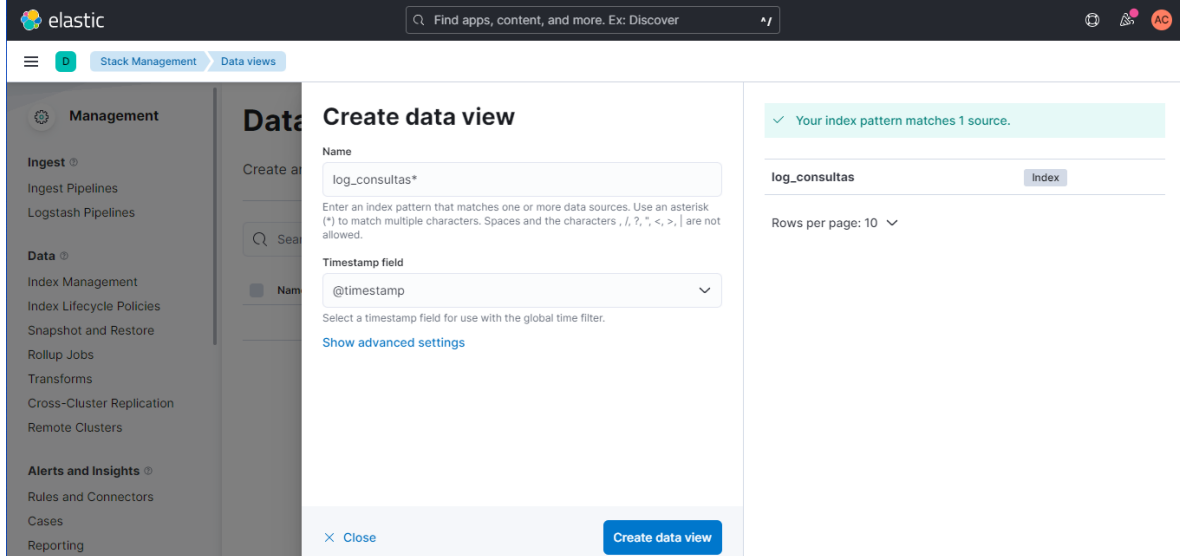


Clic en “Create data view” > Se visualizará en pantalla lo siguiente:

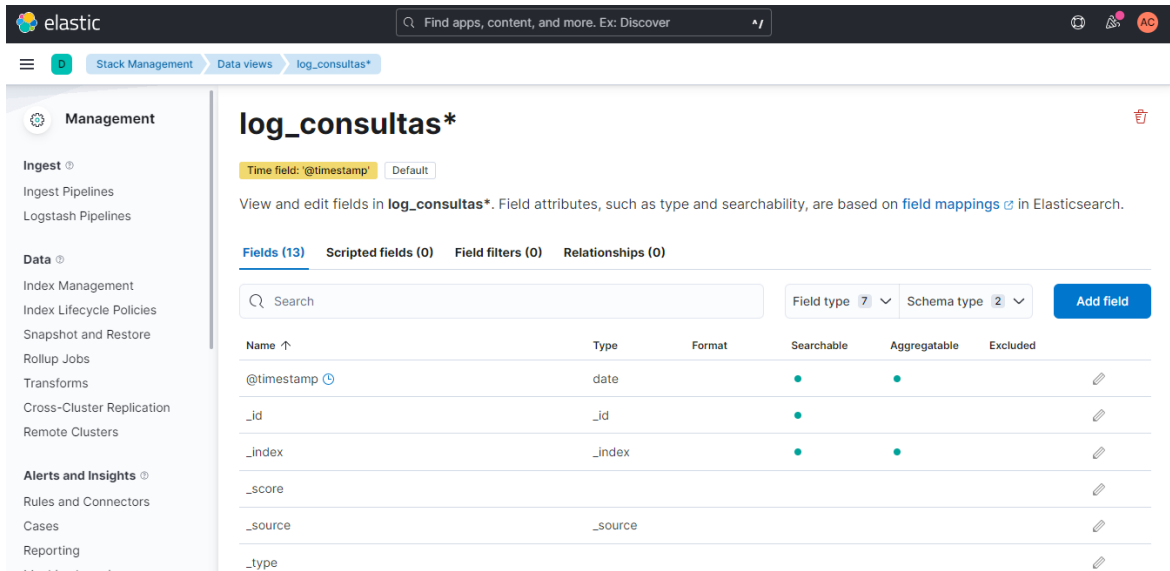


Podemos observar del lado derecho que se encuentran 3 index.

En el apartado de “Name” escribiremos “log_consultas*” (El símbolo “*” lo pondrá en automático) > Reconocerá la coincidencia y el campo de “Timestamp field” se llenará en automático > Se visualizará en pantalla:

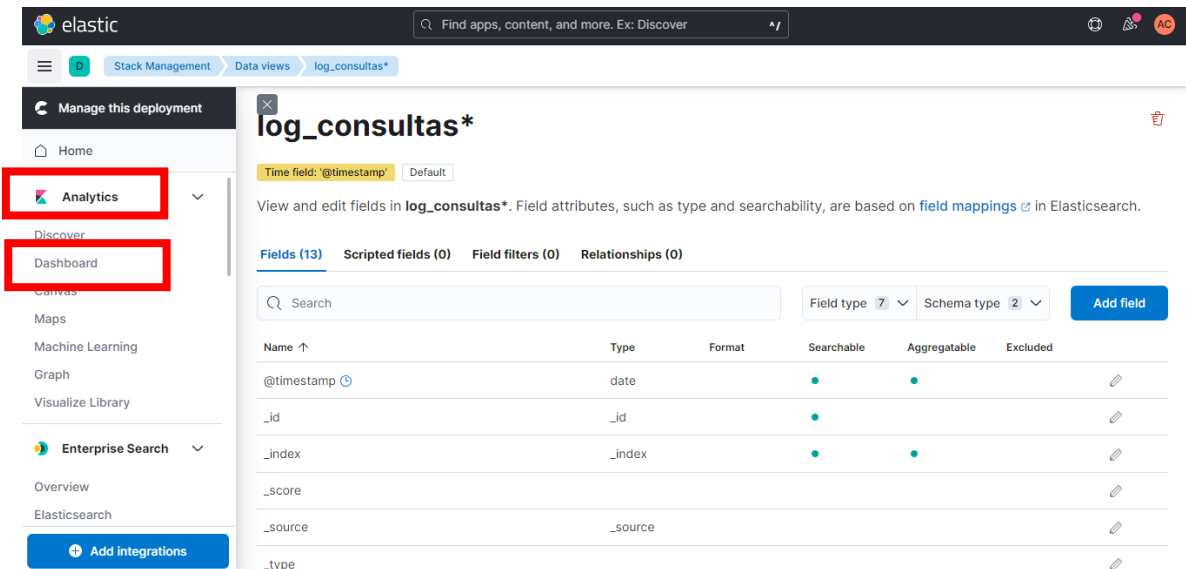


De lado derecho podemos observar que el index pattern tiene una coincidencia > Clic en “Create data view” > Se visualizará en pantalla:

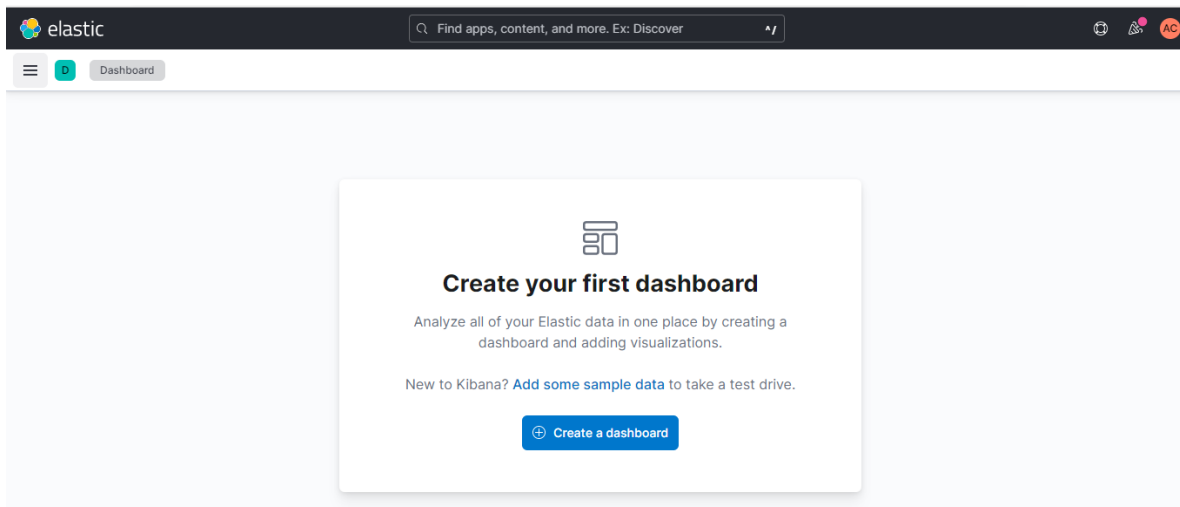


Nota: En esta versión no es necesario crear el index pattern y después el data view, se crean en simultaneo lo que nos permite ahorrar un paso.

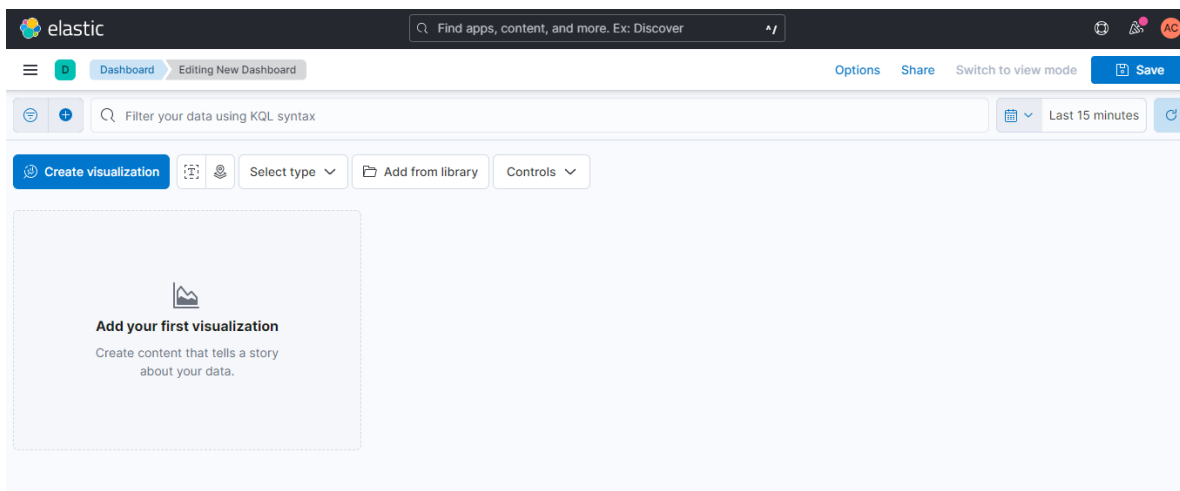
A continuación, nos dirigiremos nuevamente al menú > Buscaremos la opción de “Analytics”
> Clic en “Dashboard”



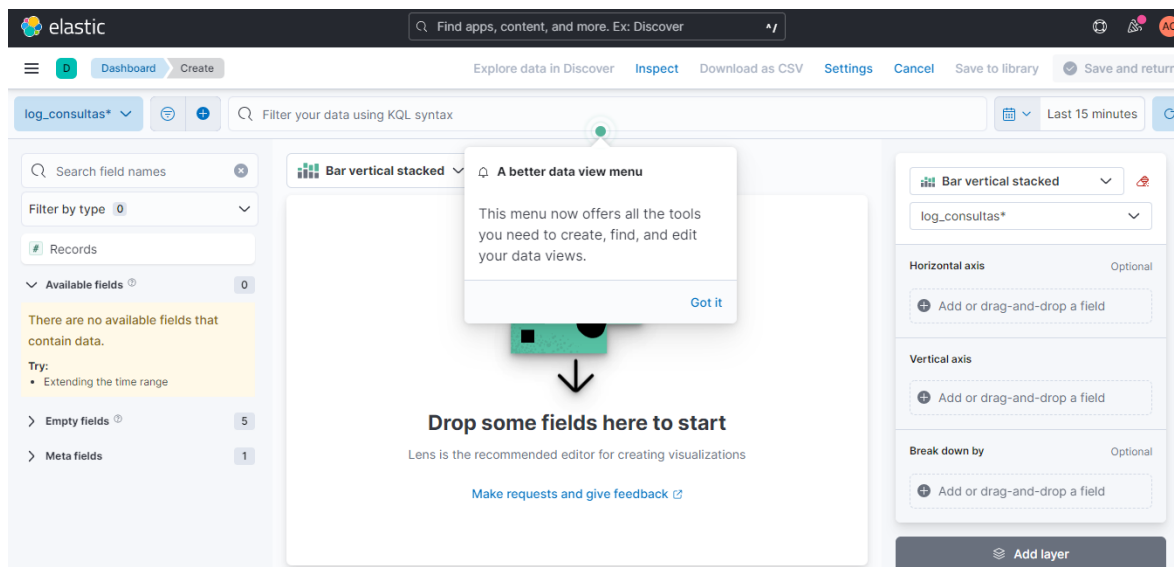
Se visualizará la siguiente pantalla:



Clic en “Create a dashboard” > Se visualizará la siguiente pantalla:

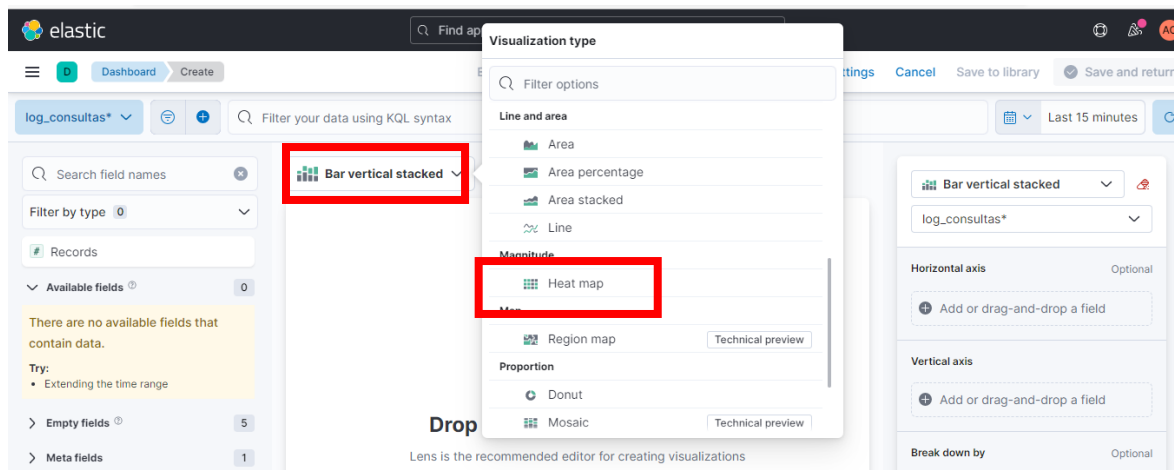


Clic en “Create visualization” > Se visualizará en pantalla:



Este apartado nos permitirá configurar las características para nuestra visualización.

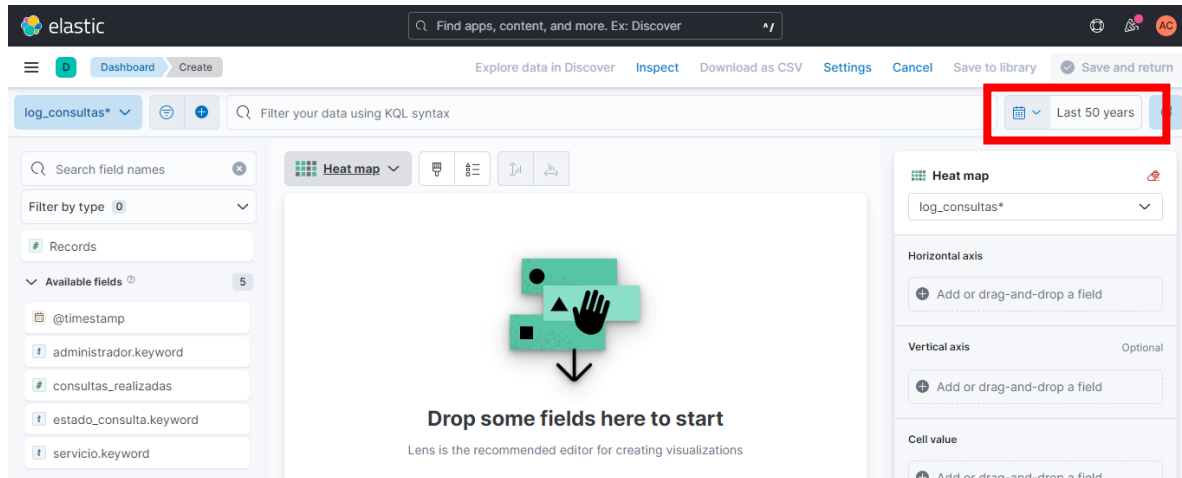
Clic en “Bar vertical stacked” > Se desplegará un menú y buscaremos la opción de “Heat map” > Clic



Ahora podemos ver que se ha cambiado el formato de la gráfica.

Es importante remarcar que se debe seleccionar un rango de fechas amplio para que nuestros datos aparezcan, de lo contrario NO saldrán disponibles los datos ⁶.

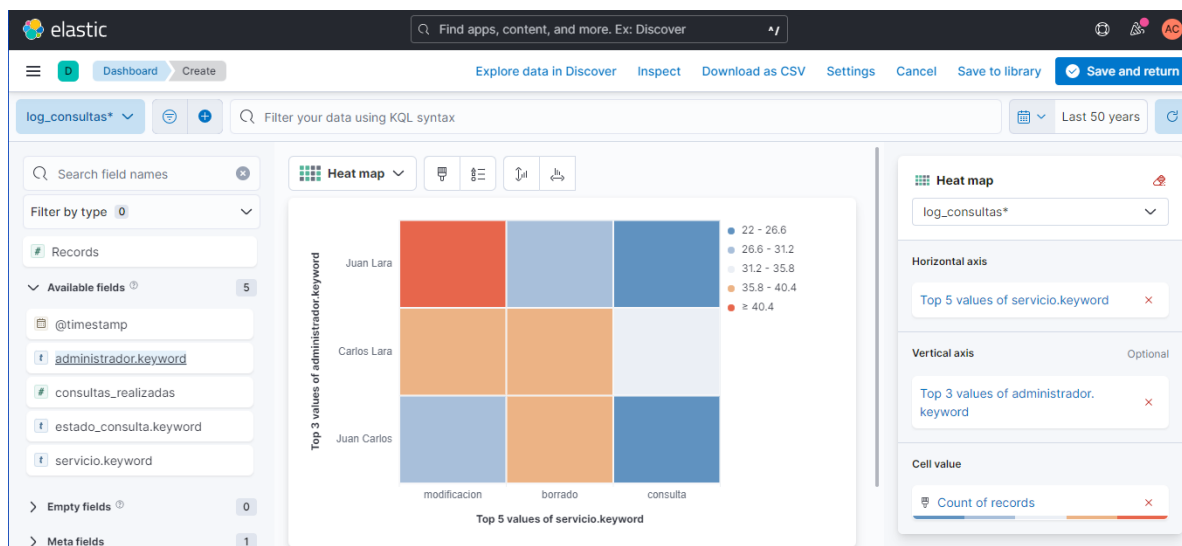
Nos ubicamos en la parte superior derecha > Clic en calendario > Configuramos fecha



Una vez aquí podremos seleccionar la configuración para nuestro mapa, es decir los axis que se muestran en la barra lateral derecha.

Siendo para el eje de las X “servicio” y para el eje de las Y “administrador”.

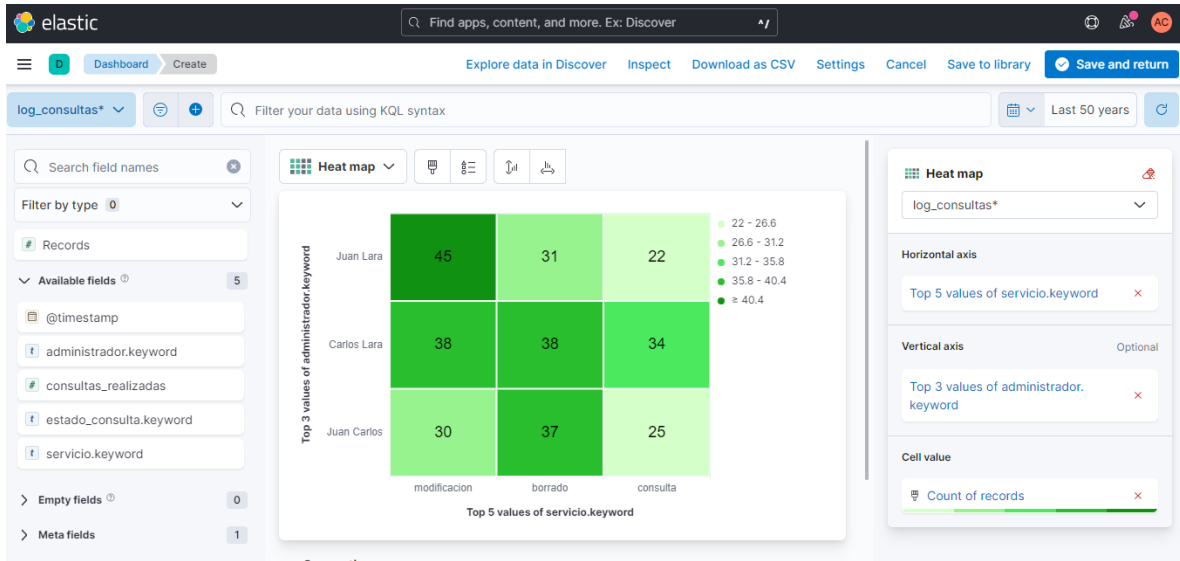
- Lo que se visualizará en pantalla:



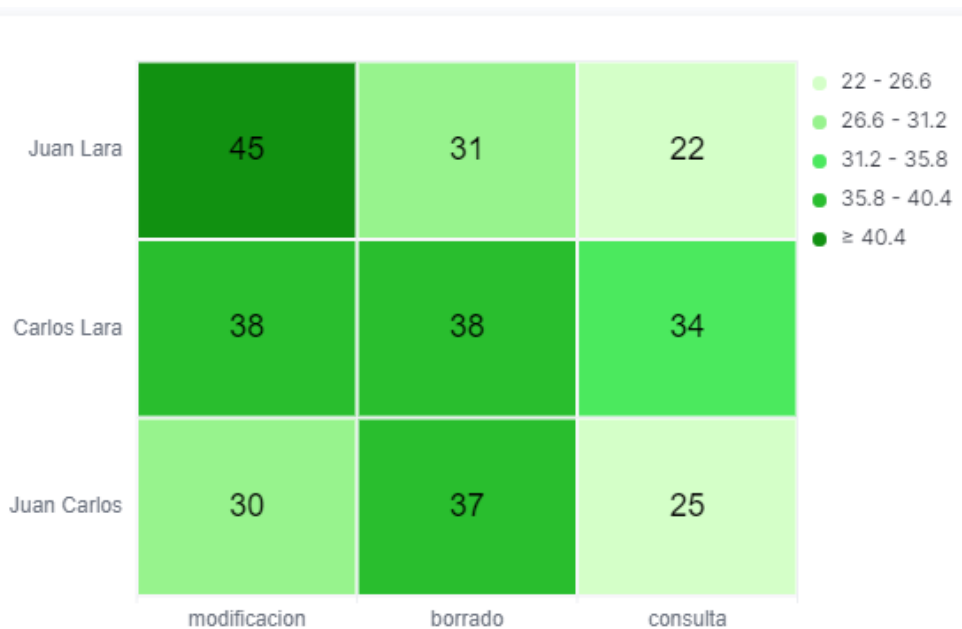
⁶ Debe existir una forma de optimizar este apartado, es necesario identificarla.

Por último, modificamos el color para generar un mapa como el solicitado.

- Se visualizará en pantalla lo siguiente:



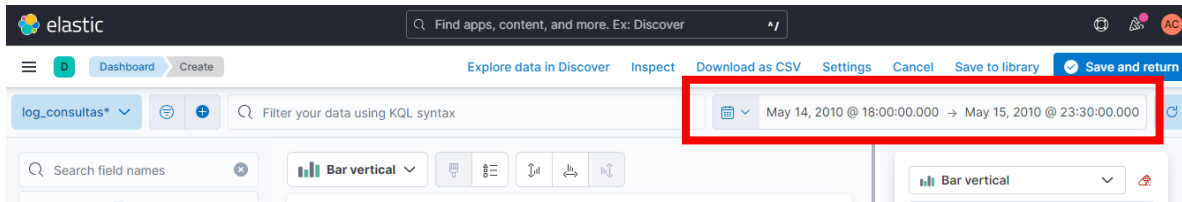
Resultado final:



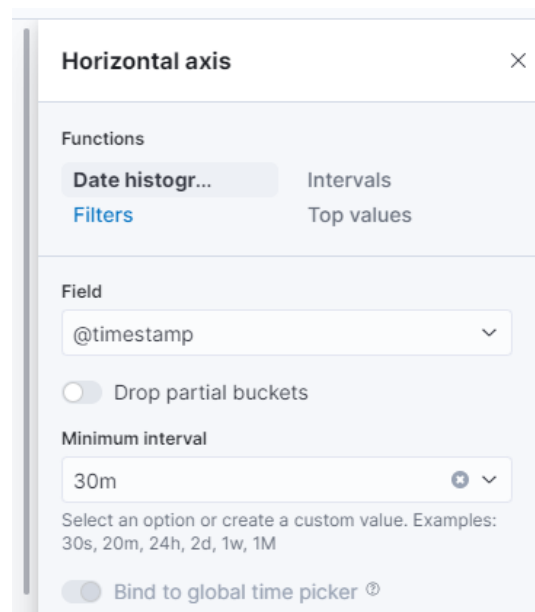
Para la creación del gráfico de barras seleccionaremos la opción de “Bar vertical”.

Ajustaremos el calendario para poder visualizar el contenido.

- La fecha correcta será:



En el axis X seleccionaremos el campo de “@timestamp” para decir que queremos la serie de tiempo > Seleccionamos “Date histogram” > Ajustamos el intervalo de tiempo como “30m”.



En el axis Y escogeremos “consultas_realizadas” debido a que necesitamos la cantidad total de consultas en el tiempo. Posterior a ello se utilizará un filtro para que solo muestren las que en “estado_consulta” sea un error.

Vertical axis

Quick functions Formula

Field

consultas_realizadas

Add advanced options

Filter by

estado_consulta : "error"

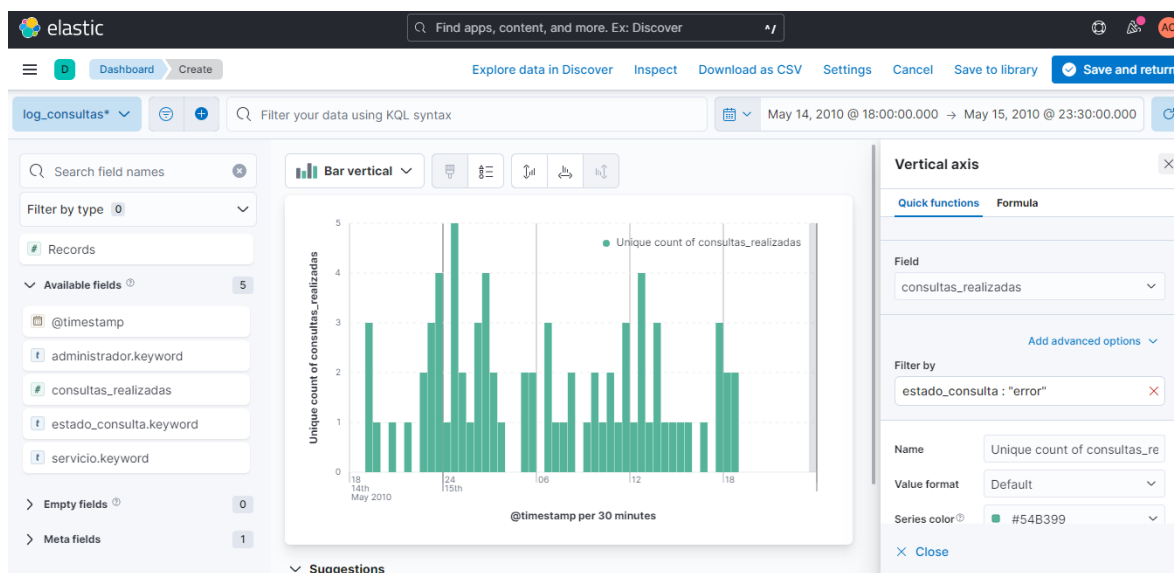
Name

Unique count of consultas_re

Value format

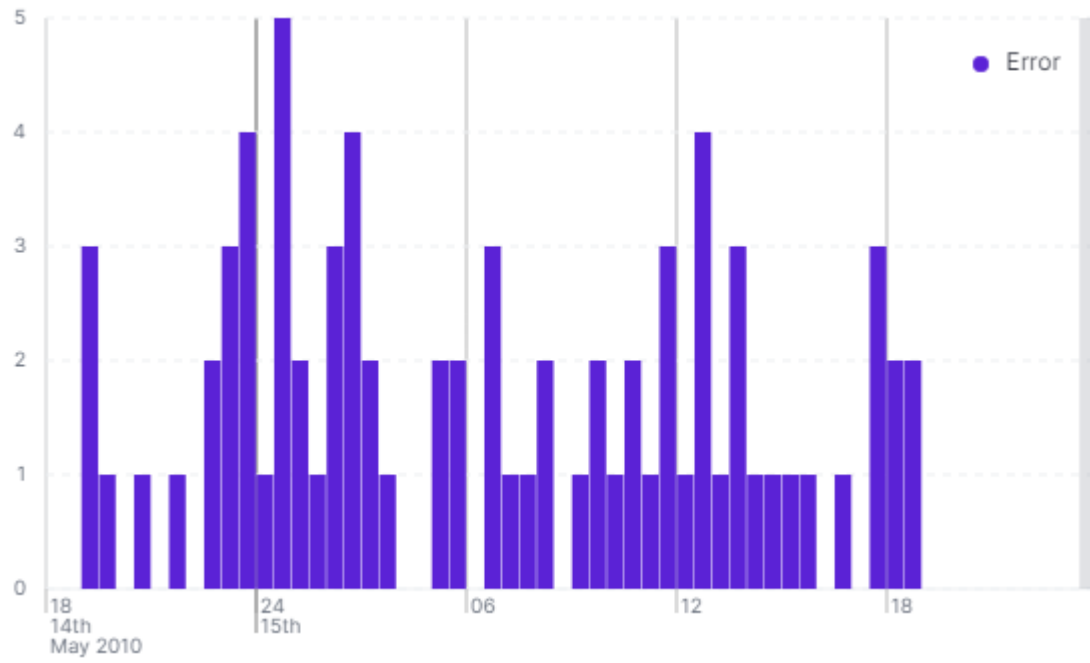
Default

- Se visualizará en pantalla:



Por último, cambiamos el color.

- El resultado final será:



Bibliografía

- *Elasticsearch* (8.2.3). (2015). [Servidor de búsqueda].
<https://www.elastic.co/guide/index.html>