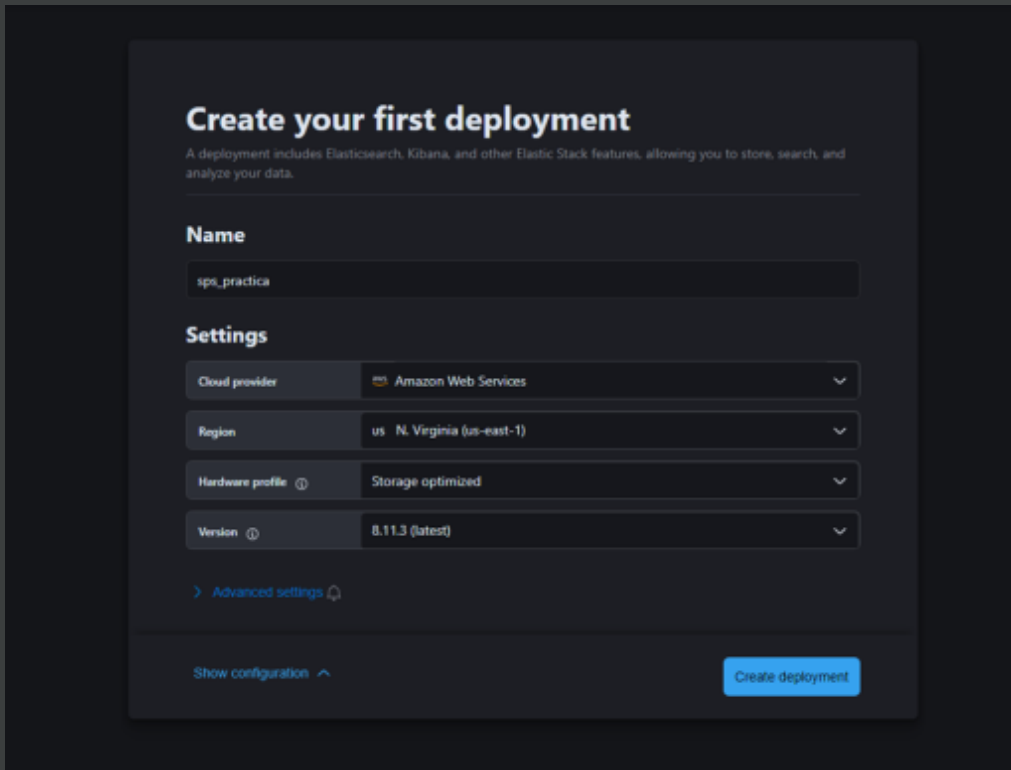


practica_sps_Santiago_Pinedo

Procedimiento tras la creación de mi perfil.

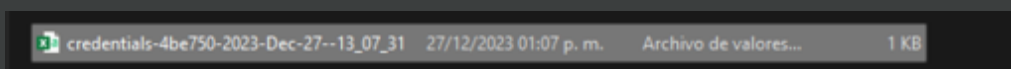
Al crear el perfil, la página me dirigió de manera diferente a la indicada en la práctica, por lo que tuve que buscar en la barra de búsqueda el lugar donde podía realizar un deploy. Una vez en la página correcta, llevé a cabo el deploy siguiendo las instrucciones proporcionadas.



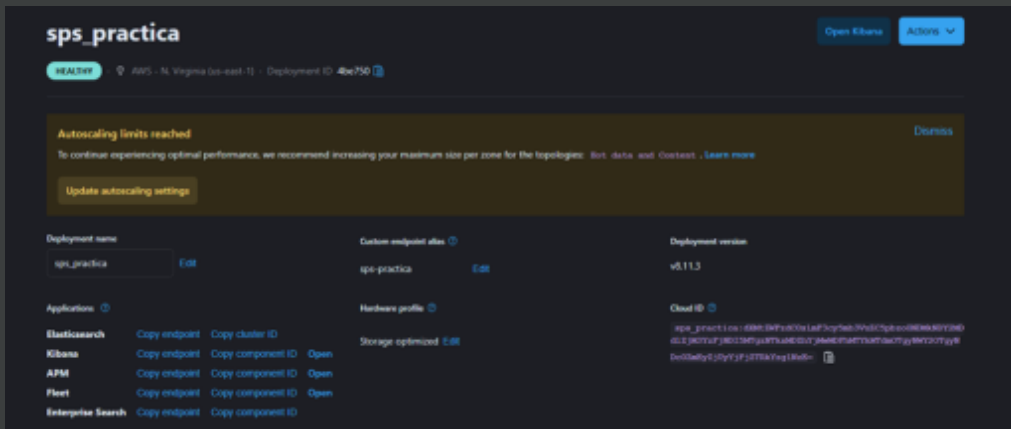
The screenshot shows the 'Create your first deployment' page in the AWS Management Console. The page has a dark theme. At the top, it says 'Create your first deployment' in bold, followed by a subtitle: 'A deployment includes Elasticsearch, Kibana, and other Elastic Stack features, allowing you to store, search, and analyze your data.' Below this is a form with the following fields:

- Name:** A text input field containing 'sps_practica'.
- Settings:** A section with four dropdown menus:
 - Cloud provider:** Amazon Web Services (with the AWS logo).
 - Region:** us N. Virginia (us-east-1).
 - Hardware profile:** Storage optimized (with a help icon).
 - Version:** 8.11.3 (latest) (with a help icon).
- Advanced settings:** A link with a right-pointing arrow and a help icon.
- Show configuration:** A link with an upward-pointing arrow.
- Create deployment:** A blue button.

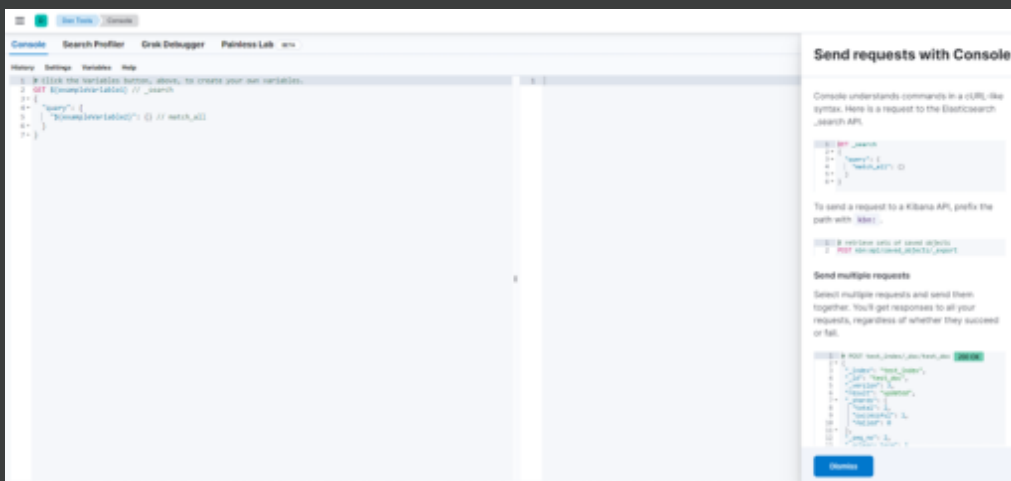
Una vez que hice clic en el botón "create deployment", se me proporcionaron las credenciales respectivas.



Al finalizar el proceso, la página me redirigió a otra ventana, por lo que tuve que buscar los deploys en mi perfil y desde allí realizar el lanzamiento de Kibana.



Una vez abierto Kibana, me dirigí a la opción de DevTools para realizar la creación de un índice.



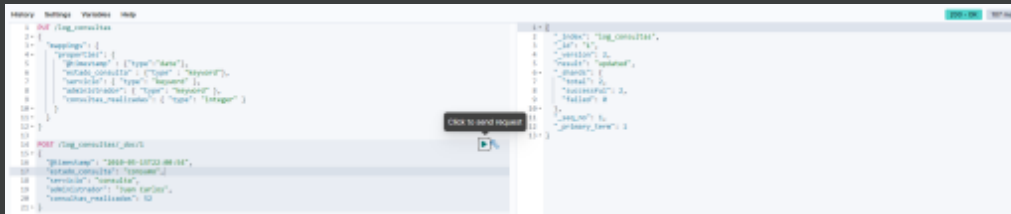
En esta sección hice uso de ChatGPT para auxiliarme en la creación del índice y la explicación del código de este.

```
/*Definicion del mapping del indice*/
PUT /log_consultas
{
  "mappings": {
    "properties": {
      "@timestamp": { "type": "date" },
      "estado_consulta": { "type": "keyword" },
      "servicio": { "type": "keyword" },
      "administrador": { "type": "keyword" },

```

Este código hace lo siguiente:

Una vez creado el indice realice un post con los datos proporcionados para el indice, lo cual me arroja un response 200 ok



Para obtener el mapping de la interpretación de Elasticsearch se utiliza el siguiente código:

```
GET /log_consultas/_mapping
```

El cual nos da el siguiente response:

```
{
  "log_consultas": {
    "mappings": {
      "properties": {
        "@timestamp": {
          "type": "date"
        },
        "administrador": {
          "type": "keyword"
        },
        "consultas_realizadas": {
          "type": "integer"
        },
        "estado_consulta": {
          "type": "keyword"
        },
        "servicio": {
          "type": "keyword"
        }
      }
    }
  }
}
```

```
}
```

Generacion del template a partir del response obtenido para el mapping.

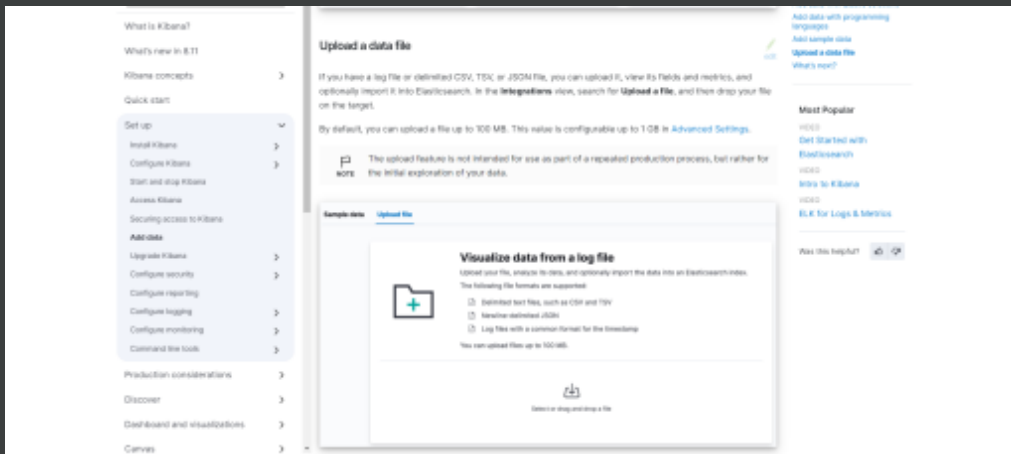
Sobre los templates en Elasticsearch, al definir una plantilla en Elasticsearch, esta se utilizará cada vez que un índice comience con "log_consultas". Esto significa que al realizar una solicitud que incluya esto en sus valores, automáticamente buscará los campos mapeados para poder obtener un código de respuesta 200 OK.

Para crear el template se hizo uso del siguiente código:

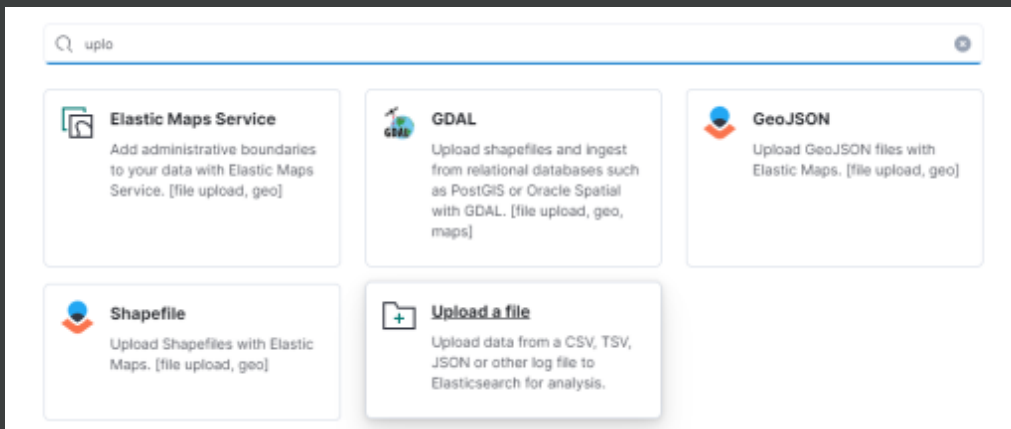
En la linea del PUT se hace uso del api para generar templates (*/_template/nombre_indice*)

```
/*Creacion del template log_consultas*/  
PUT /_template/log_consultas_template  
{  
  "index_patterns": ["log_consultas*"],  
  "mappings": {  
    "properties": {  
      "@timestamp": { "type": "date" },  
      "estado_consulta": { "type": "keyword" },  
      "servicio": { "type": "keyword" },  
      "administrador": { "type": "keyword" },  
      "consultas_realizadas": { "type": "integer" }  
    }  
  }  
}
```

Para cargar los datos del archivo proporcionado hice la búsqueda en la documentación de ElasticSearch



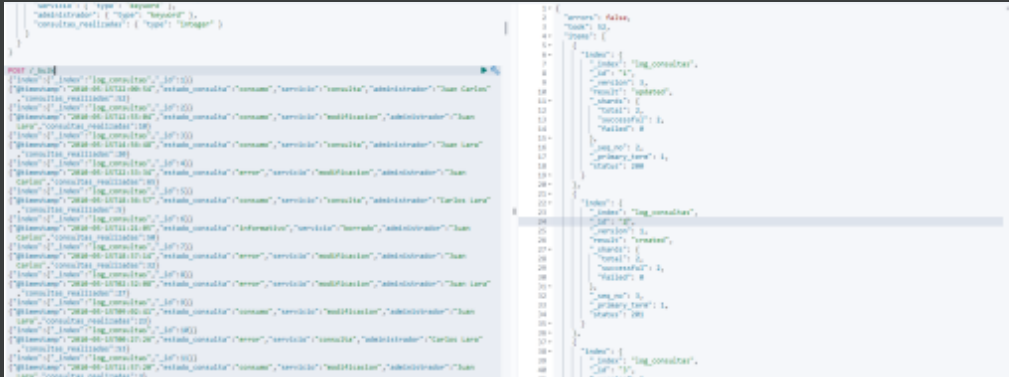
Buscando la integración de "Upload a file"



Automáticamente nos proporciona una descripción y resumen de los datos que contiene el archivo:

File stats					
All fields 6 of 6 total		Number fields 1 of 1 total		Field name 8	Field type 4
Type	Name ↑	Documents (%)	Distinct values	Distributions	
📄	@timestamp	299 (50%)	297		
📄	administrador	299 (50%)	3	2 categories	
📄	consultas_realizadas	299 (50%)	74	min	median max
				0	35 74
📄	estado_consulta	299 (50%)	3	3 categories	
📄	index	0 (0%)	0		
📄	servicio	299 (50%)	3	2 categories	

Al llegar a este punto, pude notar que este procedimiento era más bien para cargar un archivo y crear el índice para atender esos datos al mismo tiempo.



Realizar búsquedas sobre el índice.

Una vez más, realicé la búsqueda de los datos necesarios y ejemplos del uso del API en la documentación de Elasticsearch.

Search API

Returns search hits that match the query defined in the request.

```
GET /my-index-000001/_search
```

Console [Copy as curl](#) [View in Console](#)

Request

```
GET /<target>/_search
```

```
GET /_search
```

```
POST /<target>/_search
```

```
POST /_search
```

Prerequisites

- If the Elasticsearch security features are enabled, you must have the `read` [index privilege](#) for the target data stream, index, or alias. For cross-cluster search, see [Configure privileges for cross-cluster search](#).

To search a [point in time \(PIT\)](#) for an alias, you must have the `read` [index privilege](#) for the alias's data

Una vez obtenida la documentación, para visualizar un ejemplo de las búsquedas, nuevamente hice uso de ChatGPT para comprender la estructura de la primer búsqueda y así poder realizar las siguientes búsquedas

por mi cuenta.



You

me puedes ayudar con este primer ejercicio, me gustaria poder comprender como realizar busquedas con elasticsearch asi que me gustaria que comentaras el coigo del request

Haciendo uso del API de búsquedas (SEARCH), deberás realizar las siguientes consultas:

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



ChatGPT

¡Por supuesto! Vamos a desglosar la solicitud para obtener el número de registros con `estado_consulta` igual a "error" y "consumo" utilizando la API de búsquedas (SEARCH) en Elasticsearch:

json

Copy code

```
GET /log_consultas/_search
{
  "query": {
    "bool": {
      "should": [
        { "match": { "estado_consulta": "error" } },
        { "match": { "estado_consulta": "consumo" } }
      ]
    }
  }
}
```

ChatGPT:

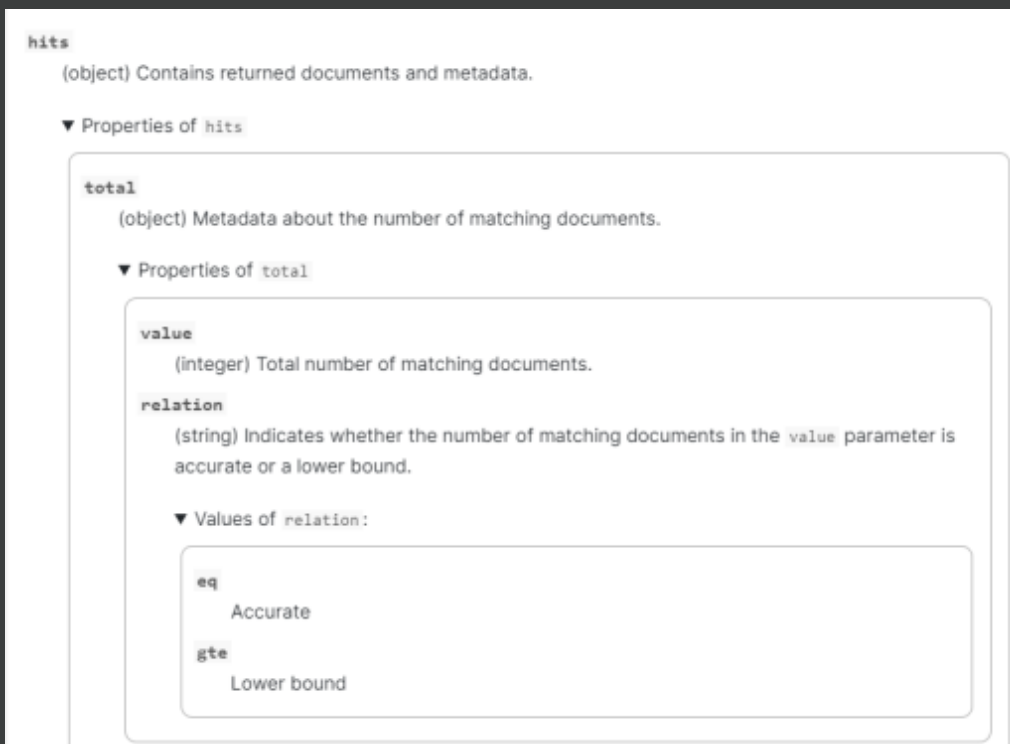
Ahora, expliquemos cada parte de esta solicitud:

- **GET /log_consultas/_search:** Indica que estás realizando una operación de búsqueda en el índice llamado "log_consultas". Puedes ajustar esto al nombre de tu índice.
- **"query":** Define la consulta de búsqueda. En este caso, estamos utilizando un filtro booleano para combinar dos condiciones con el operador OR (`should`).
- **"bool":** Indica que estamos construyendo una consulta booleana.
- **"should":** Representa las condiciones que deben cumplirse. En este caso, estamos utilizando dos condiciones: una para registros con `estado_consulta` igual a "error" y otra para registros con `estado_consulta` igual a "consumo". Como estamos utilizando el operador OR (`should`), la consulta devolverá documentos que cumplan cualquiera de estas condiciones.

Entonces, esta solicitud buscará en el índice "log_consultas" y devolverá los documentos que tengan estado_consulta igual a "error" o "consumo". El resultado incluirá todos los documentos que cumplan al menos una de estas condiciones.

Puedes ejecutar esta solicitud en tu consola de Dev Tools en Kibana o a través de cualquier interfaz que te permita interactuar con Elasticsearch.

Analizando la documentación de Elasticsearch, pude profundizar en la respuesta obtenida. Noté que el campo "Hits" nos devuelve en el campo "total" los metadatos del número de documentos que coincidieron, mientras que el "value" nos indica cuántos registros hicieron match con la consulta.



Teniendo estos datos en mente, tomamos los datos del response para dar respuesta a las preguntas planteadas.

Número de registros con estado_consulta = error y consumo = 182



Número de registros realizados por Juan Lara = 98

```
/*Obtener el número de registros realizados por el administrador Juan Lara.*/
GET /log_consultas/_search
{
  "query": {
    "match": {
      "administrador": "Juan Lara"
    }
  }
}
```


```
{
  "took": 8,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 3,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": {
      "value": 88,
      "relation": "eq"
    }
  }
}
```

Número de registros con estado_consulta = informativo y servicio = borrado = 171

```
/*Obtener el número de registros con estado_consulta igual a informativo y servicio igual a borrado*/
GET /log_consultas/_search
{
  "query": {
    "bool": {
      "must": [ { "estado_consulta": "informativo" } ],
      "filter": [ { "servicio": "borrado" } ]
    }
  }
}
```

```
{
  "took": 1,
  "timed_out": false,
  "_shards": {
    "total": 1,
    "successful": 1,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": {
      "value": 171,
      "relation": "eq"
    }
  }
}
```

Para la consulta extra, busqué cómo realizar la suma de los valores de un campo en la documentación de Elasticsearch. En dicha búsqueda, encontré algunos ejemplos para ello:

 Platform Solutions Customers Resources Pricing Docs

Sum

T-test

Top hits

Top metrics

Value count

Weighted avg

Pipeline aggregations

Geospatial analysis

EQL

ES|SQL

SQL

Scripting

Data management

Autoscaling

Monitor a cluster

Roll up or transform your data

Set up a cluster for high availability

Snapshot and restore

Secure the Elastic Stack

Watcher

Elastic Docs • Elasticsearch Guide [8.11] • Aggregations • Metrics aggregations

Sum aggregation

A single-value metrics aggregation that sums up numeric values that are extracted from the aggregated documents. These values can be extracted either from specific numeric or *histogram* fields.

Assuming the data consists of documents representing sales records we can sum the sale price of all hats with:

```
POST /sales/_search?size=0
{
  "query": {
    "constant_score": {
      "filter": {
        "match": { "type": "hat" }
      }
    }
  },
  "aggs": {
    "hat_prices": { "sum": { "field": "price" } }
  }
}
```

Console Copy as curl View in Console

Resulting in:

```
{
  ...
  "aggregations": {
    "hat_prices": {
      ...
    }
  }
}
```

On this page

Script

Missing value

Histogram fields

Most Popular

VIDEO Get Started with Elasticsearch

VIDEO Intro to Kibana

VIDEO ELK for Logs & Metrics

Was this helpful?

Analizando esto, podemos ver que hay una sección de query, la cual indicará la búsqueda que vamos a realizar. Posteriormente, está la función "aggs", la cual permite realizar la suma de valores de un campo indicado por "field". De esta manera, le indicamos con el query que deseamos que coincida con

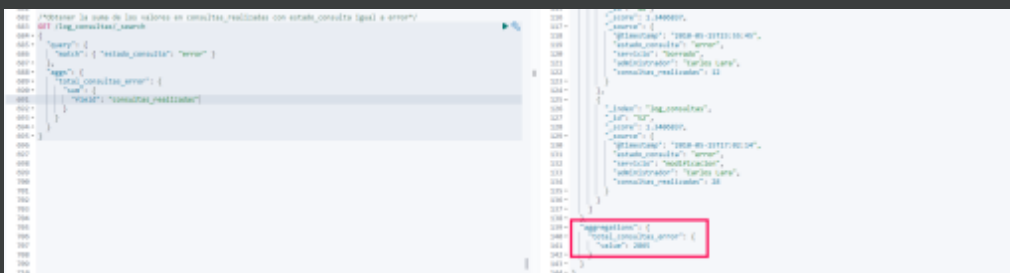
estado_consulta = error, y con "agg" le indicamos que la variable de salida será "*total_consultas_error*", la cual almacenará la suma del campo *consultas_realizadas*.

Tenemos entonces el siguiente código para esta consulta:

```
/*Obtener la suma de los valores en consultas_realizadas con estado_consulta igual a error*/  
GET /log_consultas/_search  
{  
  "query": {  
    "match": { "estado_consulta": "error" }  
  },  
  "aggs": {  
    "total_consultas_error": {  
      "sum": {  
        "field": "consultas_realizadas"  
      }  
    }  
  }  
}
```

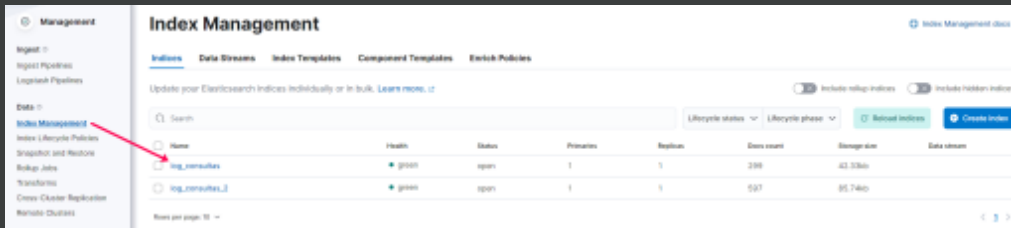
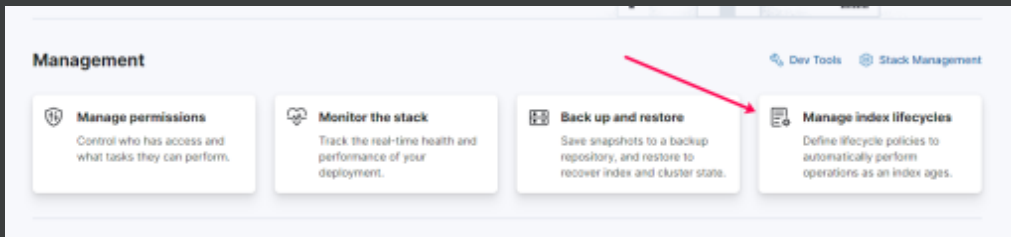
Obtenemos la salida:

Consultas realizadas con valor = error = 2865

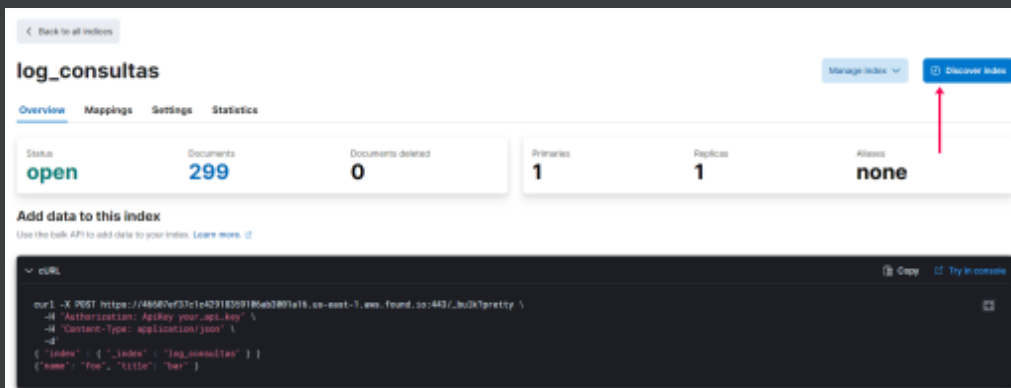


Realizar un tablero para visualizar información de empleados

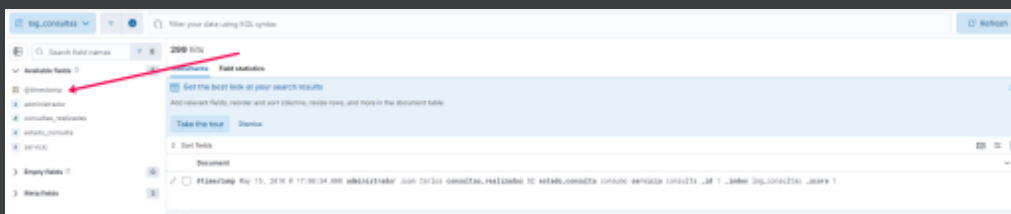
Para acceder al management index tuve que hacerlo desde otra opción

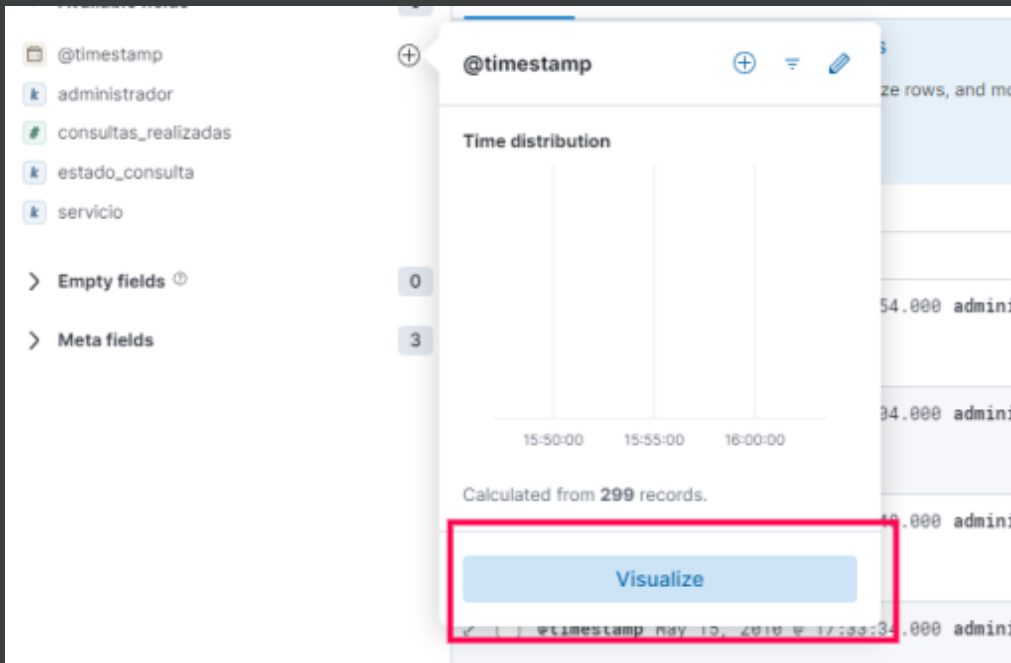


Una vez dentro del indice "log_consultas" pulse en el botón de "Discover index"



Al hacer clic en ese botón, se nos abre la siguiente vista. En ella, al hacer clic en el campo deseado, podemos abrir el menú de "visualize".





Una vez dentro de la interfaz de visualización, encontré que esta se asemejaba a otras interfaces de visualización de datos, lo que me permitió jugar con los parámetros para generar las visualizaciones deseadas.

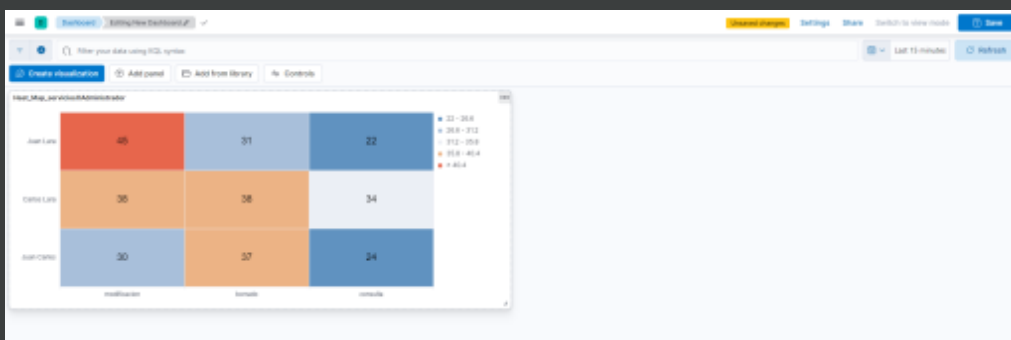
En la parte de configuración del gráfico podemos colocar los valores para los ejes, seleccionar los colores, etc.

Creación del Heat Map

En el caso del heat map solamente tuve que arrastrar los valores de *servicio* y *administrador* a los respectivos campos.



Al guardar el grafico da la opción de "guardar e ir al dashboard" lo cual nos crea un nuevo Dashboard en automatico.

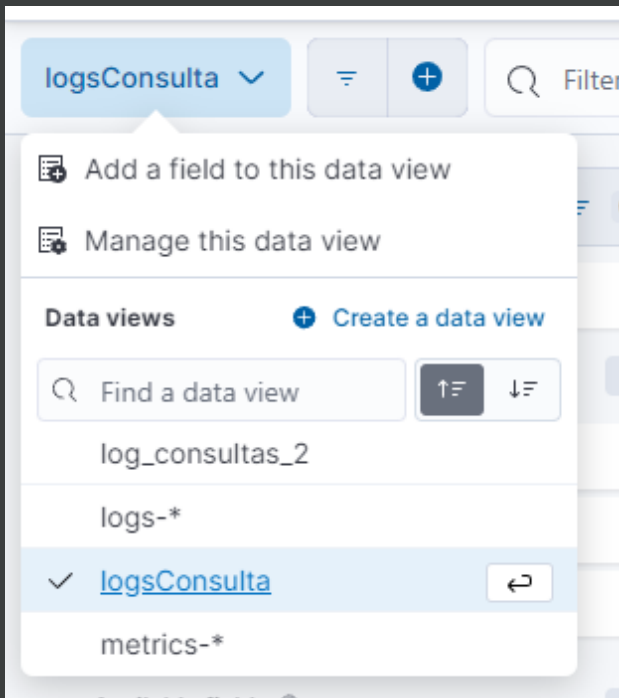


Una vez que estamos en el dashboard podemos dar clic en crear visualización y agregar la que nos hace falta.

Para agregarlo es necesario crear un data_view

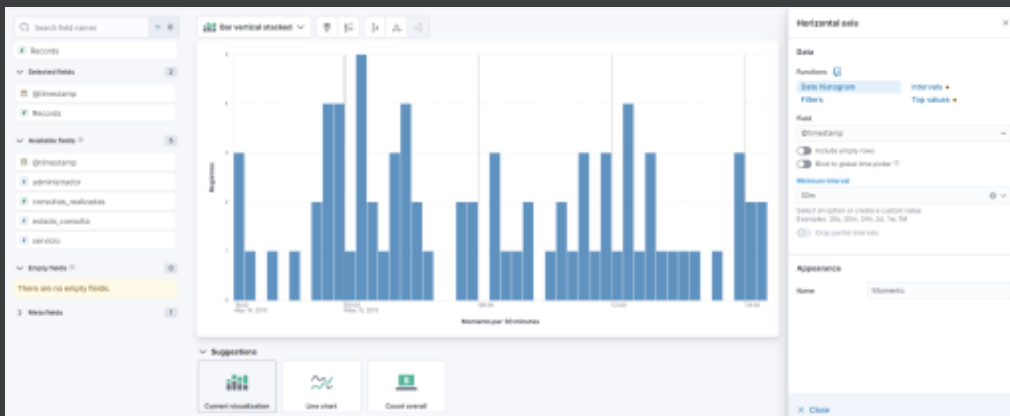
Matching sources	
log_consultas	Index

De esta manera se agrega nuestros "sets de datos"

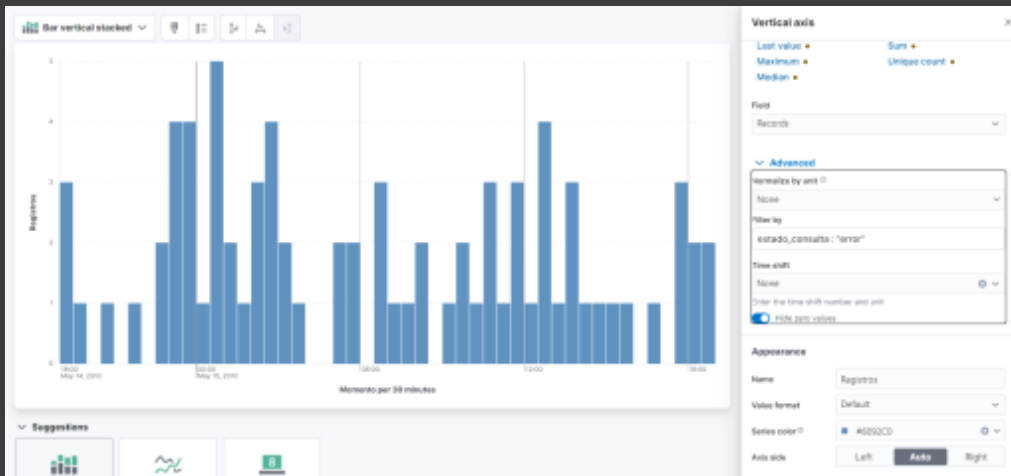


Configuración de la visualización de grafico de barras:

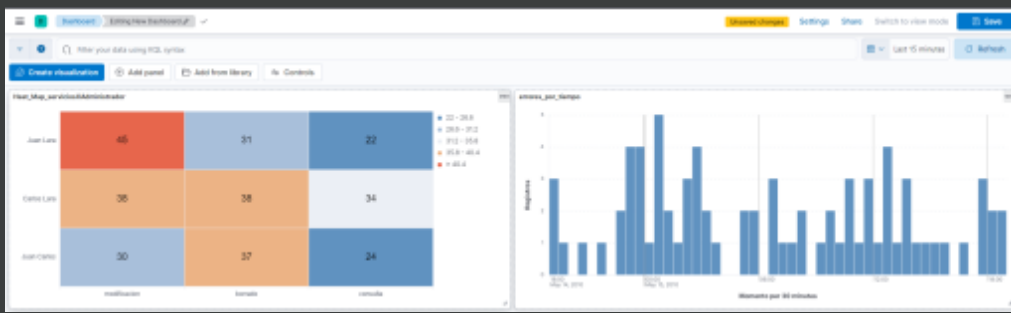
Para el eje horizontal (timestamp)



Para el eje vertical (registros con error):



Finalmente agregamos la visualización al dashboard y podemos guardar el dashboard creado:



[Enlace al dashboard](#)

[Enlace al repositorio de código](#)