Services and Processes Solutions, S.A. de C.V.

Práctica
ElasticSearch

Candidato:
Angel Uzziel Espinoza López

Fecha: 24/05/2021

Índice

Introducción

¿Qué es ElasticSearch?

Elasticsearch es un motor de analítica y análisis distribuido, gratuito y abierto para todos los tipos de datos, incluidos textuales, numéricos, geoespaciales, estructurados y no estructurados.
Conocido por sus API REST simples, naturaleza distribuida, velocidad y escalabilidad, Elasticsearch es el componente principal del Elastic Stack, un conjunto de herramientas gratuitas y abiertas para la ingesta, el enriquecimiento, el almacenamiento, el análisis y la visualización de datos.

¿Cómo funciona?

Los datos sin procesar fluyen hacia Elasticsearch desde una variedad de fuentes, incluidos logs, métricas de sistema y aplicaciones web. La ingesta de datos es el proceso mediante el cual estos datos son parseados, normalizados y enriquecidos antes de su indexación en Elasticsearch. Una vez indexados en Elasticsearch, los usuarios pueden ejecutar consultas complejas sobre sus datos y usar agregaciones para recuperar resúmenes complejos de sus datos. Desde Kibana, los usuarios crean visualizaciones poderosas de sus datos, comparten dashboards y gestionan el Elastic Stack.

¿Cuáles son sus alcances?

La velocidad y escalabilidad de Elasticsearch y su capacidad de indexar muchos tipos de contenido significan que puede usarse para una variedad de casos de uso:

- Búsqueda de aplicaciones
- Búsqueda de sitio web
- Búsqueda Empresarial
- Logging y analíticas de log
- Métricas de infraestructura y monitoreo de contenedores
- Monitoreo de rendimiento de aplicaciones
- Análisis y visualización de datos geoespaciales
- Analítica de Seguridad
- Analítica de Negocios

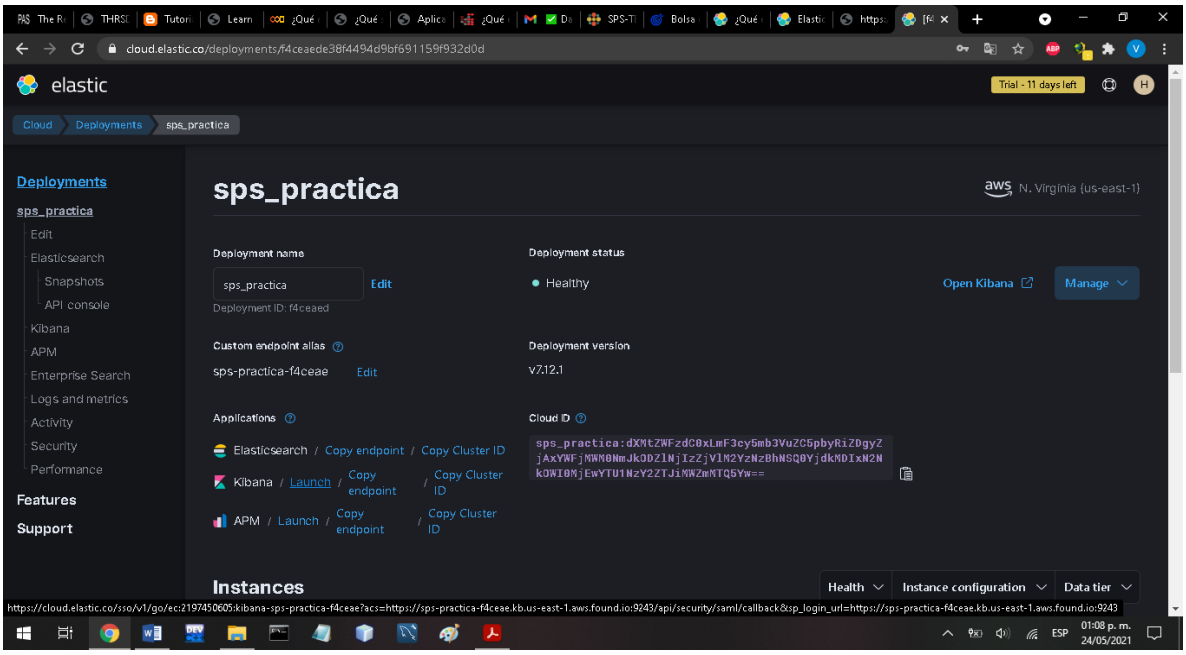
Elasticsearch también es una plataforma de búsqueda en casi tiempo real, lo que implica que la latencia entre el momento en que se indexa un documento hasta el momento en que se puede buscar en él es muy breve: típicamente, un segundo. Como resultado, Elasticsearch está bien preparado para casos de uso con restricciones de tiempo como analítica de seguridad y monitoreo de infraestructura.

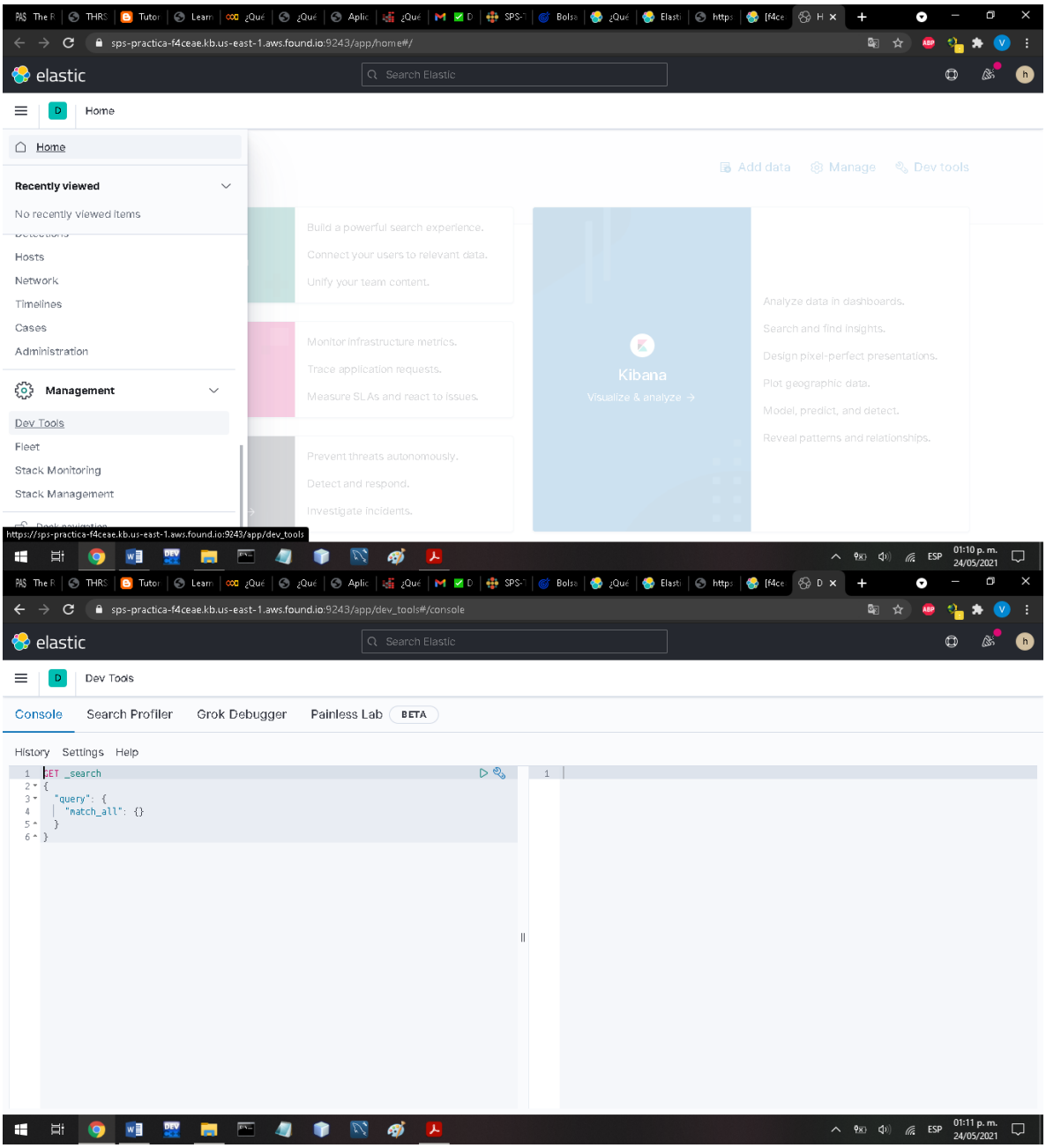
Desarrollo de la práctica

Creación de un índice.

1. Seleccionar Dev Tools.

- Para empezar se creó una cuenta en Elastic, se inició la prueba gratuita y se creó el proyecto con las características indicadas.
- Enseguida se lanza la herramienta web Kibana dando click en Launch y, después, en el menú vertical seleccionamos Dev Tools.





2. Crea un índice con el nombre de log_consultas a partir del siguiente JSON:

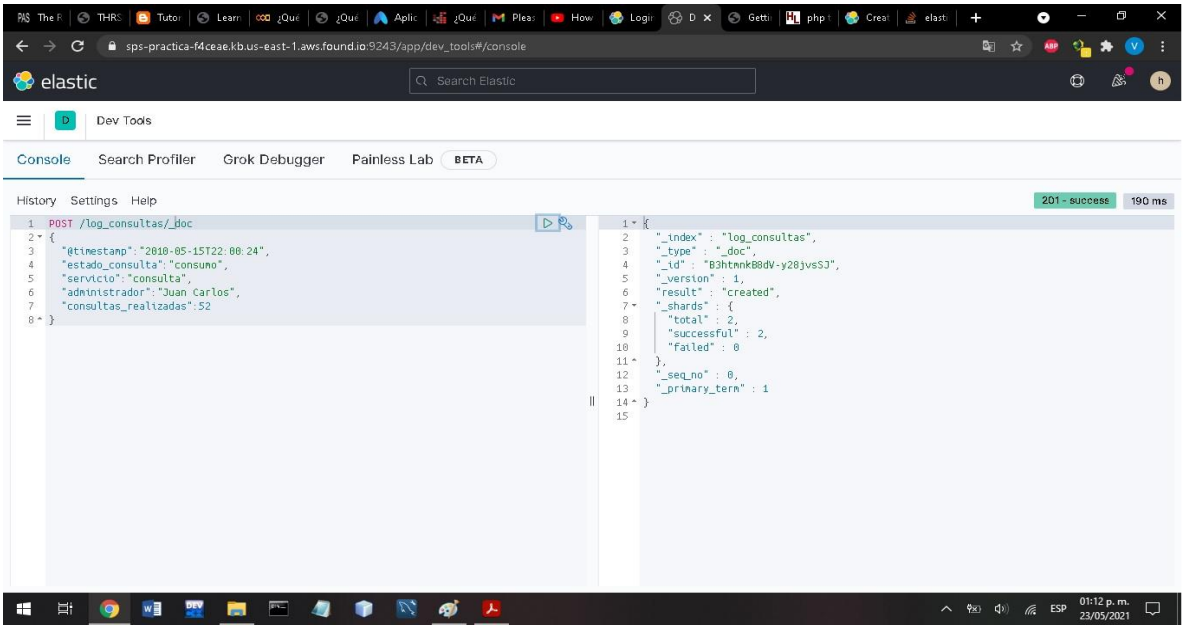
```
{
        "@timestamp":"2010-05-15T22:00:54",
        "estado_consulta":"consumo",
        "servicio":"consulta",
        "administrador":"Juan Carlos",
        "consultas_realizadas":52
}
```

- Para crear el índice con las características anteriores se procedió a escribir el comando POST seguido del nombre del índice, en este caso log_consultas, y el nombre del documento, _doc.

  Como se muestra a continuación:

  POST /log_consultas/_doc
  {
    "@timestamp":"2010-05-15T22:00:24",
    "estado_consulta":"consumo",
    "servicio":"consulta",
    "administrador":"Juan Carlos",
    "consultas_realizadas":52
  }

- Oprimimos el botón de Run(play) para ejecutar dicho comando, obteniendo una respuesta como se muestra en la siguiente imagen:

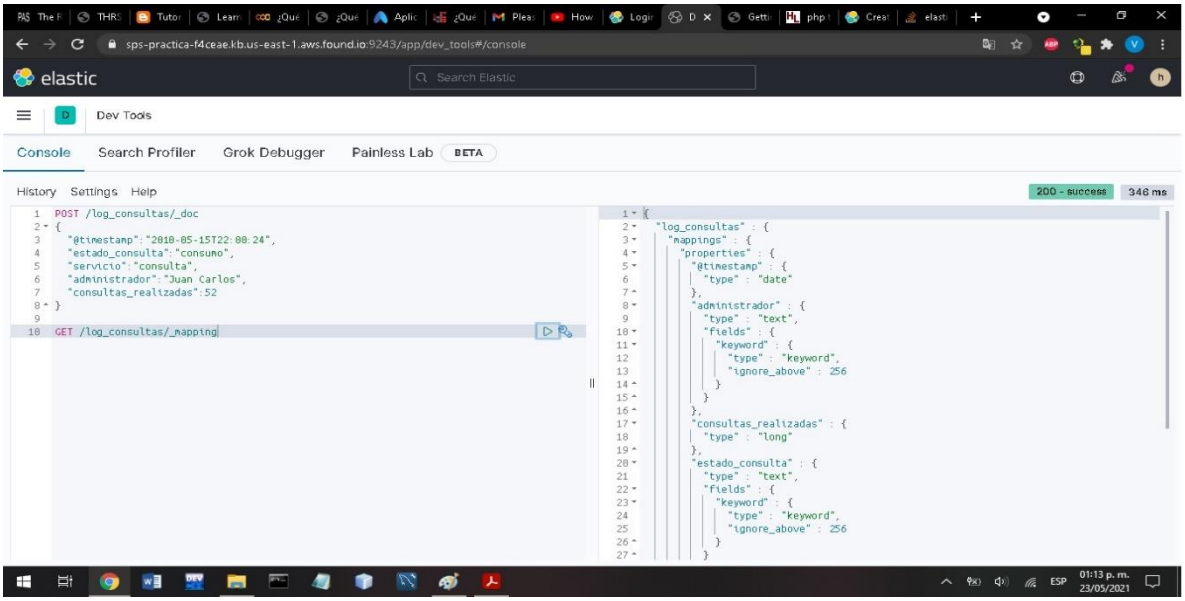


3. Obtén el mapping del índice anterior y genera un template a partir de este índice haciendo uso del API de plantillas (TEMPLATE). El patrón para el índice debe ser: “log_consultas*”. Verifica los tipos de datos hagan sentido con la información almacenada.

- Para obtener el mapeo del índice creado en el paso anterior utilizaremos el siguiente comando que utiliza la Mapping API:

GET /log_consultas/_mapping

- Mostrando la estructura del índice con nombre del campo clave y el tipo de dato del campo clave. Como se muestra a continuación:



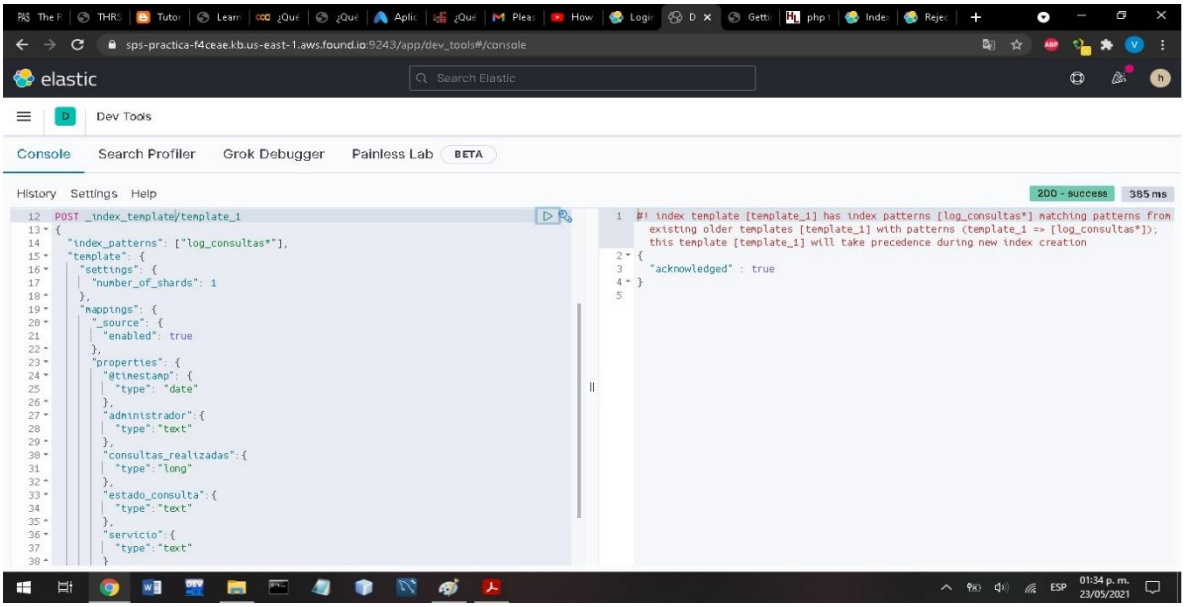
- Para crear la plantilla de índice con el patrón de índice, log_consultas*, se escribió el siguiente comando con la Template API y los valores de el mapeo del paso anterior:

```
POST _index_template/template_1
{
  "index_patterns": ["log_consultas*"],
  "template": {
    "settings": {
      "number_of_shards": 1
    },
    "mappings": {
      "_source": {
        "enabled": true
      },
      "properties": {
        "@timestamp": {
          "type": "date"
        },
        "administrador":{
          "type":"text"
        },
        "consultas_realizadas":{
          "type":"long"
        },
        "estado_consulta":{
```

```
        "type":"text"
      },
      "servicio":{
        "type":"text"
      }
      }
    }
  }
}
```

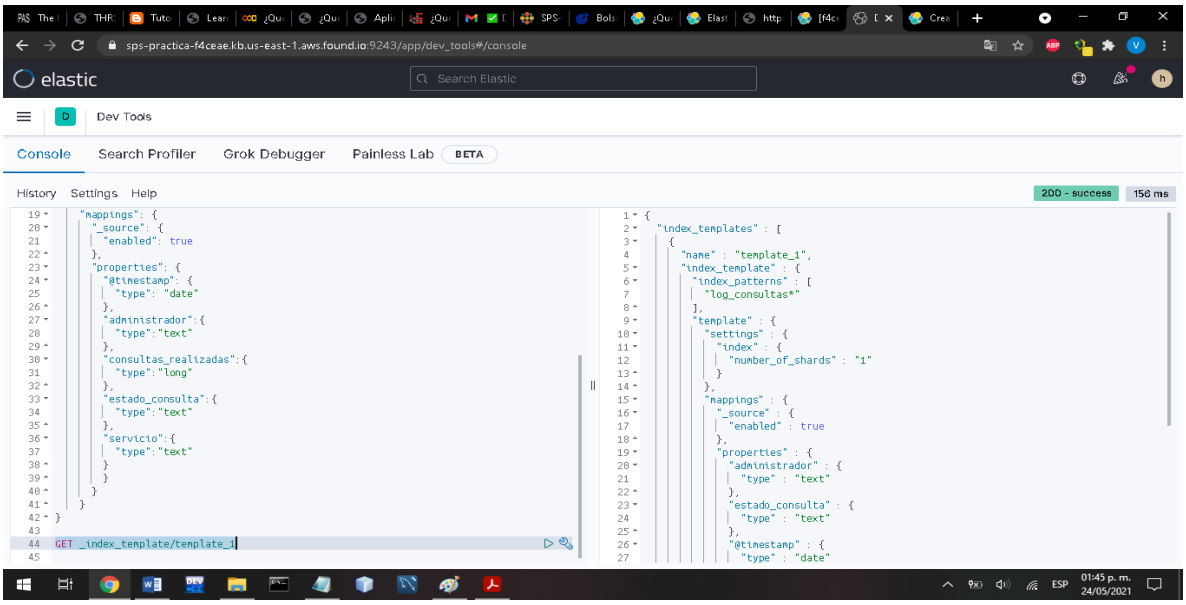
- Como se puede notar escribimos los comandos unos debajo de otros para mantener el historial de comandos que hemos ejecutado. Esto no afecta a los comandos anteriores ya que solo se ejecuta lo que hay por debajo del comando y por encima de la declaración de un nuevo comando.

- Oprimimos el botón de Run(play) para ejecutar dicho comando, obteniendo una respuesta como se muestra en la siguiente imagen:



- Para verificar los datos de la plantilla (template) utilizamos el siguiente comando:

GET _index_template/template_1

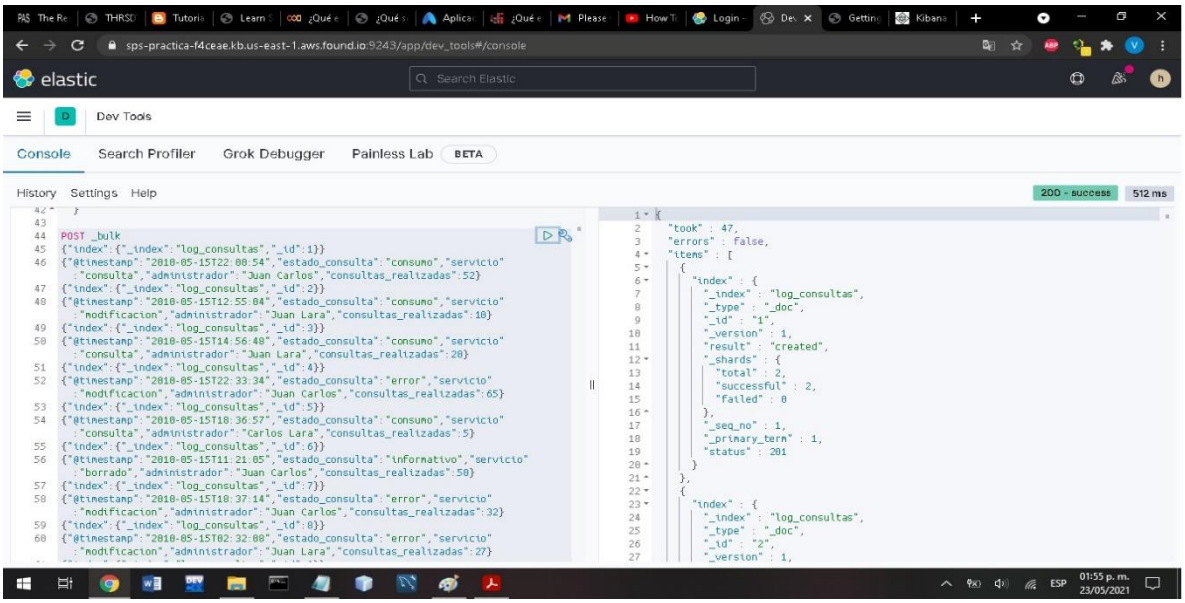


4. Una vez definido tu template cargaras una serie de documentos en tu índice utilizando el archivo que se encuentra en el escritorio: log_consultas.json. Para esto utiliza el API (BULK).

- Para realizar esto escribimos el siguiente comando con la Bulk API:

POST _bulk {Contenido del archivo Registros.json}

- Y oprimimos el icono Run (play) para ejecutar el comando, dándonos como resultado lo siguiente:



Y eso es todo con respecto a la primer parte.

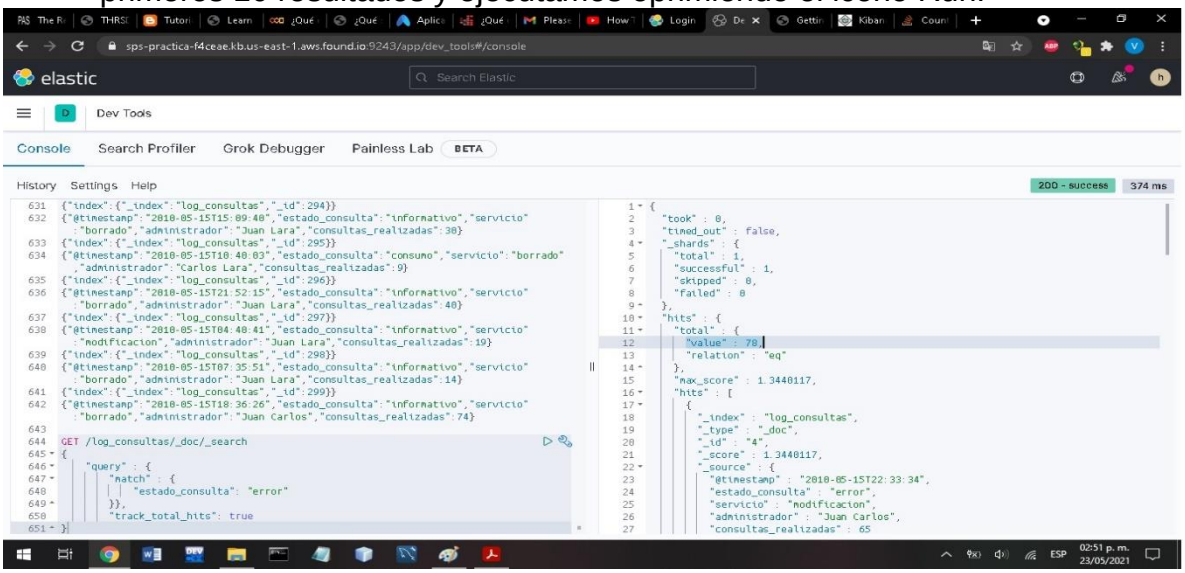
Realizar búsquedas sobre el índice

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

- Para realizar esta búsqueda con la Search API se procedió a escribir el siguiente comando, consulta al índice, con los siguientes parámetros:

```
GET /log_consultas/_doc/_search
{
   "query" : {
      "match" : {
         "estado_consulta": "error"
      }},
      "track_total_hits": true
}
```

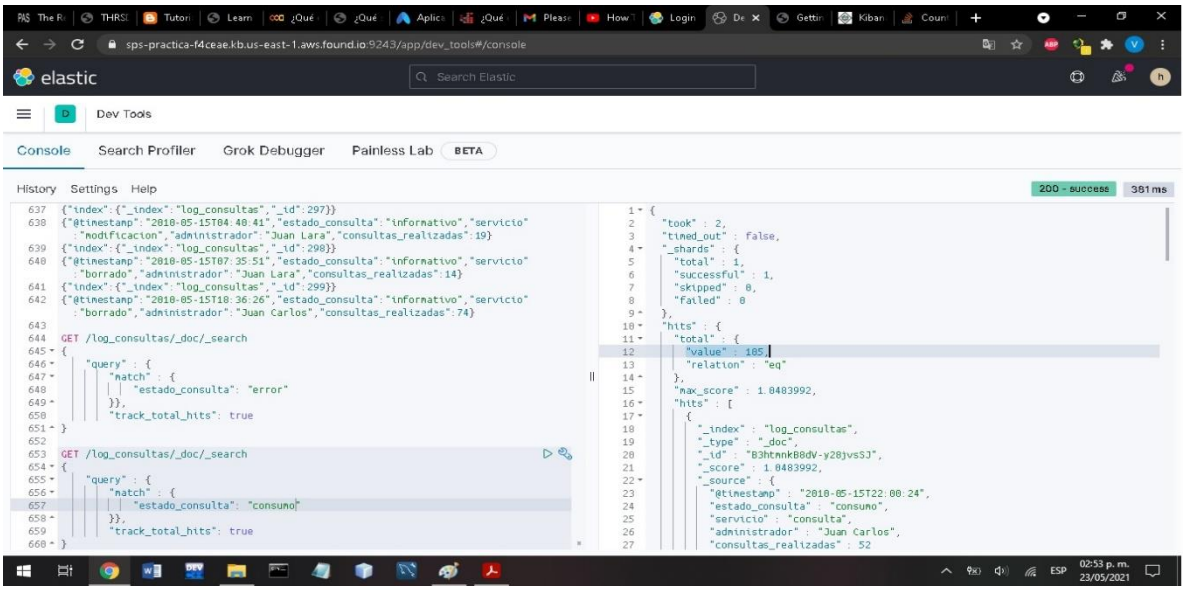
- El query anterior con notación de REST API nos dice que hay que buscar (/search) en el índice log_consultas dentro de todos los documentos (_doc) a los documentos que hagan coincidan (hagan match) con los parámetros de búsqueda que son aquellos valores en los que el campo “estado_consulta” sea igual a “error”, además se le agrega el parámetro track_total_hits: true el cual le dice a la consulta que regrese el número de documentos totales que coinciden con la búsqueda; ya que por default la búsqueda solo regresa los primeros 10 resultados y ejecutamos oprimiendo el icono Run:



- Y nos devuelve un total de 78 registros que empataron con la búsqueda como se muestra en la imagen anterior resaltado en azul con el cursor.

- Para la siguiente búsqueda fue algo similar a la anterior cambiando el valor del campo “estado_consulta” igual a “consumo”:

```
GET /log_consultas/_doc/_search
{
  "query" : {
      "match" : {
        "estado_consulta": "consumo"
      }},
      "track_total_hits": true
}
```



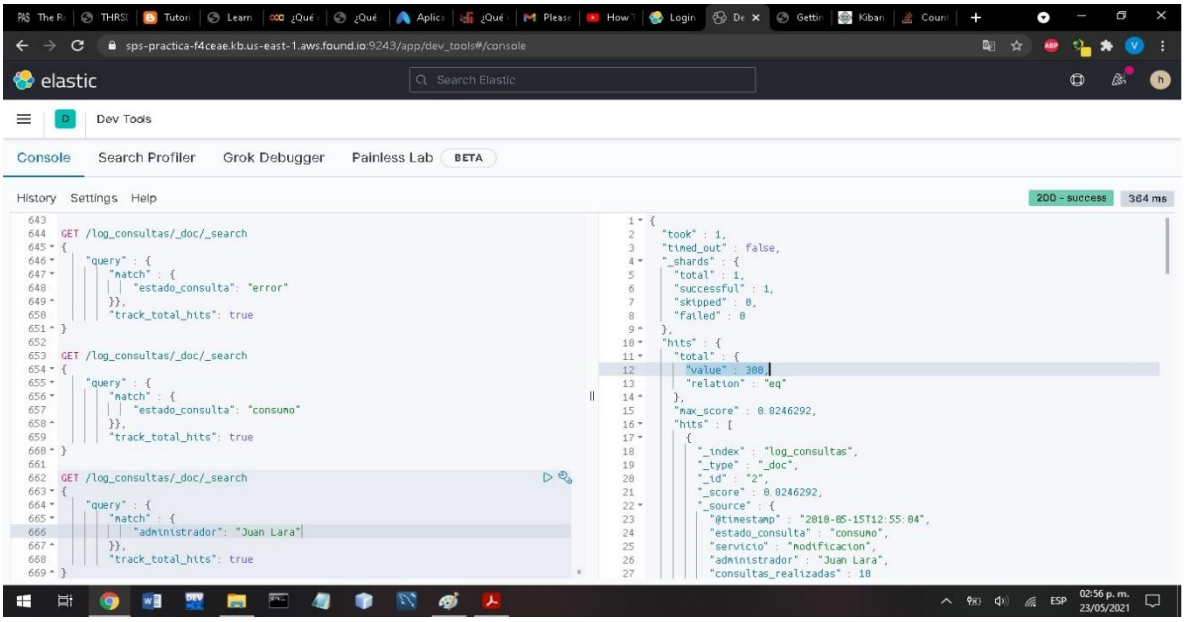
- Y nos devuelve un total de 185 registros que empataron con la búsqueda como se muestra en la imagen anterior resaltado en azul con el cursor.

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

- Para realizar la siguiente búsqueda se procedió a escribir el siguiente comando con los parámetros de búsqueda:

```
GET /log_consultas/_doc/_search
{
    "query" : {
    "match" : {
      "administrador": "Juan Lara"
    }},
    "track_total_hits": true
 }
```

- El query anterior con notación de REST API nos dice que hay que buscar (/search) en el índice log_consultas dentro de todos los documentos (_doc) a los documentos que hagan coincidan (hagan match) con los parámetros de búsqueda que son aquellos valores en los que el campo “administrador” sea igual a “Juan Lara” y oprimimos el icono Run para ejecutar la busqueda:



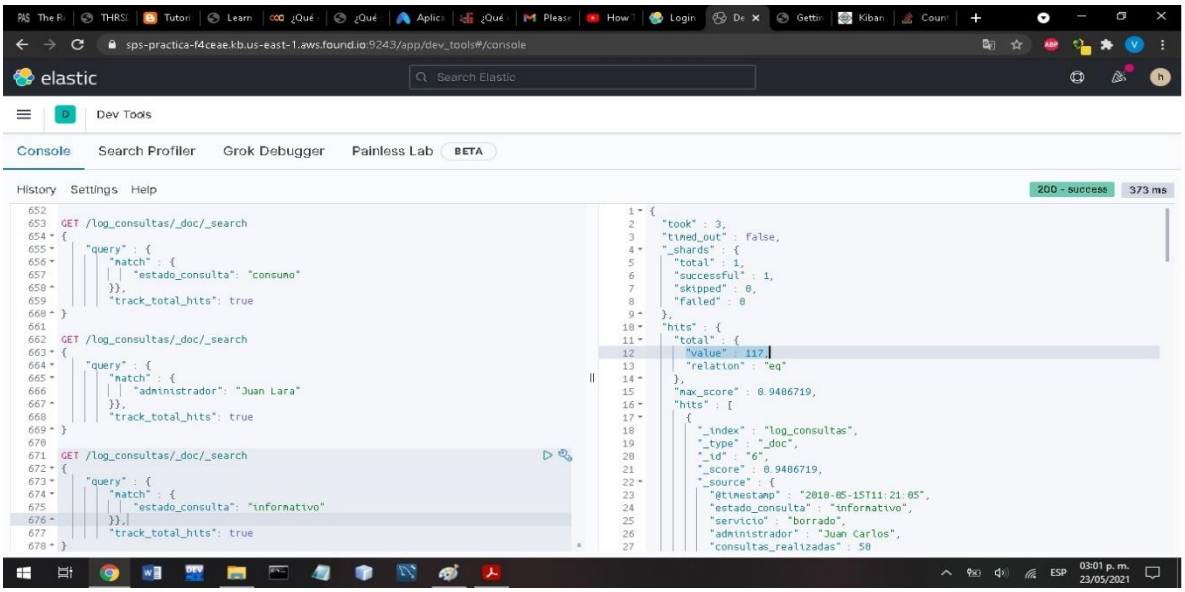
- Y nos devuelve un total de 300 registros que empataron con la búsqueda como se muestra en la imagen anterior resaltado en azul con el cursor.

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

- Para realizar la siguiente búsqueda se procedió a escribir el siguiente comando con los parámetros de búsqueda:

```
GET /log_consultas/_doc/_search
{
   "query" : {
      "match" : {
         "estado_consulta": "informativo"
      }},
      "track_total_hits": true
}
```

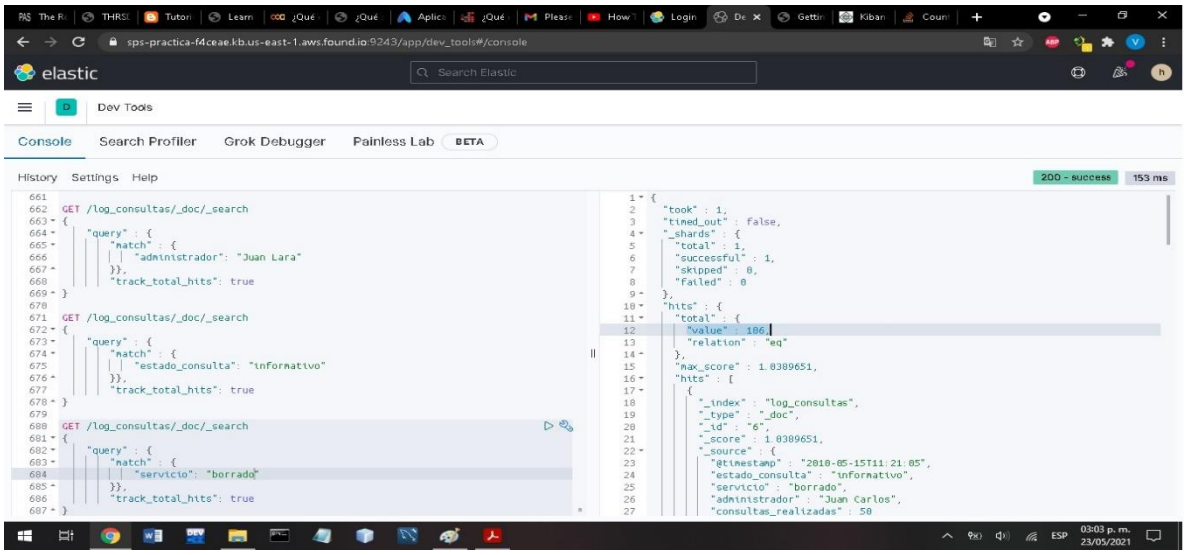
- El query anterior con notación de REST API nos dice que hay que buscar (/search) en el índice log_consultas dentro de todos los documentos (_doc) a los documentos que hagan coincidan (hagan match) con los parámetros de búsqueda que son aquellos valores en los que el campo “estado_consulta” sea igual a “informativo” y oprimimos el icono Run para ejecutar la busqueda:



- Y nos devuelve un total de 117 registros que empataron con la búsqueda como se muestra en la imagen anterior resaltado en azul con el cursor.

- Para la siguiente búsqueda fue algo similar a la anterior cambiando el valor del campo “servicio” igual a “borrado”:

```
GET /log_consultas/_doc/_search
{
   "query" : {
      "match" : {
         "servicio": "borrado"
      }},
      "track_total_hits": true
}
```



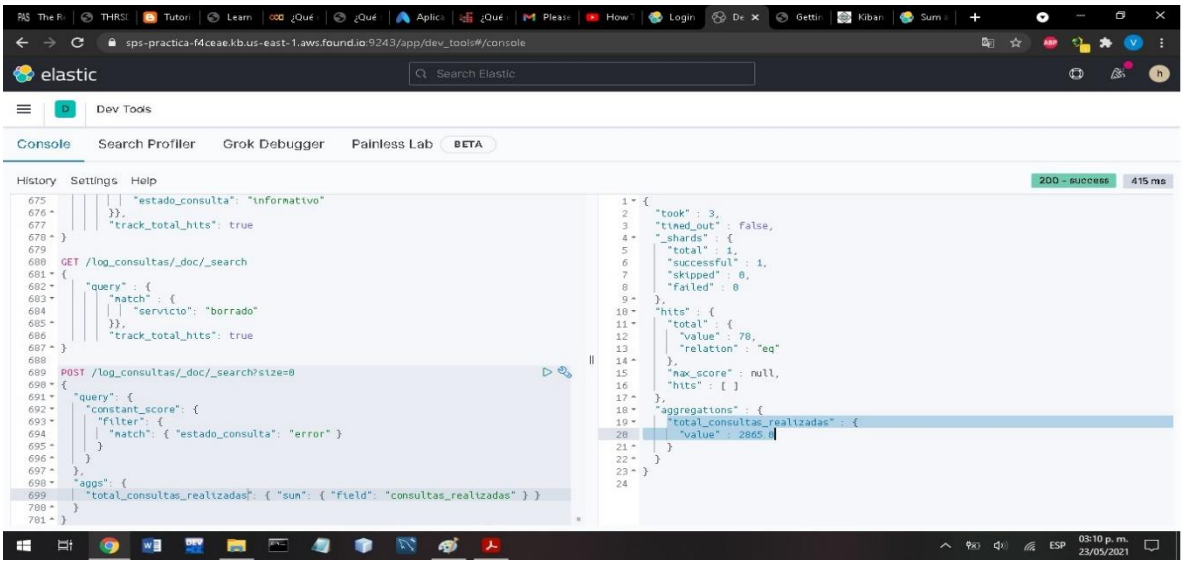
- Y nos devuelve un total de 106 registros que empataron con la búsqueda como se muestra en la imagen anterior resaltado en azul con el cursor.

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

- Para realizar esta búsqueda con la Search API se procedió a escribir el siguiente comando, consulta al índice, con los siguientes parámetros y agregaciones:

```
POST /log_consultas/_doc/_search?size=0
{
  "query": {
    "constant_score": {
      "filter": {
        "match": { "estado_consulta": "error" }
      }
    }
  },
  "aggs": {
    "total_consultas_realizadas": { "sum": { "field": "consultas_realizadas" } }
  }
}
```

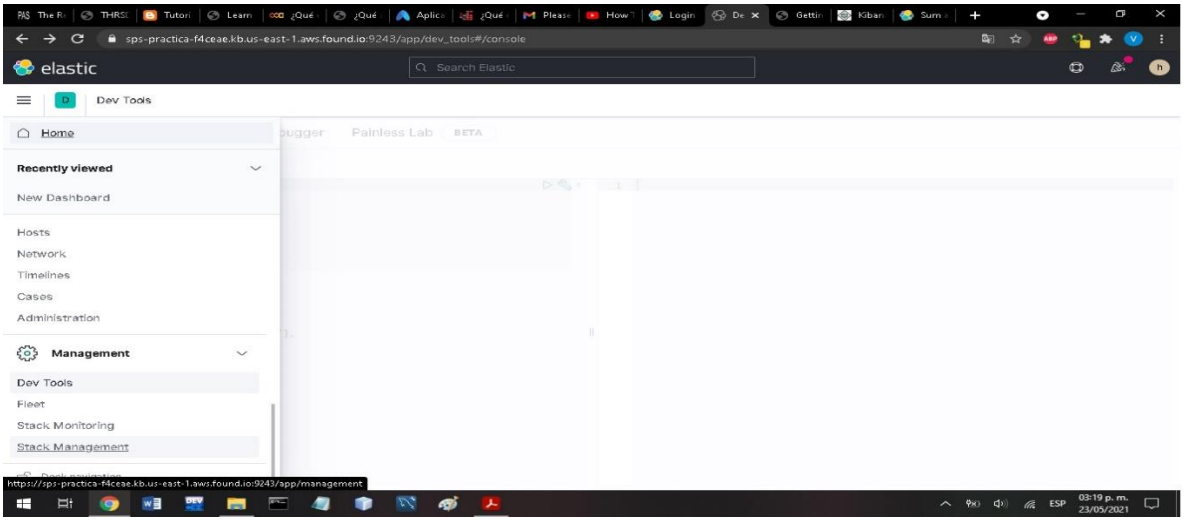
- El query anterior con notación de REST API nos dice que hay que buscar (/search) sin regreso de documentos (?size=0) en el índice log_consultas dentro de todos los documentos (_doc) a los documentos que hagan coincidan (hagan match) con los parámetros de búsqueda que son aquellos valores en los que el campo “estado_consulta” sea igual a “error” pero además le ponemos como agregación que nos dé un campo llamado total_consultas_realizadas el cual tendrá como valor la suma de todos los valores del campo de consultas_realizadas y oprimimos el icono Run para ejecutar la búsqueda:



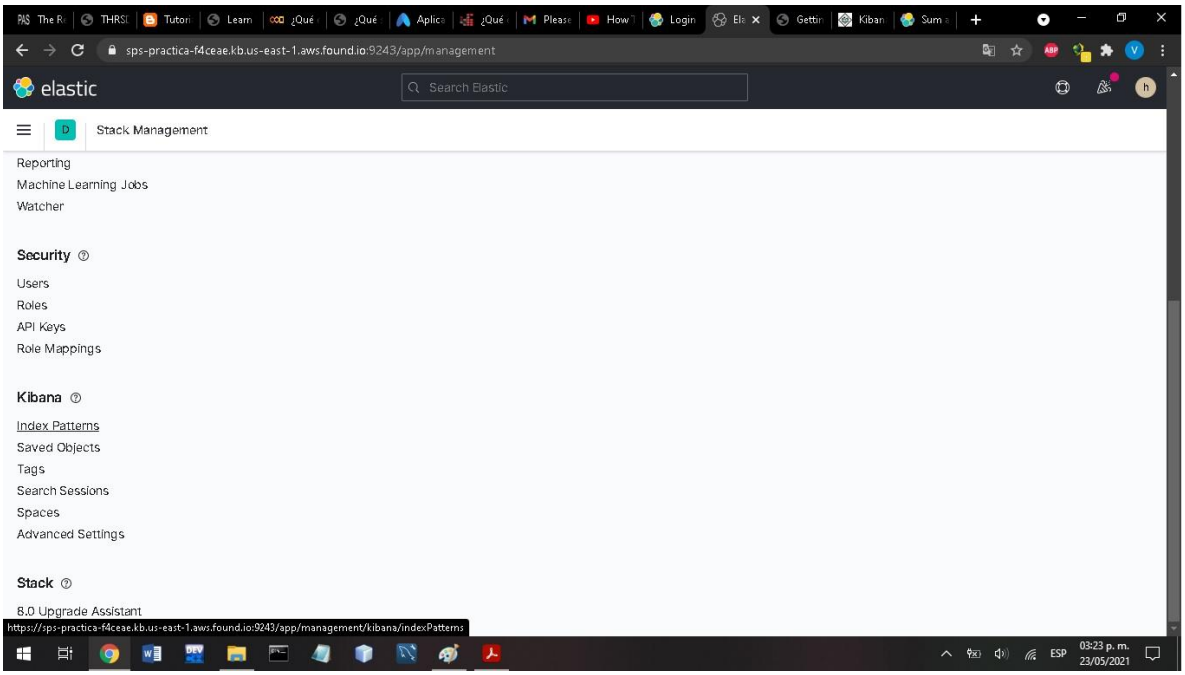
- Y nos devuelve un la suma de consultas realizadas, 2865 consultas realizadas, de los 78 documentos que empataron con la búsqueda como se muestra en la imagen anterior resaltado en azul con el cursor.

Realizar un tablero para visualizar información de empleados

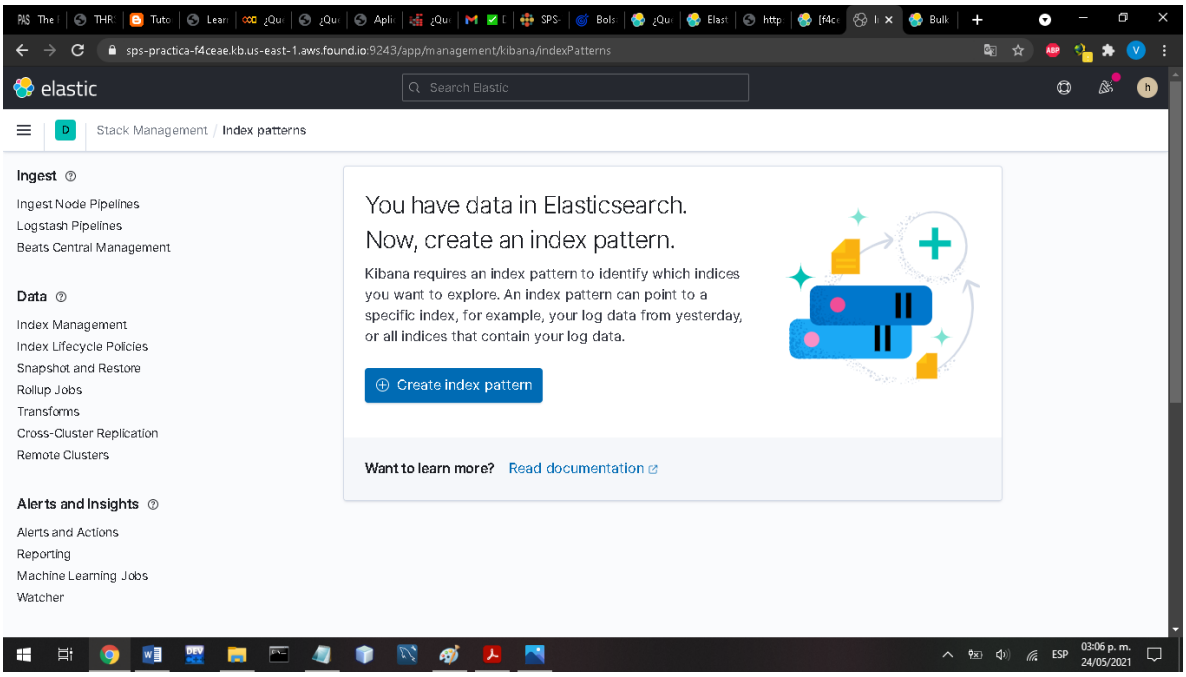
Crea un patrón de índice en Kibana, dirígete a la opción de Management y selecciona la opción Kibana > Index Patterns y selecciona el índice que creaste en los pasos anteriores log_consultas tomando como Time Filter el campo de @timestamp.



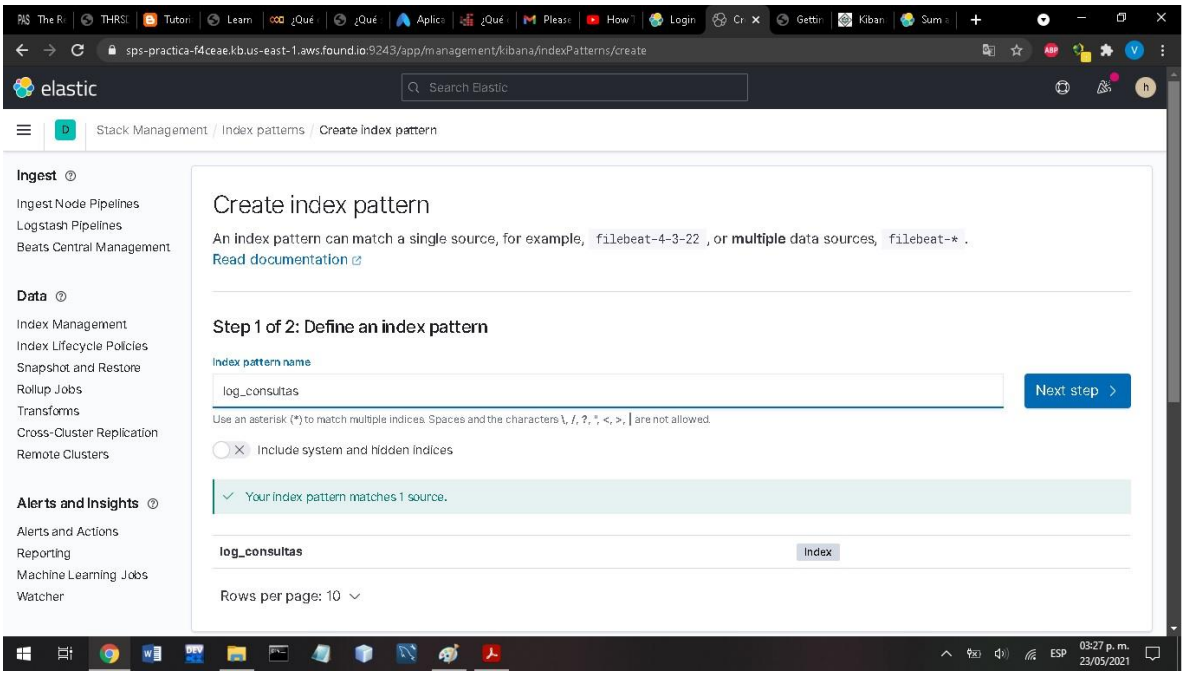
- Despues de seleccionar Stack Management en el menú vertical hacemos click en Index Patterns:



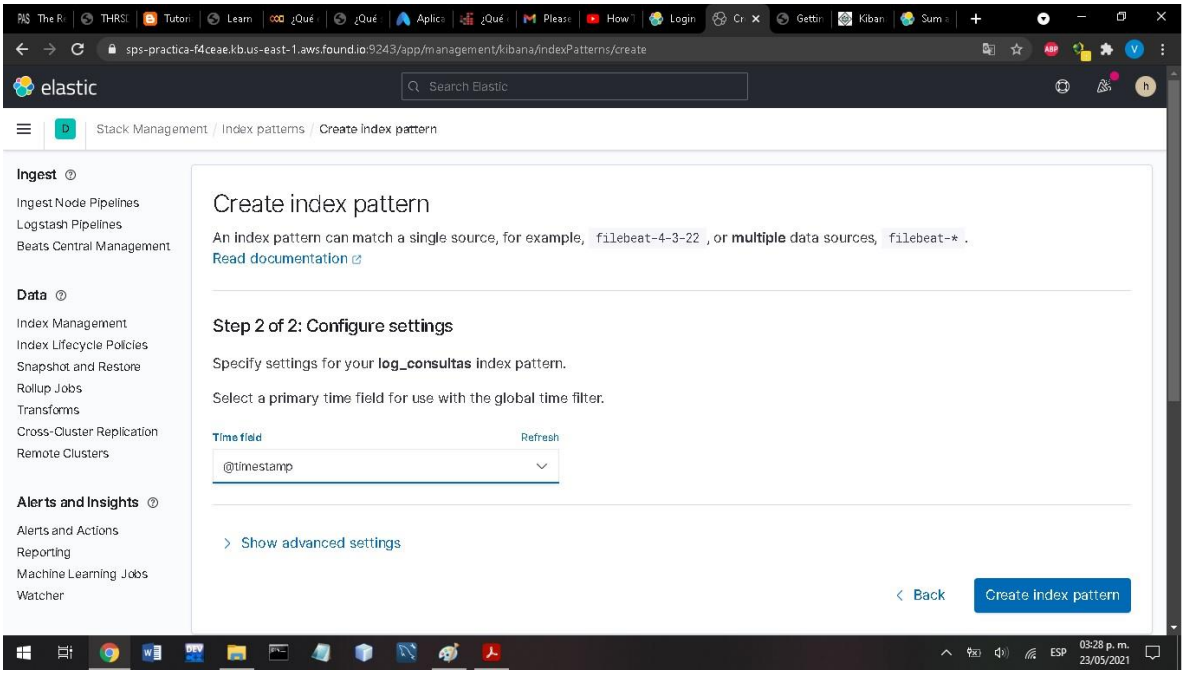
- Oprimimos en create index pattern



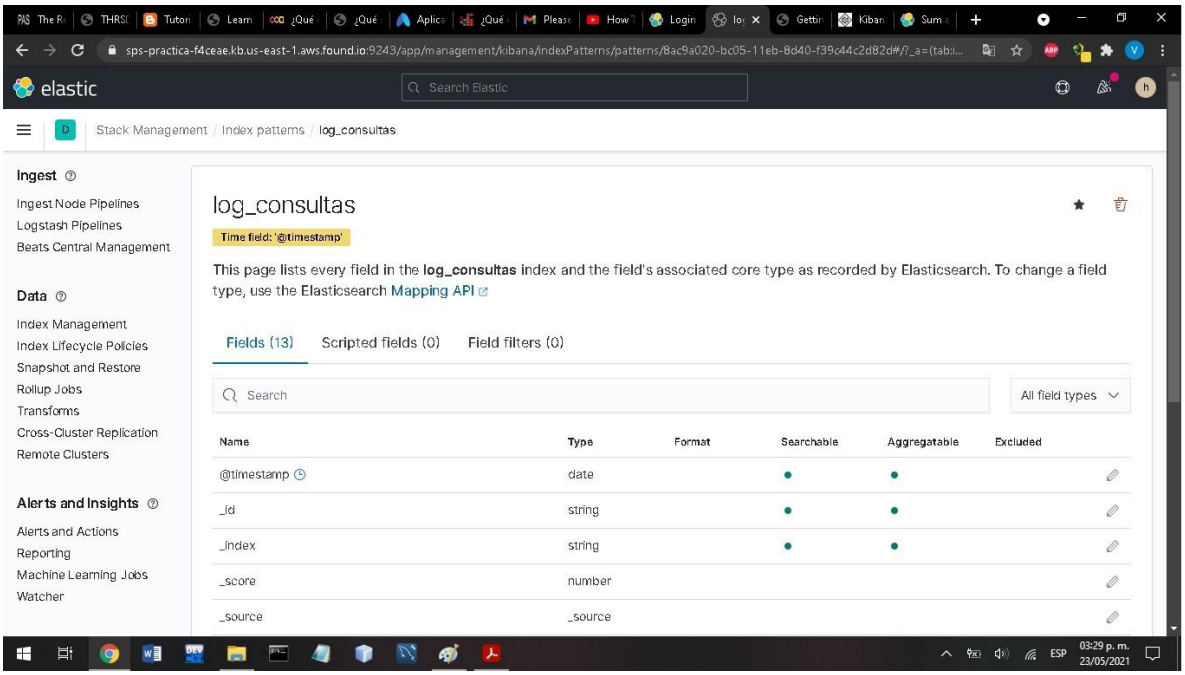
- Escribimos el nombre del patrón de índice log:consultas y damos click en Next Step como se muestra a continuación:



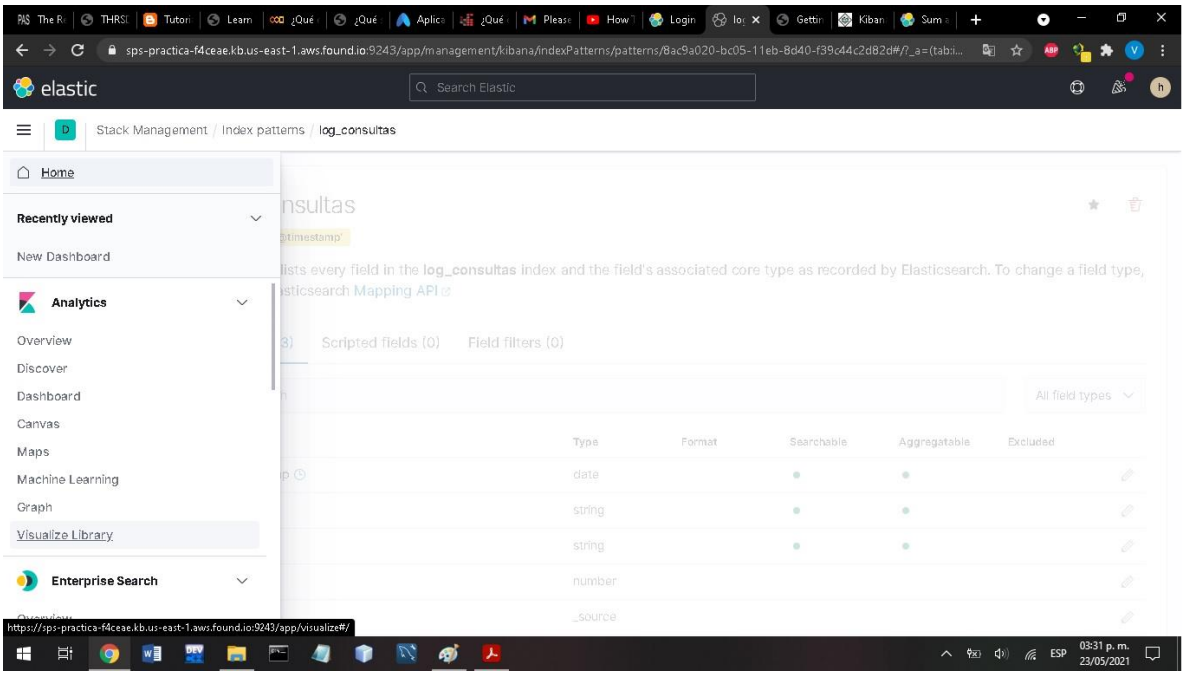
- En el selector debajo de Time Field seleccionamos @timestamp y damos click en create index pattern:



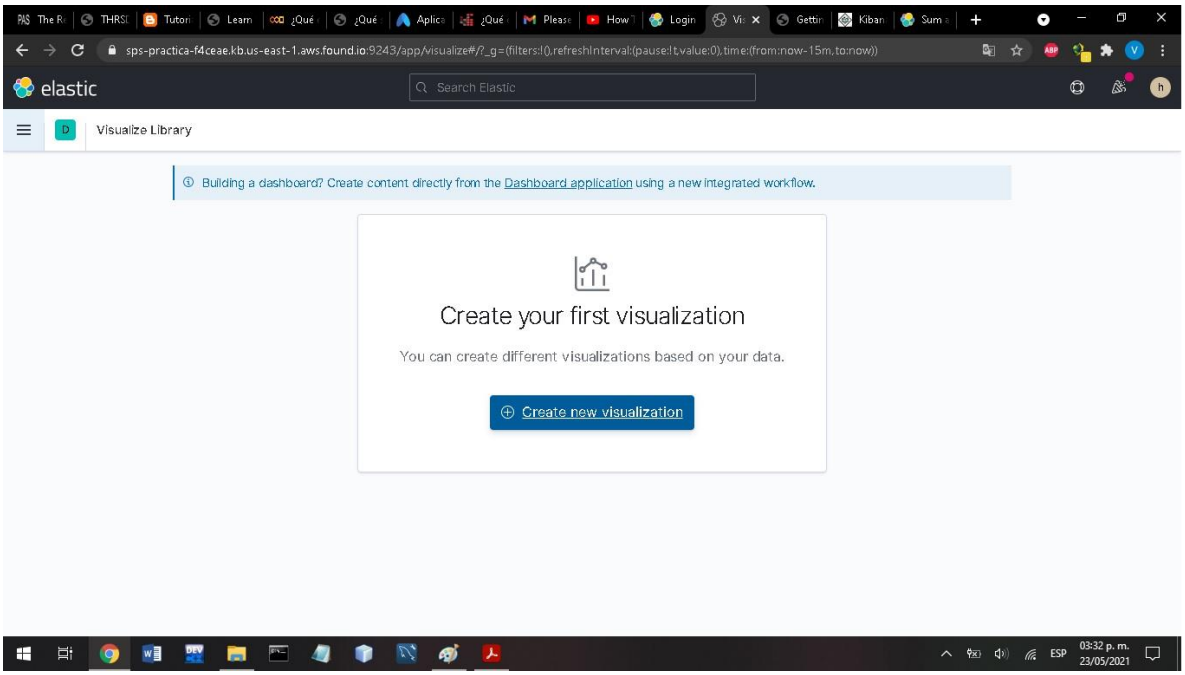
- Al terminar de crear el index patern debe observarse una pantalla como la siguiente:



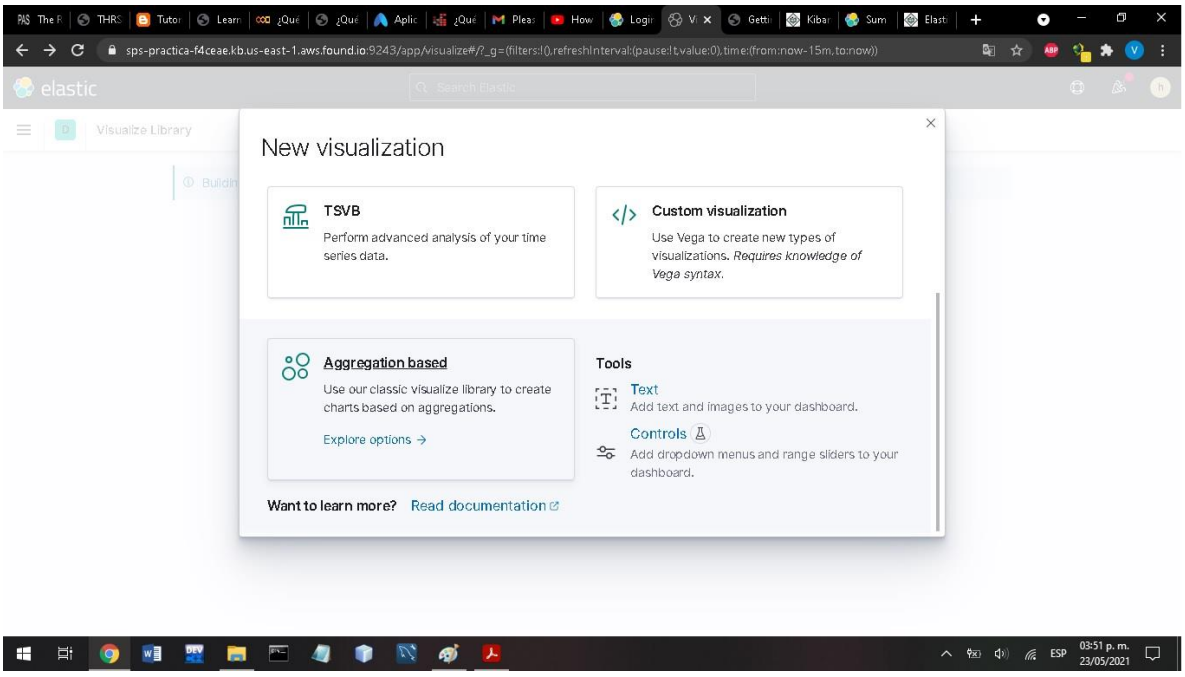
- En el menú vertical damos click en Visualize Library



- Creamos una nueva visualización.

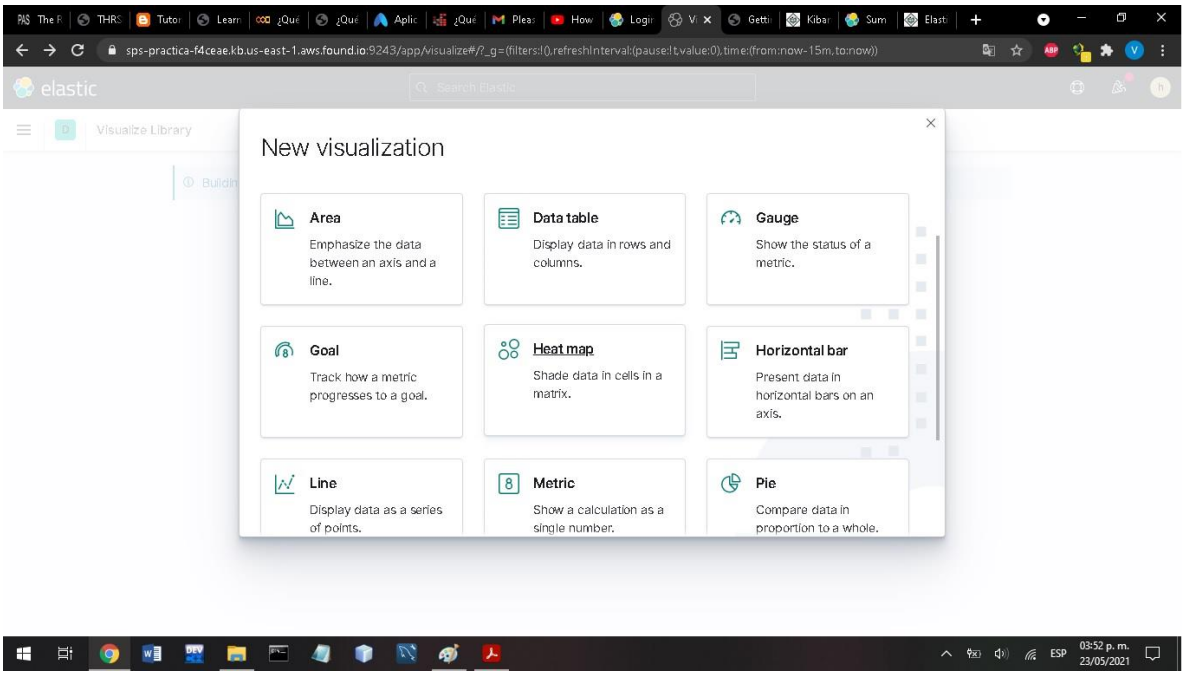


- Despues damos click en Agreggation based

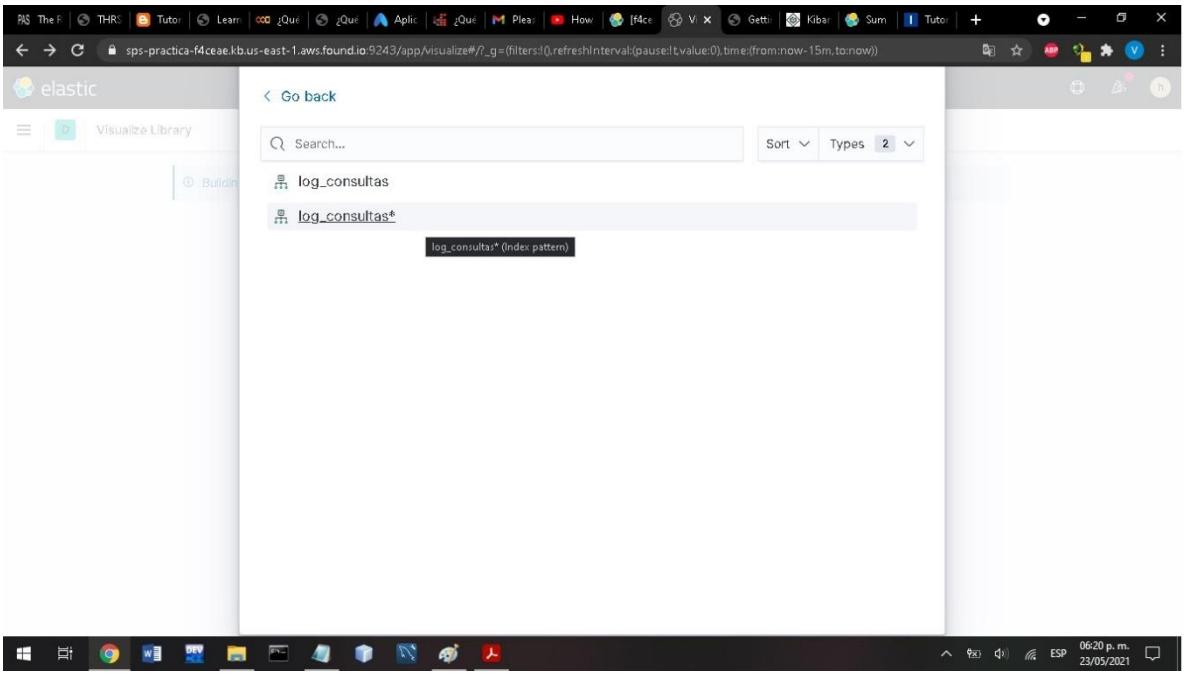


1. Vista de heat map, donde mostraras el número de servicios realizados por administrador.

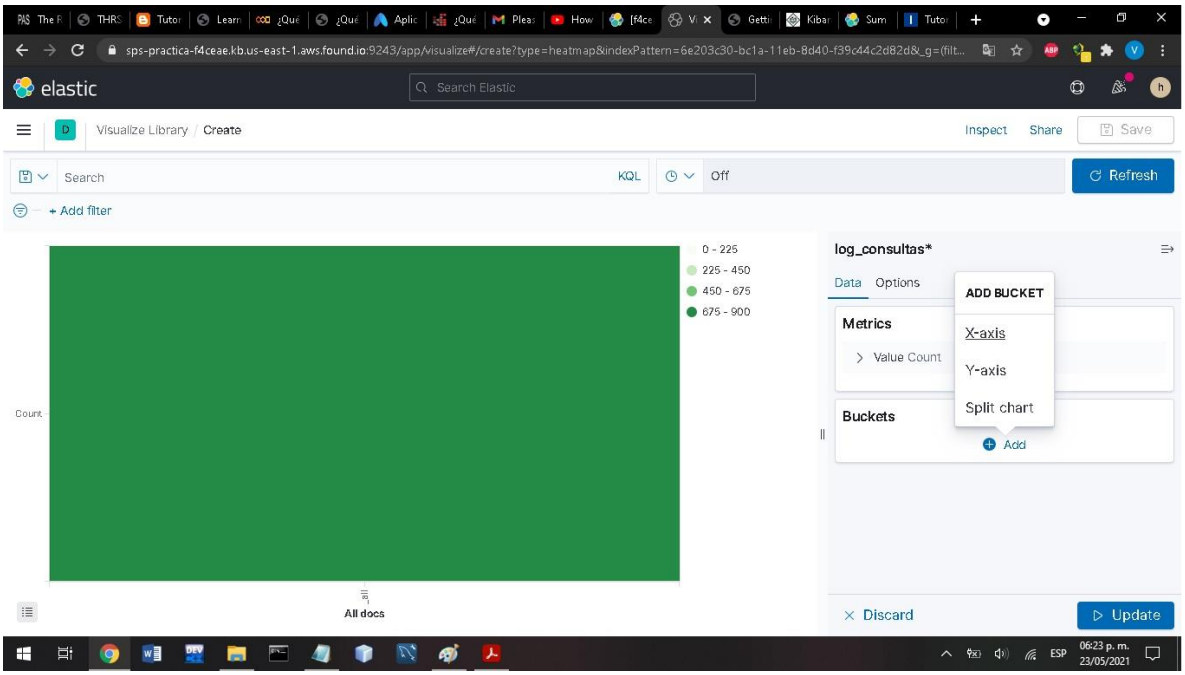
- Despues damos click en Heat Map



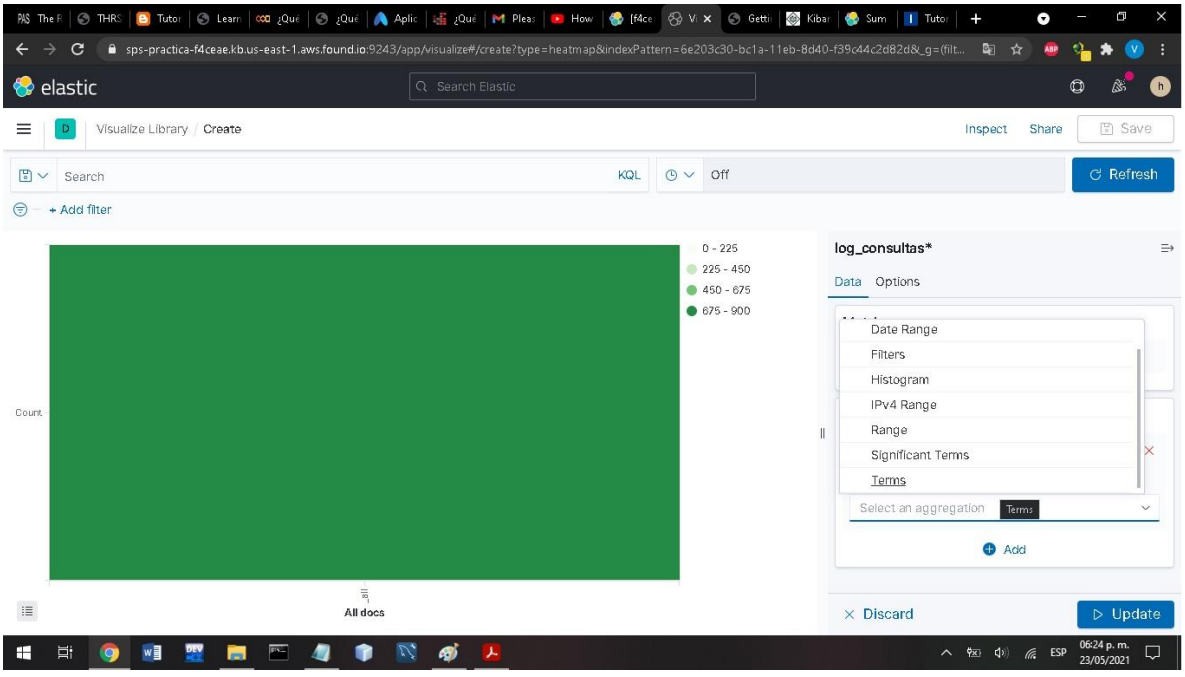
- Enseguida seleccionamos el log_consultas:



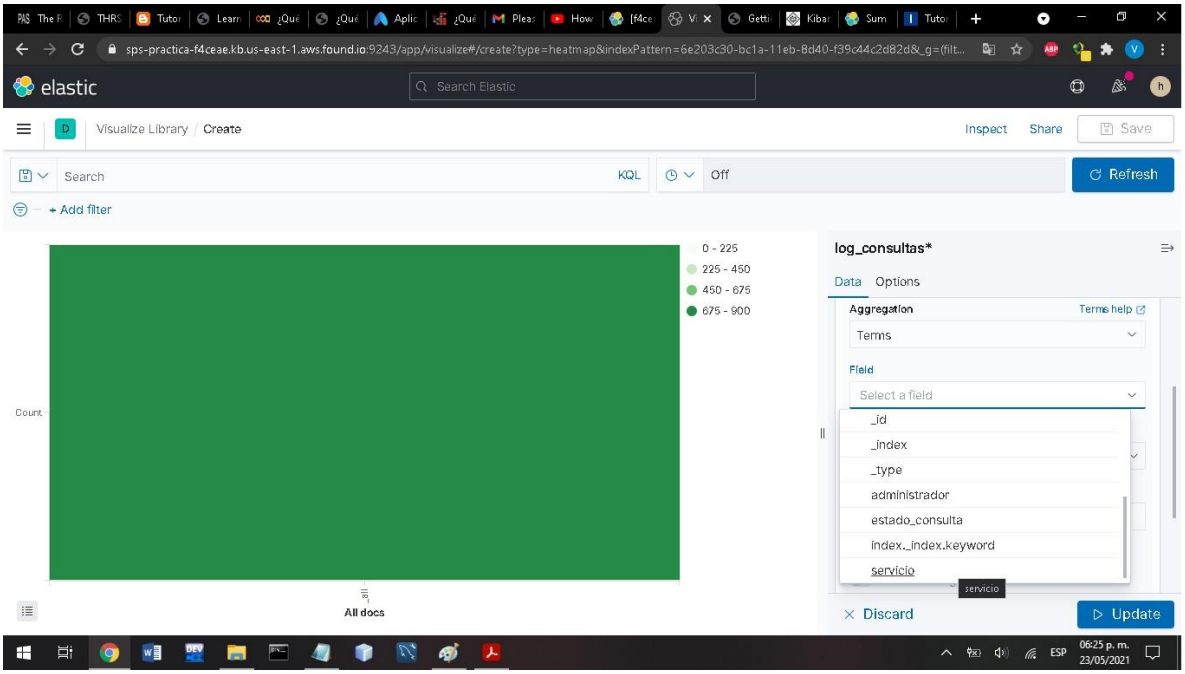
- Entonces visualizaremos una pantalla como la siguiente e iremos a la opción add y damos click en X-axis con el fin de añadir un parámetro en sobre el eje de las X:



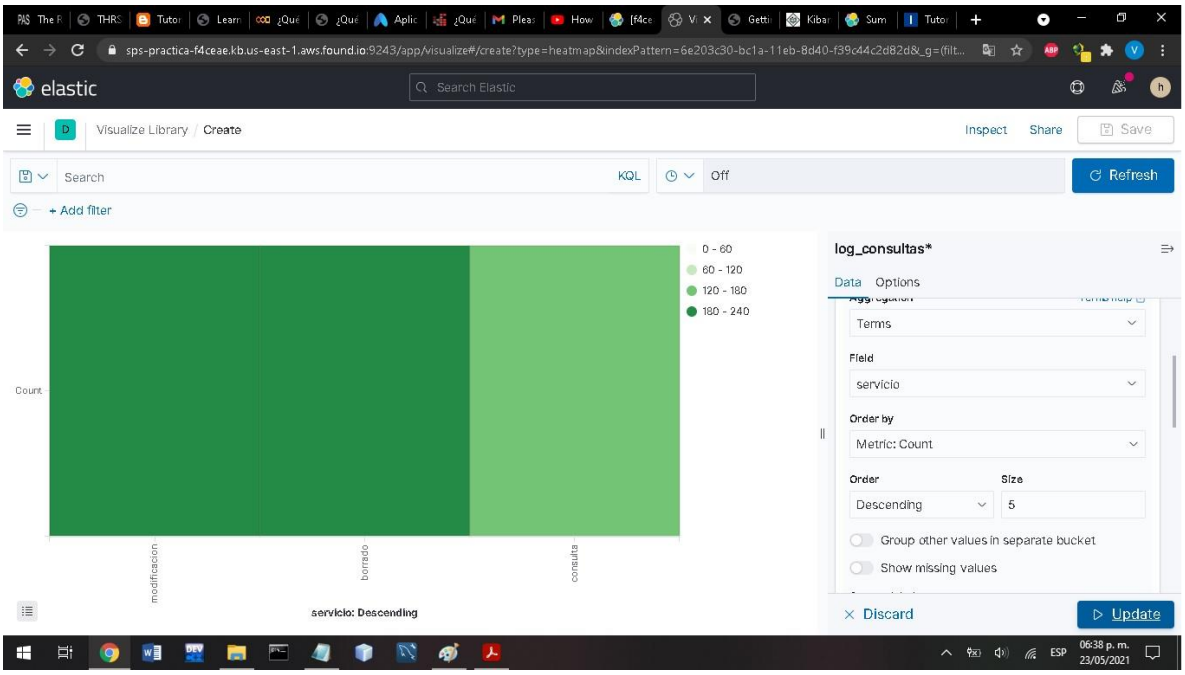
- En el campo agreggation seleccionamos Terms como se muestra a continuación:



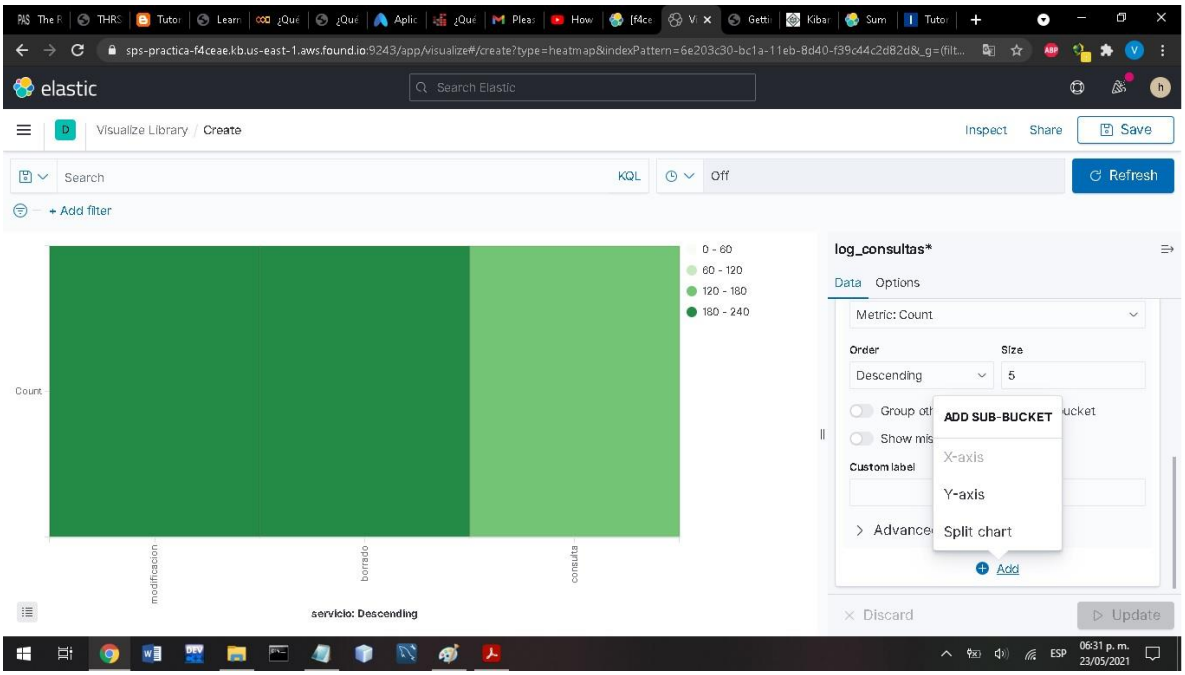
- Después en el campo Field seleccionamos servicio como se muestra a continuación:



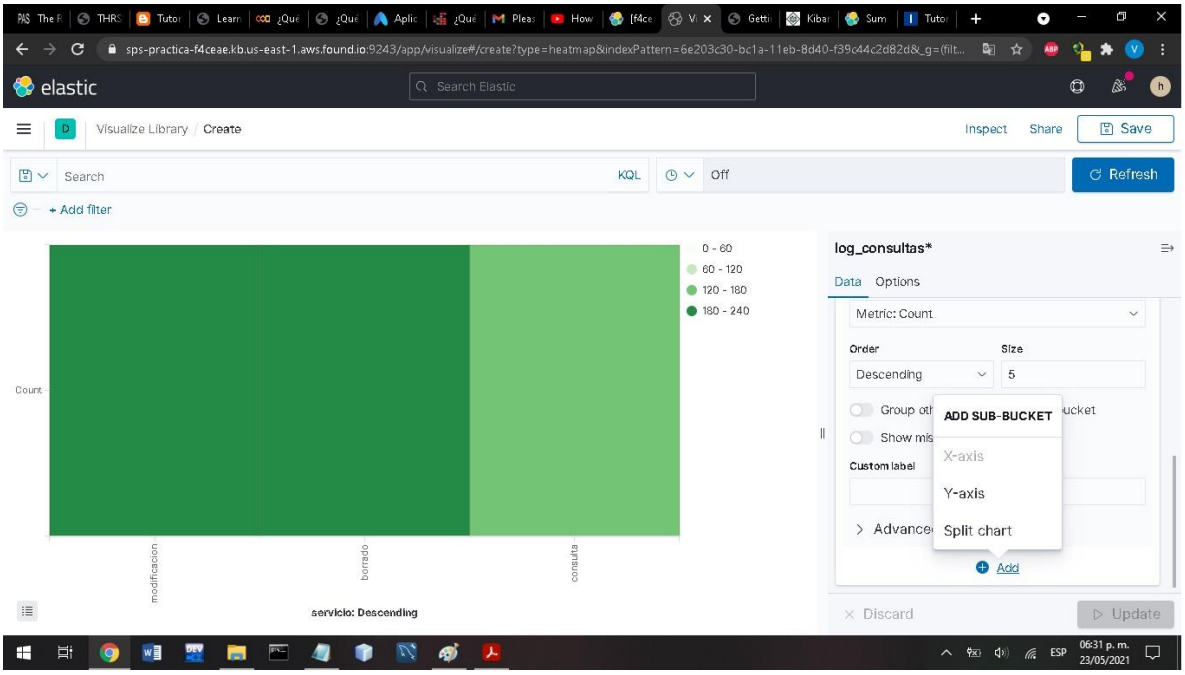
- Y damos click en Update para visualizar los tres valores claves del campo servicio de nuestro indice como sigue:



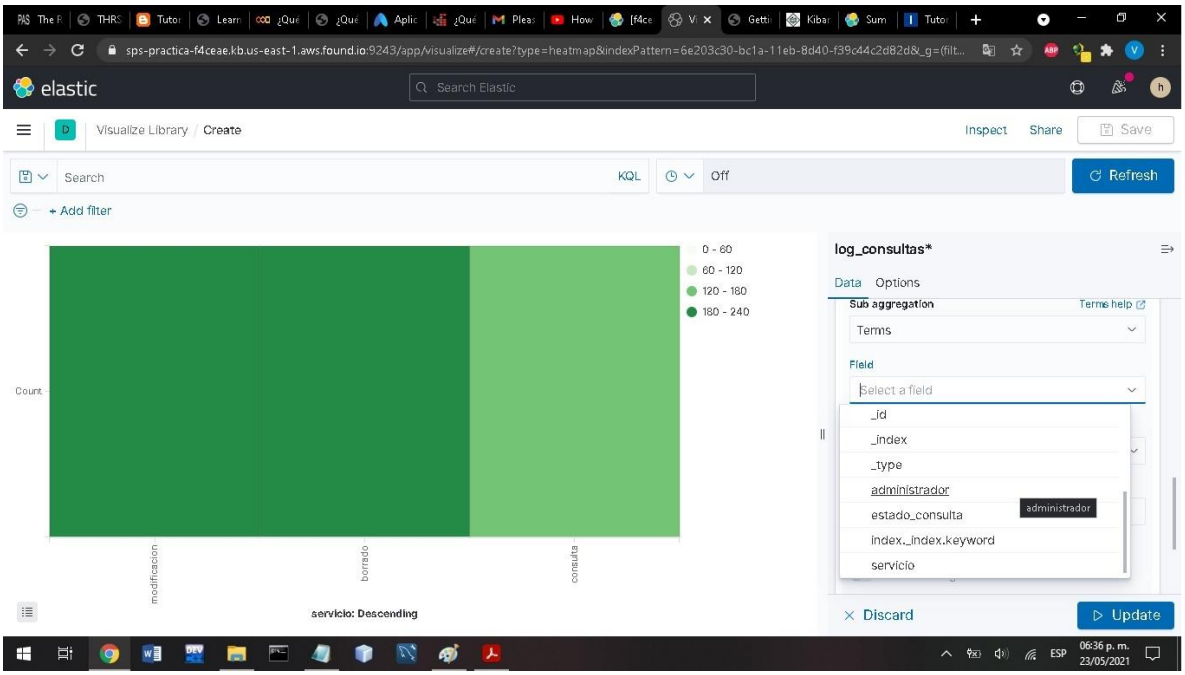
- Enseguida vamos a la opción Add para añadir Y-axis y agregar el parámetro para el eje de las Y:



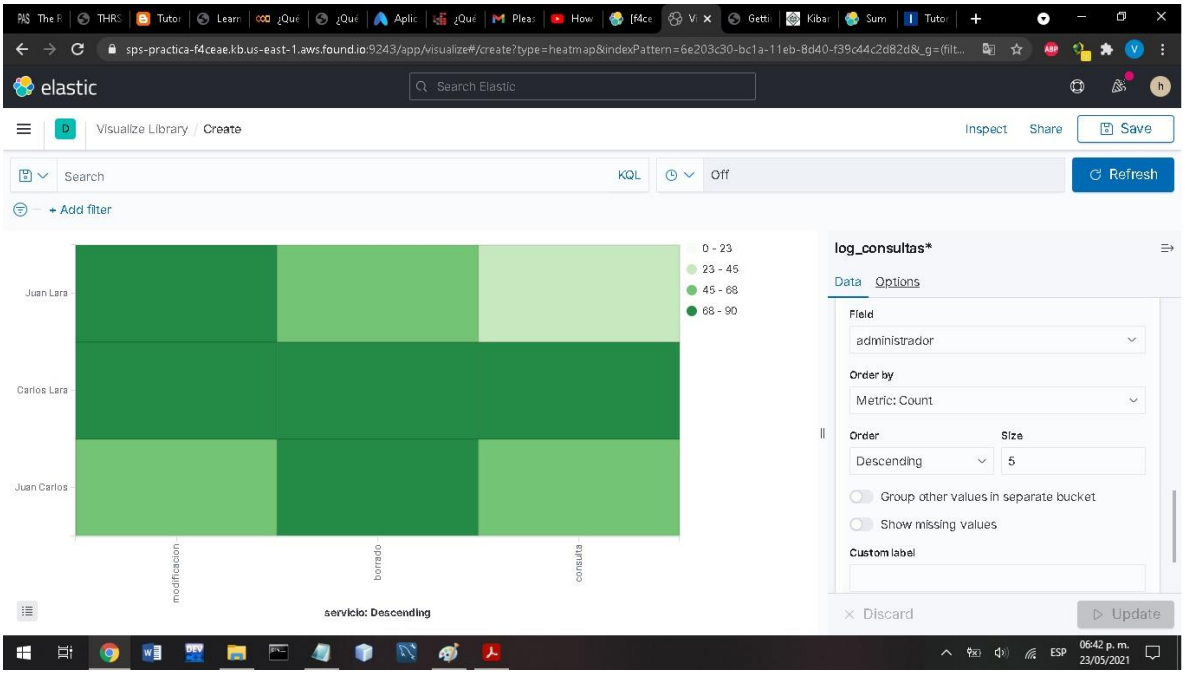
- En el campo agreggation seleccionamos Terms como se muestra a continuación:



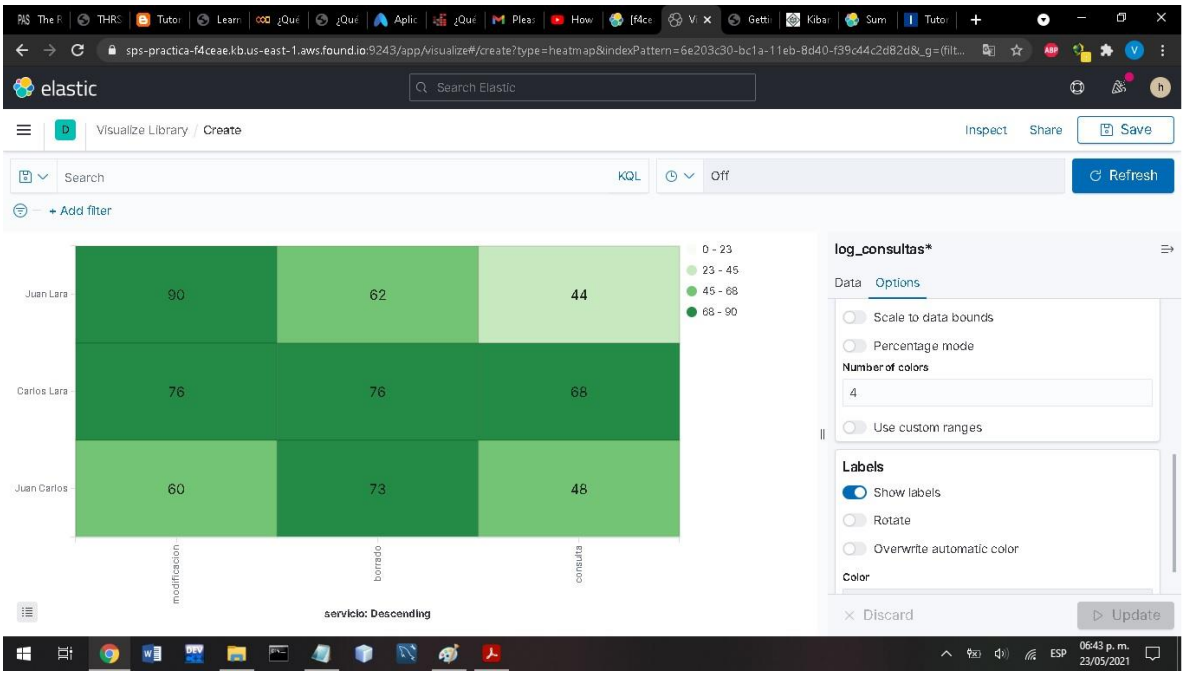
- En el campo Field seleccionamos administrador como se muestra a continuación:



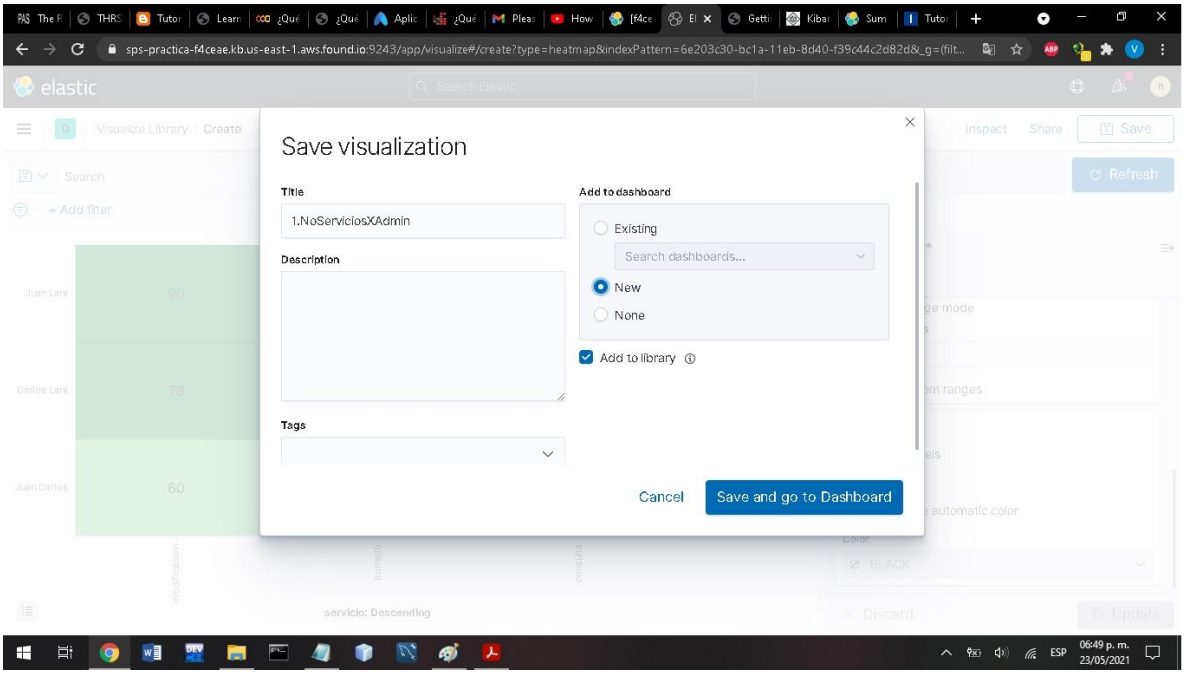
- Y damos click en Update para visualizar los tres valores claves del campo administrador de nuestro indice como sigue:



- Para mostrar el valor de cada recuadro de color vamos a la pestaña options y habilitamos la opción show labels:



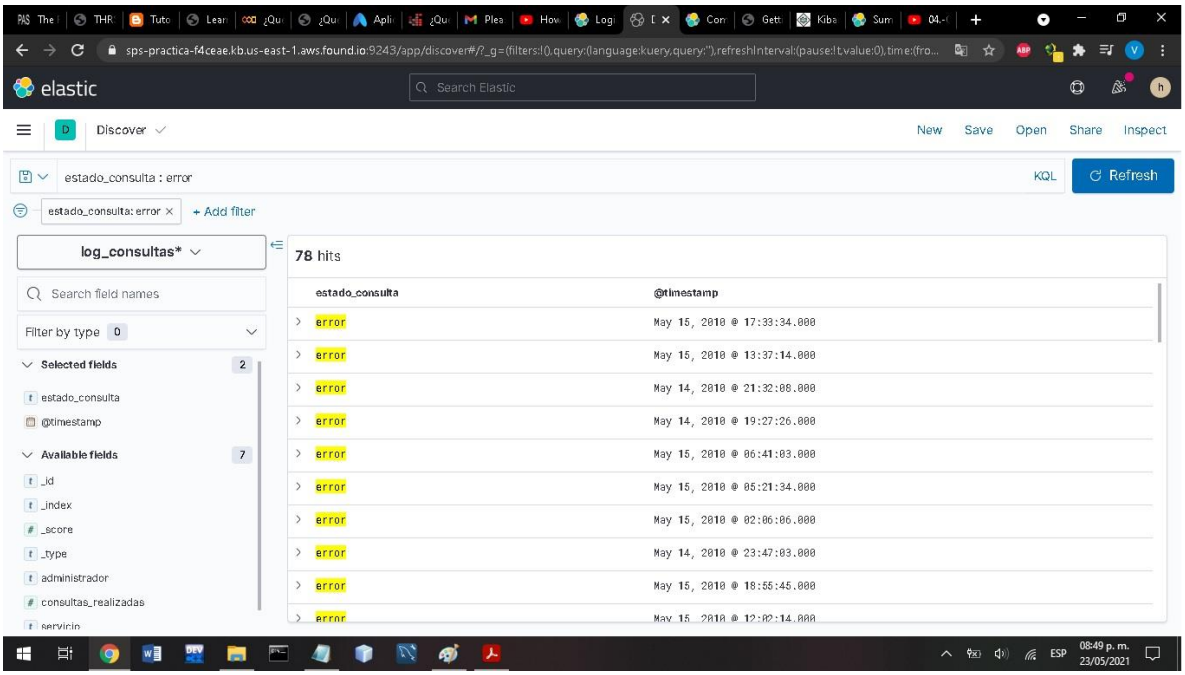
- Y por último, damos guardar en el dashboard para ponerlo en el tablero (EXTRA 1).



2. Vista de Barras, donde se grafique el número de registros con estado_consulta igual a error a través del tiempo.

- Al crear el patrón de índice como log_consultas filtrado con el timestamp, no obtenía resultados a la hora de vaciar la información en la gráfica.

- Era un error relacionado con el rango de tiempo que no pude corregir ni encontrarle solución.
- Pero a través de la opción Discover pude hacer un filtrado de la información para obtener los resultados que se esperaban en la gráfica. Por cierto en el Discover también debía mostrar una gráfica que jamás mostró como se puede ver en la siguiente imagen:



- Como se puede ver se obtienen los 78 registros con el estado de consulta igual a error y la fecha en la que se realizaron. Por encima del primer registro debio mostrarse la gráfica cosa que no pasó.
- Para lograr lo anterior se eligieron los dos campos: estado_consulta y @timestamp y, además, agregar un filtro eligiendo el campo estado_consulta con el operador is y el valor error quedando como estado_consulta:error.

Conclusión

La aplicación ElasticSearch es muy poderosa y mucho más a través de su API REST del cual se obtiene resultados más rápidamente. Al trabajar con su versión web te das cuenta que no es necesario instalar nada en el ordenador como se muestra en variedad de videos tutoriales.

Es conveniente leer la documentación de sus APIs para poder trabajar por medio de la herramienta Dev Tools. Dentro de la documentación existen ejemplos que te muestran como formular los comandos que necesitas desde creación de índices, pasando por agregar documentos en formato json y las importantísimas búsquedas. Se puede aprender en poco tiempo.

Para la parte Visualize Library con las gráficas es una herramienta muy poderosa que te simplifica mucho el trabajo al ofrecerte diferentes tipos de grafica además de que agrega la parte de los campos clave que se encuentran en el índice con sus respectivos documentos y filtros para realizar las gráficas con parámetros definidos con los valores de cada campo clave. Para la parte de las gráficas basadas en tiempo se requieren más conocimientos para agregar rangos de tiempo cosa que se me dificulto demasiado.

Referencias

https://www.elastic.co/es/what-is/elasticsearch

https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html

https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-bulk.html

https://assets.contentstack.io/v3/assets/bltefdd0b53724fa2ce/blt56ad3f4e2c755f29/5d37c1602a506857d64eff48/es_commands.txt

ElasticSearch Basics and Dev Tools Workshop - part 1 | Idan Fridman

ElasticSearch Basics and Dev Tools Workshop - part 2 | Idan Fridman

Curso Elastic Search - Capítulo I

Elasticsearch Kibana - How import JSON File to Kibana

How to load bulk data into ElasticSearch Index